МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

На правах рукописи

Булаев Алексей Александрович

РАЗРАБОТКА СИСТЕМ ПРОЕКТИРОВАНИЯ 3D ГИС И КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ ТРЁХМЕРНОЙ СИТУАЦИОННОЙ ОБСТАНОВКИ

Специальность: 05.13.12 – Системы автоматизации проектирования (промышленность)

Диссертация на соискание ученой степени кандидата технических наук

Научный руководитель – доктор технических наук, профессор,
Смагин Алексей Аркадьевич

Ульяновск – 2018

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность. Развитие информационных и телекоммуникационных технологий обусловило переход всех картографических и геоинформационных систем от 2D вида к 3D. Быстрый рост технологий разработки и создания 3D ГИС влечёт за собой их применение в различных сферах жизнедеятельности человека, таких как: гражданская и военная отрасли, телекоммуникации, здравоохранение, геодезия, нефтегазовая промышленность, городское планирование, коммунальные услуги, транспорт, экология и другие.

Лидером среди 2D и 3D ГИС систем является наиболее востребованное семейство продуктов в области картографии и геоинформационных систем ArcGIS, которое позволяет визуализировать (представить в виде цифровой карты) большие объёмы статистической информации, имеющей географическую привязку. В среде создаются и редактируются карты всех масштабов: от планов земельных участков до карты мира. Также в ArcGIS встроен широкий инструментарий анализа пространственной информации. [120]

Одной из ведущих организаций на Российском рынке геоинформационных технологий является КБ «Панорама», занимающаяся разработкой и внедрением программно-аппаратных средств с применением геоинформационных систем и технологий, которые используются как в народном хозяйстве, так и в интересах обороны страны (профессиональная ГИС "Панорама", муниципальная ГИС "Земля и Недвижимость", GIS WebServer SE, визуальные компоненты GIS ToolKit, ГИС для сельского хозяйства, ГИС для операционных систем Linux, Solaris, Windows Mobile и другое); разработкой корпоративных информационных систем; созданием картографических Интернет-сайтов. [31,51,52,110]

Среди свободно распространяемых ГИС широко известны QuantumGIS, gvSIG, GRASS GIS, но эти системы имеют ограниченный спектр возможностей (разработка сообществом программистов разного уровня, большая вероятность наличия ошибок в новых модулях, трудная расширяемость новыми функциями и форматами геоинформационных данных) для 3D визуализации.

Проектирование 3D ГИС вызывает определенные сложности в силу необходимости работы с пространственными данными, 3D моделями и отображением изменений ситуационной обстановки. Всё это требует высокой квалификации проектировщика, который знаком с проектированием информационных систем и одновременно с технологиями геоинформатики. Использование 3D ГИС отображения обстановки позволяет улучшить качество принимаемых решений за счёт её визуализации в трёхмерном виде и возможностей её моделирования. [23,117]

В силу того, что 3D ГИС вошли в нашу жизнь, к ним возрос интерес со стороны тех организаций, которым необходимо их практическое применение. Само проектирование 3D ГИС представляет собой некоторый барьер для большинства пользователей, которым необходимо спроектировать качественную 3D ГИС. [23]

Трудности проектирования 3D ГИС могут быть уменьшены благодаря использованию средств автоматизации проектирования, готовых разработок, опыта проектировщика и используемых на практике методов к проектированию 3D ГИС.

Одним из эффективных средств автоматизации проектирования 3D ГИС является CASE-средство проектирования 3D ГИС, которое позволяет уменьшить временные затраты и ресурсы на создание информационной системы, сохраняя при этом необходимые качества. Такое средство носит специализированный характер, ориентируется на свою область применения, позволяет осуществлять генерацию проектных решений для создания и практического использования 3D ГИС в разнообразных областях применения. CASE-средство проектирования 3D ГИС позволяет устранить трудности проектирования и сделать 3D ГИС более доступными. [23]

В частности, CASE-средство проектирования 3D ГИС имеет возможность построения 3D ГИС отображения ситуационной обстановки с использованием множества библиотек из открытых ресурсов Интернет, позволяющих частично реализовать требуемую функциональность, а также их совмещение между собой и собственными разработками.

Цель и задачи исследования. Целью диссертации является анализ возможностей и разработка систем и средств автоматизации проектирования и моделирования 3D ГИС отображения ситуационной обстановки для снижения временных и финансовых затрат, уменьшения сложности проектирования и уровня знаний проектировщика.

Достижение поставленной цели потребовало решения следующих задач:

1. Исследование методов создания и функционирования 3D ГИС отображения морской, наземной и воздушной обстановок с использованием современных информационных технологий и широко применяемых форматов геоинформационных данных;
2. Разработка двухкомпонентной функционально-ресурсной модели проектирования 3D ГИС, обеспечивающей формирование функциональных покрытий проектируемой системы и поиск среди них наиболее оптимального по выбранным проектировщиком критериям.
3. Разработка системы автоматизации проектирования 3D ГИС на основе двухкомпонентной функционально-ресурсной модели, обеспечивающей снижение ресурсных затрат (время, финансы) и требований к разработчику, выполняющей

генерацию и оценку проектных решений для последующей их программной реализации.

4. Разработка алгоритмов проектирования 3D ГИС, моделей и диаграмм проектных решений, формирующих базу для создания специализированного CASE-средства проектирования 3D ГИС отображения ситуационной обстановки и генерации проектных решений и исходных кодов файлов заголовков для последующей разработки на их основе 3D ГИС.
5. Разработка специализированного CASE-средства проектирования 3D ГИС, обеспечивающего формирование и оценку готовых проектных решений, отвечающих требованиям на разработку 3D ГИС.
6. Разработка алгоритмов функционирования 3D ГИС отображения ситуационной обстановки, реализующих многорежимный приём и обработку данных и обеспечивающих отображение морской, наземной и воздушной ситуационных обстановок с заданной точностью.
7. Разработка системы 3D моделирования и оценки проектных решений, обеспечивающих визуализацию трёхмерной обстановки, взаимодействие с внешними системами и источниками обстановки и имитацию движения трёхмерных объектов.

Объект исследования – геоинформационные системы и технологии их проектирования в различных сферах деятельности.

Предметом исследования являются инструментальные средства проектирования и создания 3D ГИС отображения ситуационной обстановки.

Теоретико-методологическую основу исследования составили труды отечественных и зарубежных ученых, занимающихся теоретическими и практическими вопросами проектирования и разработки 3D ГИС.

Методы исследования. При решении поставленных в работе задач использовались методы системного анализа и математического моделирования, проектирования и разработки информационных систем, а также методы программирования**.**

Научная новизна:

1. разработана двухкомпонентная модель для проектирования 3D ГИС отображения ситуационной обстановки с использованием свободно распространяемых ресурсов (библиотек), включающая – функциональную модель формирования покрытий заданной функциональности 3D ГИС на основе матрицы соответствия доступных для использования функций и реализующих их библиотек, и – геометрическую модель в виде круговой диаграммы для оценки затрат на его разработку;

2. разработан комплекс моделей, диаграмм, которые формируют базу для создания ориентированного средства проектирования 3D ГИС отображения ситуационных обстановок, который в отличие от известных позволяет снизить требования к уровню подготовки проектировщика и получать достаточно эффективные проектные решения;
3. разработан алгоритм проектирования 3D ГИС на базе свободно-распространяемых ресурсов и собственных разработок с использованием функциональной и геометрической моделей, обеспечивающих формирование функциональных покрытий, их оценку и выбор среди них оптимального покрытия с последующей его программной реализацией;
4. разработана модель и программная реализация специализированного средства проектирования 3D ГИС, позволяющего использовать современные информационные технологии в виде свободно-распространяемых ресурсов: библиотек, текстур, моделей высот и глубин, 3D моделей объектов, а также формировать готовые проектные решения, отвечающие требованиям на разработку 3D ГИС и делать их оценку качества для последующей реализации;
5. создана методика и система 3D моделирования проектных решений, которая позволяет оценивать адекватность полученных проектных решений исходным требованиям на создание 3D ГИС и возможность коррекции для их улучшения.

Результаты работы получены автором при выполнении следующих проектов для ФНПЦ АО «НПО «МАРС»:

- НИР «Анализ и исследование возможностей 3D-ГИС движков для представления морской обстановки» - 2013 г.;
- ОКР «Разработка комплекса 3D визуализации морской, наземной и воздушной обстановки» - 2014 г.

Теоретическая и практическая значимость работы. Основные положения и выводы диссертационного исследования могут быть использованы при проектировании и создании 3D ГИС на основе собственных разработок и трансформируемых готовых решений (свободно-распространяемых библиотек).

Апробация работы и использование результатов.

Результаты работы использовались:

- при создании системы 3D моделирования проектных решений;
- при программной реализации комплекса 3D визуализации морской, наземной и воздушной обстановок для ФНПЦ АО «НПО «МАРС»;

– при выполнении государственного задания Министерства образования и науки РФ №2.1816.2017/ПЧ по теме «Исследование и разработка интегрированной автоматизированной системы управления производственно-технологическим планированием авиастроительного предприятия на базе цифровых технологий».

Публикации. По теме исследования опубликовано 10 работ, отражающих основные положения исследования, 4 публикации в журналах, рекомендованных ВАК Минобрнауки России.

Структура и объем работы диссертации. Диссертационная работа состоит из введения, пяти глав, заключения, списка используемой литературы и приложений. Общий объем диссертации составляет 225 страниц, основной текст изложен на 175 страницах.

Глава 1.Анализ современных геоинформационных систем и пространственных данных

§ 1.1. Анализ и классификация существующих геоинформационных систем

1.1.1. Классификация геоинформационных систем

В связи с быстрым развитием современных информационных технологий и широкой применимостью геоинформационных систем (ГИС) в настоящее время разработано большое количество ГИС, различающихся функциональностью и сферой использования. Современные ГИС применяются в таких областях как картография, геология, метеорология, землеустройство, экология, муниципальное управление, транспорт, экономика, оборона и многие другие. [17,26,47,88,89,125]

Одни системы разрабатываются крупными компаниями и имеют коммерческую лицензию, другие поддерживаются Интернет-сообществом, дорабатываются энтузиастами и любой пользователь может воспользоваться функциональностью этих ГИС. [93]

Все геоинформационные системы можно разделить на двумерные (2D) и трёхмерные (3D) ГИС. [19]

По причине быстрого увеличения компьютерных мощностей рабочих станций обычного пользователя 2D ГИС функционально устарели. Многие разработчики таких систем пытаются совершенствовать свои продукты и добавляют некоторые возможности 3D, такие как добавление высоты для улучшенной визуализации поверхности, придание объёма карте. Однако, добавляемая Z-координата не является независимой, и такие системы называются 2.5D ГИС.

Классификация геоинформационных систем по различным признакам представлена на рисунке 1.1.

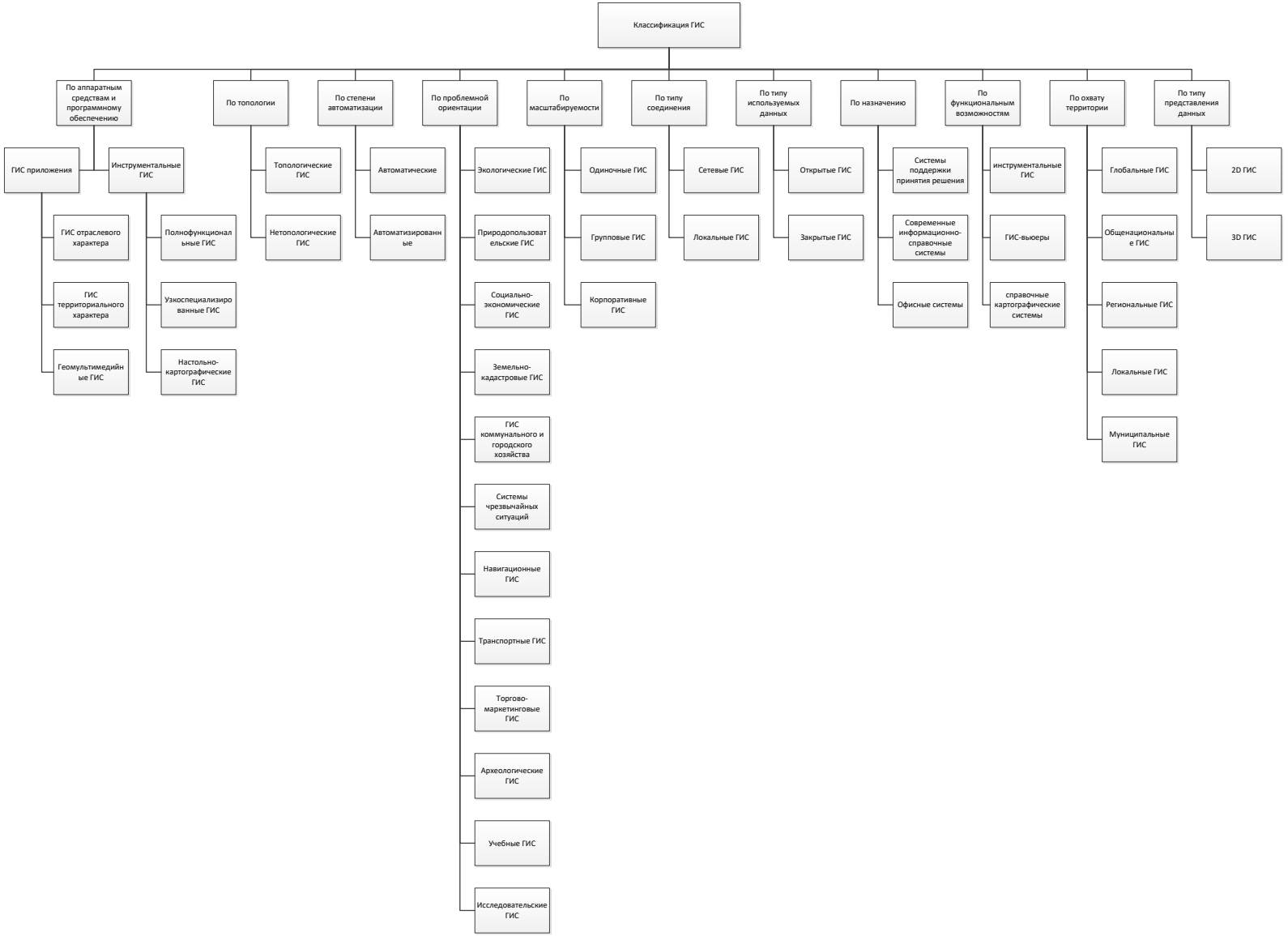

Рисунок 1.1. Классификация ГИС

В зависимости от степени автоматизации геоинформационные системы делятся на автоматические и автоматизированные. Автоматические геоинформационные системы обеспечивают обработку геоинформации без участия оператора. Автоматизированные геоинформационные системы предполагают непрерывное взаимодействие оператора и самой системы.

Системы поддержки принятия решения представляют собой геоинформационные системы, в которых с помощью запросов производятся отбор и анализ данных по временным, географическим и прочим показателям. Информационно-справочные системы представляют собой набор гипертекстовых документов и мультимедиа. Офисные системы обеспечивают перевод в электронный вид документов, представленных на бумажной основе, для целей автоматизации делопроизводства. [71]

С учетом функциональных возможностей ГИС выделяются следующие группы:

- инструментальные ГИС – это системы с наиболее широкими возможностями, включающие подсистемы ввода данных, подсистемы пространственного моделирования и анализа данных, мощные средства запросов, средства вывода информации на твердые носители, средства расширения возможности систем; [71]
- ГИС-вьюеры (вьюверы), которые представляют собой системы сопровождения инструментальных ГИС и предназначены для просмотра информации; [32]
- справочные картографические системы, которые содержат встроенные базы данных без возможности редактирования и в которых отсутствуют возможности расширения функциональности. [9,10,18,71]

По охвату описываемой территории выделяются глобальные, общенациональные, региональные, локальные и муниципальные ГИС, обеспечивающие пользователей информацией по указанным районам. [42,71]

По топологии ГИС подразделяются на топологические и нетопологические. Топологические ГИС используются для решения задач пространственного анализа (определение расстояний между объектами, видимости между ними и т. д.). [71,113]

Все ГИС подразделяются по типу соединения на сетевые и локальные.

По визуальному отображению различают двухмерные и трёхмерные ГИС. В 3D ГИС пользователям предоставляются широкие возможности по формированию объемных изображений объектов, изучению рельефа и анализу пространственной информации. [71,114]

По типу используемых аппаратных средств и программного обеспечения выделяются инструментальные ГИС и ГИС-приложения.

Инструментальные ГИС обеспечивают формирование сложных запросов и пространственный анализ данных.

ГИС-приложения ориентированы на одну из предметных областей и обычно создаются на базе инструментальных ГИС. [71]

По типу используемых пространственных данных выделяются ГИС, взаимодействующие с векторными и растровыми форматами карт или отображающие смешанную информацию.

Все ГИС подразделяются на два типа: открытые и закрытые.

В открытых ГИС имеется возможность доработки функциональности для решения специфических задач с использованием встроенных в них языков программирования, а в закрытых ГИС такие возможности отсутствуют.

По масштабируемости выделяются: одиночные, групповые и корпоративные ГИС.

Одиночные геоинформационные системы используются на единичном персональном компьютере, рассчитаны на обслуживание одного пользователя и создаются на основе настольных систем управления базами данных. [71]

Групповые ГИС применяются в пределах одной или нескольких локальных сетей для взаимодействия пользователей внутри одной организации.

Корпоративные ГИС используются в крупных компаниях и имеют возможность распределения и взаимодействия между их филиалами.

По области применения выделяются: экологические, природопользовательские, социально-экономические, земельно-кадастровые системы, системы коммунального и городского хозяйства, чрезвычайных ситуаций, навигационные, транспортные, торгово-маркетинговые, археологические, учебные, исследовательские и прочие ГИС. [71,76,77,111,112,124]

1.1.2. Анализ геоинформационных систем

Одной из самых распространённых **2D ГИС** является MapInfo Professional. MapInfo – это географическая информационная система, предназначенная для сбора, хранения, отображения, редактирования и анализа пространственных данных. Сферы применения ГИС MapInfo: бизнес и наука, образование и управление, социологические, демографические и политические исследования, промышленность и экология, транспорт и нефтегазовая индустрия, землепользование и кадастр, службы коммунального хозяйства и быстрого реагирования, армия и органы правопорядка, а также многие другие отрасли хозяйства. [79,80,96,126]

Геоинформационная система MapInfo работает без конвертации с графическими данными в форматах ArcView Shape File, ESRI ArcSDE, ESRI Geodatabase (mdb), ARC/INFO E00, AutoCAD DXF/DWG, Intergraph/MicroStation Design DGN, SDTS, VPF и табличными данными в форматах Access, Excel, Lotus 1-2-3, xBASE и ASCII. Транслятор MapInfo позволяет осуществлять импорт и экспорт данных в другие ГИС и САПР системы (ESRI Shape

File, AutoCAD DXF/DWG, Intergraph/MicroStation Design DGN, AtlasGIS, ARC/INFO E00). MapInfo имеет возможность работы с данными в растровых форматах GIF, JPEG, TIFF, GEO TIFF, PCX, BMP, TGA, BIL и др., включая форматы сжатого растра – ECW, MrSID, JPEG2000. [96]

Основным недостатком MapInfo Professional является отсутствие возможности отображения полноценных трёхмерных карт и 3D объектов.

Большое распространение в настоящее время получили **Web-ГИС**, которые ориентированы на отображение геоинформации в Web-браузерах пользователей. Такие системы имеют клиент-серверную архитектуру, где на стороне сервера выполняется большинство операций по математическим расчётам, генерации пространственной информации и отображаемых объектов местности, а на стороне клиента (Web-браузера) – её отображение. Дополнительно сервер обеспечивает хранение картографической информации в специализированных геопространственных базах данных. Для функционирования таких систем необходимо постоянное подключение к локальной сети или Интернет. [64]

Наиболее популярными Web-ГИС являются: Google Maps, NASA Maps, Yandex карты, Геопортал Роскосмоса, карты Спутник и т.п.

Представленные выше Web-ГИС используют собственную картографическую информацию, полученную со спутников или летательных аппаратов, либо взаимодействуют с открытыми источниками пространственных данных, такими как OpenStreetMap и др.

Основными преимуществами Web-ГИС являются:

- отсутствие расходов на покупку настольных геоинформационных систем;
- использование клиент-серверной архитектуры с целью централизованного хранения пространственных данных;
- повышение доступности геоинформации благодаря возможности доступа с различных устройств (персональные компьютеры, мобильные устройства, планшеты и т.д.);
- кроссплатформенность и отсутствие привязки к определенной операционной системе.

В качестве недостатков Web-ГИС выделяются высокая сложность отображения трёхмерной информации, небольшой объём функциональных возможностей по управлению и редактированию обстановки из-за ограничений Web-браузеров и низкая скорость взаимодействия между клиентом и сервером по сравнению с настольными ГИС.

Среди **3D ГИС** можно выделить следующие наиболее известные системы: ArcGIS, SharpMap, MapAround, MapWindow GIS, gvSIG, Quantum GIS, GRASS, wxGIS.

Самой распространенной системой среди 3D ГИС является **ArcGIS** — семейство геоинформационных программных продуктов американской компании ESRI с возможностями создания карт, глобусов и моделей в настольных системах, публикации и использования их в настольных приложениях, веб-браузерах и мобильных устройствах. Система ArcGIS имеет множество различных модулей и необходимые инструменты для создания собственных приложений, но является коммерческим программным продуктом. [4,120]

Аналогами ArcGIS с открытым исходным кодом являются такие системы как: Quantum GIS и gvSIG.

Система **QGIS (Quantum GIS)** — свободная кроссплатформенная геоинформационная система, имеющая в наличии большинство функциональных возможностей проприетарных ГИС, например:

- просмотр данных форматов: GeoTIFF, Erdas IMG, ArcInfo ASCII Grid, JPEG, PNG, shape-файлы ESRI, MapInfo, SDTS, GML, KML, SpatiaLite, GRASS, OpenStreetMap и другие.
- исследование данных и компоновка карт;
- управление данными: создание, редактирование и экспорт;
- анализ данных;
- публикация карт в сети Интернет;
- расширение функциональности QGIS с помощью специальных модулей на языке Python. [96]

Система **MapAround** — это трёхмерная геоинформационная система, которая содержит инструментарий для решения большинства типовых задач, возникающих при разработке ГИС. Система MapAround используется в качестве платформы для разработки на уровне предприятия, многопользовательских географических информационных систем с эффективными инструментами для преобразования, хранения и изменения пространственных данных, а так же для разработки настольных ГИС приложений, позволяет разработчикам дополнить ГИС-характеристиками классические информационные системы управления и мониторинга, добавляя графическую визуализацию, обработку и анализ данных и другие характеристики.

Функциональные возможности:

- быстрый рендеринг;
- тематические карты;
- наличие виджетов для windows forms, asp.net и javascript;

- простая и расширяемая объектная модель;
- вычисление длин, расстояний, площадей и периметров;
- решение прямой и обратной геодезических задач.

Основным недостатком MapAround является её функционирование только в операционных системам семейства Windows. [97]

Система **MapWindow GIS** — это ГИС с открытым исходным кодом, которая обладает набором программных и программируемых библиотек и разработана университетом GeoSpatial Software Lab штата Айдахо. Ядро данной системы написано на языке программирования C++ и представляет собой библиотеку ActiveX, которую можно использовать отдельно от MapWindow для разработки собственных приложений.

Основные преимущества MapWindow GIS:

- является бесплатным при использовании как в коммерческом, так и не в коммерческом режимах;
- проект предоставляется с открытым исходным кодом, что дает возможность разработчикам создавать дополнительные модули и библиотеки;
- возможность программирования прямо из приложения при помощи встроенного редактора.

Основные недостатки MapWindow GIS:

- ограничение форматов векторных данных;
- отсутствие встроенного редактора компоновок;
- отсутствие совместной работы. [97]

Система **gvSIG** — свободно-распространяемая геоинформационная система с открытым исходным кодом, предназначенная для сбора, хранения, обработки, анализа и развёртывания любой географически привязанной информации для решения комплексных проблем управления и планирования.

Преимуществами gvSIG являются:

- дружественный к пользователю интерфейс;
- доступ к наиболее распространенным векторным и растровым форматам данных;
- обширный набор средств для работы с географической информацией (инструменты запросов, создание макетов, геообработка, сетевой анализ и т.д.).

Недостатками gvSIG являются:

- отсутствие поддержки некоторых популярных систем координат (EPSG:3857 и др.);
- использование собственного браузера, недостаточно качественно отображающего интернет-страницы;
- необходимость устанавливать готовые проекты в определенную директорию на жёстком диске. [97]

Система **SharpMap** – свободно-распространяемая трёхмерная геоинформационная система для использования в настольных и веб-приложениях, написанная на языке C# и фреймворке .NET 2.0.

Недостатками SharpMap являются:

- медленная скорость рендеринга карт;
- ограниченный набор форматов карт без подключения дополнительных компонентов;
- функционирование только в операционных системах семейства Windows. [97]

Ниже представлена сравнительная таблица наиболее распространенных ГИС с открытым исходным кодом (Таблица 1.1) по поддерживаемым операционным системам, лицензии и языку программирования.

Таблица 1.1. Сравнительная таблица наиболее распространенных ГИС с открытым исходным кодом

Система	**Кроссплатформенность**	**Лицензия**	**Язык**
SharpMap	Windows	GNU LGPL	C#, .NET 2.0
MapAround	Windows	GNU GPL	.NET
MapWindow GIS	Windows	MPL	C#, C++
gvSIG	Windows, Linux, Mac OS	GNU GPL	Java
Quantum GIS	Cross-platform	GPL	C++, Qt
GRASS	POSIX (включая GNU/Linux и Apple Mac OS X), Microsoft Windows	GNU GPL	C++
wxGIS	Cross-platform	GNU GPL v3	C++

Большинство современных геоинформационных систем обладает широкими возможностями для трёхмерной визуализации, отображения рельефа местности, текстур и вектор-

ных объектов, но имеет ряд ограничений при расширении функциональности или взаимодействии с узкоспециализированными форматами карт и 3D моделями объектов.

§ 1.2. Пространственные данные в ГИС

1.2.1. Общая информация о пространственных данных

Пространственные данные (пространственная информация) - цифровые данные о пространственных объектах, включающие сведения об их местоположении и свойствах, пространственных и непространственных атрибутах. [67,85]

Пространственные данные являются информационной составляющей ГИС, в связи с чем способы их формализации и хранения являются важной частью технологии географических информационных систем. [86,110]

Пространственные данные состоят из двух взаимосвязанных частей: координатных (метрических) и атрибутивных (семантических) данных, а установление связи между ними называется геокодированием. [66,90]

Координатные данные определяют позиционные характеристики объекта. Они описывают его местоположение в установленной системе координат. [67]

Атрибутивные данные представляют собой совокупность непозиционных характеристик (атрибутов) объекта, определяют их смысловое содержание (семантику) и содержат качественные или количественные значения. [67]

Пространственные типы данных для СУБД должны быть:

- замкнутыми на множестве операций;
- иметь формально-определенную семантику;
- быть определенными в компьютерном представлении;
- предлагать адекватные возможности для представления реальных пространственных объектов;
- быть независимыми от конкретной модели СУБД, но взаимодействовать с любой. [94]

Геометрия геоданных - это совокупность координатной геометрии и системы привязки.

Координатная геометрия состоит из следующих элементов:

1. последовательности координатных точек, заданных в одной и той же системе привязки;
2. набора других геометрий, заданных в одной и той же системе привязки;

3. алгоритма интерпретации, который использует эти геометрии и точки, для построения геометрического объекта;
4. пространственно-временной системы привязки, которая определяет соответствие между координатной геометрией и расположением объекта. [94]

В большинстве пространственных СУБД поддерживаются три основных типа координатной геометрии: точечные, линейные, площадные. Точечные объекты представляются одной или несколькими точками, линейные - одной или несколькими линиями, площадные - одной или несколькими компонентами, состоящими из внешней границы и произвольного количества внутренних границ. [116]

1.2.2. Цифровая модель местности в ГИС

Цифровая модель местности является формой представления информации о рельефе и пространственной изменчивости различных характеристик, предназначена для автоматизации предоставления данных о рельефе в численных реализациях гидрологических моделей и позволяет избежать трудоёмких процедур ручной подготовки данных для программ, использующих особенности геометрии объекта. [15,37,68]

Исходными данными для построения трехмерной цифровой модели местности являются: векторные данные, матрица высот и глубин, триангуляционная модель рельефа, библиотека трехмерных моделей объектов, цифровые фотоснимки местности и цифровые фотографии объектов и их частей (Рисунок 1.2). [84]

Рисунок 1.2. Трёхмерная цифровая модель местности в ГИС

Векторная карта (векторные данные) – это совокупность описания паспортных данных о листе карты (масштаб, проекция, система координат, прямоугольные и геодезические координаты углов листа и т. д.), метрических данных объектов карты (координаты объектов на местности) и семантических данных объектов карты (различные свойства объектов). Векторные карты имеют расширения: MAP, SIT, SXF, DXF, MIF, SHP, DGN и др. [46,57,58]

Матрица высот и глубин содержит абсолютные высоты рельефа местности и может быть создана по данным векторной карты, либо загружена из специализированных баз данных высот и глубин. [46,60]

Триангуляционная модель рельефа содержит треугольники нерегулярной сети, описывающие поверхность местности. [46]

Библиотека 3D объектов содержит описания объемного вида трёхмерных объектов и их семантическую информацию.

Цифровые фотографии объектов (в формате BMP, TIFF, JPEG) содержат изображения объектов или частей объектов и используются в качестве текстур.

Цифровые фотоснимки местности содержат изображения местности в растровых форматах и могут быть использованы для наложения на поверхность рельефа. [46]

1.2.3. Форматы локальных пространственных данных

Все форматы геоданных делятся на векторные и растровые. [128]

Растровые данные – данные, представляющие собой набор пикселей. Каждый растровый слой в ГИС имеет пиксели фиксированного размера, которые определяют его пространственное разрешение. [12,14]

Два наиболее распространенных способа создания растровых данных – аэрофотосъемка и спутниковая съемка. [65,70,127]

В первом случае фотографирование территории осуществляется с определённой высоты от поверхности Земли при помощи аэрофотоаппарата, установленного на атмосферном летательном аппарате (самолёте, вертолёте, дирижабле и пр. или их беспилотном аналоге). [7,8,100]

Спутниковые снимки создаются с использованием искусственных спутников, вращающихч вокруг Земли по определенным орбитам. [1,36]

Процесс получения растровых данных с помощью летательных аппаратов или искусственных спутников называется дистанционным зондированием. [27,28]

Наиболее распространённые растровые форматы геоданных:

- Microsoft Windows Device Independent Bitmap (.bmp);
- ERMapper (.ers);
- Graphics Interchange Format (.gif);
- GRASS Raster Format;
- TIFF / BigTIFF / GeoTIFF (.tif);
- HF2/HFZ heightfield raster;
- JPEG JFIF (.jpg);
- NetCDF;
- Portable Network Graphics (.png). [39]

Большинство растровых форматов поддерживается свободно распространяемой библиотекой GDAL, которая легко подключается ко множеству современных ГИС.

Векторные данные – данные, в которых способ представления объектов и изображений основан на математическом описании элементарных геометрических объектов, таких как: точки, линии, полигоны. Векторные данные являются способом представления объектов реального мира в среде ГИС. [16,74]

Векторный объект имеет форму, записанную в виде геометрии. Геометрия состоит из одной или бóльшего числа связанных вершин. Вершина описывает позицию в пространстве, используя оси X, Y и Z. [74]

Если геометрия объекта состоит из единственной вершины, этот объект называется точечным. Когда геометрия состоит из двух и более вершин, формируется полилиния. Если первая вершина равна последней и вершин четыре и более, они составляют замкнутый полигон. [74]

Наиболее распространённые векторные форматы геоданных:

- ESRI Shapefile;
- GeoJSON;
- GRASS Vector Format;
- KML;
- LIBKML;
- OpenStreetMap XML and PBF;
- S-57 (ENC);
- SVG;
- Storage and eXchange Format. [40,41]

Большинство векторных форматов поддерживается дополнением OGR к свободно распространяемой библиотеке GDAL.

1.2.4. Интернет источники пространственных данных

Все рассмотренные в предыдущем параграфе форматы геоданных являются файлами, расположенными локально на жёстком диске компьютера. Но многие современные ГИС имеют поддержку подключения к геоданным из различных Интернет источников по соответствующим протоколам.

Данный способ работы с геоданными, в первую очередь, позволяет сократить затраты на хранение текстур и рельефа локально. При необходимости получения карты местности система отправляет запрос на сервер, получает требуемые данные и сохраняет их в оперативной памяти на время работы с ними. Существует возможность сохранения данных в кэш и дальнейшее их использование. [13]

Основным недостатком данного способа является обязательное подключение к Интернет для загрузки требуемых данных. В случае использования сервера в локальной сети – наличие достаточно мощного сервера с большим количеством свободной памяти на жестком диске для хранения всей базы карт. [11]

Наиболее распространенные Интернет-ресурсы и протоколы:

- **Web Map Service** – стандартный протокол для обслуживания через Интернет географически привязанных изображений, генерируемых картографическим сервером на основе данных из БД ГИС. Web Map Service предоставляет простой интерфейс http-запросов для получения геопривязанных изображений карт нескольких распределённых пространственных баз данных. В ответе сервиса на запрос будет одно или более изображений карты (в формате JPEG, PNG и др.), которые могут быть показаны в браузере или настольном приложении.
- **Web Feature Service** – веб-сервис пространственных объектов, определяющий интерфейсы и операции, которые позволяют запрашивать и редактировать векторные пространственные данные, такие как дороги или береговые линии. [95]
- **Tile Map Service** – сервис доступа к хранилищам «тайлов» (элементов карты в виде небольших прямоугольников). TMS-сервис может функционировать в одном из двух режимов - статическом и динамическом. В первом случае это набор файлов, организованных определённым образом в файловой системе, так называемый «тайловый кэш». Второй режим работы TMS сервиса – динамический – предполагает, что запрашиваемые клиентом «тайлы» генерируются по запросу. [95]
- **OpenStreetMap** — некоммерческий веб-картографический проект по созданию силами сообщества участников-пользователей Интернета подробной свободной и бесплатной географической карты мира. [95]

Для использования ГИС в связке с удаленными источниками данных, но при отсутствии подключения к Интернет в большинстве случаев есть возможность создания «зеркала» (копии) сервера в локальной сети. База пространственных данных в данном случае хранится в одном из СУБД (MySQL, PostgreSQL и др.).

Выделяют четыре наиболее распространённых формата (Интернет-ресурса) пространственных данных высот и глубин:

- CGIAR – данные рельефа высотой 90 м., полученные из форматов SRTM и ETOPO;
- SRTM30+ – карта рельефа Земли, включающая батиметрические данные;
- GLCF – данные о рельефе суши;
- GEBCO – данные о батиметрии Мирового Океана.

Shuttle radar topographic mission (SRTM) - Радарная топографическая съемка большей части территории земного шара, за исключением самых северных (>60), самых южных широт (>54), а также океанов. [87]

Существует три версии данных: предварительная (unfinished, версия 1), окончательная (finished, версия 2) и обработанная. Окончательная версия прошла дополнительную обработку, выделение береговых линий и водных объектов, фильтрацию ошибочных значений. Обработанная версия производится CGIAR и включает сборку мозаик в более крупные фрагменты (6000x6000 пикселей или 5x5, а не 1x1 градус) и исправление областей с отсутствующими значениями. [87,119]

Результирующие данные соответствуют спецификации интерферометрических данных о рельефе (Interferometric Terrain Height Data (ITHD)-2), а именно, размер элемента 30x30 метров, <=20 метров точность по высоте. [87]

Исходные данные распространяются квадратами размером 1x1 градус, при максимальном доступном разрешении 3 арксекунды. Такой квадрат является матрицей размером 1201x1201 элементов (пикселей). Один дополнительный ряд (нижний) и одна колонка (правая) являются дублирующими и повторяются на соседней матрице. [87]

Данные являются простым 16 битным растром (без заголовка), значение пиксела является высотой над уровнем моря в данной точке, оно также может принимать значение 32768, что соответствует значению no data (нет данных). [87]

Другой тип данных – *GEBCO*. GEBCO - это General Bathymetric Chart of the Oceans, то есть данные, по которым можно строить карту дна океана. Разрешение «сетки» - одна минута. Основной формат GEBCO – netCDF.

§ 1.3. Методы построения современных 3D ГИС

В настоящее время существует несколько методов создания геоинформационных систем:

1. разработка дополнительных функционально-программных модулей, работающих совместно с существующим ядром ГИС;
2. разработка ГИС «с нуля»;
3. разработка WEB-ГИС на основе клиент-серверной архитектуры;
4. разработка ГИС на основе готовых решений.

Первый метод заключается в создании внешних программных модулей, работающих в среде универсальной ГИС. Как правило, модули реализуются с помощью специализированных макроязыков, интерпретаторы которых встроены в ядро универсальной ГИС. Часто возможностей макроязыка не достаточно для решения тех или иных задач, поэтому макроязыки должны иметь средства для встраивания программ, написанных па языках другого уровня. [4,5,53]

Преимуществом данного метода является наличие готового ядра ГИС, реализующего основные функциональные возможности любой подобной системы, удобные встроенные языки программирования для разработки собственных модулей и наличие программных интерфейсов, обеспечивающих взаимодействие ядра ГИС и созданных модулей.

Основной недостаток метода заключается в ограничении функциональности разрабатываемых программных модулей, обусловленном возможностями ядра ГИС и программных интерфейсов. [29]

В основе второго метода лежит разработка геоинформационной системы без использования готовых движков и модулей. В данном случае разработчик на выбранном языке программирования выполняет программную реализацию ГИС, полностью описывая все математические операции по преобразованию карт и объектов и реализуя их визуальное отображение.

Преимуществами метода являются наличие только тех функций, которые необходимы в рамках поставленной задачи, что обеспечивает увеличение скорости работы системы и уменьшение потребляемых ресурсов, а также отстутствие скрытых функциональных возможностей.

В качестве недостатков выделяются сложность реализации такой ГИС, высокий уровень знаний разработчика, большие сроки разработки, наличие ошибок и уязвимостей.

В третьем методе используется технология клиент-сервер. Клиентом выступает программа, которая решает производственные задачи, например, обработки данных. Данная программа делает запросы к другой программе - серверу. В качестве сервера используется программное обеспечение универсальной ГИС. Сервер выполняет запросы клиента и передает ему результаты.

Частным случаем этого метода являются Web-ГИС, которые отображаются у пользователя в специализированных программных средствах – Web-браузерах и взаимодействуют с удаленным сервером, формирующим для отображения слои местности, объекты и другую метрическую и семантическую информацию.

Преимущества: гибкость, кроссплатформенность, возможность использования на мобильных устройствах.

Недостатки:

- необходимость иметь мощный сервер, выполняющий большинство операций для отображения;
- постоянное соединение по локальной сети или с Интернет;
- необходимость разработки двух программных средств: сервера и клиента, каждый из которых обладает своими функциональными возможностями;
- ограниченные клиентом возможности отображения пространственных данных на местности и взаимодействия с ними.

Четвёртый метод заключается в разработке ГИС с использованием готовых решений, реализующих одну или несколько функциональных возможностей создаваемой системы.

Преимуществами являются:

- наличие готовых программных решений по реализации основных функций ГИС, таких как: загрузка и отображение растровых и векторных карт, визуализация объектов, решение математических и геометрических задач на местности, аналитика и др.;
- снижение уровня знаний программиста за счёт отсутствия необходимости выполнения большого объёма математических расчётов при отображении карт, объектов и т.п.;
- широкая доступность готовых решений, т.к. в настоящее время большинство таких решений поставляется по свободно-распространяемых лицензиям и даёт возможность подключения и встраивания в другие проекты.

Основным недостатком является сложность совмещения готовых решений между собой и с разрабатываемой ГИС из-за разных программных интерфейсов, форматов ввода-вывода пространственных данных и операционных систем, на которых они функционируют. Этот недостаток легко решается разработкой дополнительных средств (адаптеров), обеспечивающих конвертацию данных из готового решения в ГИС и наоборот.

§ 1.4. Постановка задачи

Целью данной диссертации является анализ возможностей и разработка систем и средств автоматизации проектирования и моделирования 3D ГИС отображения ситуационной обстановки для снижения временных и финансовых затрат, уменьшения сложности проектирования и уровня знаний проектировщика.

Пусть имеется множество различных информационных ресурсов, множество способов их использования на практике (проектирования) и множество вариантов ситуационной обстановки.

Требуется разработать такую систему автоматизации проектирования, которая бы смогла обеспечивать построение 3D ГИС отображения ситуационной обстановки с наименьшими ресурсными и временными затратами и выполнением требований по точности с возможностями отображения воздушной, наземной и морской обстановок.

Исходя из проведенного в первой главе анализа существующих 3D ГИС, можно выделить следующие **основные требования** к разрабатываемым системам:

1) система должна работать в режиме, приближённом к реальному времени;
2) система должна обладать возможностью отображения различных ситуационных обстановок и объектов, участвующих в них;
3) система должна иметь возможности современных геоинформационных систем, такие как: 3D визуализация местности, рельефа и трехмерных объектов, работа с источниками обстановки, взаимодействие с внешними системами, имитация движения объектов.

Ограничения для 3D ГИС:

1) количество объектов обстановки конечно;
2) типы объектов отображения – морские, наземные, воздушные объекты, векторные и растровые карты, матрицы высот и глубин;
3) отображение трехмерной обстановки с точностью, достаточной для однозначного определения типа объекта, его траектории движения и принятия решения оператором 3D ГИС.

Критериями достижения цели являются**:**

- создание 3D ГИС с заданными характеристиками;
- снижение затрат на проектирование;
- уменьшение времени на проектирование;
- снижение требований к проектировщикам и разработчикам;
- возможность использования в смежных предметных областях.

Выводы

На основании проведенного анализа, классификации существующих геоинформационных систем и методов их построения:

1. определены основные характеристики трёхмерных геоинформационных систем, такие как: использование матриц высот и глубин, отображение векторных и растровых карт с учётом рельефа местности, построение 3D моделей объектов и местности, получение информации о текущей ситуации и её анализ с учётом третьей координаты, которые не позволяют использовать методы проектирования 2D ГИС для 3D ГИС;
2. рассмотрены используемые в современных 3D ГИС пространственные данные и особенности их хранения и отображения, представлена схема построения трехмерной цифровой модели пространственных данных, которая включает:
   - матрицу высот;
   - библиотеки 3D объектов;
   - растровые снимки местности;
   - триангуляционную модель рельефа;

   и позволяет учитывать и моделировать трехмерную обстановку.
3. Проведён анализ используемых в 3D ГИС форматов карт, хранимых на локальных носителях и в сети Интернет, представлены различия между векторными и растровыми видами карт по типам хранимых в них объектов (полигонов, линий, точек), из чего возникает необходимость создания процедур для чтения, отображения и записи этих карт и конвертеров для их преобразования при разработке 3D ГИС.
4. Проведён анализ Интернет-источников, который указал на возможности использования трансформируемых готовых решений, применимых к проектированию 3D ГИС, что позволяет снизить трудоёмкость и затраты на их создание.
5. Определены задачи, ограничения и основные критерии 3D ГИС отображения ситуационной обстановки, выделены трудности при их разработке, вследствие чего возникает необходимость формирования новой эффективной системы автоматизации проектирования и разработки современных трехмерных геоинформационных систем, позволяющей понизить временные и ресурсные затраты на проектирование и повысить надёжность 3D ГИС за счёт использования проверенных на практике трансформируемых готовых решений.

Глава 2. Проектирование 3D ГИС на основе CASE-средства

§ 2.1. Функционально-геометрическое покрытие заданной функциональности 3D ГИС

В практике проектирования информационных систем широко применяются комплексные модели, включающие в свой состав несколько модельных компонентов, которые имеют собственные области представления и преобразования данных, отображающие процесс разработки проектируемой системы с разных сторон. Такие модели отличает комплексность, особенностью которой является их взаимодействие между собой по определенным правилам в процессе разработки. [98,99]

Например, часто требуется решение связанных между собой задач: одной – с затратами на проектирование, а второй – с обеспечением заданной функциональности, при этом представление затрат обычно отождествляется с размером площадей геометрических объектов, выражаемых в определенных единицах измерения и пропорциях, а функциональность – с обеспечением полноты реализуемых функций.

Представление в виде геометрических объектов позволяет варьировать формой и размером площадей, а функциональную модель удобно представлять в виде матриц, которые имеют широкий диапазон операций по их преобразованию.

Комплексность моделирования позволяет работать попеременно в двух средах: геометрической и функциональной, и результаты, полученные с помощью геометрического моделирования проектных решений, используются при функциональном моделировании и обратно.

Целью использования комплексной (двухкомпонентной) функционально-ресурсной модели является получение эффективных решений, приводящих к полному покрытию заданной функциональности программного комплекса свободно распространяемыми ресурсами (библиотеками) и собственными разработками, минимизирующими затраты на его разработку. [107]

2.1.1 Двухкомпонентная функционально-ресурсная модель покрытия функциональности 3D ГИС

Двухкомпонентная функционально-ресурсная модель (М) покрытия функциональности 3D ГИС в общем виде представлена на рисунке 2.1 и состоит из функциональной модели (Ф) и геометрической модели (G):

$$M = \Phi \cup G. \quad (2.1)$$

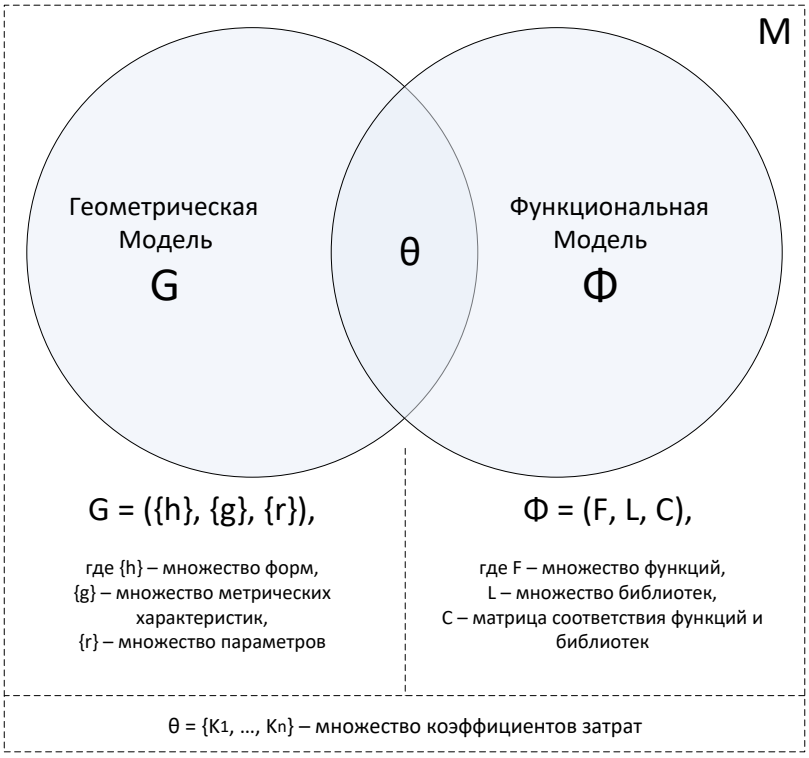

Рисунок 2.1. Двухкомпонентная функционально-ресурсная модель покрытия функциональности 3D ГИС

К геометрической модели относятся: совокупность форм {h}, метрические характеристики {g}, которые определяют размеры форм, параметры {r}, задающие местоположение форм в соответствующем геометрическом пространстве. [107]

Представим геометрическую информацию как:

$$G = (\{h\}, \{g\}, \{r\}). \quad (2.2)$$

Функциональная модель представляет собой множество функций, доступных для реализации (*F)*, множество библиотек (*L)*, которые покрывают эти функции, и матрицу их соответствия (*C)*:

$$\Phi = (F, L, C). \quad (2.3)$$

Пересечением (θ) геометрической и функциональной моделей является множество коэффициентов (K_i), определяющих собственные затраты на программную реализацию i-й функции:

$$G \cap \Phi = \theta. \quad (2.4)$$

Такой процесс взаимодействия двух моделей обеспечивает контроль его состояния и позволяет вносить корректировки на разных этапах проекта.

2.1.2 Функциональная модель 3D ГИС

3D ГИС может быть построена как полностью собственная разработка, которая включает в себя участие разработчиков разной квалификации с использованием определенных языков программирования, платформ, информационных технологий, а также на основе свободно распространяемых ресурсов – библиотек.

Построение такой 3D ГИС представляет собой разработку отдельных или связанных между собой подсистем, реализующих функции, которые образуют функциональное пространство 3D ГИС (в дальнейшем – функциональность). Под понятием функциональности 3D ГИС понимается его способность выполнять набор функций, удовлетворяющих заданным потребностям пользователей, а под покрытием функциональности – подбор известных решений с возможной их модернизацией. [107]

Для получения функционального покрытия необходимо предварительно составить список всех функций, входящих в функциональность 3D ГИС, и одноимённый список функций, реализуемых свободно распространяемыми библиотеками.

Для получения функционального покрытия заданной функциональности предлагается использовать следующий подход:

1. Строится обобщённая бинарная матрица соответствия (C) функций, которые можно реализовать в рамках заданной области применения, и библиотек, которые частично их покрывают.

$$C = \begin{array}{c|cccc} & \boldsymbol{f_1} & \boldsymbol{f_2} & \boldsymbol{\dots} & \boldsymbol{f_n} \\ \boldsymbol{l_1} & c_{1,1} & c_{1,2} & \dots & c_{1,n} \\ \boldsymbol{l_2} & c_{2,1} & c_{2,2} & \dots & c_{2,n} \\ \boldsymbol{\dots} & \dots & \dots & \dots & \dots \\ \boldsymbol{l_m} & c_{m,1} & c_{m,2} & \dots & c_{m,n} \end{array}$$

В этой матрице строки представляют собой множество библиотек $L = \{l_1, \dots, l_m\}$, а столбцы – множество функций $F = \{f_1, f_2, \dots, f_n\}$ заданной области применения. В ячейку $c_{i,j}$ обобщённой матрицы соответствия библиотеки l_i и функции f_j записывается единица, если библиотека l_i реализует функцию f_j, в противном случае – ноль.

2. Формируется бинарный вектор $\bar{f}$ функций заданной области применения, каждый элемент которого принимает значение *1*, если он принадлежит заданной функциональности 3D ГИС, иначе – *0*.
3. С целью отсеивания из множества функций тех, которые не входят в заданную функциональность, используется операция умножения матрицы (C) на вектор функций $\bar{f}$.

В полученной в результате перемножения матрице D определяются столбцы, содержащие все нули, после чего эти столбцы удаляются.

Задача поиска библиотек, реализующих заданную функциональность, сводится к нахождению кратчайшего покрытия полученной на предыдущем шаге булевой матрицы *D,* то есть нахождения такой минимальной совокупности строк матрицы, которая содержала бы не менее одной единицы в каждом столбце матрицы.

Под покрытием матрицы D размера $m \times n$ в данной работе понимается такое подмножество ее строк $i_1, \dots, i_k$, что для каждого j, где $j = 1, \dots, n$, найдется такой номер $s = s(j)$, $1 \le s(j) \le k$, что $d_{s(j),\,j} = 1$. Другими словами, подмножество строк $i_1, \dots, i_k$ матрицы D является ее покрытием, если в подматрице, образованной этими строками, нет нулевых столбцов.

Представляет интерес поиск кратчайшего покрытия матрицы, под которым понимается покрытие матрицы наименьшей мощности (наименьшее количество строк).

Для нахождения покрытий бинарной матрицы можно использовать следующие методы: метод Патрика, метод Закревского строчного и столбцового покрытий и другие. [2,49,61]

Одним из эффективных методов получения покрытий, широко применяемым на практике, является метод Патрика, который обеспечивает получение полного списка покрытий заданной матрицы. [2]

Для нахождения всех функциональных покрытий матрицы D методом Патрика необходимо представить каждый столбец матрицы в виде дизъюнкции библиотек, которые реализуют данную функцию. [2]

Рассмотрим пример.

Пусть имеется 5 библиотек и 4 функции, которые необходимо реализовать с помощью этих библиотек.

Матрица D выглядит следующим образом:

$$D = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} \text{а} \\ \text{б} \\ \text{в} \\ \text{г} \\ \text{д} \end{matrix}$$

Первый столбец покрывается строками б и г, второй столбец – б и г, третий столбец – а и в, четвертый столбец – а, б, д.

Составим конъюнкцию перечисленных дизъюнкций:

(б ∪ г) ∩ (б ∪ г) ∩ (а ∪ в) ∩ (а ∪ б ∪ д) = ба ∩ бад ∩ бва ∩ бвд ∩ бга ∩ бгад ∩ бгва ∩ бгв ∩ бгвд ∩ га ∩ гад ∩ гва. (2.5)

Из уравнения следует, что для реализации 3D ГИС формируется 12 проектных решений по числу комбинаций библиотек, покрывающих все столбцы матрицы *D*. Среди них наиболее эффективными с точки зрения количества библиотек будут комбинации библиотек {а, б} и {а, г}.

2.1.3 Геометрическая модель 3D ГИС

При всяком проектировании важнейшую роль играет минимизация затрат на разработку, поэтому подбирается некоторое множество критериев, выполнение которых ее обеспечивает. Однако многокритериальность затрудняет во многих случаях процесс анализа их достижимости, и тогда прибегают к выделению основного критерия, а остальные являются дополнительными и служат для исключения неопределенности выбора решения, когда знания значения основного критерия является недостаточными.

Представим геометрическую информацию как:

- форма круга, метрическими характеристиками которого являются: длина радиуса *R*, площадь *S*;
- форма сектора круга, метрическими характеристиками которого являются: длина дуги *L*, угол α', площадь S';
- форма треугольника, метрической характеристикой которого является длина дуги *L*, угол α'', площадь S'';
- параметры: угол, размеры и местоположение сектора относительно начала системы отсчёта.

Пусть в двумерном пространстве имеется круг радиусом *R* и покрывающие его геометрические объекты (секторы и составляющие их сегменты и треугольники), где *M* – общее количество заданных покрывающих объектов (секторов круга с заданными метрическими характеристиками и их теоретико-множественными комбинациями).

Требуется расположить геометрические объекты (секторы) на покрываемой площади круга таким образом, чтобы вся площадь была покрыта целиком.

В качестве геометрической модели выбрана круговая диаграмма (Рисунок 2.2), представляющая собой круг с дискретизацией *n*-секторами одинакового размера, число которых равно $\frac{2*\pi}{n}$, где *n* – количество функций, требуемых для покрытия заданной функциональности проектируемого 3D ГИС.

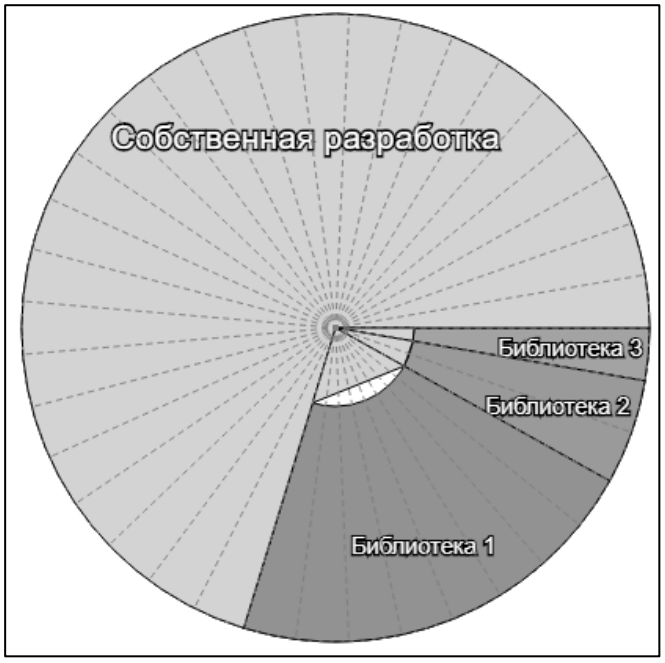

Рисунок 2.2. Круговая диаграмма 3D ГИС

Основным элементом круговой диаграммы выступает геометрическая фигура – единичный сектор, которым представляется функция из заданной функциональности 3D ГИС.

Объединением секторов обозначается библиотека или собственная разработка, реализующая набор функций из заданной функциональности, а треугольником соответствующего этой библиотеки сектора круга меньшего радиуса (R_1) – затраты на собственные доработки средств подключения и взаимодействия библиотек.

Библиотека представляет собой информационный объект, реализующий одну или несколько функций, которые могут входить в состав заданной функциональности. Отсюда библиотека, реализующая *m*-функций, будет отображаться в виде *m*-единичных секторов.

Единичные секторы, отображающие библиотеку с *m*-функциями, занимают соседние позиции круговой диаграммы и имеют определённую площадь. У библиотек с разным количеством функций площади соответствующих им объектов круговой диаграммы будут различными.

Две библиотеки называются пересекающимися, если у них имеется хотя бы одна одинаковая функция. Заполнение исходной функциональности 3D ГИС может производиться с помощью пересекающихся библиотек, когда одна часть необходимых функций содержится в одной библиотеке, а другая – во второй. На круговой диаграмме обе библиотеки занимают два соседних сектора и отображаются суммарной площадью $S_1 + S_2$. Но на практике одинаковые функции должны быть реализованы только одной из этих библиотек. Поэтому угол сектора библиотеки, которая не реализует одну из повторяющихся функций, на круговой диаграмме должен быть уменьшен. В противном случае при большом количестве пересека-

ющихся библиотек геометрическое покрытие будет избыточным, что приведёт к наложению секторов друг на друга. [107]

Полное покрытие функциональности 3D ГИС характеризуется количеством затрат (Z_{API}) на разработку средств подключения и взаимодействия библиотек, выражаемых площадями треугольников соответствующих им секторов круга меньшего радиуса, и затрат ($Z_{\text{соб.раз.}}$) на собственные разработки непокрытых заданных функций:

$$Z = Z_{API} + Z_{\text{соб.раз}}. \tag{2.6}$$

При выборе библиотек встречаются случаи, когда требуется из нескольких библиотек, покрывающих одинаковые функции, выбрать лучшую по некоторому критерию. Таким критерием может служить коэффициент затрат на разработку средств подключения и взаимодействия библиотеки с 3D ГИС.

Экспериментальным путём получена зависимость коэффициента затрат на подключение библиотеки от количества функций, которые она покрывает. Данная зависимость вычисляется как отношение площади треугольника затрат библиотеки (S_2), покрывающей n-функций, к сумме площадей треугольников (S_1) затрат на подключение каждой из этих функций отдельно:

$$K(n) = \frac{S_2}{n * S_1} = \frac{\sin(n*\varphi)}{n*\sin(\varphi)}, \tag{2.7}$$

где n – количество функций, покрываемых библиотекой, φ – угол единичного сектора.

На графике (Рисунок 2.3) представлена зависимость коэффициента затрат на создание средств подключения и взаимодействия библиотеки с 3D ГИС от количества функций, покрываемых этой библиотекой. На основе расчётов выявлено, что с увеличением количества функций, покрываемых библиотекой, уменьшается коэффициент затрат на подключение этой библиотеки.

Рисунок 2.3. Зависимость коэффициента затрат на разработку средств подключения и взаимодействия библиотеки с 3D ГИС от количества покрываемых функций

Рабочий диапазон коэффициента затрат: $[1, \frac{N}{2})$, где N – суммарное количество функций из заданной функциональности. В случае, если библиотека покрывает одну функцию, то коэффициент затрат на разработку средств подключения и взаимодействия библиотеки с 3D ГИС равен *1*. При покрытии библиотекой $\frac{N}{2}$ функций коэффициент затрат стремится к нулю.

Сложность собственных разработок покрытия и обеспечения подключения библиотеки к 3D ГИС имеет важное значение в процессе его проектирования с учётом использования свободно распространяемых библиотек. Реализация каждой функции, полученной из библиотеки или собственной разработки, имеет свою сложность разработки.

На практике наиболее часто используются количественные метрики программного обеспечения, такие как: количество строк кода (SLOC), цикломатическая сложность, количество операторов, анализ функциональных точек, среднее число строк для функций (классов, файлов, модулей).

В настоящей работе предлагается оценка сложности функции по количеству реализующих её операторов как наиболее удобная для практических применений. В случае, если *i*-я функция покрывается какой-либо библиотекой, то сложность функции (Z_i) определяется количеством операторов в составе функции этой библиотеки.

Тогда сложность библиотеки будет вычисляться как сумма сложностей функций, которые она покрывает:

$$Z' = \sum_{i=1}^{k} Z_i, \tag{2.8}$$

где k – количество функций, покрываемых библиотекой.

Зная сложность функций, реализуемых библиотеками, и суммарный угол, который они занимают, можно определить среднее значение сложности на единицу угла, отождествляемое с единицей площади на геометрической модели для измерения затрат на собственные разработки:

$$Z_{\text{ср.}} = \frac{\alpha * \sum_{i=1}^{n} Z_i}{360^\circ} \text{ операторов}, \tag{2.9}$$

где α – угол сектора, занимаемого библиотеками, n – количество функций, покрываемых библиотеками.

Сложность собственных разработок функций, которые не покрываются библиотеками, предлагается оценивать как произведение угла (β) сектора, который они занимают, на среднее значение сложности ($Z_{\text{ср}}$):

$$Z_{\text{соб.раз.}} = \beta * Z_{\text{ср.}} \text{ операторов}. \tag{2.10}$$

2.1.4 Взаимодействие моделей при формировании дерева библиотек

Для наглядного представления функциональных покрытий наиболее удобна их визуализация в виде пространства состояний – дерева библиотек с возможностью поиска по нему оптимального покрытия по заданному критерию. Глубина «ветки» дерева библиотек (без начальной вершины) соответствует количеству библиотек в покрытии. Количество «листьев» в дереве библиотек соответствует количеству допустимых покрытий для текущего набора функций.

В пространстве состояний, отображаемом в виде дерева библиотек, встречаются библиотеки с различным количеством покрываемых функций. Их выбор влияет на длину пути в дереве и на количество собственных затрат. [118,121]

Отсюда возникают два варианта построения дерева:

I. Первый вариант: построение дерева по минимальному количеству библиотек. Поиск в дереве идёт сверху вниз, отталкиваясь от корневого узла, а листьями являются множество библиотек, покрывающих заданную функциональность.

Алгоритм построения дерева библиотек имеет вид:

1) Обозначается корневая вершина.
2) Преобразовывается матрица *D*, исключаются из неё все столбцы, описывающие функции, не входящие в вектор $\bar{f}$.
3) Проводится поиск множества функций $\{f_i\}$, которые реализованы меньшим количеством библиотек, т. е. столбцы с минимальной суммой элементов в преобразованной матрице *D*:

$$\sum_j^m c_{i,j} \to min. \qquad (2.11)$$

4) Для каждой функции из полученного множества формируется множество библиотек $\{l_j\}$, реализующих найденные функции в пункте 3.
5) Для каждой из них строится вершина в дереве библиотек и преобразовывается матрица *D*, из которой исключаются все столбцы, описывающие функции, входящие в библиотеку l_j.
6) Рекурсивно осуществляется переход на шаг 3 для каждой из выбранных библиотек, если в матрице *D* остались столбцы, в противном случае – строится конечная вершина дерева.

II. Второй вариант: построение дерева по коэффициенту затрат на разработку средств по подключению и взаимодействию библиотек с 3D ГИС. В основу алгоритма заложен итерационно-рекурсивный процесс, в котором рекурсия используется для поиска минимального коэффициента затрат в матрице затрат *W*, а итерация обеспечивает поша-

говый подбор библиотеки с минимальными затратами и последующее уменьшение размеров матрицы. Поиск оптимального пути в этом дереве начинается от корня, листьями являются библиотеки с характеризующими их коэффициентами собственных затрат на разработку средств подключения и взаимодействия библиотек с 3D ГИС.

Алгоритм построения дерева библиотек с использованием коэффициента затрат:

1) Формируется матрица (W) коэффициентов затрат, в которой столбцами обозначаются функции из заданной функциональности, а строками – библиотеки, их покрывающие. На пересечении строки и столбца матрицы записывается 0, если библиотека не покрывает данную функцию, в противном случае – коэффициент затрат (K_j) на разработку средств подключения и взаимодействия j-й библиотеки с 3D ГИС.

   Матрица затрат выглядит следующим образом:

$W =$	f_1	f_2	…	f_n
l_1	$w_{1,1}$	$w_{1,2}$	…	$w_{1,n}$
l_2	$w_{2,1}$	$w_{2,n}$	…	$w_{2,n}$
…	…	…	…	…
l_m	$w_{m,1}$	$w_{m,2}$	…	$w_{m,n}$

   где $w_{i,j} = K_j$, если библиотека l_j покрывает функцию f_i, в противном случае – 0.
2) Задаётся корневая вершина дерева.
3) Выполняется поиск ячейки с минимальным коэффициентом затрат на разработку средств подключения и взаимодействия библиотеки с 3D ГИС.
4) К корневой вершине добавляется библиотека, соответствующая выбранной на шаге 3 ячейке матрицы W. Если найдено несколько ячеек с одинаковым коэффициентом затрат, то записываются все библиотеки, им соответствующие.
5) Рекурсивно для каждой выбранной библиотеки выполняется удаление из матрицы W строки, соответствующей этой библиотеке, и всех столбцов, которые она покрывает, библиотека указывается в качестве корневой вершины текущего поддерева, и, если матрица W непустая, повторяются шаги 3–5.

Использование второго дерева актуально при создании 3D ГИС с ограниченными затратами на разработку, которые можно контролировать с помощью геометрической модели по площади секторов, отражающих собственные разработки и связанных с затратами на программную реализацию.

В случае подбора библиотек может оказаться, что две библиотеки являются эквивалентными, т. е. покрывают одни и те же функции. Для выбора одной из них предлагается делать оценку по дополнительным критериям:

- однородность языков программирования у библиотек;
- платформа, на которой библиотеки функционируют;
- системные и аппаратные требования библиотек;
- критерий неизбыточности кода покрытия библиотек (минимум мощности разности множеств функций из библиотек покрытия и требуемых функций):

$$\left(\sum_i^l \sum_j^n \left(\left|c, c_{i,j} = 1\right| + \left|\bar{f}\right| - 2\left|c, c_{i,j} = 1 \cap \bar{f}\right|\right)\right) \to min, \quad (2.12)$$

где *l* – количество библиотек в покрытии;

- критерий максимальной расширяемости из *l* библиотек (максимум мощности множества объединения всех функций библиотек покрытия):

$$\left|\overline{f_1} \cup .. \cup \overline{f_l}\right| \to max, \quad (2.13)$$

где *l* – количество библиотек в покрытии.

Поиск оптимального пути на дереве библиотек возможен следующими способами:

- в глубину с учетом выбранного критерия;
- в ширину с учетом выбранного критерия;
- эвристический (эвристика определяется разработчиком).

На практике с большим количеством библиотек возможно использование дополнительных эвристик, которые позволяют сократить вычислительные затраты и не строить всё дерево библиотек. Например, в качестве эвристики можно использовать максимум реализованных функций в библиотеке (для критерия расширяемости). Тогда в алгоритме построения дерева библиотек будут формироваться вершины не для всех библиотек, а только для тех, у которых сумма элементов в строке матрицы *D* максимально:

$$\sum_i^n c_{i,j} \to max. \quad (2.14)$$

Использование дополнительных критериев и дерева библиотек обеспечивает разработчику гибкость управления процессом проектирования.

В целом, схема взаимодействия функциональной и геометрической моделей в процессе получения проектного решения отображается на рисунке .4, которая показывает, что входными данными в функциональную модель являются 3D ГИС и база библиотек, которые полностью или частично её покрывают. Между функциональной и геометрической моделями организовано двухстороннее взаимодействие, где из функциональной модели в геометриче-

скую передаётся информация об используемых библиотеках и функциях в виде матрицы соответствия *D*, а в обратную сторону – вычисленные с помощью геометрической модели коэффициенты затрат на разработку средств по подключению и взаимодействию библиотек с 3D ГИС. С использованием функциональной и геометрической моделей имеется возможность построения дерева библиотек двух вариантов: по минимуму функций и по минимуму коэффициентов затрат на подключение библиотек.

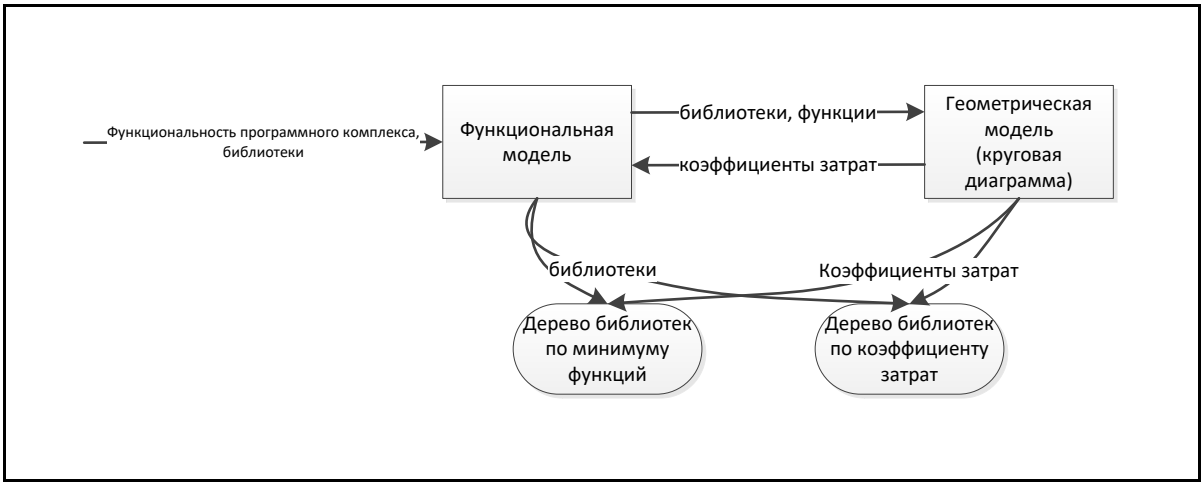

Рисунок 2.4. Схема взаимодействия функциональной и геометрической моделей

Процесс формирования дерева библиотек с помощью функциональной и геометрической моделей по первому алгоритму представлен на рисунке .5. Построение дерева начинается с корня с обращением к матрице *D*, и последовательном обходе узлов. На круговой диаграмме, расположенной слева, проход каждого узла отображается добавлением сектора с указанием имени библиотеки и количества реализуемых функций. В центре круга выделены площади треугольников, охватывающих секторы соответствующих библиотек и обозначающих необходимость дополнительных затрат (*K*) по подключению и взаимодействию библиотек с 3D ГИС. [107]

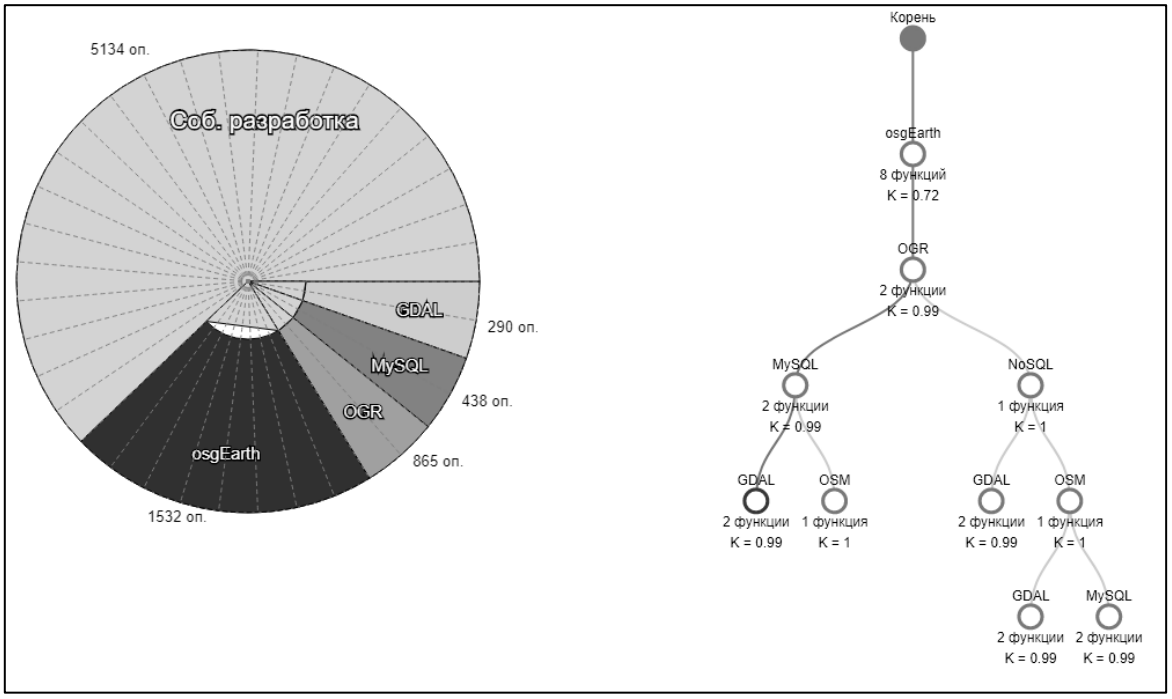

Рисунок 2.5. Дерево библиотек и круговая диаграмма по первому алгоритму

§ 2.2. Структурно-компонентная модель CASE-средства проектирования 3D ГИС

Проектирование 3D ГИС является сложной задачей из-за некоторых особенностей в функциональных возможностях таких систем.

К особенностям функционирования 3D ГИС относятся:

- обработка видеоизображений;
- преобразование трёхмерных растровых изображений в векторные графические модели;
- обработка картографической информации с учётом рельефа;
- обработка неоднородной информации;
- построение 3D моделей объектов или местности;
- анализ 3D моделей ГИС;
- получение информации о текущей ситуации;
- получение решений на основе геоинформации. [90]

Любая 3D ГИС взаимодействует с особым видом данных – пространственными данными. Основной отличительной особенностью пространственных данных является то, что они состоят из двух взаимосвязанных частей: метрических и семантических данных. Для хранения таких данных необходимо использовать специальные базы данных, средства работы с трехмерной графикой и 3D моделями объектов и средства анализа геоинформации. Все

эти отличительные черты влияют на компонентный состав CASE-средств для проектирования 3D ГИС.

На основе функциональных покрытий и дерева библиотек (§2.1), а также особенностей функционирования 3D ГИС в целом предлагается структурно-компонентная модель CASE-средства для проектирования 3D ГИС отображения ситуационной обстановки (Рисунок 2.6).

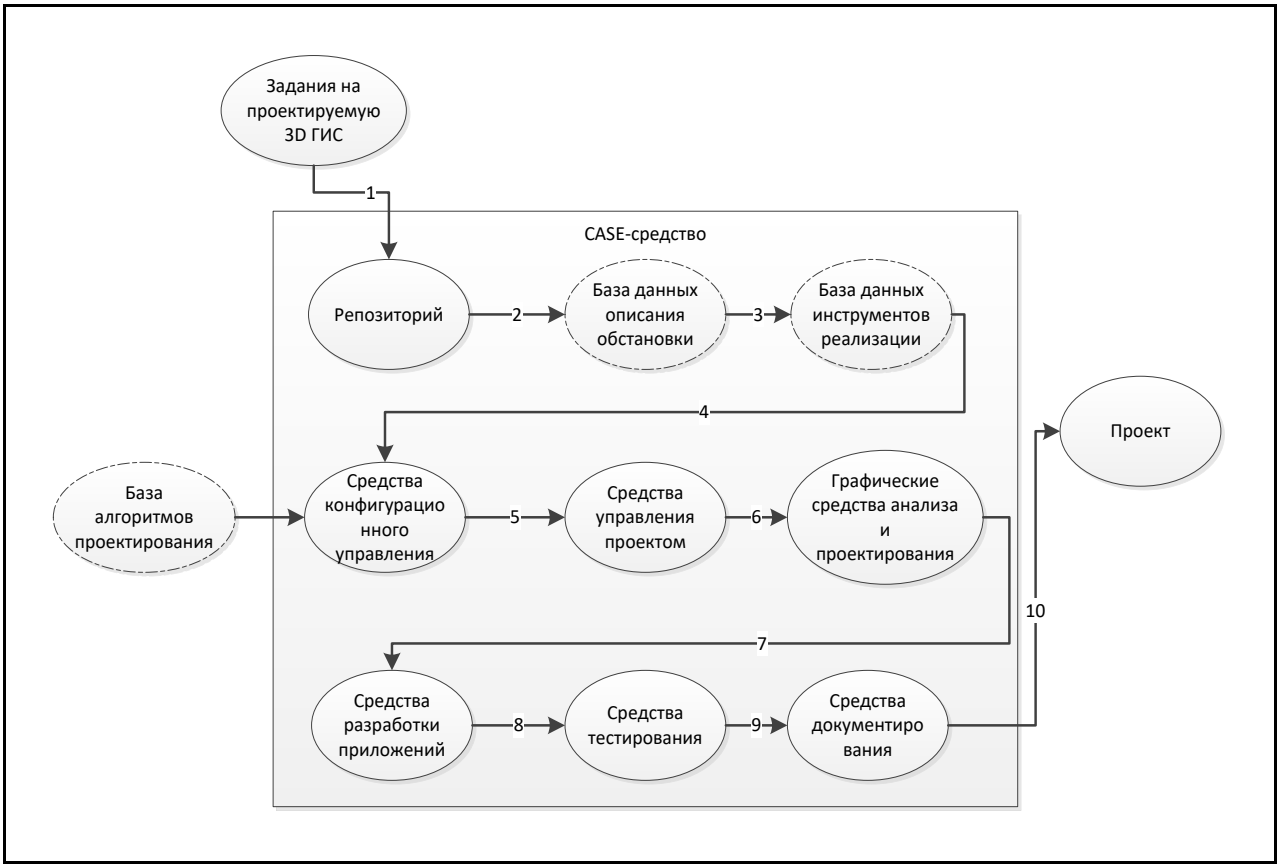

Рисунок 2.6. Структурно-компонентная модель CASE-средства для проектирования 3D ГИС

Для решения задач проектирования 3D ГИС введены следующие компоненты, уникальные для таких систем (выделены пунктирными линиями):

- база данных описания обстановки, которая хранит информацию обо всех объектах обстановки и ситуациях, которые могут происходить;
- база данных инструментов реализации, которая содержит набор библиотек для разработки 3D ГИС и инструментов для создания моделей объектов и картографических основ системы;
- база алгоритмов проектирования, которая содержит готовые алгоритмы, необходимые для создания проекта 3D ГИС.

На вход CASE-средства поступают задания на проектируемую 3D ГИС. В качестве внешних ресурсов используется база алгоритмов проектирования. Результатом функционирования CASE-средства является проект на создаваемую 3D ГИС. Числа на связях между компонентами на рисунке 2.6 введены для указания последовательности вызовов средств при проектировании 3D ГИС.

§ 2.3. Система автоматизации проектирования 3D ГИС на основе CASE-средства

В основу САПР 3D ГИС положено CASE-средство для генерации проектных решений, которое представляет собой систему взаимосвязанных программно-реализованных процедур, алгоритмов, моделей и диаграмм, образующих эти решения, для реализации ситуационной обстановки, 3D глобуса, трёхмерных объектов, модели рельефа и местности. [38]

При помощи созданного CASE-средства проектирования 3D ГИС отображения ситуационной обстановки разработана система моделирования проектных решений, которая апробирована на практике и позволяет оценивать сформированные CASE-средством проектные решения. В случае неудовлетворения требованиям разработанное проектное решение пересматривается с помощью CASE-средства, изменяется и вновь передаётся в систему 3D моделирования для его повторного анализа. Процесс носит цикличный характер и заканчивается после устранения замеченных недостатков. [25,36,59,108]

Структурно-функциональная модель САПР 3D ГИС на основе CASE-средства представлена на рисунке 2.7.

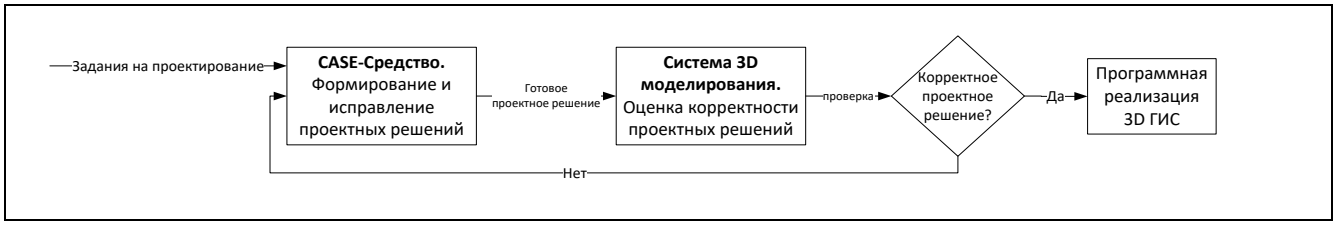

Рисунок 2.7. Структурно-функциональная модель системы автоматизации проектирования 3D ГИС на основе CASE-средства

Предложенная система автоматизации проектирования 3D ГИС отображения ситуационной обстановки содержит 5 укрупнённых этапов.

1. Предварительный этап, включающий формирование заданий на проектируемую 3D ГИС отображения ситуационной обстановки, создание базы инструментов реализации, базы алгоритмов проектирования и функционирования 3D ГИС, базы данных описания обстановки. Этап заканчивается определением полной функциональности проектируемой си-

стемы и формированием инструментария в виде набора удовлетворяющих этой функциональности инструментов разработки.

2. Этап проектирования. На этом этапе используется CASE-средство проектирования 3D ГИС отображения ситуационной обстановки, которое на основе введенных проектировщиком требований к системе формирует проектные решения, включающие модели и диаграммы 3D ГИС, удовлетворяющие требованиям инструменты реализации и исходные коды файлов заголовков.
3. Этап моделирования проектных решений. На этом этапе при помощи разработанной с использованием CASE-средства системы 3D моделирования проводится визуальная оценка проектного решения и принимается решение о его программной реализации.
4. Этап программной реализации проектного решения, на котором программистами разрабатываются необходимые функции, в том числе:
    - построение трехмерной модели поверхности Земли в каркасном или текстурированном виде;
    - масштабирование трёхмерной модели с автоматической генерализацией объектов;
    - навигация по трёхмерной сцене в режиме реального времени;
    - выбор режима перемещения по трёхмерной модели;
    - отображение векторных и растровых карт;
    - поиск и выделение объектов по различным критериям отбора;
    - фиксация участка местности в заданном масштабе;
    - отображение пути по рельефу местности и перемещение виртуальной камеры по нему;
    - отображение классификатора 3D моделей.
5. Этап проведения моделирования и оценки качества выполняемых требований согласно поставленной задаче. На этом этапе осуществляется:
    - компоновка программных средств;
    - выбор среды моделирования;
    - подготовка тестовых данных;
    - запуск и пошаговый контроль системы;
    - загрузка тестовых данных (указание координат района Земли, загрузка векторных и растровых данных, загрузка 3D объектов);
    - навигация по 3D глобусу (масштабирование местности, переход по координатам);
    - управление слоями и объектами обстановки;
    - поиск и выделение объектов;

- взаимодействие 3D ГИС отображения ситуационной обстановки с внешними системами;
- моделирование движения 3D объектов.

С точки зрения реализации 3D ГИС отображения ситуационной обстановки из готового проектного решения с использованием предложенной САПР 3D ГИС осуществляется разработка алгоритмов, описывающих как функционирование системы целиком, так и отдельных её частей. [69,115]

Преимуществом использования алгоритмов является то, что при проектировании информационной системы можно полностью формализовать её функционирование, а в дальнейшем на основе построенных алгоритмов разработать рабочую программу на одном из языков программирования. Каждый алгоритм представляет собой изолированный модуль, который может разрабатываться отдельно и при необходимости подключаться к работающей системе, дополняя её новой функциональностью.

Все разрабатываемые алгоритмы представляют собой набор процессов, которые они выполняют, входных данных и условий, при которых выполняются те или иные процессы.

Модель реализации 3D ГИС отображения ситуационной обстановки на базе алгоритмов функционирования представлена на рисунке 2.8.

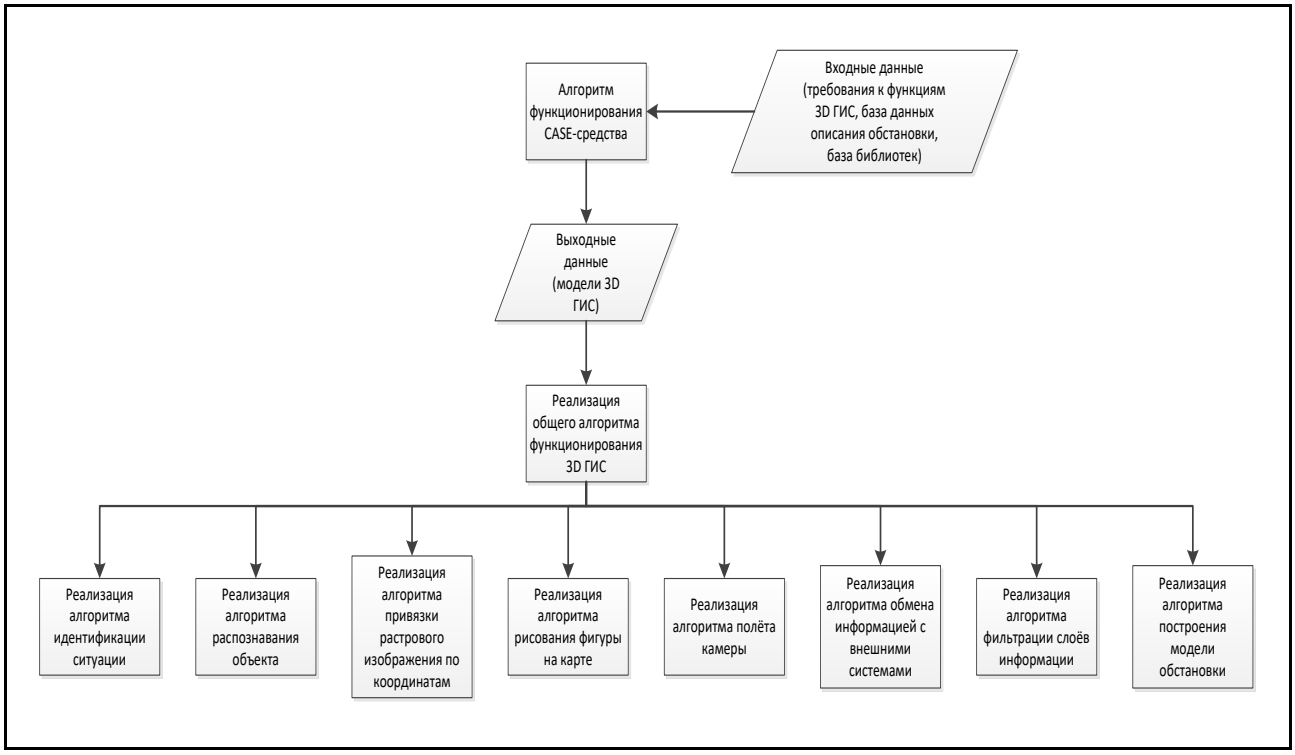

Рисунок 2.8. Модель реализации 3D ГИС на базе алгоритмов функционирования

В начале проектирования 3D ГИС используется CASE-средство, которое позволяет построить модели 3D ГИС: функциональную модель, модель обстановки, модель библиотек и др., а также сгенерировать файлы заголовков, которые в дальнейшем будут необходимы при реализации системы на конкретном языке программирования. Входными данными CASE-средства являются функциональные требования к проектируемой системе, база обстановки и база библиотек.

Выходные данные (проектные решения) CASE-средства оцениваются системой 3D моделирования и в случае удовлетворения требованиям используются при реализации общего алгоритма функционирования 3D ГИС. Данный алгоритм описывает основное поведение всей системы и функции, которые она выполняет. В дальнейшем для данного алгоритма разрабатываются дополнительные алгоритмы, описывающие функционирование каждого требования, введенного проектировщиком в качестве входного параметра CASE-средства.

Таким образом, замкнутая система «CASE-средство + система 3D моделирования» позволяет в итерационном режиме формировать необходимые по точности проектные решения и реализовывать их в виде законченных программных процедур после окончательного шага моделирования.

§ 2.4. CASE-средство проектирования 3D ГИС

Представленные в §2.2 компоненты системы проектирования 3D ГИС не используются в существующих CASE-средствах, из чего возникает необходимость разработки CASE-средства 3D ГИС отображения ситуационной обстановки, включающего эти компоненты и реализующего особенности функционирования 3D ГИС необходимой степени сложности и широкой области применения. [78,91]

Структурная модель CASE-средства для проектирования 3D ГИС отображения ситуационной обстановки представлена на рисунке 2.9.

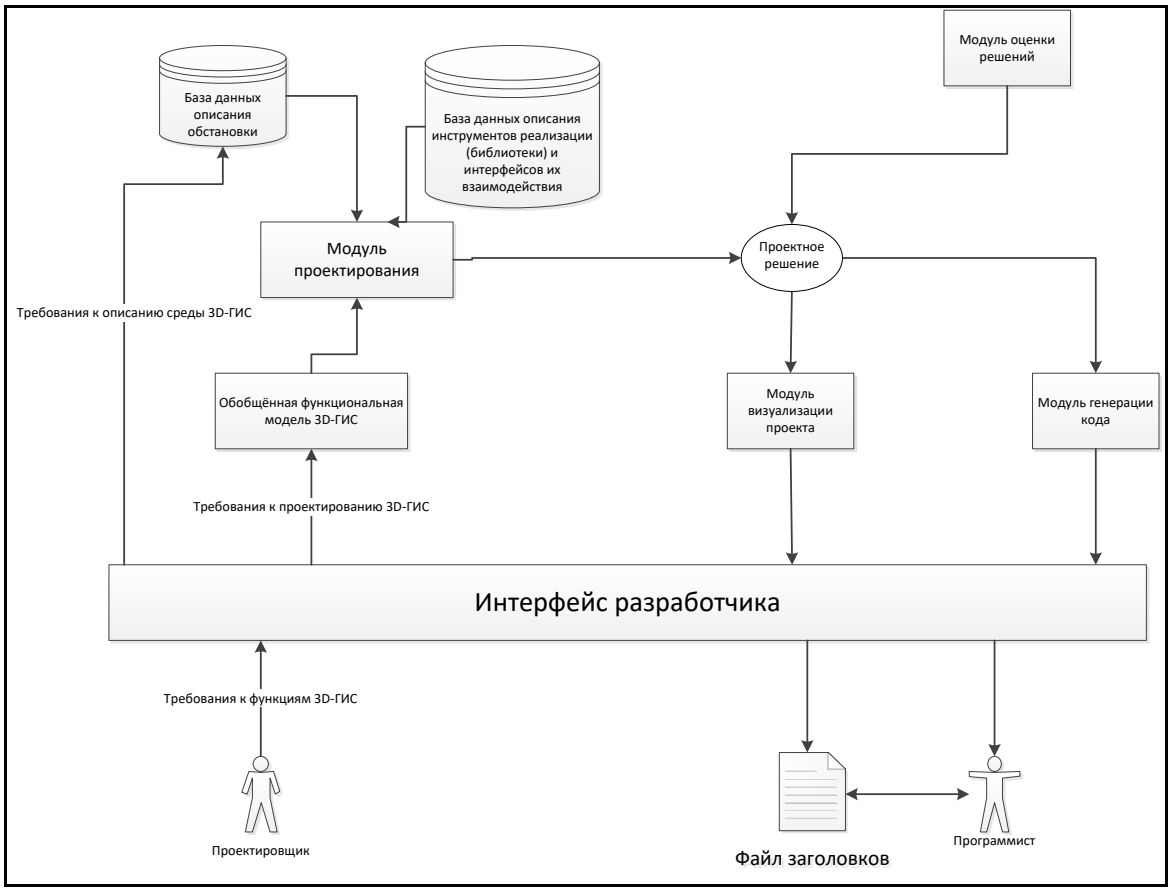

Рисунок 2.9. Структура CASE-средства проектирования 3D ГИС

CASE-средство имеет трехуровневое представление.

Первый уровень – уровень хранения данных – содержит базу данных описания обстановки и базу данных инструментов реализации. Эти базы строятся на основе двух реляционных моделей представления данных: модели обстановки и модели описания инструментов (Рисунок 3.5). Данные модели используются для описания функций и объектов проектируемой 3D ГИС, а именно на уровне представления информации находится интерфейс разработчика, который позволяет использовать модели и предоставить проектировщику средства для формализации функциональных требований. Также через интерфейс проектировщик и программист могут получать доступ к результатам проектирования, а именно: просматривать проектные решения в виде диаграмм и получать файлы заголовков на выбранном языке программирования. [22]

На втором уровне – уровне бизнес-логики – находятся функциональные модули, позволяющие на основе пересечения множеств функций обобщенной функциональной модели 3D ГИС (Рисунок 3.4) и функций, описанных проектировщиком, определять доступные для данного проекта библиотеки и на основе описания библиотек формировать проектные решения. [22]

В проектные решения входят:

- структурная модель, в которой библиотеки объединяются в функциональные подсистемы и обозначаются модули, требуемые для собственной разработки, и с учётом выбранных библиотек определяется архитектура (клиент-серверная, web-сервис-ориентированная и т.д.) 3D ГИС;
- функциональная модель проектируемой 3D ГИС на основе требований проектировщика, которая представляется в виде дерева функций или в виде UML-диаграммы прецедентов использования;
- диаграмма интерфейсов, которая описывает схему API между свободно распространяемыми библиотеками и собственными разработками в виде диаграмм классов UML.

На основе указанных требований может быть сформировано несколько вариантов проектных решений, которые автоматизированно оцениваются в модуле оценки решений по следующим критериям: количество строк кода в библиотеке, количество связей между библиотеками, однородность технологий разработки (языка описания, протоколов взаимодействия, архитектуры и т. д.).

На третьем уровне – уровне представления – находятся интерфейс разработчика и модуль визуализации. На базе полученных оценок пользователь может выбрать лучшее для него проектное решение и просмотреть его в виде диаграмм в модуле визуализации, а также на базе диаграммы классов получить заголовочные файлы с помощью модуля генерации кода. [22]

Предлагаемое CASE-средство осуществляет информационную поддержку проектировщику и разработчику на этапах анализа и проектирования 3D ГИС, а файлы заголовков являются результатом проектирования, используемым на этапе реализации 3D ГИС.

Процесс функционирования CASE-средства проектирования 3D ГИС отображения ситуационной обстановки, т.е. преобразования входных данных в выходные, представлен в виде модели BPMN (Рисунок 2.10). Пользователь поэтапно должен выполнить 4 шага: выбрать тип 3D ГИС, ее функции, отображаемые объекты и требования к реализации. После этого CASE-средство перейдет к построению и оценке проектных решений. На диаграмме отображена зависимость процесса построения моделей и диаграмм 3D ГИС от действий пользователя. [22]

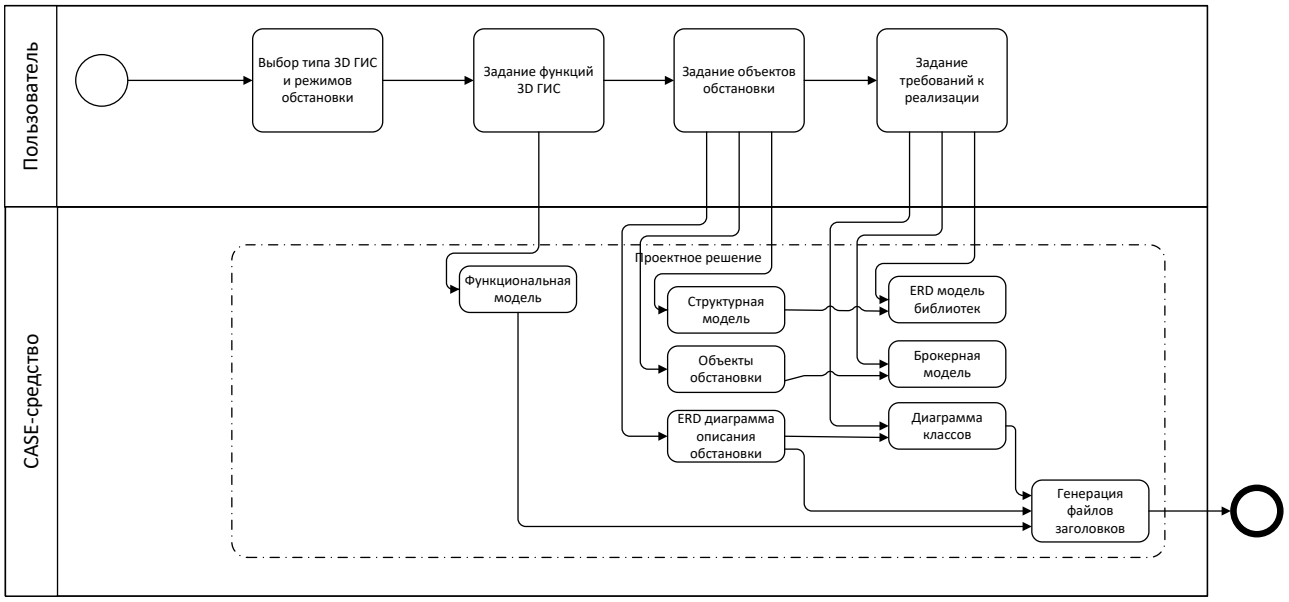

Рисунок 2.10. Модель бизнес-процесса функционирования CASE-средства проектирования 3D ГИС

Набор моделей и диаграмм является достаточным для запуска процедуры генерации файлов заголовков. Файл заголовков представляет собой указания классов используемых библиотек или алгоритмов функционирования 3D ГИС, которые реализованы в данной работе и могут быть реально применены на практике при реализации 3D ГИС. [22]

§ 2.5. Модель функционирования CASE-средства проектирования 3D ГИС

В основу проектирования 3D ГИС положено ядро, состав которого является необходимым атрибутом всякой аналогичной системы и отражает такие функции, как визуализация обстановки, имитация движения трехмерных объектов, работа с событиями внешнего мира, интерфейсная часть. Ядро 3D ГИС представляет собой самостоятельный программный компонент, который может взаимодействовать с дополнительными модулями, такими как: модуль определения пересечений объектов, модуль определения взаимного расположения двух объектов, модуль экспорта и печати обстановки и 3D объектов на местности. [22]

CASE-средство проектирования 3D ГИС отображения ситуационной обстановки выполняет две основные задачи:

1. проектирование ядра 3D ГИС;
2. проектирование проектных решений 3D ГИС на базе выбранного ядра.

На этапе проектирования ядра 3D ГИС проектировщику необходимо указать несколько критериев, на основе которых CASE-средство предложит набор подходящих ядер 3D ГИС.

Модель функционирования CASE-средства 3D ГИС представлена на рисунке 2.11.

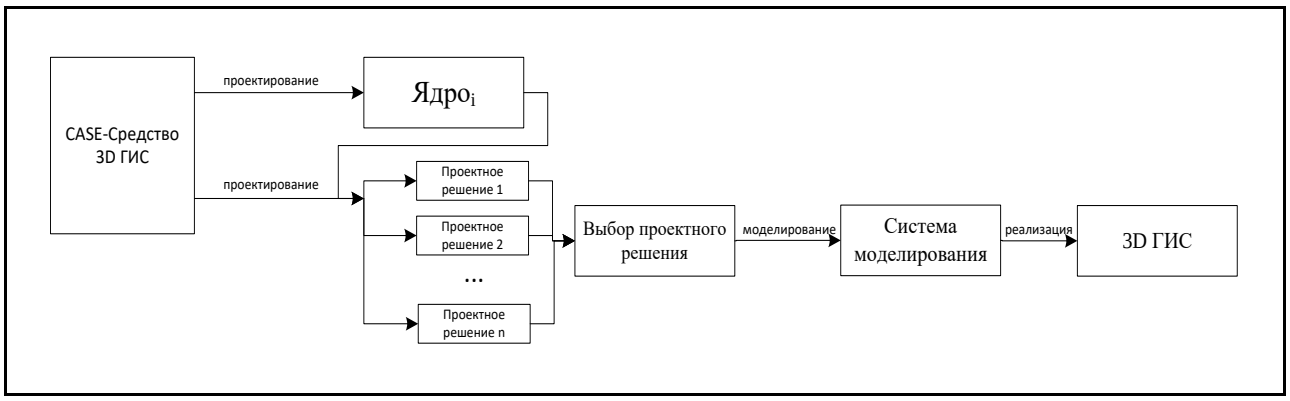

Рисунок 2.11. Модель функционирования CASE-средства 3D ГИС

Критерии выбора ядра 3D ГИС:

- тип 3D ГИС;
- платформа;
- архитектура;
- язык разработки.

Основным критерием при проектировании ядра является выбор типа разрабатываемой 3D ГИС. Предлагаются два типа ядер: базовое ядро и специализированное ядро.

Базовые ядра содержат только те функции, которые необходимы для разработки любой 3D ГИС. Ниже представлена функциональность базового ядра.

$F_{\text{ядра}} = \{f_1, \dots, f_n\}$, где n – количество функций базового ядра. (2.15)

Для каждой функции из базы библиотек выбираются те библиотеки, которые её реализуют. Набор библиотек, выполняющих одну функцию можно представить как:

$L_{f_i} = \{L_{f_i,1}, \dots, L_{f_i,m}\}$, где m – количество выбранных библиотек. (2.16)

На основе выбранных для каждой функции библиотеки из всевозможных сочетаний формируются ядра 3D ГИС. Полный набор библиотек ядра можно представить:

$L_{\text{ядра}} = \{L_{f_1}, \dots, L_{f_n}\}$, где n – количество функций ядра. (2.17)

Кроме базовых ядер 3D ГИС CASE-средство позволяет спроектировать специализированные ядра. Такие ядра 3D ГИС имеют набор библиотек с функциями по реализации 3D ГИС в конкретной области применения, например, геологическая ГИС, метеоГИС, архитектурная ГИС и т.д. После выбора области применения проектировщик аналогично генерации базового ядра может выбрать критерии для специализированного ядра (платформа, структура, язык разработки). [62,63]

Выбор платформы для ядра задаёт операционные системы, для которых разрабатывается 3D ГИС. Наиболее распространёнными являются: Windows, Linux, UNIX, Mac OS, An-

droid, IOS. Некоторые библиотеки обеспечивают кроссплатформенность 3D ГИС, т.е. возможность функционирования на нескольких операционных системах.

Архитектура ядра определяет технологию передачи данных оператору 3D ГИС.

Выделяют 3 основных технологии:

- файл-сервер;
- клиент-сервер;
- терминал-сервер.

Файл-сервер является исторически первой архитектурой, в которой происходит извлечение данных из файлов и передача их клиенту для дальнейшей обработки. В процессе работы из базы данных клиенту передаются большие объемы информации. Значительный сетевой трафик иногда особенно сильно сказывается при одновременной работе даже уже нескольких клиентов.

Информационные системы с *клиент-серверной архитектурой* позволяют избежать проблем файл-серверных приложений. При такой архитектуре сервер базы данных, расположенный на компьютере-сервере, обеспечивает выполнение основного объема обработки данных. Клиентское приложение формирует запросы к серверу базы данных, как правило, в виде инструкций языка SQL. Сервер извлекает из базы запрошенные данные и передает на компьютер клиента. Главное достоинство такой архитектуры — значительно меньший объем передаваемых данных. [45]

Технология *терминал-сервер* принципиально отличается от двух предыдущих тем, что конечному пользователю по сети передаются не сами интересующие его данные, а изображение этих данных. Логика процесса такова: пользователь подключается к так называемому "серверу терминалов" или "терминальному серверу" и сервер предоставляет пользователю свой «рабочий стол», свои программы и т.д. Фактически пользователь работает за другим компьютером, физически удаленным от него, получая по сети только изображение «рабочего стола» с запущенными программами с заданной частотой. [10945]

Выбор языка программирования определяется тем, на каком языке программисты на этапе реализации 3D ГИС смогут построить её ядро, при необходимости написать исходный код адаптеров и собственных разработок.

После того, как проектировщик при помощи CASE-средства спроектировал ядра 3D ГИС, ему необходимо произвести анализ предложенных вариантов и выбрать наиболее подходящий. Проектировщик может выбрать ядро вручную на основе экспертных оценок или своего собственного опыта. В свою очередь, CASE-средство предлагает свои собственные оценки эффективности того или иного ядра 3D ГИС на основе значений некоторых критериев.

Такими критериями являются:

- количество строк кода библиотек;
- суммарное время выполнения функций библиотек;
- среднее количество входных переменных;
- количество адаптеров (специальных модулей, которые позволяют библиотекам взаимодействовать между собой);
- наличие API у библиотек;
- количество библиотек;
- различия в языках программирования, на которых реализованы библиотеки;
- кроссплатформенность.

Необходимо отметить, что несколько функций ядра могут быть обеспечены одной и той же библиотекой, что может привести к их дублированию. Так, если одну и ту же функцию реализуют две библиотеки и одна из этих библиотек уже используется для обеспечения другой функции, то целесообразнее для первой функции использовать эту же библиотеку.

После того, как проектировщик провёл анализ предложенных ядер 3D ГИС и выбрал наиболее подходящее, программисты могут приступать к программной реализации выбранного ядра и в дальнейшем получить полностью функционирующую 3D ГИС.

§ 2.6. Проектные решения 3D ГИС

После формирования ядра 3D ГИС существует возможность дальнейшего проектирования 3D ГИС на базе выбранного ядра с целью расширения её функциональности. Проектировщик в CASE-средстве выбирает дополнительные функции, объекты, которые необходимо добавить в 3D ГИС. После выбора всех необходимых данных CASE-средство генерирует перечень проектных решений, готовых к реализации. [34,92]

Для каждого проектного решения CASE-средство расчитывает приблизительную сложность. Оценка сложности производится по формуле:

$$S = a * n_{custom} + b * n_{line} + c * n_{time} + d * n_{input} + e * n_{lib} + f * n_{ad},$$

(2.18)

где n_{custom} – количество собственных разработок,

n_{line} – общее количество строк кода всех библиотек,

n_{time} – общее время выполнения функций библиотек,

n_{input} – среднее количество входных данных API библиотек,

n_{lib} – количество библиотек,

n_{ad} – количество адаптеров,

a, b, c, d, e, f – веса соответствующих характеристик.

Все веса оцениваются и задаются экспертами в зависимости от практического использования CASE-средства и опыта проектирования 3D ГИС.

Проектные решения можно частично смоделировать на разработанном ядре 3D ГИС, которое в данном случае является системой 3D моделирования обстановки. Данная система позволяет отобразить большинство объектов 3D ГИС, а также те функции, которые реализованы в ядре. Для моделирования функций, не заданных в ядре, программистам необходимо подключить с помощью API или специальных адаптеров библиотеку, которая обеспечивает требуемую функцию или реализовать собственную разработку в зависимости от «указаний» моделируемого проектного решения.

Проектное решение включает в себя модели и диаграммы 3D ГИС, а также исходные коды файлов заголовков. Структура проектного решения 3D ГИС представлена на рисунке 2.12.

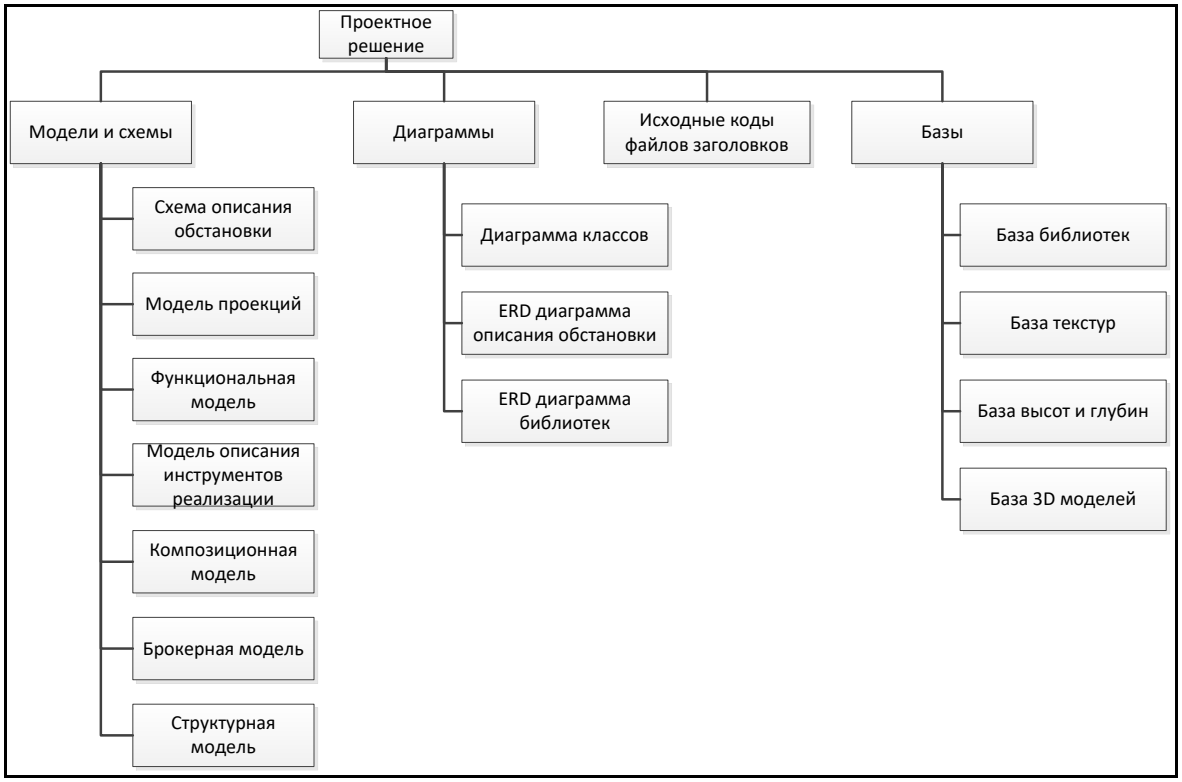

Рисунок 2.12. Структура проектного решения 3D ГИС отображения ситуационной обстановки

Схема описания обстановки включает в себя местность, модели текстур и рельефа, динамические и статические объекты, которые взаимодействуют между собой, и ситуации.

Функциональная модель 3D ГИС включает в себя такие работы, как: работа с графикой, работа с Интернет, работа с базами данных, работа с геоданными. Отдельно выделяется

работа с файловой системой, без которой невозможно обеспечить основные функции 3D ГИС.

Модель проекций представляет собой отображение функциональных возможностей библиотек на компоненты ядра. Связи между библиотеками и компонентами ядра отражают его функциональную нагрузку. Функции отдельного компонента ядра могут быть обеспечены несколькими библиотеками. Некоторые библиотеки обладают большим количеством функций и могут охватить несколько требований того или иного компонента ядра. Построение модели проекций завершается при возможности полным покрытием функций компонентов ядра библиотеками.

Модель описания инструментов реализации отображает библиотеки, которые используются при разработке 3D ГИС и интерфейсы их взаимодействия между собой.

Композиционная модель 3D ГИС представляет собой функциональность 3D ГИС, которая подвергается разбиению на множество составных функций и часть этих функций может быть реализована с помощью библиотек, а функции, для которых нет библиотек, реализуются собственными разработками. Наращиваемость 3D ГИС осуществляется на базе реализаций составных функций с помощью библиотек и собственных разработок.

Структурная модель 3D ГИС описывает компоненты системы и их взаимосвязи. Каждый компонент может быть разбит на более мелкие составляющие – модули.

Брокерная модель отображает способы взаимодействия ядра 3D ГИС с библиотеками и собственными разработками. В случае несоответствия интерфейсов библиотеки интерфейсу ядра модель предлагает разработку адаптера – специального элемента, который обеспечивает преобразование данных из формата, поддерживаемого библиотекой, в формат ядра 3D ГИС и наоборот.

Диаграмма классов описывает модель предметной области, в которой присутствуют только классы прикладных объектов. Каждый объект в диаграмме классов имеет уникальный идентификатор, название, атрибуты и методы. На основе диаграммы классов CASE-средство генерирует *файлы заголовков*, которые в дальнейшем могут использоваться при программной реализации 3D ГИС.

§ 2.7. Режимы функционирования CASE-средства

В CASE-средстве проектирования 3D ГИС предусмотрено два режима функционирования:

– ручное управление;

– автоматизированное управление с использованием мастера генерации проектных решений.

Ручное управление CASE-средством производится на его панели управления и позволяет пользователю задавать функции 3D ГИС, объекты обстановки, ситуации и требования к реализации 3D ГИС. После того, как все требования к проектируемой 3D ГИС заданы, пользователь вручную запускает генерацию проектных решений активацией соответствующего элемента на панели управления CASE-средства. Дополнительной возможностью является выбор типа ядра 3D ГИС и режимов обстановки. В зависимости от выбранного типа 3D ГИС CASE-средство автоматически задаёт обязательные функции и объекты обстановки и по умолчанию формирует 3D ГИС на базовом ядре, включающем минимальный набор обязательных функций. Дополнительно существует возможность сохранения выбранных настроек в файл и последующая их загрузка.

В CASE-средстве для пользователя доступны режимы морской, воздушной и наземной обстановки. Каждый режим задаёт свой набор объектов обстановки, местности и ситуаций. Возможен выбор нескольких режимов для отображения смешанной обстановки.

В CASE-средство встроен мастер генерации проектных решений – инструмент, помогающий пользователю CASE-средства быстро и наглядно осуществлять выбор всех необходимых требований 3D ГИС и формировать проектные решения. Мастер генерации проектных решений отображается на экране как последовательно сменяющие друг друга диалоговые окна.

Мастер генерации проектных решений помогает осуществлять следующие процедуры:

- выбор типа 3D ГИС;
- выбор режимов отображения;
- выбор функций 3D ГИС;
- выбор объектов обстановки;
- выбор языка программирования;
- выбор платформ;
- выбор генерируемых моделей 3D ГИС;
- генерацию проектных решений.

Формально эти процедуры представлены в виде дерева шагов на рисунке 2.13.

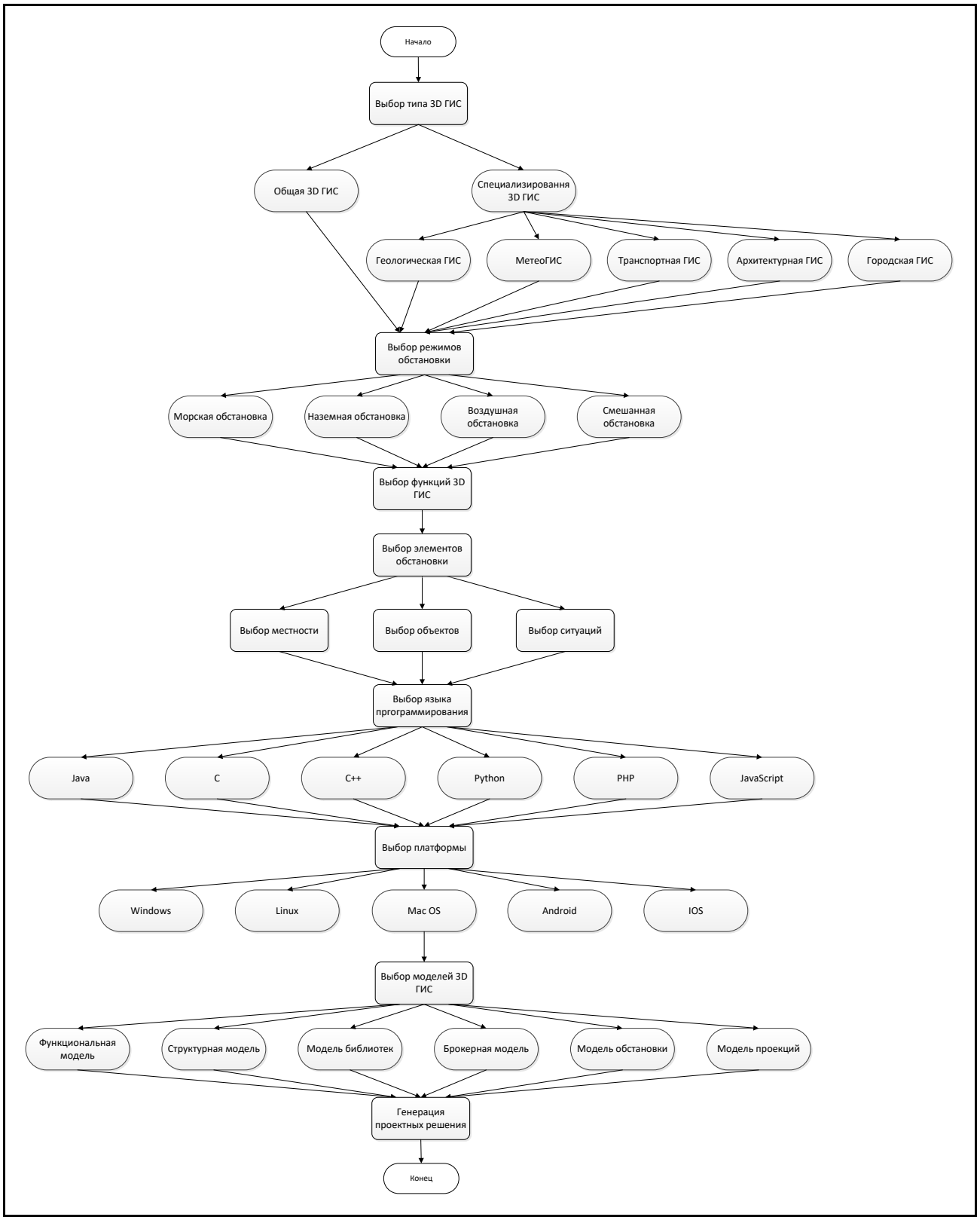

Рисунок 2.13. Дерево шагов выбора процедур мастера генерации проектных решений 3D ГИС

Мастер генерации проектных решений позволяет пользователям CASE-средства максимально быстро и качественно задать все функциональные требования проектируемой 3D

ГИС, и на их основе CASE-средство генерирует проектные решения для последующей их реализации.

Выводы

1. Разработана двухкомпонентная функционально-ресурсная модель проектирования 3D ГИС отображения ситуационной обстановки с использованием свободно распространяемых ресурсов (библиотек), включающая функциональную модель, которая используется для формирования покрытий заданной функциональности 3D ГИС на основе матрицы соответствия доступных для использования функций и реализующих их библиотек, и геометрическую модель – для оценки затрат на её разработку в виде круговой диаграммы.
2. Предложена система автоматизации проектирования 3D ГИС отображения ситуационной обстановки на основе специализированного CASE-средства проектирования 3D ГИС и системы 3D моделирования проектных решений, обеспечивающая снижение ресурсных затрат (время, финансы) и требований к разработчику, использование современных информационных технологий и разработок, генерирующая и оценивающая проектные решения для последующей их реализации.
3. Разработано специализированное CASE-средство проектирования 3D ГИС, обеспечивающее снижение ресурсных затрат (время, финансы) и требований к разработчику, а также позволяющее использовать современные информационные технологии в виде свободно-распространяемых библиотек, формировать готовые проектные решения, отвечающие требованиям на разработку 3D ГИС и делать их оценку качества для последующей реализации.
4. Предложена формула для оценки сложности проектных решений по таким характеристикам как:
   - количество собственных разработок,
   - общее количество строк кода всех используемых библиотек,
   - общее время выполнения функций библиотек,
   - среднее количество входных данных API библиотек,
   - общее количество библиотек,
   - количество адаптеров.
5. Разработан мастер генерации проектных решений, помогающий пользователю CASE-средства быстро и наглядно осуществлять выбор всех необходимых требований к проектированию 3D ГИС отображения ситуационной обстановки и формировать на их основе проектные решения.

Глава 3. Разработка алгоритма проектирования 3D ГИС отображения ситуационной обстановки на основе CASE-средства

§ 3.1. Модели построения 3D ГИС отображения ситуационной обстановки

Перечень моделей формируется в результате рассмотрения взаимодействия 3D ГИС с внешней средой и содержательной частью задания на проектирование.

CASE-средство проектирования 3D ГИС отображения ситуационной обстановки базируется на использовании следующих моделей и схем:

- схемы описания обстановки (внешняя среда);
- физической модели базы данных хранения информации об обстановке;
- обобщённой функциональной модели 3D ГИС;
- модели описания свободно распространяемых библиотек;
- ERD-модели библиотек;
- модели проекций;
- брокерной модели;
- композиционной модели.

Использование схемы описания обстановки и обобщённой функциональной модели 3D ГИС позволяет формализовать требования к проектируемой 3D ГИС. Модель описания свободно распространяемых библиотек и модель проекций обеспечивают для полученных формализованных требований необходимый подбор среди свободно распространяемых библиотек, подходящих для построения конкретной 3D ГИС. На основе брокерной модели разрабатывается схема интеграции выбранных библиотек, а на базе композиционной модели строится структура 3D ГИС.

Схема описания обстановки должна иметь универсальный способ отображения её составных элементов и удобства при проектировании 3D ГИС. Схема описания обстановки (Рисунок 3.1) включает в себя саму местность, модели текстур и рельефа, динамические и статические объекты, которые взаимодействуют между собой.

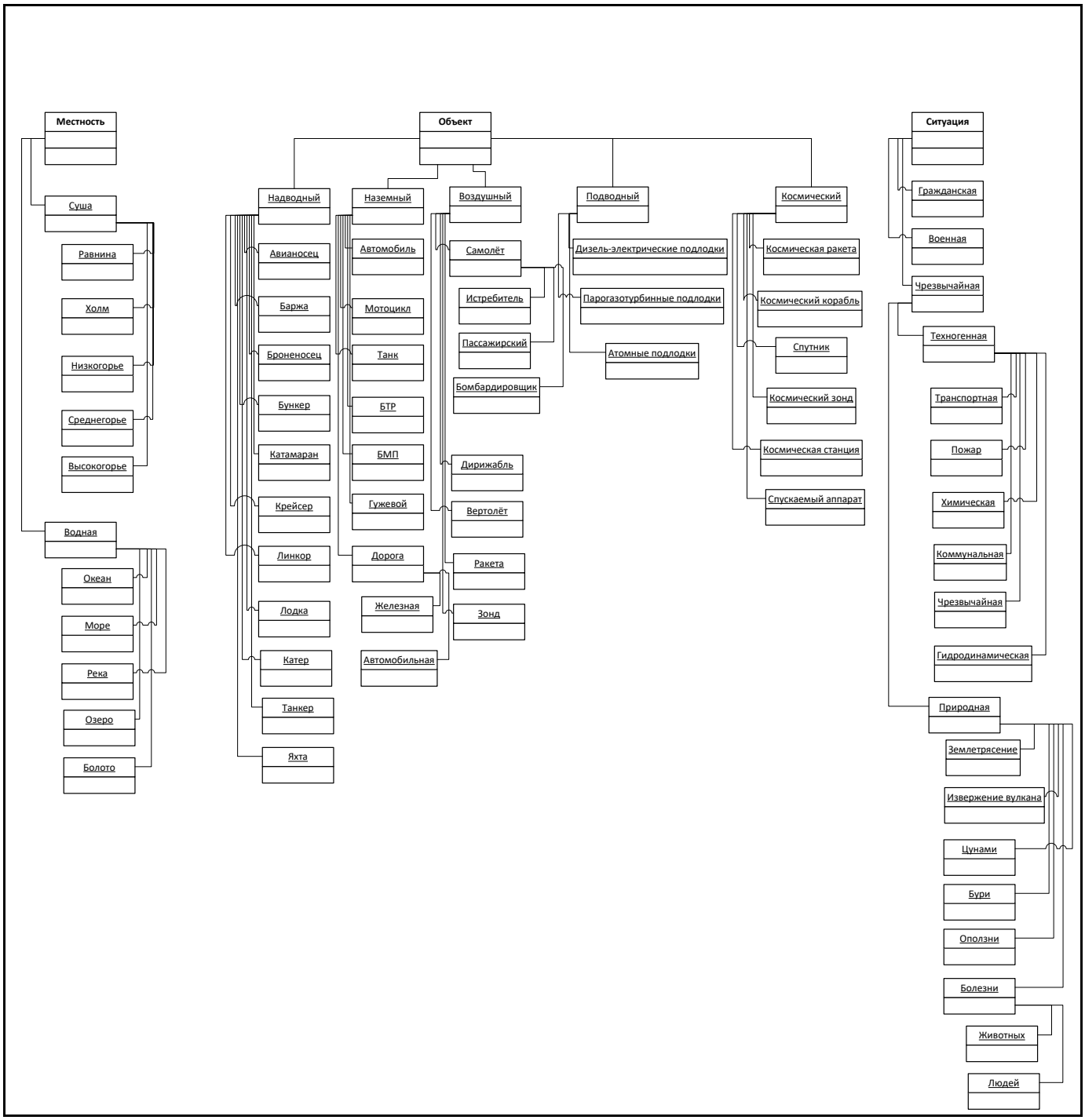

Рисунок 3.1. Обобщённая схема описания обстановки 3D ГИС

Вся обстановка состоит из местности, объектов и ситуаций. Местность делится на сушу и воду, которые, в свою очередь, уточняются конкретными особенностями местности. Объекты конкретизируются по месту применения: водные, наземные, воздушные, подводные, космические. Ситуация бывает природной и антропогенной.

Для удобства хранения схемы описания обстановки и дальнейшего проектирования конкретной 3D ГИС был разработан алгоритм кодирования выбранных для проектирования

элементов обстановки. Алгоритм позволяет закодировать последовательность выбранных элементов схемы описания обстановки в строку символов.

При проектировании схемы описания обстановки 3D ГИС рекомендуется придерживаться следующего требования. Для более точной структуризации и кодирования схемы в 4-х битовом формате у каждого узла используется не более 15 дочерних элементов (от 0000 до 1110 в бинарном виде). В случае, если в алгоритме применяется не 4-х битовый формат, а отличный от него, то дочерних элементов должно быть $n^2 - 1$, где n – количество разрядов, необходимых для кодирования одного элемента модели. Это необходимо по той причине, что число $n^2 - 1$ является информационным и не используется при кодировании элемента. Если у элемента модели обстановки имеется более $n^2 - 1$ дочерних элементов, то в данном случае можно либо увеличить разрядность алгоритма, либо в качестве элемента $n^2 - 2$ задавать обобщающий элемент, который будет хранить в себе оставшиеся незакодированными дочерние элементы.

Кодирование модели обстановки в строку.

1. Выбрать в обобщённой схеме описания обстановки те элементы, которые необходимы при дальнейшем проектировании.
2. Последовательно слева направо для каждого дочернего элемента задать порядковый номер от 0 до $n^2 - 1$.
3. Выбрать родителем корневой элемент схемы.
4. Если у текущего родителя есть выбранные дочерние элементы, то записать текущую позицию родителя в *выходную строку* и опуститься на уровень ниже.
5. Если на текущем уровне есть выбранный элемент, еще незаписанный в *выходную строку*, то записать его порядковый номер в *выходную строку*.
6. Если среди дочерних элементов сохранённого родителя есть еще один выбранный элемент, то записать его, иначе вернуться на уровень выше, указав информационный код $n^2 - 1$, обозначающий переход на уровень выше, и запомнить в качестве родителя текущий элемент.

Декодирование строки.

1. Разделить входную строку на массив элементов размером, задаваемым выбранной разрядностью.
2. Выполнить цикл по каждому элементу массива.
3. Если текущий элемент равен $n^2 - 1$, то подняться на уровень выше и перейти к шагу 2.
4. Если текущий элемент является конечным и не имеет дочерних, то добавить его в выходной массив выбранных элементов и перейти к шагу 2.

5. Если текущий элемент имеет дочерние, то перейти на уровень ниже и вернуться к шагу 2.

Пример реализации алгоритма кодирования:

```
Функция ПОЛУЧИТЬ_КОД  (ИДЕНТИФИКАТОР)
    ВЫХОД = “”
ЭЛЕМЕНТЫ = дочерние от ИДЕНТИФИКАТОР
Цикл ЭЛЕМЕНТ из ЭЛЕМЕНТЫ
    Если ЭЛЕМЕНТ имеет дочерние
        БУФЕР = ПОЛУЧИТЬ_КОД (ЭЛЕМЕНТ)
        Если БУФЕР не пустой
            ВЫХОД += БУФЕР
    В противном случае
        Если ЭЛЕМЕНТ есть в выбранных
            ВЫХОД += БУФЕР
Вернуть ВЫХОД
```

Пример реализации алгоритма декодирования:

```
Функция ПОЛУЧИТЬ_МАССИВ  (СТРОКА)
    ВЫХОДНОЙ_МАССИВ = []
ПУТЬ = “”
ЭЛЕМЕНТЫ = Разбить СТРОКА по N символов
Цикл ЭЛЕМЕНТ из ЭЛЕМЕНТЫ
    Если ЭЛЕМЕНТ равен “1111”
        ПУТЬ = часть ПУТЬ без N последних символов
    В противном случае
        ПУТЬ += ЭЛЕМЕНТ
        Если ЭЛЕМЕНТ не имеет дочерних
            Записать ПУТЬ в ВЫХОДНОЙ_МАССИВ
Вернуть ВЫХОДНОЙ_МАССИВ
```

Приведённый алгоритм был реализован на языке программирования CoffeeScript с возможностью визуализации полного дерева и дерева, построенного для конкретного режима отображения (Рисунок 3.2).

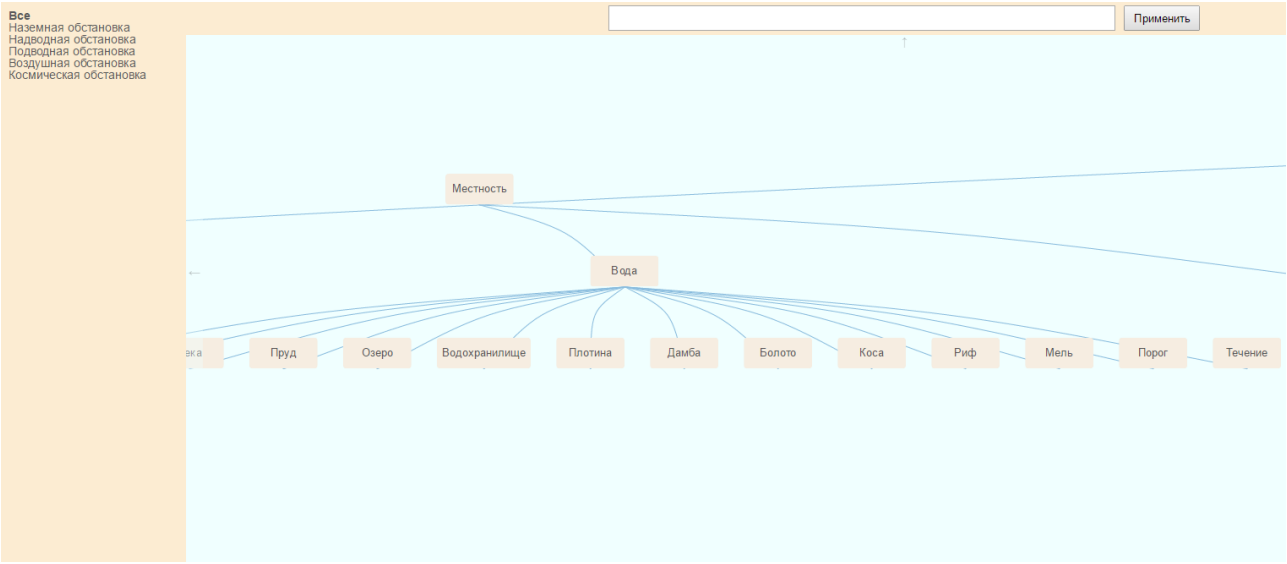

Рисунок 3.2. Интерфейс программы кодирования обстановки

В данной программе заданы основные режимы отображения обстановки (наземная, воздушная, надводная, подводная, космическая). Нажатие клавиши мыши по каждому из этих режимов перерисовывает граф с учётом только участвующих в данном режиме элементов обстановки.

Дополнительно имеется возможность выделения любого элемента обстановки на графе с возможностью получения закодированной строки выделенного подмножества и сохранения в качестве нового режима.

На основе схемы описания обстановки строится *физическая модель базы данных*, в которую загружается информация из внешней среды и оператора (Рисунок 3.3). В базе данных описания обстановки выделяются следующие таблицы: местность, объект, ситуация, маршрут, а также такая информация, как растровые карты и составные элементы векторных карт. В каждой таблице имеется уникальный идентификатор, позволяющий однозначно определить тот или иной элемент таблицы.

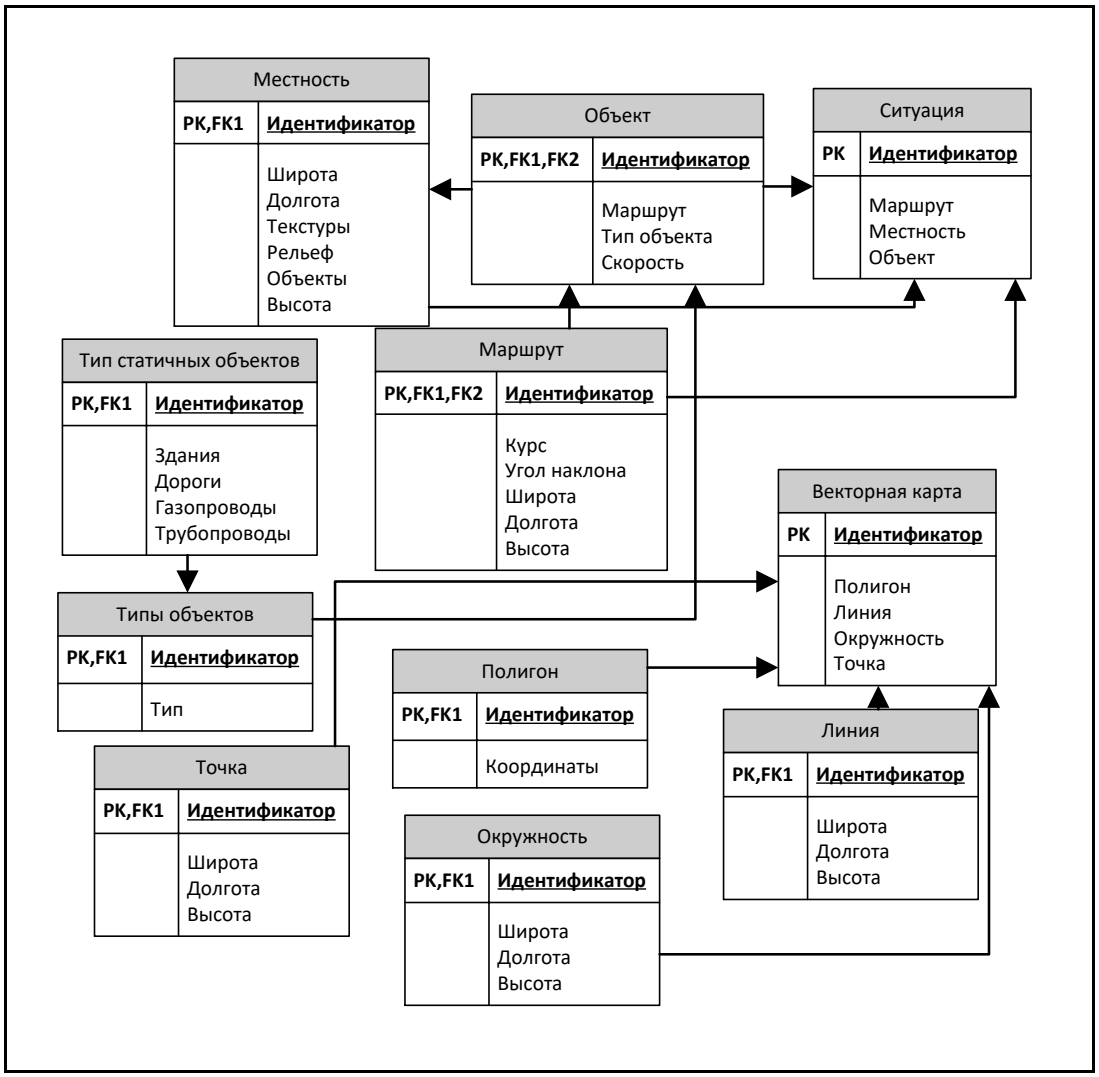

Рисунок 3.3. Физическая модель базы данных хранения информации об обстановке

Обобщённая функциональная модель.

Требования, формируемые в процессе анализа задачи, интерпретированы множеством алгоритмов, описанных формальным образом. Организация функционирования 3D ГИС представлена на рисунке 3.4, на котором отображаются процессы и функции, применяемые как для проектирования, так и для реализации 3D ГИС в реальных условиях. Здесь под понятием «работа» подразумеваются процессы, которые необходимо выполнить как на стадиях предварительного и последующего проектирования (работа с графикой: построение 3D-модели Земли, отображение векторных и растровых форматов, отображение 3D-моделей объектов, отображение рельефа местности), так и на стадии непосредственного создания системы (программирование: разработка компонентов ядра; работа с базами данных: PostgreSQL, SQLite, LevelDB). Для обеспечения работы с файловой системой, работы с Интернет и работы с геоданными осуществляется обоснованная выборка необходимых свободно распространяемых библиотек с требуемыми функциями.

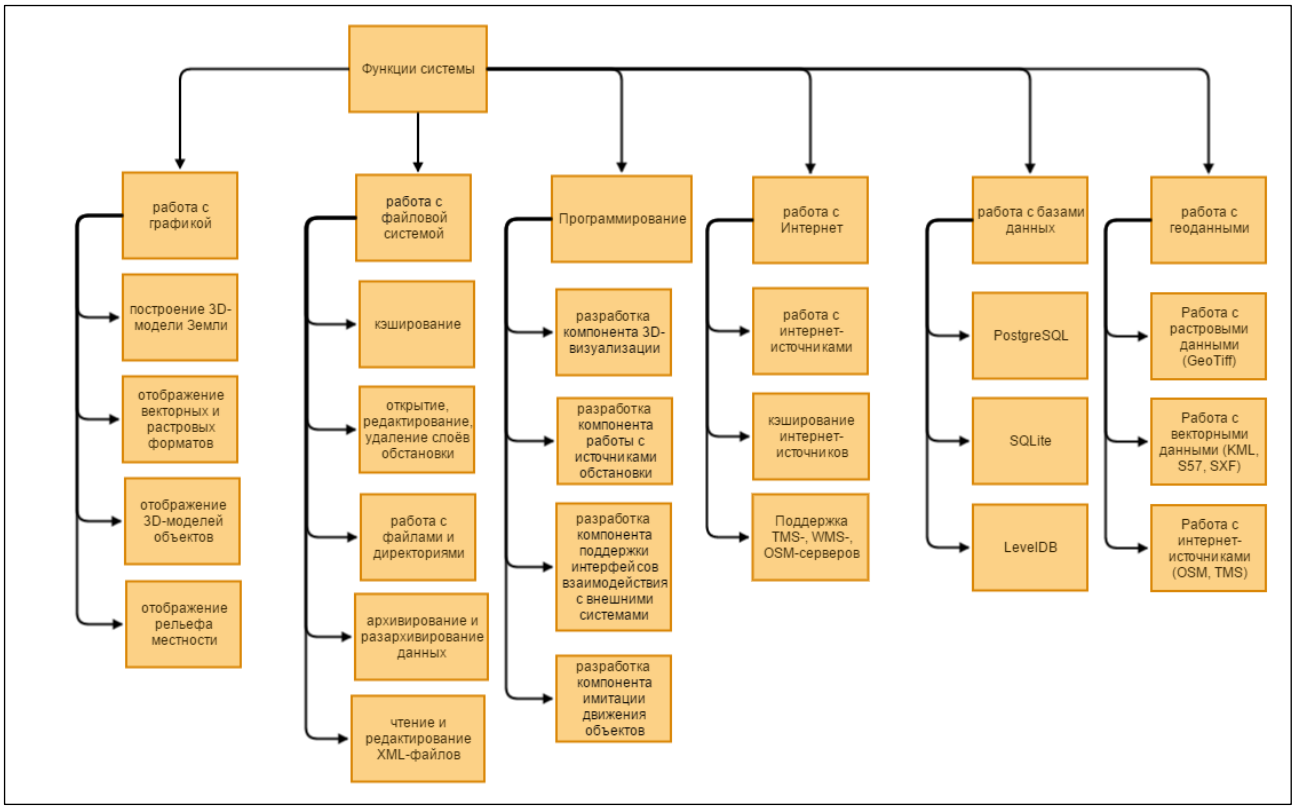

Рисунок 3.4.Функциональная модель 3D ГИС отображения ситуационной обстановки

Функции, обеспечивающие работу с графикой, визуализируют 3D-глобус, обстановку на карте, 3D-объекты, текстуры и рельеф. Первоначальная векторная пространственная модель описывает при помощи уравнений положение объектов на 3D-сцене. Далее эта модель преобразовывается в растровое изображение и при помощи пикселей отображается на экране.

Функции для работы с Интернет обеспечивают взаимодействие с Интернет-источниками обстановки, векторными (WMS, WFS) и растровыми форматами данных.

В базу данных заносится информация о текстурах, высотах, глубинах и 3D-объектах, используемых в 3D ГИС в различных СУБД.

Функции, обеспечивающие работу с геоданными, дают возможность создания, чтения и редактирования растровых и векторных данных. Все геоданные хранятся в базе данных, что позволяет обеспечить быстрый поиск и доступ к ним.

Для обеспечения выполнения перечисленных выше работ необходимо иметь полный набор библиотек, который мог бы решать все поставленные задачи. При этом все ресурсы в наборе должны быть совместимы с языком программирования, на котором создаётся 3D ГИС. Однако на практике такой набор найти не представляется возможным.

Модель описания свободно распространяемых библиотек.

Построение ядра 3D ГИС отображения ситуационной обстановки опирается на использование свободно распространяемых библиотек (3D-движков), которых существует достаточно большое количество и каждая из которых обладает своими функциональными возможностями. Необходимо сформировать базу библиотек и задать им необходимые функциональные характеристики с возможностью оценивания их применимости в конкретных задачах. Отсюда возникает потребность их оперативного сбора, упорядоченного хранения и быстрого поиска для использования при проектировании 3D ГИС. База данных библиотек должна быть пополняемой, т. к. их количество неуклонно увеличивается и появляются библиотеки с новыми функциональным возможностями. Необходимо учитывать совместимость технологий программирования, на которых построены библиотеки. В случае наличия несовместимой библиотеки и отсутствия её подходящего аналога требуется разработать собственный вариант, реализующий заданный алгоритм.

Один из возможных наборов библиотек, который может быть использован для проектирования 3D ГИС отображения ситуационной обстановки, состоит из таких библиотек, как OpenSceneGraph, osgEarth, GDAL, CURL, GEOS, MiniZip, OGR, Qt, CMake, OpenGL, LevelDB, SQLite.

На основе анализа свободно распространяемых библиотек и их взаимодействия между собой разработана *ERD модель библиотек* (рис. 3.5), которая позволяет накапливать информацию обо всех проанализированных библиотеках и их функциях. У библиотек выделяются следующие атрибуты, которые хранятся в таблице «Библиотеки»: название библиотеки, язык разработки, операционная система (кроссплатформенность). В таблице «Связи» размещены пары библиотек, у которых реализовано взаимодействие между собой средствами API. Для каждой библиотеки определена иерархия классов функций, и все функции сохраняются в базе данных с указанием следующих атрибутов: название, класс, входные данные, выходные данные.

Рассмотренная структура хранения библиотек и их функций позволяет классифицировать все свободно распространяемые библиотеки и обеспечивает их удобный поиск по различным критериям при использовании CASE-средства проектирования 3D ГИС отображения ситуационной обстановки.

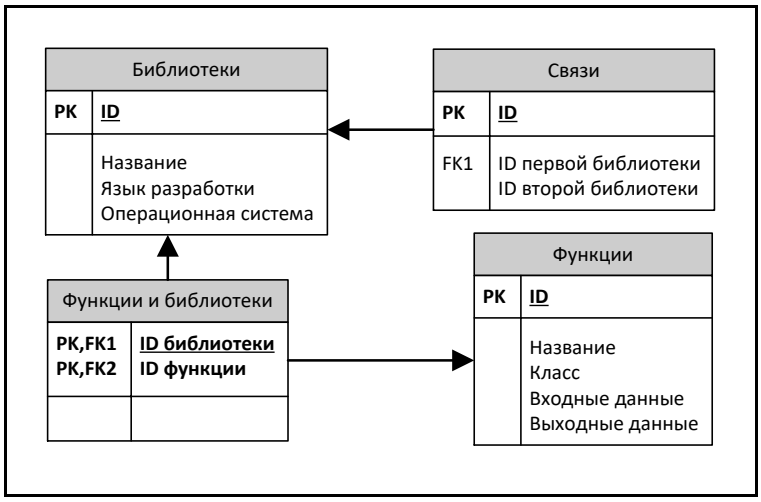

Рисунок 3.5. ERD-модель библиотек

Модель проекций представляет собой отображение функциональных возможностей библиотек на компоненты ядра. Связи между библиотеками и компонентами ядра отражают его функциональную нагрузку. Функции отдельного компонента ядра могут быть обеспечены несколькими библиотеками. Некоторые библиотеки обладают большим количеством функций и могут охватить несколько требований того или иного компонента ядра. Построение модели проекций завершается при возможности полным покрытием функций компонентов ядра библиотеками, в противном случае, недостающие функции покрываются собственными разработками.

Для реализации компонента визуализации системы 3D моделирования проектных решений из рассмотренных библиотек выделяются такие, которые обеспечивают отображение трехмерной графики и 3D-модели Земли: OpenGL, osgEarth. Библиотека OpenGL представляет собой программный интерфейс для создания приложений, использующих двумерную и трехмерную графику. Библиотека osgEarth является набором инструментов для визуализации земной поверхности в трехмерном виде.

При разработке компонента работы с источниками обстановки используются библиотеки взаимодействия с геоданными: OGR, GDAL. Библиотека GDAL обеспечивает чтение и редактирование карт растровых форматов, а расширение OGR – карт векторных форматов.

Для компонента поддержки интерфейсов взаимодействия с внешними источниками системы 3D моделирования проектных решений требуются библиотеки, обеспечивающие работу с файловой системой и ресурсами Интернет. Библиотека CURL позволяет взаимодействовать с большим количеством серверов по множеству различных протоколов с синтаксисом URL.

Компонент имитации движения объектов системы 3D моделирования проектных решений реализуется при помощи функций, позволяющих изменять координаты 3D-объектов, с использованием библиотеки OpenSceneGraph.

Отдельно можно выделить библиотеку Qt, которая является кроссплатформенным инструментарием разработки ПО на языке программирования C++ и реализует основную функциональность всех компонентов ядра 3D ГИС.

Ниже представлен один из возможных наборов библиотек, который может быть использован для проектирования 3D ГИС отображения ситуационной обстановки (Таблица 3.1).

Таблица 3.1. Набор свободно распространяемых библиотек для проектирования 3D ГИС

Библиотека	Описание	Выполняемые функции
OpenSceneGraph	кроссплатформенная свободно распространяемая библиотека для разработки высокопроизводительных 3D-приложений	• Работа с графикой
osgEarth	масштабируемый набор инструментов для визуализации земной поверхности на основе OpenSceneGraph.	• Работа с графикой • Программирование движка
GDAL	библиотека для работы с растровыми географическими форматами файлов данных	• Работа с графикой • Взаимодействие с файловой системой • Открытие и чтение геоданных
CURL	свободная, кроссплатформенная библиотека, позволяющая взаимодействовать с множеством различных серверов по множеству различных протоколов с синтаксисом URL	• Взаимодействие с Интернет источниками
GEOS	библиотека для работы с геометрическими фигурами	• Открытие и чтение геоданных
Minizip	библиотека для работы с ZIP/GZIP файлами	• Взаимодействие с файловой системой
OGR	Свободно распространяемая библиотека для работы с векторными данными	• Работа с графикой • Взаимодействие с файловой системой • Открытие и чтение геоданных
Qt	кроссплатформенный инструментарий разработки ПО на языке программирования C++.	• Взаимодействие с файловой системой • Программирование движка
CMake	это кроссплатформенная библиотека автоматизации сборки программного обеспечения из исходного кода.	• Программирование движка
OpenGL	платформонезависимый программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику.	• Работа с графикой
LevelDB	Библиотека для работы с базами данных LevelDB	• Работа с базой данных
SQLite	Библиотека для работы с базами данных SQLite	• Работа с базой данных

На рисунке 3.6 представлено отображение в виде проекций функциональных возможностей библиотек (Таблица 3.1), применимых для реализации компонентов ядра 3D ГИС.

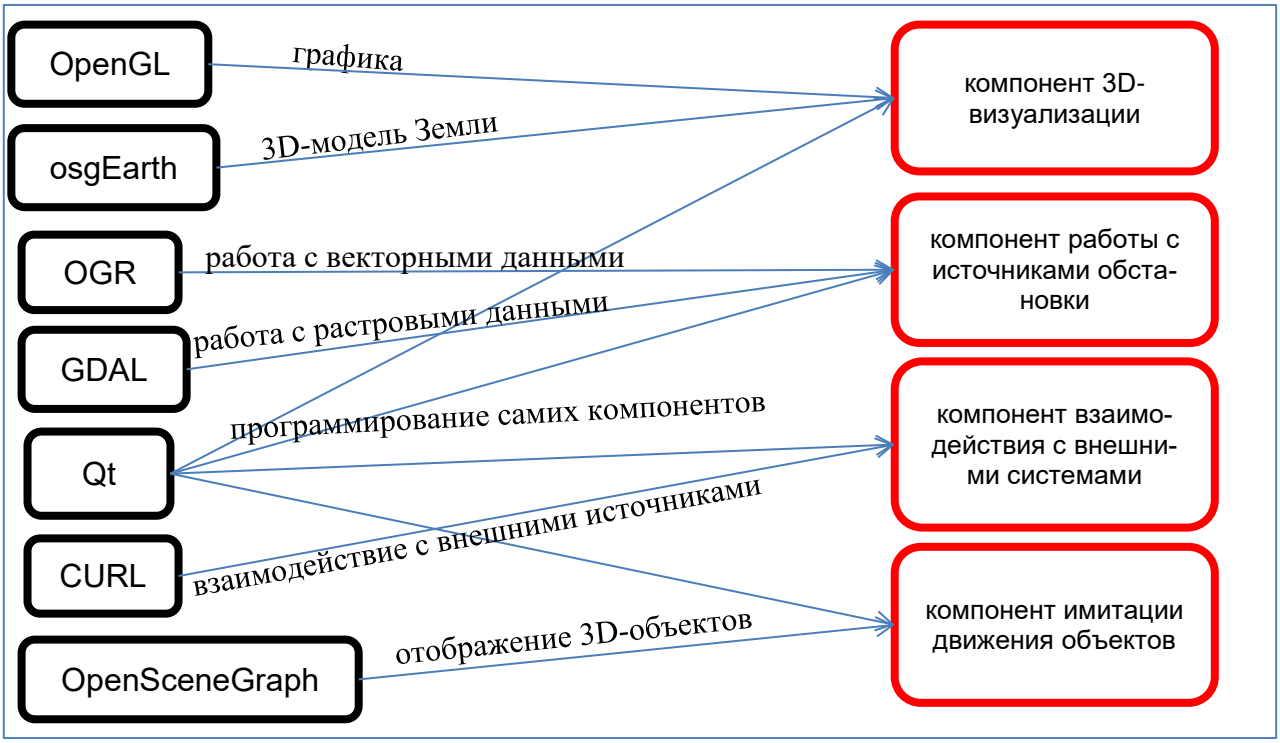

Рисунок 3.6. Проекции свободно распространяемых библиотек на компоненты ядра

Среди рассмотренных свободно распространяемых библиотек работу с геоданными обеспечивают две библиотеки: GDAL и OGR. В библиотеке GDAL реализована возможность чтения, записи и редактирования геоданных наиболее широко используемых растровых форматов: GeoTiff, JPEG, MAP, WMS и др. Библиотека OGR является дополнительным инструментарием к библиотеке GDAL, который позволяет работать с векторными форматами карт: KML, OSM, S57, SXF, WFS и др.

Брокерная модель.

Основная трудность работы с библиотеками состоит в том, что необходимо устанавливать их взаимодействие между собой и с собственными разработками. Другими словами, возникает необходимость интеграции библиотек и собственных разработок, которая опирается на три известных подхода: использование брокера, использование шины и использование Web-сервисов. [50]

Наиболее подходящей моделью для создания 3D ГИС является брокер, представляющий собой центральный узел, к которому обращаются интегрируемые подсистемы и библиотеки и на который возлагаются функции адресации и трансформации передаваемых данных.

На рисунке 3.7 представлена «брокерная модель» взаимодействия 3D ГИС. На связях «брокер-библиотека» выделены адаптеры, которые обеспечивают двунаправленную передачу данных согласованных форматов между библиотекой и брокером.

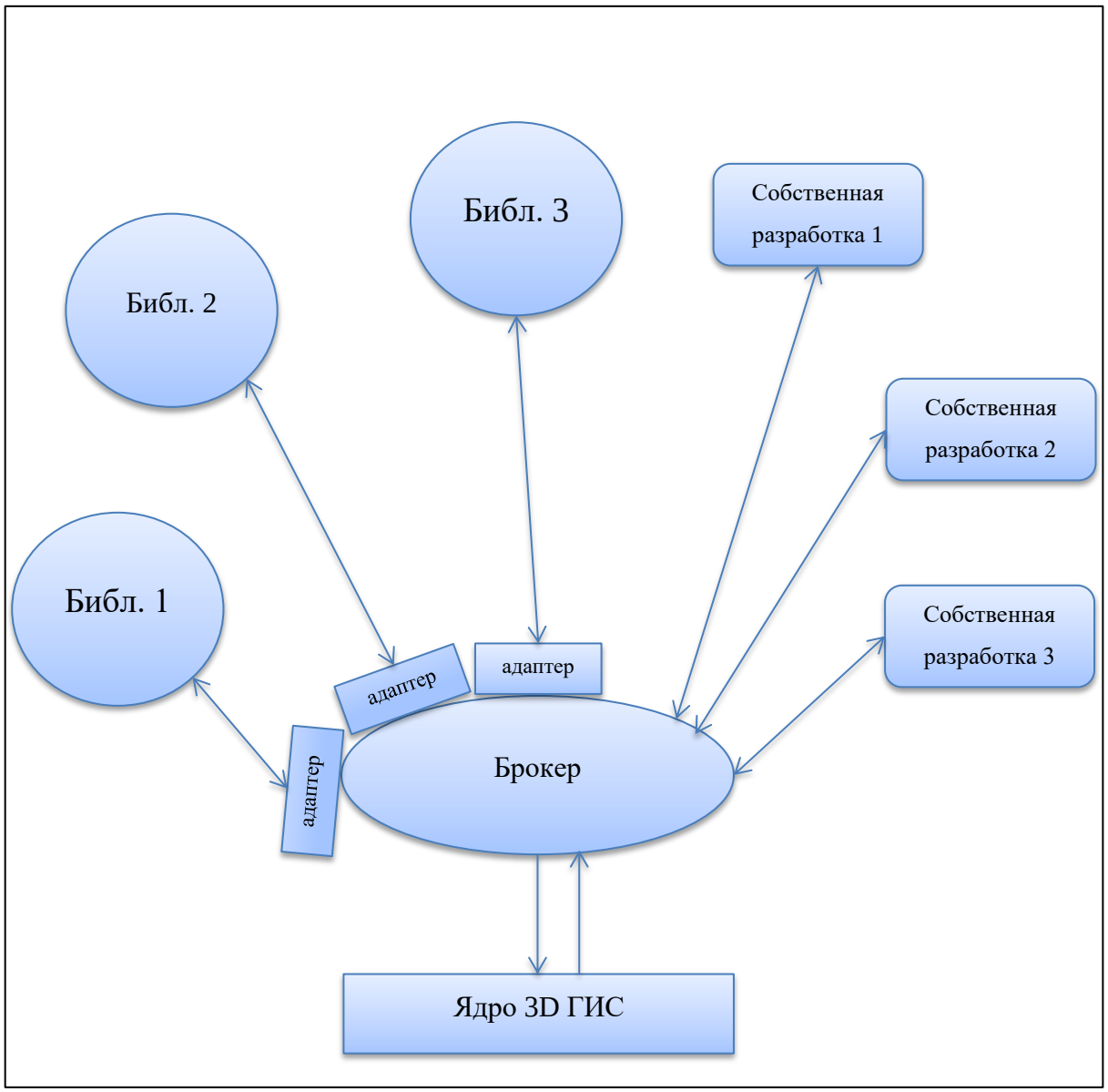

Рисунок 3.7. Брокерная модель 3D ГИС

Брокер реализуется в виде модуля, который обеспечивает подключение через адаптер выбранной библиотеки или собственной программной разработки.

Количество входов у брокера равно сумме количества библиотек и количества собственных разработок, для которых не требуется адаптеры.

Композиционную модель 3D ГИС целесообразнее рассмотреть на примере построения 3D ГИС отображения ситуационной обстановки.

Композиция включает в себя две составляющие:

1) общую функциональность 3D ГИС, получаемую из анализа задачи;
2) общую модель ядра 3D ГИС.

Функциональность подвергается разбиению на множество составных функций и часть этих функций может быть реализована с помощью библиотек, а функции, для которых нет библиотек, реализуются собственными разработками.

Наращиваемость ядра 3D ГИС осуществляется на базе реализаций составных функций с помощью библиотек и собственных разработок. В результате формируется расширенное и наполненное необходимыми функциями модифицированное ядро, представляющее собой разрабатываемую 3D ГИС.

На рисунке 3.8 представлен пример композиционной модели модифицированного ядра 3D ГИС отображения ситуационной обстановки.

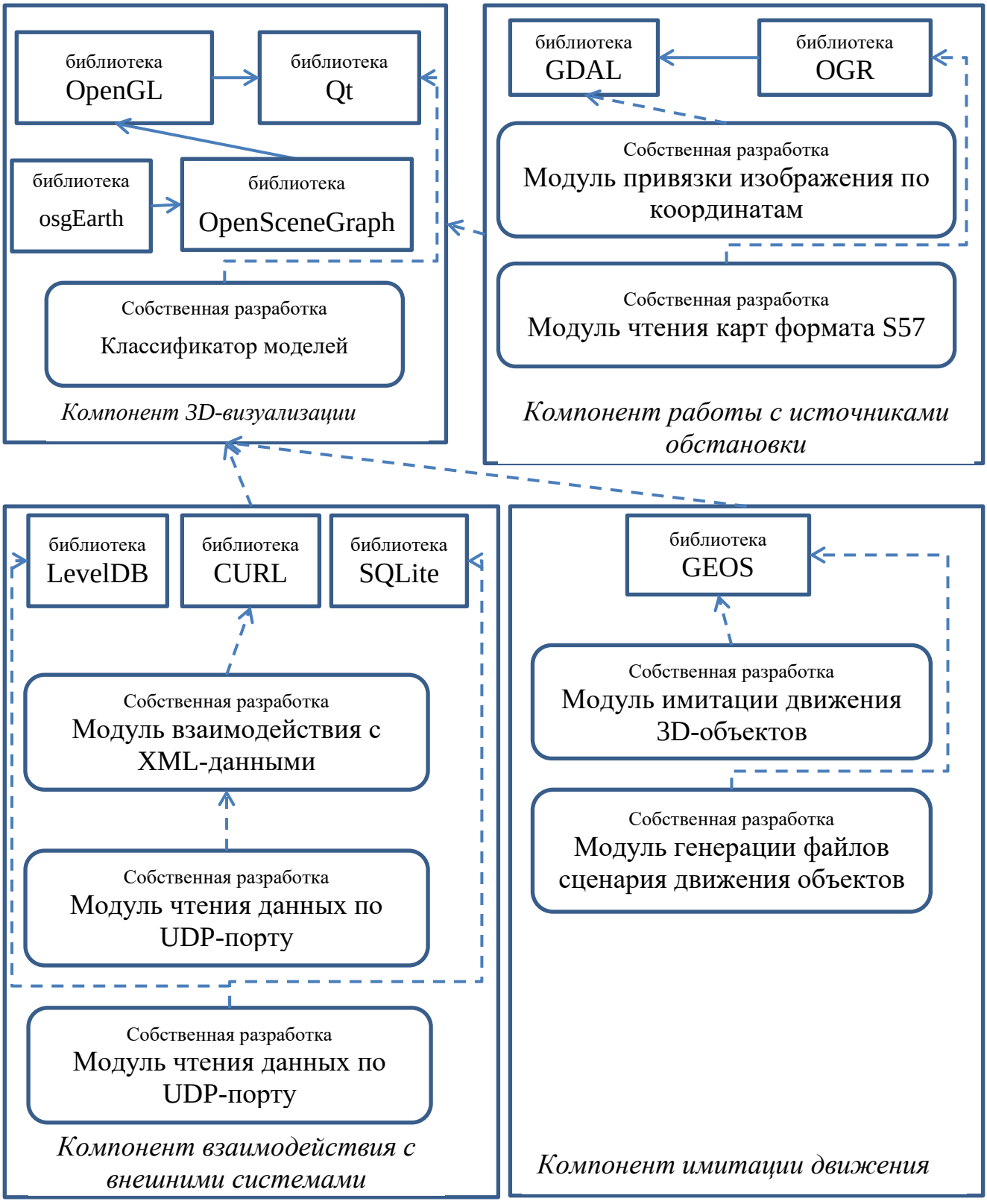

Рисунок 3.8. Пример композиционной модели ядра 3D ГИС

Описанные 7 моделей являются необходимой модельной базой проектирования 3D ГИС, которая позволяет создавать программную реализацию базовых компонентов ядра 3D

ГИС и после этого переходить к моделированию и оценке качества выполняемых требований согласно поставленной задаче.

§ 3.2. Алгоритмы проектирования 3D ГИС отображения ситуационной обстановки на базе CASE-средства

При подготовке к программной реализации CASE-средства необходимо построить его общий алгоритм функционирования, который описывает действия проектировщика при использовании CASE-средства.

Перед началом разработки 3D ГИС заказчиком выдаются задания на геоинформационную систему, которые преобразуются проектировщиком в поддерживаемый CASE-средством вид (функциональные требования). Также необходимыми атрибутами функционирования CASE-средства являются требования на поставленные задания и условия, при которых функционирует система. Структура задания на проектирование 3D ГИС представлена на рисунке 3.9.

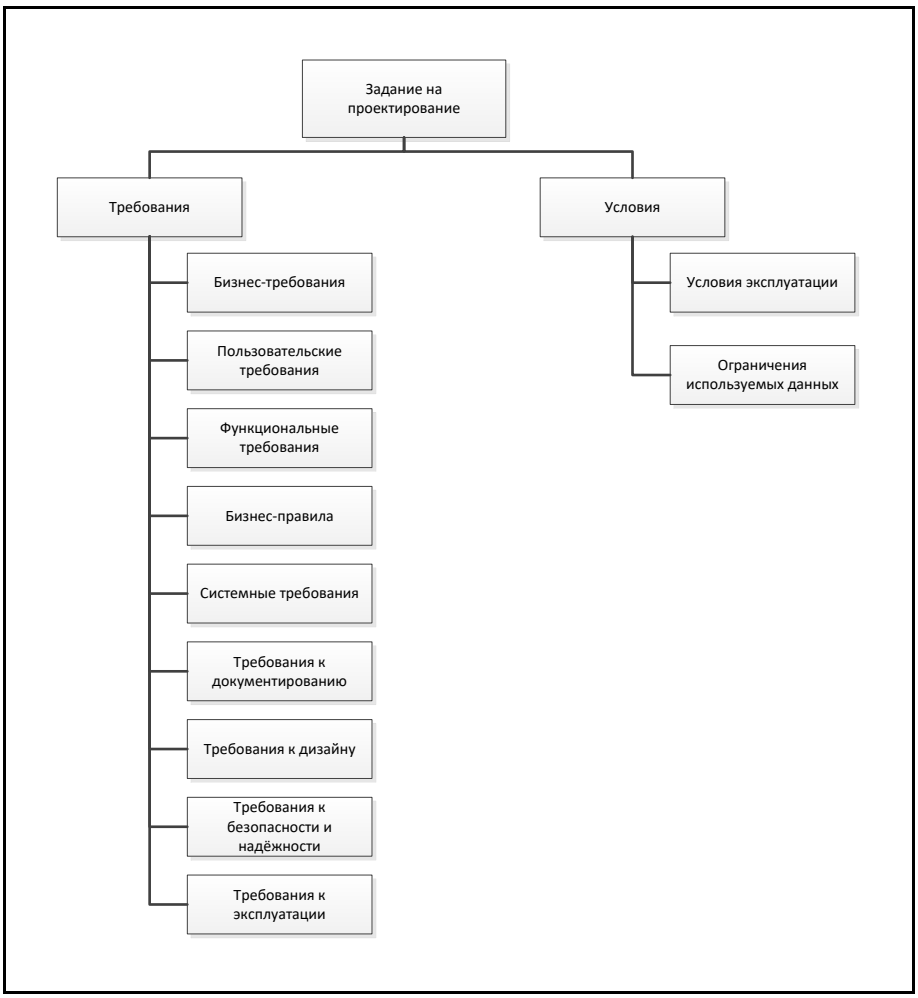

Рисунок 3.9. Структура задания на проектирование 3D ГИС

CASE-средство выполняет подключение к базе данных описания обстановки и базе данных описания инструментов реализации, загрузку обобщённой функциональной модели 3D ГИС и обобщённой модели обстановки. В ходе работы с CASE-средством проектировщик выбирает согласно требованиям необходимые в 3D ГИС функции и объекты обстановки.

Далее CASE-средство генерирует проектные решения согласно выбранным данным. Проектировщик вручную или автоматически выбирает наиболее подходящее проектное решение и в дальнейшем по моделям и диаграммам выбранного проектного решения программисты выполняют реализацию 3D ГИС (Рисунок 3.10).

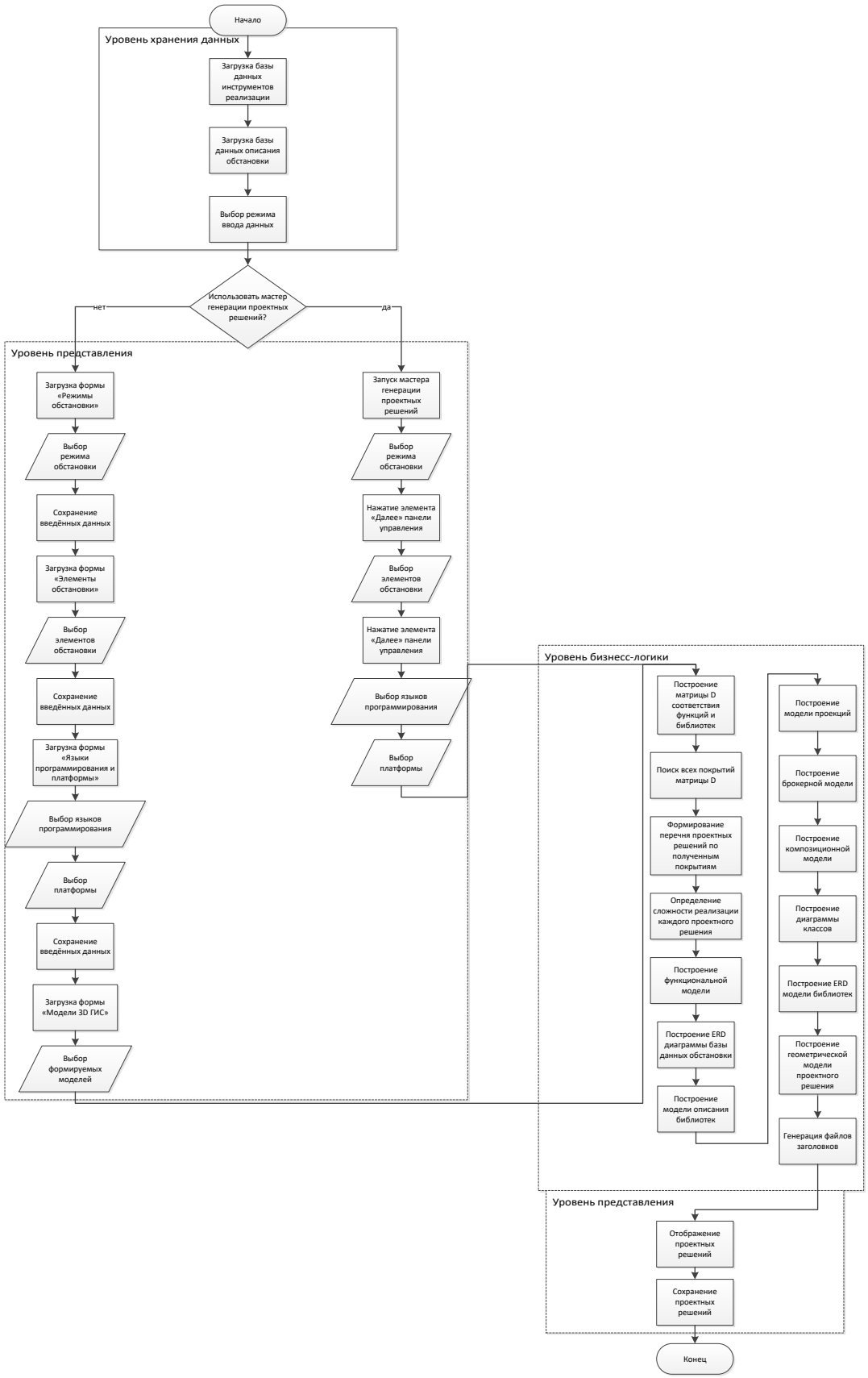

Рисунок 3.10. Алгоритм проектирования 3D ГИС отображения ситуационной обстановки на базе CASE-средства

Из общего алгоритма функционирования CASE-средства формируется база алгоритмов проектирования.

Алгоритмы проектирования реализуют каждый компонент CASE-средства:

- алгоритм работы с базой данных описания обстановки;
- алгоритм работы с базой данных описания инструментов реализации;
- алгоритм функционирования средства управления проектом;
- алгоритм функционирования средства конфигурационного управления;
- алгоритм тестирования;
- алгоритм создания документации;
- алгоритм построения обобщённой функциональной модели 3D ГИС;
- алгоритм построения модели описания обстановки;
- алгоритм построения модели библиотек;
- алгоритм построения модели проекций;
- алгоритм построения композиционной модели 3D ГИС;
- алгоритм генерации файлов заголовков.

§ 3.3. Программная реализация CASE-средства проектирования 3D ГИС отображения ситуационной обстановки

3.3.1. Выбор среды разработки CASE-средства проектирования 3D ГИС отображения ситуационной обстановки

При разработке CASE-средства был выбран язык программирования Python с использованием графического интерфейса Qt. Преимуществами Python являются: простота в изучении, удобочитаемый синтаксис, возможность объектно-ориентированного программирования (ООП), большое количество дополнительных библиотек. Программы, разработанные на языке Python, функционируют на большинстве современных операционных систем, в том числе Windows, Mac OS, Linux, Android.

Библиотека PyQt, входящая в комплект установки языка Python, позволяет создавать приложения, основным интерфейсным элементом которых является форма – стандартный элемент управления «окно» с уже определенным стилем. Разрабатываемое приложение проектируется как совокупность таких форм, на которых располагаются остальные элементы управления.

3.3.2. Описание интерфейса CASE-средства проектирования 3D ГИС отображения ситуационной обстановки

При инициализации CASE-средства отображается форма (Рисунок 3.11), где сосредоточены основные элементы управления. Через эту форму осуществляется доступ ко всем функциям CASE-средства.

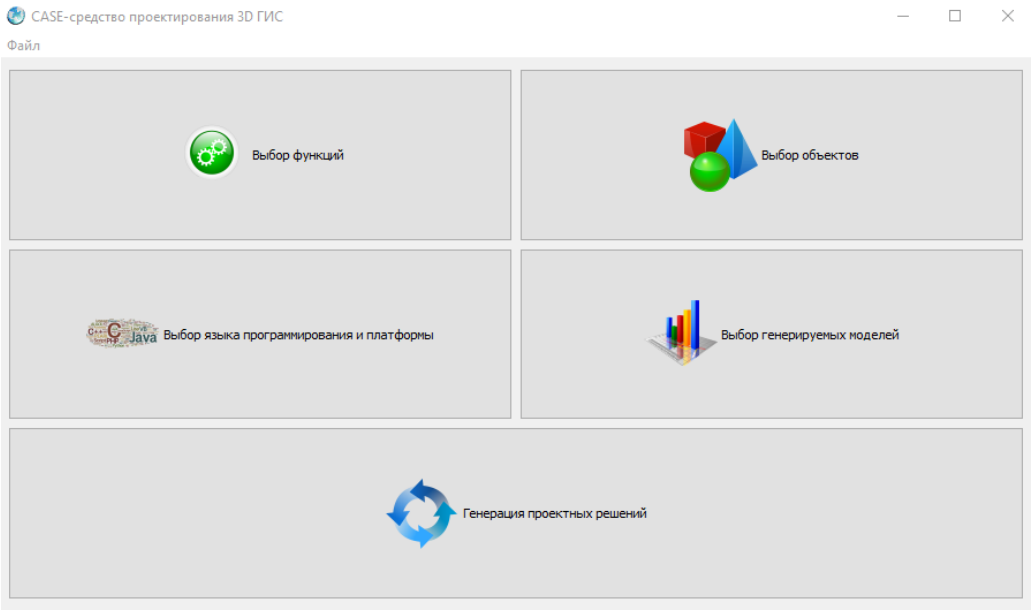

Рисунок 3.11. Панель управления CASE-средством 3D ГИС

Первый пункт панели управления «Выбор функций» позволяет пользователю CASE-средства задавать функции, которые требуются от проектируемой 3D ГИС. Форма «Выбор функций 3D ГИС» представлена на рисунке 3.12. Все функции системы группируются по типам для более удобного отображения. По умолчанию выбраны все доступные функции, но пользователь CASE-средства может убрать ненужные. При необходимости пользователь имеет возможность добавления собственных функций.

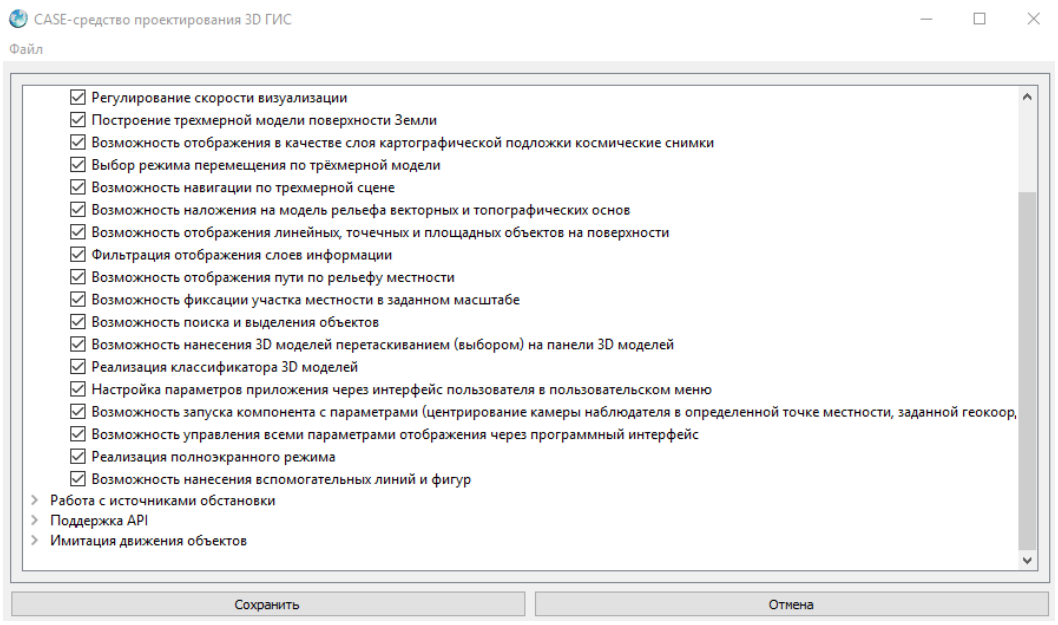

Рисунок 3.12. Выбор функций 3D ГИС

Следующий пункт панели управления «Выбор объектов» позволяет пользователю CASE-средства выбрать объекты обстановки, местность и ситуации, которые должна отображать проектируемая 3D ГИС. В форме «Выбор объектов обстановки 3D ГИС» (Рисунок 3.13) все объекты группируются по трём типам: местность, объекты и ситуации, а они, в свою очередь, иерархически содержат конкретные объекты.

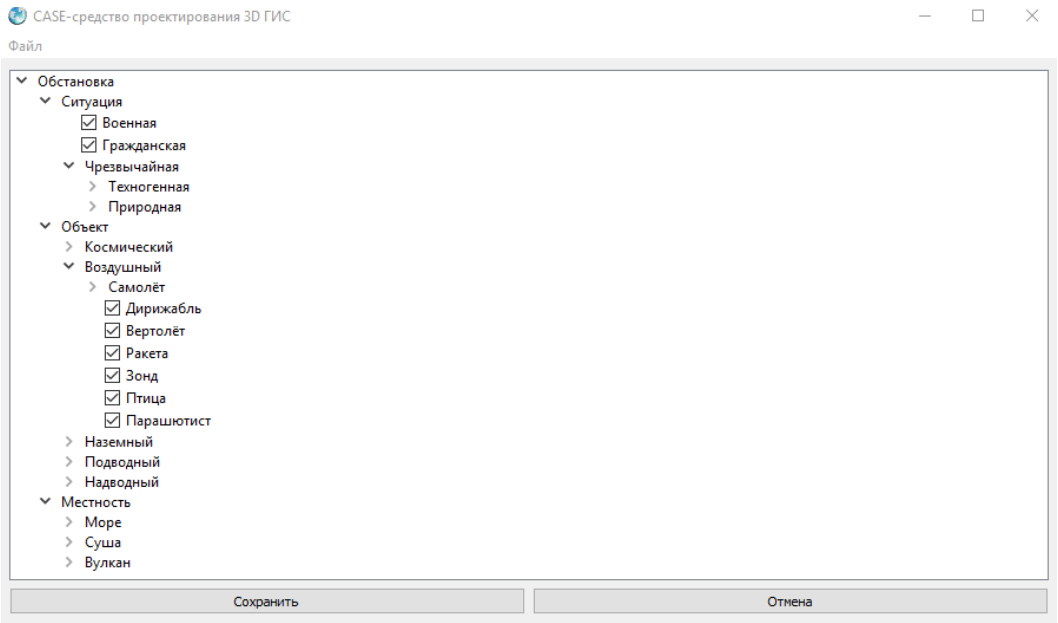

Рисунок 3.13. Выбор объектов обстановки 3D ГИС

Пункт панели управления «Выбор языка программирования и платформы» позволяет проектировщику задать те языки программирования, по которым CASE-средство выбирает

из базы необходимые для разработки 3D ГИС библиотеки, а также указать основный язык программирования, на котором будет реализована 3D ГИС. Дополнительной возможностью является выбор платформы 3D ГИС, на которой будет функционировать 3D ГИС. Форма «Выбор языка программирования и платформы» представлена на рисунке 3.14.

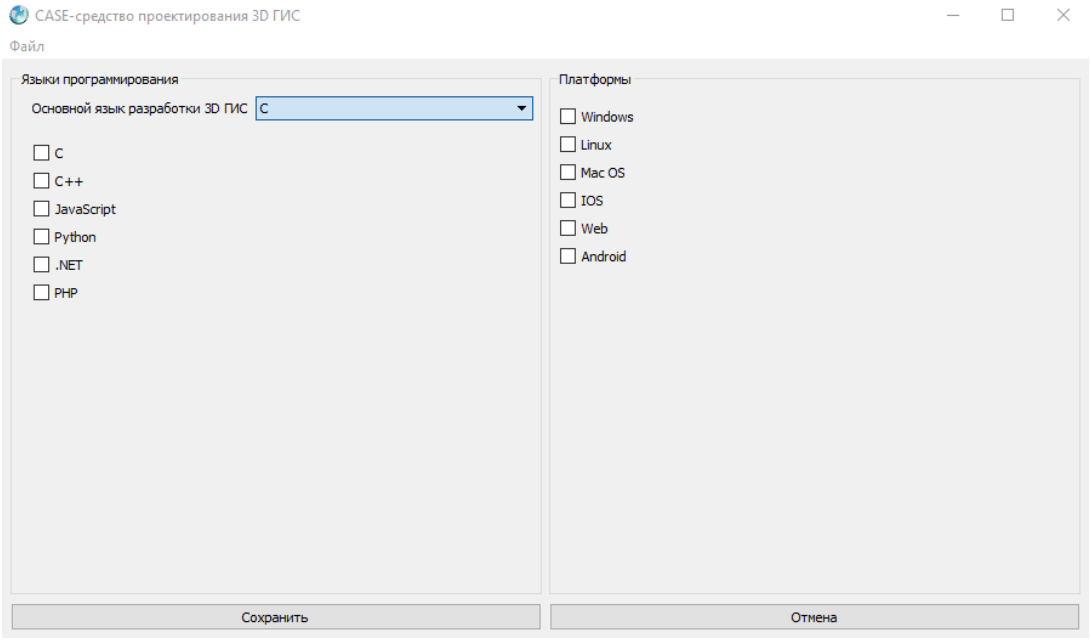

Рисунок 3.14. Выбор языка программирования и платформы

После выбора проектировщиком функций 3D ГИС, объектов обстановки, поддерживаемых языков программирования и платформ становится доступной возможность генерации проектных решений. Для этого необходимо выбрать пункт «Генерация проектных решений» на панели управления CASE-средства. После этого CASE-средство автоматически определяет все библиотеки, реализующие заданные функции, и генерирует все возможные комбинации с ними. Форма «Проектные решения 3D ГИС» представлена на рисунке 3.15. В левой части формы отображаются все полученные проектные решения. Для каждого проектного решения CASE-средство расчитывает приблизительную сложность (§ 2.5).

При выборе проектного решения в правой части формы «Проектные решения 3D ГИС» отображается информация по данному решению и оценка сложности в реализации данного проектного решения.

Для каждого проектного решения есть возможность просмотра моделей 3D ГИС при нажатии на соответствующий пункт.

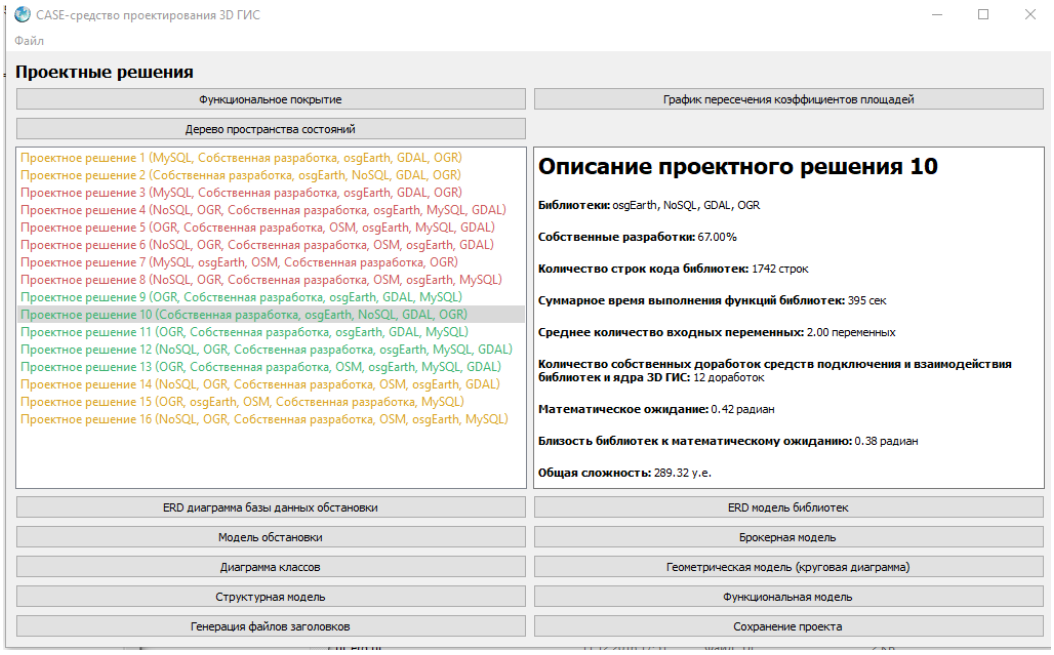

Рисунок 3.15. Проектные решения 3D ГИС

Примеры ERD модели диаграммы базы данных обстановки, ERD модели библиотек, модели обстановки 3D ГИС, брокерной модели, диаграммы классов библиотек, структурной и функциональной моделей, а также сгенерированных файлов заголовков представлены на рисунках 3.16-3.23.

На рисунке 3.16 представлена ERD диаграмма базы данных обстановки, которая отображает форму хранения всех объектов обстановки в СУБД.

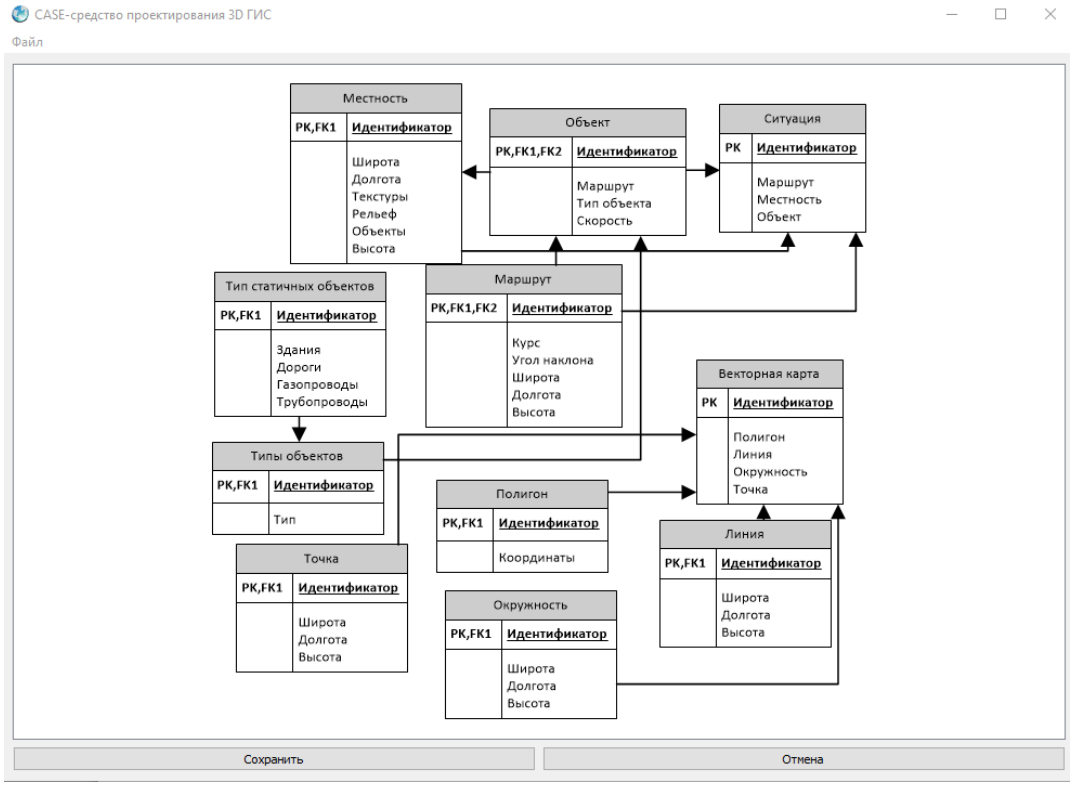

Рисунок 3.16. ERD диаграмма базы данных обстановки

На рисунке 3.17 представлена ERD диаграмма библиотек, отображающая возможность удобного хранения библиотек в базе данных и быстрого поиска по ним.

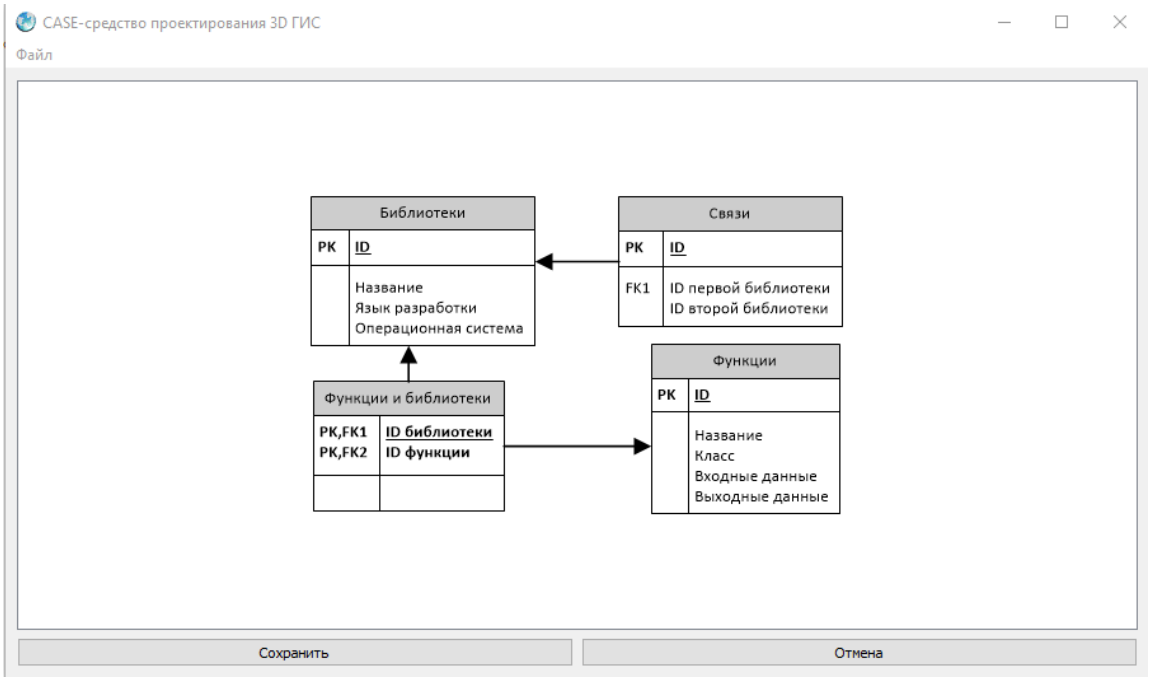

Рисунок 3.17. ERD диаграмма библиотек

На рисунке 3.18 представлена схема описания обстановки 3D ГИС конкретного проектного решения, включающая только те объекты и ситуации, которые необходимы в данном конкретном проектном решении исходя из требований, указанных проектировщиком.

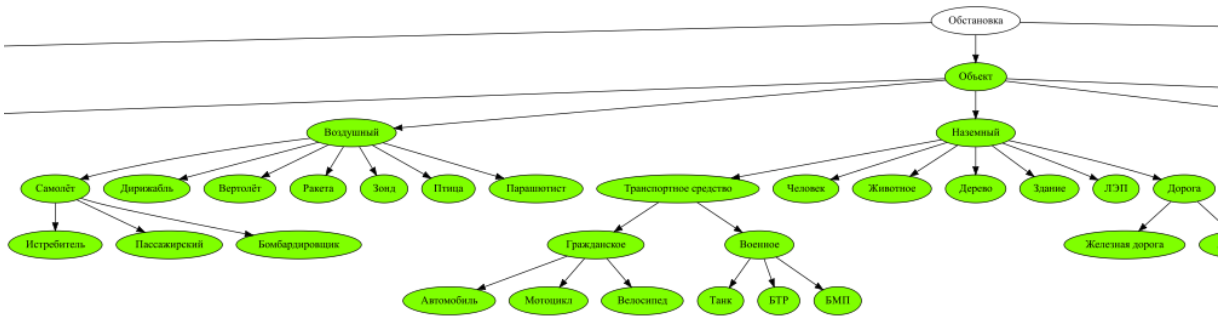

Рисунок 3.18. Схема описания обстановки 3D ГИС

На рисунке 3.19 представлена брокерная модель 3D ГИС, описывающая используемые в проектном решении библиотеки и возможности их подключения к ядру 3D ГИС.

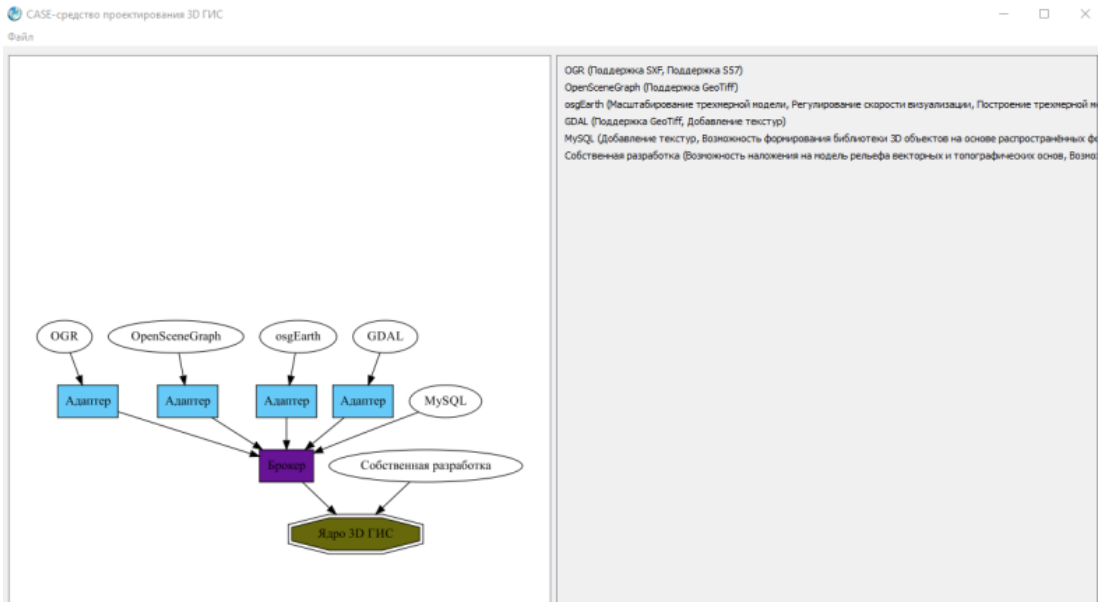

Рисунок 3.19. Брокерная модель 3D ГИС

На рисунке 3.20 представлена диаграмма классов библиотек, на основе которой в дальнейшем формируются исходные коды файлов заголовков.

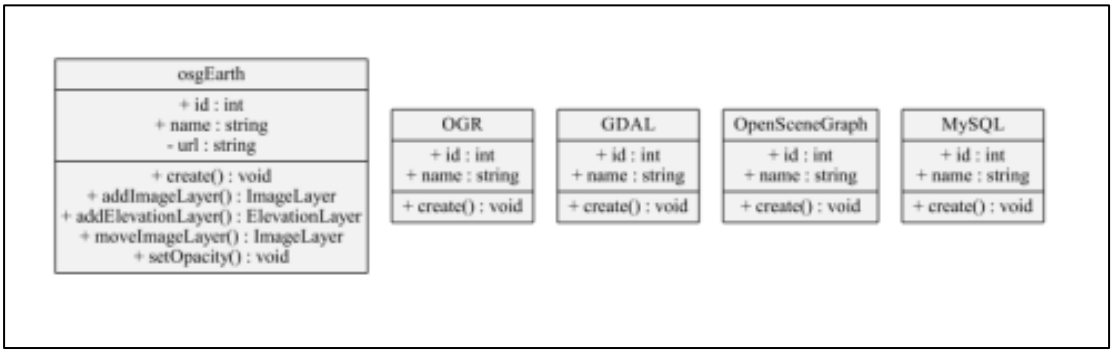

Рисунок 3.20. Диаграмма классов библиотек

На рисунке 3.21 представлена структурная модель 3D ГИС, включающая ядро 3D ГИС с компонентами, входящими в него, а также базы текстур, высот и объектов 3D ГИС.

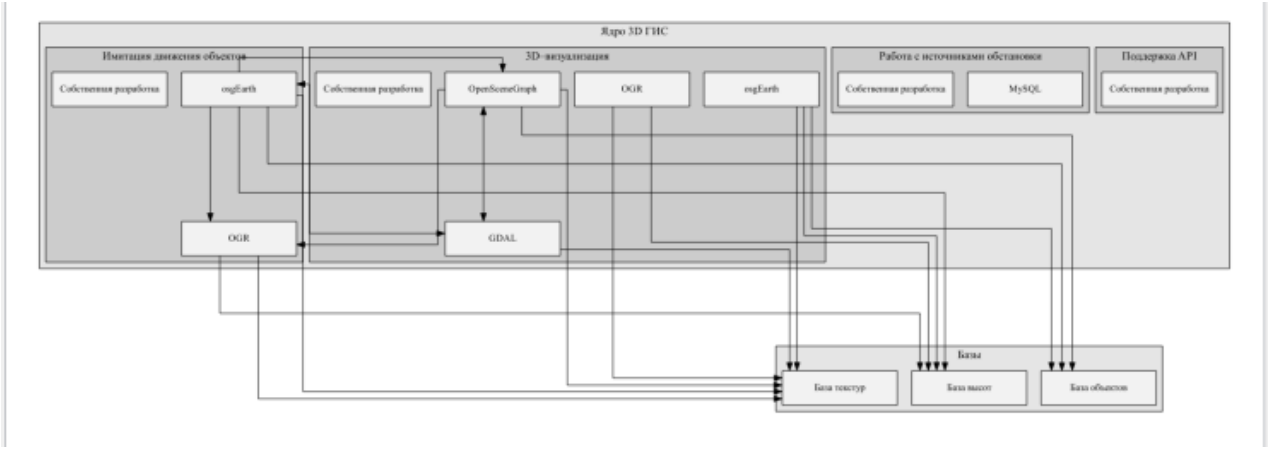

Рисунок 3.21. Структурная модель 3D ГИС

На рисунке 3.22 представлена функциональная модель 3D ГИС, отображающая дерево функций полученного проектного решения.

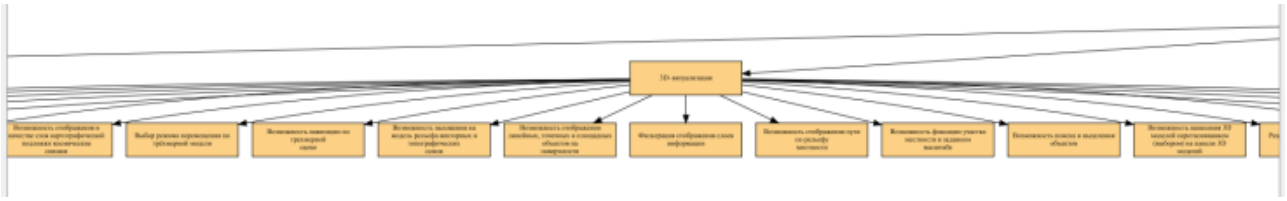

Рисунок 3.22. Функциональная модель 3D ГИС

На рисунке 3.23 представлен пример сформированных на основе диаграммы классов библиотек исходных кодов файлов заголовков на выбранном языке программирования, которые можно использовать для программной реализации проектного решения.

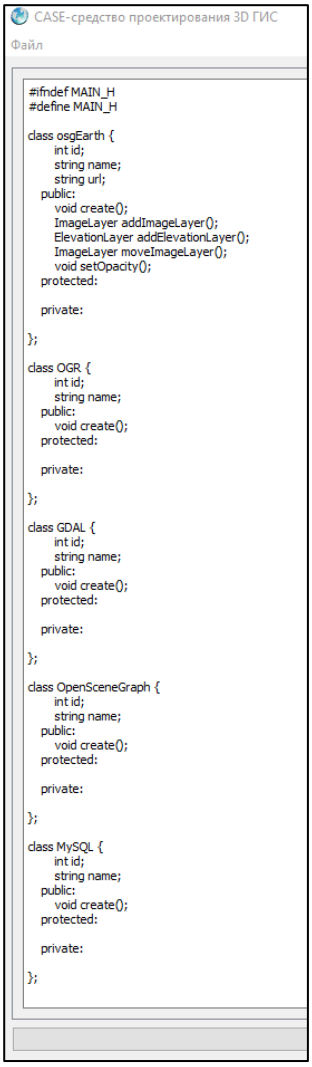

Рисунок 3.23. Исходные коды файлов заголовков

Любое проектное решение, а также весь набор сгенерированных проектных решений можно сохранить в файл конфигурации на жёстком диске или съёмном носителе. В дальнейшем файл конфигурации можно загрузить в CASE-средство для повторного использования.

Выводы

1. Разработан комплекс моделей, диаграмм и алгоритма проектирования 3D ГИС, которые формируют базу для создания ориентированного CASE-средства проектирования 3D ГИС отображения ситуационной обстановки и позволяют генерировать проектные решения и исходные коды файлов заголовков для последующей разработки на их основе 3D ГИС.
2. Разработана программная реализация CASE-средства проектирования 3D ГИС отображения ситуационной обстановки, позволяющий по введенным проектировщиком заданиям на проектирование формировать проектные решения для последующей реализации на их основе 3D ГИС широкой области применения.

Глава 4. Разработка алгоритма функционирования системы 3D моделирования проектных решений

Предлагаемое в главе 2 CASE-средство формирует проектные решения, которые включают следующие функциональные возможности: представление в 3D ГИС обстановки, ситуации, событий, состояний объектов и процессов внешнего мира.

Проектные решения должны быть перед их использованием на практике экспериментально проверены и промоделированы для того, чтобы установить их адекватность реальной обстановке и ситуациям, событиям.

Для осуществления этой задачи необходимо иметь средство, которое позволит моделировать проектные решения, т.е. «анализировать их на практике». Таким средством может служить система 3D моделирования проектных решений, обладающая функциональной полнотой, т.е. способностью выполнять большинство функций типовых 3D ГИС и работать с пространственными данными и 3D моделями объектов и ситуаций.

Таким образом, будет иметься возможность апробировать проектные решения, вырабатываемые CASE-средством, для установления их качества и возможности применения в конкретных предметных областях.

§ 4.1. Режимы функционирования 3D ГИС отображения ситуационной обстановки

4.1.1. Обстановка внешнего мира и её отображение в 3D ГИС

Виртуальное пространство 3D ГИС отображает реальный мир в той степени, которая устраивает пользователей и обеспечивает функционирование моделируемой системы. Виртуальное пространство должно быть удобным и не сильно изменять традиционный и привычный образ жизни, иметь перед ним преимущества.

Построение виртуального пространства опирается на идеи и опыт проектировния реальных систем с учетом возможностей использования передовых достижений в области информационных технологий.

Реальный мир включает в себя пространства объектов, состояний, обстановок, событий и ситуаций.

Дадим определения понятиям, которые используются при моделировании проектных решений.

Внешний (реальный) мир – множество состояний объектов окружающего мира, например, перечень рассматриваемых объектов, их координаты, семантические свойства.

Состояние – множество параметров, которые характеризуют объект в конкретный момент времени со статической и динамической точки зрения.

Обстановка – множество объектов, возможных связей между ними, их состояний. Примером обстановки может служить отображение побережья около полуострова Крым.

Событие – получение информации об изменении обстановки от внешних систем по доступным каналам. Примером события является изменение положения морского судна в пространстве.

Ситуация – отображение обстановки внешнего мира в текущий момент времени. Примером ситуации является пролёт воздушного судна (самолёта) над рассматриваемой местностью.

Пусть $\hat{O}$ *– пространство объектов,*

$\hat{S}$ *– пространство ситуаций (событий),*

тогда $\hat{R} = \hat{O} \cup \hat{S}$*, где* $\hat{R}$ *– реальный мир.*

Модель ситуации – это объединение текущей ситуации и подмножества пространства объектов, принимающих участие в ней.

$W_i = S_i \cup \hat{O}_i$,

где W_i *– модель i-ой ситуации,*

S_i *– i-ая ситуация,*

$\hat{O}_i$ *– объекты, участвующие в i-ой ситуации.*

Виртуальное пространство состоит из базы готовых цифровых моделей и базы инструментов для их построения.

$V = \{M\} \cup \{T\}$,

где V – виртуальное пространство,

M – база цифровых моделей,

T – база инструментов.

База цифровых объектов, в свою очередь, содержит модели таких объектов, как: рельеф, текстуры поверхности Земли, динамические объекты (транспортные средства, животные, люди), статические объекты (здания, ЛЭП, горы, впадины и т.д.).

База инструментов включает программные средства для создания цифровых моделей (3ds Max, AutoCAD, GDAL, OGR), CASE-средства для проектирования 3D ГИС и библиотеки, реализующие ту или иную часть функциональности 3D ГИС.

Функция отображения представляет собой цифровизацию реального мира в виртуальный. Цифровизация - переход на цифровой способ связи, записи и передачи данных с помощью цифровых устройств. Реальные объекты и ситуации посредством создания цифровых моделей и указания их атрибутов добавляются в виртуальное пространство.

4.1.2. Режимы отображения обстановки в 3D ГИС

Для управления объектами и ситуациями в 3D ГИС, а также обстановкой в целом вводится понятие режима 3D ГИС.

Режим 3D ГИС – определенный набор объектов, ситуаций и элементов местности, который позволяет моделировать реальную обстановку.

В 3D ГИС можно выделить несколько основных режимов отображения обстановки. Выбор режима зависит от требований пользователя 3D ГИС и задач, которые система должна решать. Как правило, 3D ГИС разрабатывается для решения конкретного класса задач определенной предметной области. Так можно выделить такие виды 3D ГИС, как экологические, геологические, транспортные, муниципальные и другие. В зависимости от вида 3D ГИС определяются и её режимы отображения, а они, в свою очередь, задают местность, объекты обстановки и ситуации, которые необходимы для дальнейшей визуализации. Например, в геологических ГИС важную роль играет местность, её рельеф, месторождения горных пород и полезных ископаемых и их виды, а также объекты, используемые для исследований поверхности планеты и раскопок. При исследовании местности большое значение имеют те или иные природные явления, в том числе, чрезвычайные ситуации природного характера, например, ураганы, землетрясения, извержения вулканов, которые могут сильно повлиять на ход исследований и внешний вид местности. [72,73]

Режим отображения обстановки определяет, какие ситуации и объекты система может моделировать, а какие не доступны, т.е. накладывает определенные ограничения на обстановку, в рамках которой 3D ГИС функционирует.

В 3D ГИС отображения ситуационной обстановки выделяется 4 основных режима отображения:

- морской режим отображения обстановки;
- наземный режим отображения обстановки;
- воздушный режим отображения обстановки;
- смешанный режим отображения обстановки.

При моделировании возможно сочетание и образование смешанных режимов обстановки, таких как:

- режим «наземная обстановка» представляет собой один из вариантов возможной обстановки, в которой событие происходит на суше (например, на территории города), рисунок 4.1А;
- режим «воздушная обстановка» представляет вариант обстановки, в которой событие происходит в воздухе (например, полёт самолёта), рисунок 4.1Б;
- режим «морская обстановка» представляет собой вариант обстановки, в которой событие происходит на море, рисунок 4.1В;
- режим «наземная и морская обстановка» представляет собой вариант обстановки, в которой событие происходит и на суше, и на море при участии объектов: кораблей, гражданских судов, моделей местности: берега, моря, рисунок 4.1Е;
- режим «наземная и воздушная обстановка» представляет собой вариант обстановки, в которой событие происходит и на суше, и в воздухе (воздушное пространство над сушей), рисунок 4.1Г;
- режим «воздушная и морская обстановка» представляет собой вариант обстановки, на которой событие происходит и на море, и в воздухе (воздушное пространство над морем), рисунок 4.1Д;
- режим «наземная, воздушная и морская обстановка» представляет собой вариант обстановки, на которой событие происходит в один и тот же интервал времени и на суше, и на море, и в воздухе (например, в районе побережья полуострова Крым), рисунок 4.1Ж.

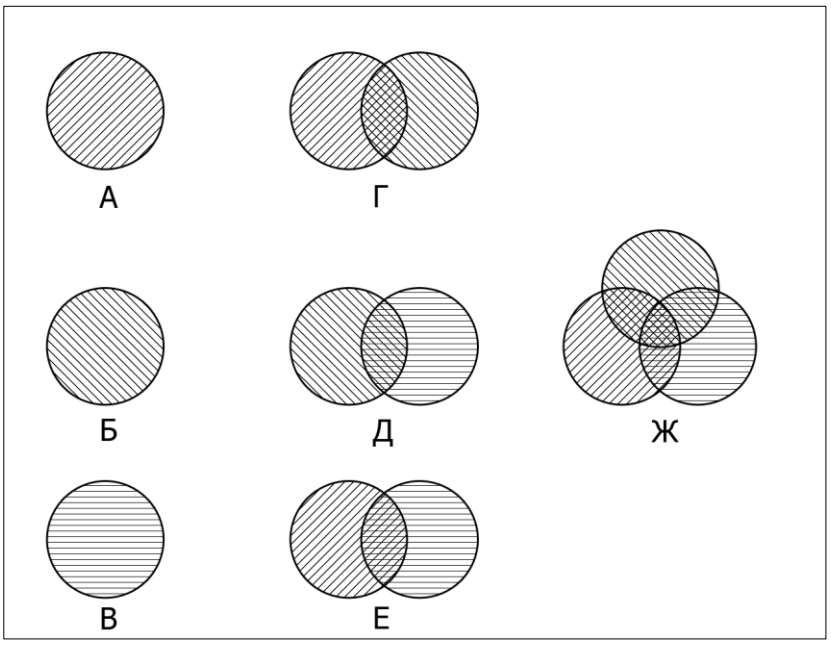

Рисунок 4.1. Схема пересечений режимов отображения обстановки

Любой режим отображения обстановки включает в себя определенный набор объектов, принимающих участие в обстановке. Одни объекты являются уникальными и используются только в одном режиме (например, подводные вулканы только в морском режиме), другие объекты взаимодействуют сразу с несколькими режимами. Схема соответствия объектов и режимов отображения ситуационных обстановок представлена на рисунке 4.2.

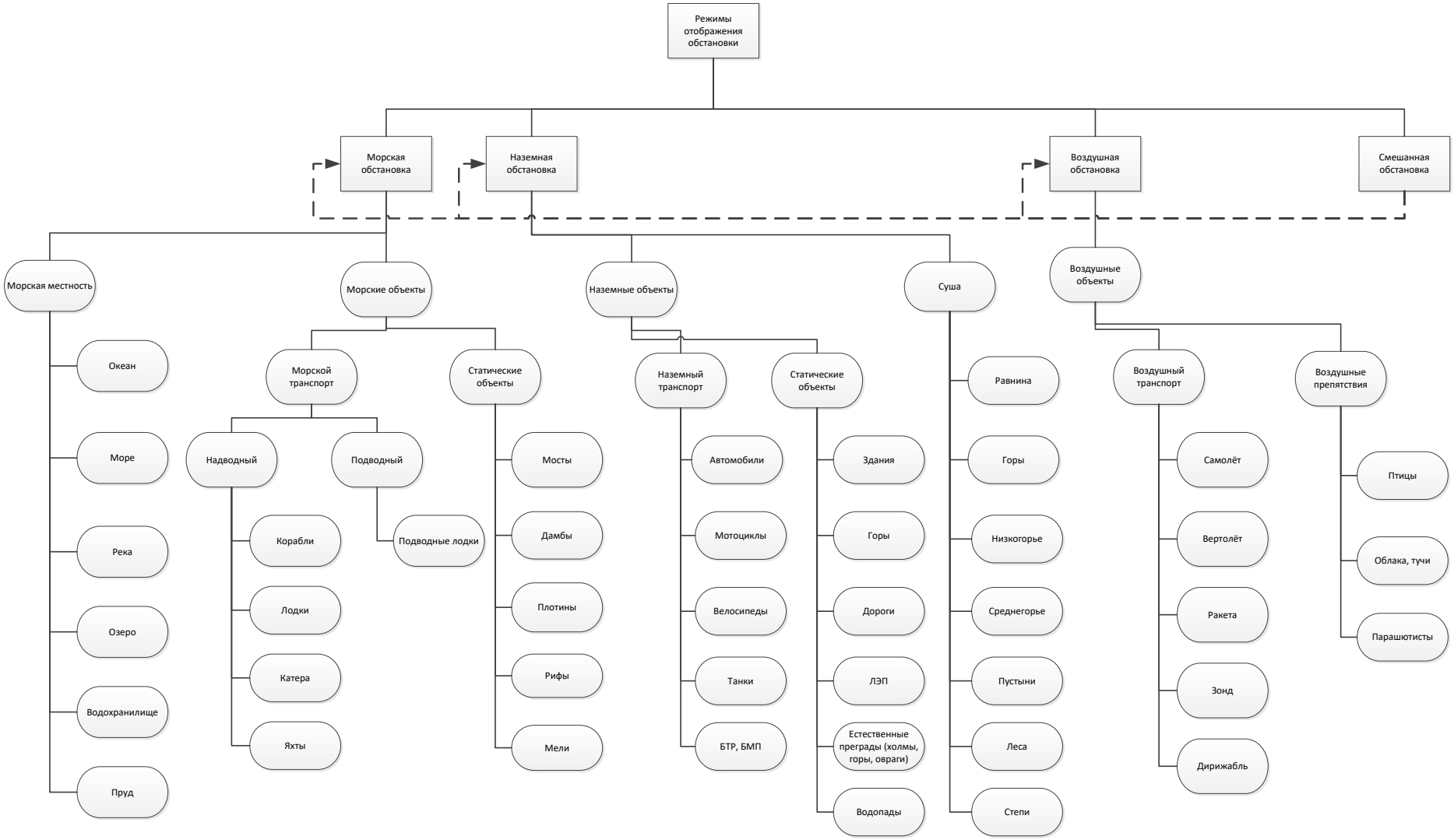

Рисунок 4.2. Схема соответствия объектов и режимов отображения ситуационных обстановок

На основе анализа существующих 3D ГИС, таких как ArcGIS, QuantumGIS, GRASS, КБ «Панорама», а также особенностей функционирования 3D ГИС в целом выявлен перечень функций, обязательных для любой 3D ГИС. Эти функции реализуют основное поведение системы и должны присутствовать при разработке новых 3D ГИС. Для описания этих функций используются соответствующие им алгоритмы функционирования, которые образ

Алгоритмическая база включает следующие алгоритмы:

- алгоритм обмена информацией с внешними системами;
- алгоритм обмена информацией по UDP-порту;
- алгоритм рисования геометрических фигур;
- алгоритм загрузки и отображения 3D модели объекта;
- алгоритм распознавания объекта;
- алгоритм идентификации ситуации.

Все перечисленные алгоритмы представлены в «скелетной» модели алгоритмов функционирования 3D ГИС (Рисунок 4.3).

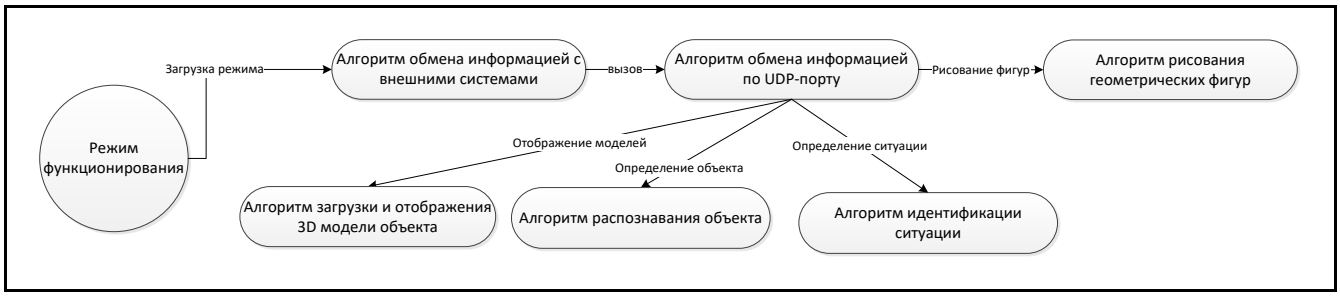

Рисунок 4.3. «Скелетная» модель алгоритмов функционирования 3D ГИС

«Скелетная» модель содержит алгоритмы, которые являются необходимой базой для обеспечения функционирования любого режима 3D ГИС. Эти алгоритмы задают базовый набор процедур, реализующих основную функциональность 3D ГИС. Такими процедурами являются:

- отображение местности;
- размещение текстур;
- наложение рельефа;
- размещение 3D моделей объектов;
- манипуляции с подсистемой навигации по трёхмерной модели;
- имитация движения объектов;
- отображение векторных основ (точек, линий, полигонов).

Представленная алгоритмическая база является неотъемлемым элементом 3D ГИС, необходимым при подключении любого режима (Рисунок 4.4). На её основе возможно моделирование как уже существующих режимов: морского, наземного, воздушного, смешанного режимы, так и создание новых. Алгоритмическая база взаимодействует с любым режимом при помощи специального API, который обеспечивает передачу входных и выходных данных алгоритмов между собой и с готовыми библиотеками.

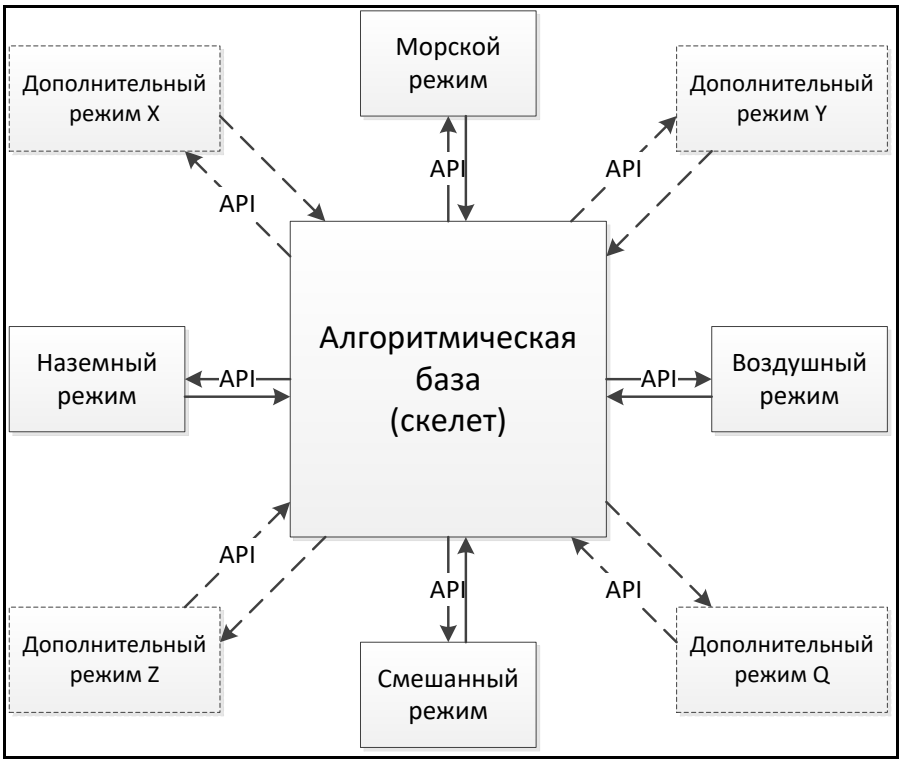

Рисунок 4.4. Матрица подключения режимов

Рассмотрим подробнее каждый режим отображения ситуационной обстановки.

Морской режим

Морской режим отображения обстановки позволяет моделировать такие элементы местности, как: океаны, моря и реки с учётом их глубины и рельефа дна, побережья; объекты: морские суда, рифы, скалы, айсберги; ситуации: штормы, морские землетрясения, цунами с отображением волн, цвет и степень загрязненности воды, военные действия на море.

Для моделирования морского режима из базы алгоритмов визуализации используются следующие алгоритмы:

– алгоритм отображения рельефа дна;
– алгоритм привязки растрового изображения по координатам;
– алгоритм моделирования водной поверхности;

– алгоритм загрузки и отображения 3D модели морского объекта;

– алгоритм фильтрации морских слоёв информации.

Модель алгоритмов функционирования морского режима 3D ГИС представлена на рисунке 4.5.

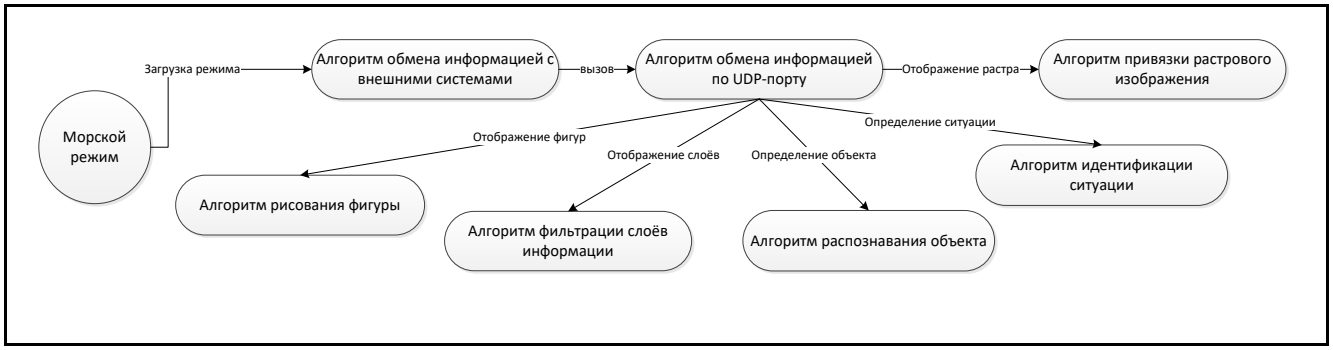

Рисунок 4.5. Модель алгоритмов функционирования морского режима 3D ГИС

У всех объектов морского режима используются такие пространственные данные, как: координаты на местности (широта, долгота), глубина (для подводных объектов и рельефа дна).

Пространственные данные могут включать географические объекты, представляемые:

- точками;
- линиями;
- полигонами.

При моделировании морского режима точки представляют морские суда, скалы, айсберги или несколько объектов, расположение которых описывается единственной точкой.

Дугами являются те реальные объекты, которые можно рассматривать как линии. Дуга состоит из отрезков линий и дуг окружностей. В морском режиме дуга может быть рекой, мостом, берегом или дамбой.

Полигоны - это замкнутые области, которые представляют однородные по некоторым критериям участки. Полигонами обозначают опасные морские участки, области со штормами, штилем или территории обитания морских животных.

Во время функционирования морского режима 3D ГИС имеет ряд ограничений: система не может моделировать сушу и воздушную обстановку.

Используемые библиотеки: OpenSceneGraph, osgEarth, OpenGL, GDAL, OGR, MarineTraffic.

Местность: береговая линия, айсберги, утёсы, вулканы, течение, порог, мель, риф, коса, болото, дамба, плотина, водохранилище, озеро, пруд, река, море, океан, вулкан.

Форматы карт: GeoTIFF, JPG, PNG, KML, S57.

Рельеф: глубина Мирового океана.

Отображаемые объекты: баржа, броненосец, катамаран, крейсер, линкор, лодка, катер, танкер, яхта, затонувшее судно, атомная подлодка, парогазотурбинная подлодка, дизельная подлодка.

Ситуации: землетрясение, извержение вулкана, цунами, штиль, шторм, ураган, буря, ветер.

Наземный режим

Наземный режим отображения обстановки позволяет моделировать местность: равнины, горы, леса, степи; объекты: наземный транспорт, здания, дороги, ЛЭП; ситуации: ураганы, смерчи, пожары, наводнения, военные действия на море.

В наземном режиме используются следующие алгоритмы:

– алгоритм отображения рельефа суши;
– алгоритм привязки растрового изображения по координатам;
– алгоритм загрузки и отображения 3D модели наземного объекта;
– алгоритм рисования геометрических фигур (линий, полигонов) на суше;
– алгоритм фильтрации наземных слоёв информации.

Модель алгоритмов функционирования наземного режима 3D ГИС представлена на рисунке 4.6.

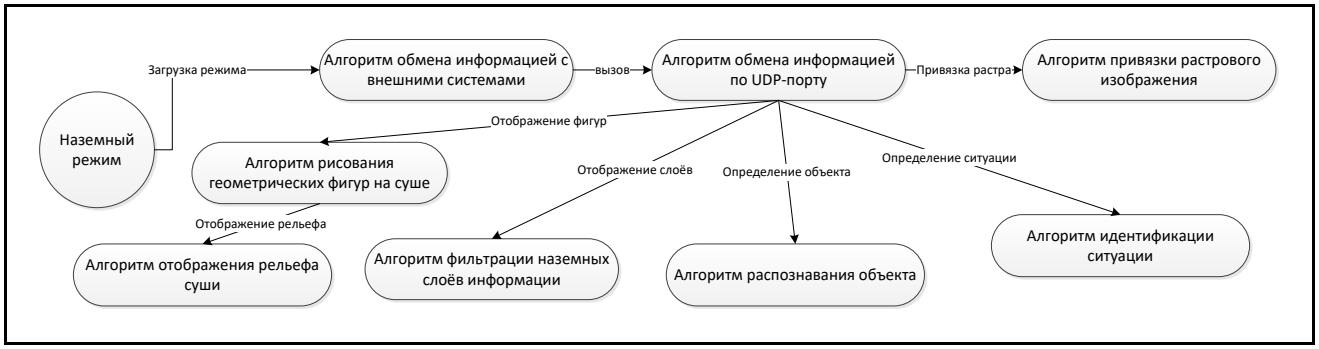

Рисунок 4.6. Модель алгоритмов функционирования наземного режима 3D ГИС

У объектов наземного режима используются такие пространственные данные, как: координаты на местности (широта, долгота), высота или глубина (для рельефа местности и объектов, расположенных на возвышенностях или в низинах).

В наземном режиме точками являются: наземный транспорт, здания, вышки; линиями: дороги, ЛЭП; полигонами: горы, города, деревни, луга, поля.

Используемые библиотеки: openSceneGraph, osgEarth, OpenGL, GDAL, OGR, osGeo.

Местность: пустыня, берег, водопад, высокогорье, среднегорье, низкогорье, холм, равнина, вулканы, горы.

Форматы карт: GeoTIFF, JPG, PNG, KML, SXF.

Рельеф: возвышенность или низменность.

Отображаемые объекты: транспортное средство, человек, животное, дерево, здание, лэп, дорога, гражданское, военное, автомобиль, мотоцикл, велосипед, танк, бтр, бмп, железная дорога, автодорога. [81,82,83]

Ситуации: землетрясение, извержение вулкана, пожар, химическая, коммунальная, чрезвычайная, гидродинамическая, буря, ветер, оползни, болезни, животных, людей.

Воздушный режим

Во время функционирования воздушного режима 3D ГИС имеет ограничения: отсутствует возможность отображения рек, озер, морских и воздушных объектов.

Воздушный режим отображения обстановки отличается от других тем, что не моделирует никакую местность, а визуализирует только объекты: воздушный транспорт, птиц, автономные летательные аппараты, облака; и ситуации: траектории воздушных перелетов, ураганы, дожди, грозы, авиакатастрофы.

В воздушном режиме используются следующие алгоритмы:

– алгоритм загрузки и отображения 3D модели воздушного объекта;

– алгоритм рисования геометрических фигур (линий, полигонов) в воздухе;

– алгоритм моделирования воздушных перелетов.

Модель алгоритмов функционирования воздушного режима 3D ГИС представлена на рисунке 4.7.

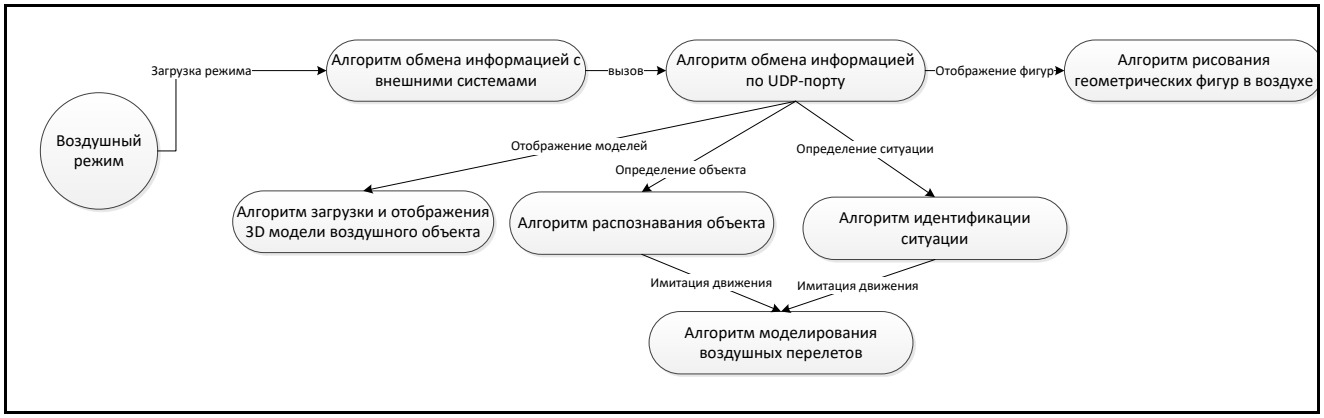

Рисунок 4.7. Модель алгоритмов функционирования воздушного режима 3D ГИС

У объектов воздушного режима используются такие пространственные данные, как: координаты (широта, долгота), высота над уровнем Мирового океана.

В воздушном режиме точками являются: воздушные суда, птицы, высокие наземные объекты; линиями: воздушные маршруты; полигонами: циклоны, антициклоны, облака, грозовые тучи, другие опасные участки.

Во время функционирования воздушного режима 3D ГИС имеет ограничения: отсутствует возможность отображения какой-либо местности, морских и наземных объектов.

Используемые библиотеки: openSceneGraph, osgEarth, OpenGL, GDAL, OGR, FlightRadar.

Местность: -

Форматы карт: GeoTIFF, JPG, PNG, KML.

Рельеф: высота.

Отображаемые объекты: самолёт, дирижабль, вертолёт, ракета, зонд, птица, парашютист, истребитель, пассажирский, бомбардировщик.

Ситуации: бури, ураганы, грозы, авиакатастрофы.

В 3D ГИС для всех режимов отображения обстановки происходит взаимодействие с внешними системами и обновление отображаемой информации.

Универсальным режимом является *смешанный режим* (Рисунок 4.8) отображения обстановки, который включает частично или полностью другие режимы отображения обстановки. Смешанный режим позволяет отображать, как морскую местность, так и сушу, моделировать воздушные, наземные и морские объекты, а также ситуации, в которых они принимают участие.

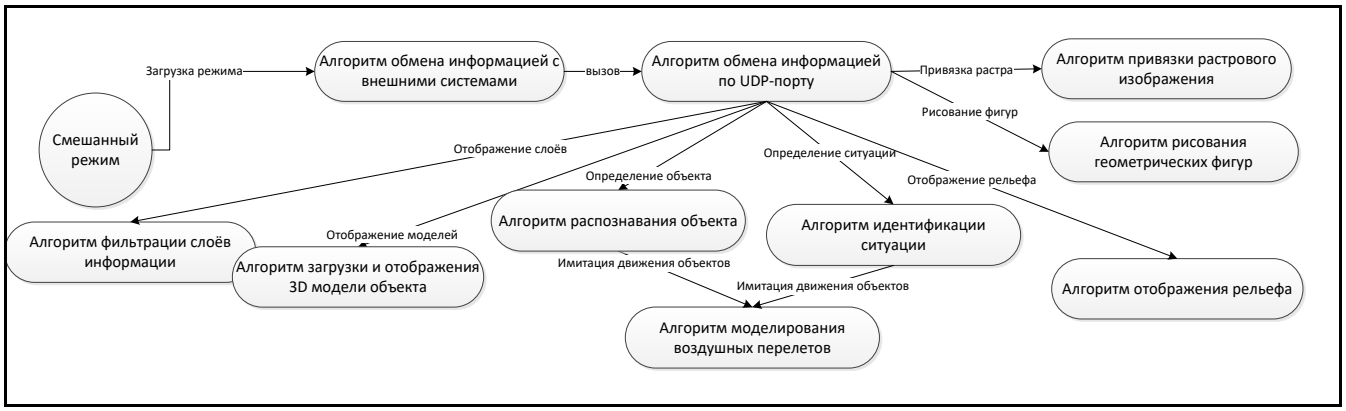

Рисунок 4.8. Модель алгоритмов функционирования смешанного режима 3D ГИС

Для обеспечения любого режима необходимо выполнять следующие действия.

1. Выбрать на панели управления элемент с названием соответствующего режима. После нажатия активируется группа алгоритмов (Рисунок 4.5, Рисунок 4.6, Рисунок 4.7, Рисунок 4.8), которые обеспечивают функциональность этого режима.
2. Определить источник обстановки, выбрав на панели инструментов элемент соответствующего способа ввода (XML-файл, ручной ввод, по UDP-порту).
3. Выбрать элемент «Загрузка входных данных» на панели управления системы 3D моделирования.

   В случае взаимодействия с XML-файлом для каждого режима уже сформулированы соответствующие карты. Для морского режима: карты S-57, TMS, для наземного: SXF, OpenStreetMap, для воздушного: OpenStreetMap с отображением слоёв погодной обстановки.

При ручном вводе необходимо выбрать пункт меню «Загрузка растровых и векторных карт», далее выбрать файл карты на жестком диске компьютера и нажать пункт меню «Загрузить».

4. Далее необходимо задать координаты района Земли путём указания их в текстовых полях на панели инструментов. Нажать кнопку перехода по этим координатам (кнопка «Перейти»). Открывается заданный район.
5. Наложить слои текстур, высот и глубин, которые подключаются автоматически. На экране появляется район с «наложенными» слоями текстур, глубин и высот.
6. Размещение объектов в заданном районе может выполняться автоматически и вручную. В автоматическом режиме объекты добавляются из базы данных через используемые файлы в формате XML. В ручном режиме в XML файл при помощи текстового редактора добавляются новые объекты и редактируются существующие.
7. Каждому объекту, помещаемому в выбранный район, «прикрепляется» 3D модель в формате 3DS из базы 3D объектов.
8. При наведении курсора на 3D объект появляется информация об идентификаторе и атрибутах объекта, а также его координаты. Если это летательный аппарат, то дополнительно высота, если подводный – глубина.
9. На местности можно определить высоты рельефа, морские глубины путём подведения курсора к интересуемому месту.
10. Имеется возможность имитации движения объектов с регулированием частоты обновления местоположения объектов.
11. Имеется возможность фильтрации слоёв отображения 3D объектов. Например, для отображения только подводной лодки. Для этого необходимо нажать на панели инструментов на кнопку «Скрыть» для других объектов.
12. Для отображения полигональных, линейных и точечных объектов нужно нажать на кнопку «Символы» и на 3D глобусе появятся полигональные, линейные и точечные объекты.
13. При необходимости можно увеличивать количество объектов в районе. При этом ограничением служат аппаратно-программные ресурсы.
14. Для отображения полигональных, линейных и точечных объектов нужно нажать на кнопку «Символы» и на 3D глобусе появится выбранный объект.

Таким образом, выбор режима обеспечивает отображение реальной обстановки для оценки проектного решения.

4.1.3. Способы взаимодействия с внешними системами в 3D ГИС отображения ситуационной обстановки

Для задания режима отображения обстановки 3D ГИС, изменения объектов, рельефа и текстур местности оператору необходимы специальные средства взаимодействия с 3D ГИС. Такие средства могут использовать 4 основных способа ввода данных:

- ручной ввод;
- обмен через XML файл;
- обмен через базу данных;
- обмен по порту.

Ручной ввод является самым медленным видом обмена информацией в 3D ГИС. Оператор системы при помощи мыши и клавиатуры вводит обновлённые данные об обстановке и удаляет устаревшие. Для внесения изменений в обстановку оператор выбирает на панели инструментов 3D ГИС необходимый элемент (например, загрузка растрового слоя), указывает дополнительные параметры (путь к файлу) и применяет указанные изменения. Данный способ не является автоматизированным и не позволяет в реальном времени обновлять и отображать ситуационную обстановку.

Обмен через XML файл подразумевает хранение всей текущей информации в файле формата XML. 3D ГИС средствами файловой системы периодически проверяет данный файл на изменения (частота проверки задаётся в настройках 3D ГИС) и в случае нахождения каких-либо новых данных загружает их в оперативную память и в дальнейшем отображает на 3D модели Земли. В XML файл изменения могут вноситься как вручную операторами, так и автоматически записываться из внешних систем. Данный способ является автоматизированным, позволяет быстро отобразить обновлённую обстановку, но имеет погрешность в актуальности обстановки, равной периоду проверки XML файла на изменения. Вторым недостатком является то, что XML файл может быть отредактирован в одно и то же время разными внешними системами, что приведет к затиранию и потере информации.

Обмен через базу данных похож на обмен через XML файл, но его преимуществом является возможность одновременного обновления информации от нескольких внешних систем за счёт использования транзакций. Недостатком также как при обмене XML файла является погрешность в актуализации обстановки.

Наиболее оперативным способом обмена является прямое взаимодействие между 3D ГИС и внешними системами через порт. В настройках 3D ГИС задается UDP или TCP порт, по которому система «ожидает» входящие сообщения и при получении нового сообщения автоматически обновляет обстановку. Разница между протоколами TCP и UDP – в так называемой

«гарантии доставки». TCP требует отклика от клиента, которому доставлен пакет данных, подтверждения доставки, и для этого ему необходимо установленное заранее соединение. Также протокол TCP считается надежным, тогда как UDP не гарантирует корректную доставку пакетов. TCP исключает потери данных, дублирование и перемешивание пакетов, задержки. UDP все это допускает, и соединение для работы ему не требуется.

§ 4.2. Процесс моделирования проектных решений

Рассмотренные режимы отображения обстановки и способы взаимодействия с внешними системами обеспечивают моделирование любых проектных решений, генерируемых CASE-средством проектирования 3D ГИС.

Процесс моделирования проектных решений начинается с выбора режима отображения обстановки. Такая обстановка может быть задана одним из четырёх способов: через ручной ввод, через XML файл, через базу данных и по порту. Для ручного ввода на панели управления системы 3D моделирования имеется элемент «Режимы обстановки» со списком всех необходимых режимов.

После этого необходимо задать координаты местности, на которой моделируется развиваемая ситуация. На панели управления это задание отображается в виде текстовых полей для ввода необходимых координат.

Следующим шагом является заполнение местности, загрузка карт, рельефа местности, объектов, принимающих участие в ситуации. В системе 3D моделирования есть возможность задания формата карт: растровых (GeoTIFF, PNG, JPG, BMP, OpenStreetMap) и векторных (SHAPE, KML, S57, SXF).

После задания всех необходимых составляющих обстановки пользователь может приступать к визуальной оценке проектного решения. Для этого в системе 3D моделирования имеются режимы визуализации:

- режим отображения динамики движения объектов путём внесения координат, для этих целей в меню объекта задаются начальные и конечные координаты, между которыми происходит плавное перемещение 3D модели объекта;
- режим облёта камеры, который предназначен для обеспечения более качественного просмотра построенного участка местности;
- режим переключения отображаемых слоёв обстановки, который осуществляется для временного отключения ненужных в данный момент слоёв, что позволяет более наглядно рассмотреть интересующие элементы местности;

- каркасный режим – в этом режиме производится анализ рельефа местности с целью принятия решения о направлении движения динамических объектов;
- режим наложения различных типов карт, который предназначен для интеграции, отображения и сравнения объектов с разных карт на одной местности (дороги, строения, ЛЭП);
- режим полноэкранного отображения, который служит для большей наглядности и имеет максимальное рабочее пространство;
- режим добавления векторных объектов - в данном режиме пользователь имеет возможность ручного добавления линий и полигонов на местности при помощи клавиатуры и мыши;
- режим фильтрации объектов – режим, позволяющий временно скрывать некоторые, ненужные в данный момент объекты для более удобного просмотра местности;
- режим слежения за объектом – это динамический режим, который обеспечивает анализ изменения местоположения объектов, заданных трёхмерными моделями.

Данных возможностей системы 3D моделирования вполне достаточно для того, чтобы проектировщик смог оценить полученное проектное решение и принять решение по его дальнейшей доработке или реализации.

§ 4.3. Алгоритмы функционирования 3D ГИС

Перед созданием алгоритма необходимо выяснить, какие задачи он должен решать. Постановка задачи алгоритма предполагает подробное описание самой задачи, описание входной и выходной информации, формулирование конечной цели, которую необходимо достигнуть при решении задачи. Одним из способов постановки задачи является математическая формулировка, которая заключается в записи условия задачи с помощью математических обозначений, формул, зависимостей, в определении исходных данных и формы выдачи результатов вычислений.

После построения математической модели необходимо выбрать метод решения задачи на ЭВМ. Выбранный метод является основой построения алгоритма решения задачи. Разработка алгоритма решения задачи заключается в установлении необходимой последовательности арифметических и логических действий, строгое выполнение которых приводит к решению задачи.

Дополнительно необходимо указать требования к самому алгоритму:

- скорость выполнения поставленной перед алгоритмом задачи;
- объём оперативной памяти, необходимый для корректной работы алгоритма;
- объём дискового пространства, который занимает алгоритм;
- требования к другим компонентам компьютера (частота процессора, видеопамять);
- требования к операционной системе;
- особенности входных данных;
- особенности выходных данных;
- язык программной реализации алгоритма.

Для обеспечения функционирования в системе 3D моделирования всех режимов визуализации (§4.2), режимов отображения обстановки (4.1.2) и способов взаимодействия с внешними системами (4.1.3) была разработана база алгоритмов функционирования 3D ГИС, которая включает:

- алгоритм идентификации ситуации;
- алгоритм распознавания объекта;
- алгоритм привязки растрового изображения по координатам;
- алгоритм рисования фигуры на карте;
- алгоритм полёта камеры;
- алгоритм обмена информацией по UDP-порту;

- алгоритм обмена информацией с внешними системами, который описывает возможности взаимодействия 3D ГИС с внешними системами, распознавание команд и их выполнение;
- алгоритм фильтрации слоёв информации, который позволяет изменять список слоёв информации, видимых на глобусе, а также их порядок;
- алгоритм построения модели обстановки, который позволяет описывать различные режимы отображения обстановки (морская, наземная, воздушная, смешанная), а также кодировать значения элементов выбранного режима в строку символов для дальнейшего хранения.

Данные алгоритмы образуют необходимое ядро системы 3D моделирования для корректной визуальной оценки проектных решений. Также представленные выше алгоритмы могут образовывать базовое ядро любой 3D ГИС, разрабатываемой на основе сгенерируемых CASE-средством проектных решений, и реализуют все основные функциональные требования.

Для взаимодействия алгоритмов функционирования 3D ГИС между собой, а также с другими алгоритмами, расширяющими основную функциональность, был разработан общий алгоритм функционирования 3D ГИС, который описывает последовательность действий и вызовов алгоритмов. Общий алгоритм функционирования 3D ГИС отображает все возможные процедуры, которые выполняет система от запуска и до завершения (Рисунок 4.9).

Требования к общему алгоритму функционирования:

- объём оперативной памяти, необходимый для корректной работы алгоритма – от 2 ГБ и выше;
- объём дискового пространства, который занимает алгоритм - от 100 МБ и выше (объём увеличивается за счёт загрузки растровых и векторных картографических данных, а также 3D моделей объектов);
- требования к другим компонентам компьютера (частота процессора – от 1.5 ГГц и выше, видеопамять – дискретная видеокарта с памятью от 512 МБ и выше);
- требования к операционной системе: Windows XP/Vista/7/8/8.1/10/Server 2008/Server 2012, Debian 7-8/Ubuntu 14.04-17.04, Mac OS 10;
- особенности входных данных (на входе алгоритма передаются: конфигурационный файл, координаты отображаемой местности, угол наклона, азимут, необходимость загрузки моделей неба и океана, файлы растровых и векторных карт, высот и глбуин, 3D моделей объектов);

– особенности выходных данных (на выходе алгоритма: 3D модель местности, с текстурами, высотами, глубинами, 3D моделями объектов; формуляры, содержащие справочную информацию по объектам; динамика перемещения объектов).

Общий алгоритм функционирования 3D ГИС отображения ситуационной обстановки обладает всеми необходимыми свойствами:

– дискретностью (представлены все этапы функционирования 3D ГИС от запуска до завершения);
– определенностью (представленные этапы имеют чёткий порядок вызова и взаимодействия друг с другом);
– массовостью (алгоритм может быть применён при построении 3D ГИС широкой области применения);
– результативностью (алгоритм достигает окончания за конечное число операций);
– формальностью (программная реализация алгоритма чётко выполняет всю последовательность операций, указанных в алгоритме).

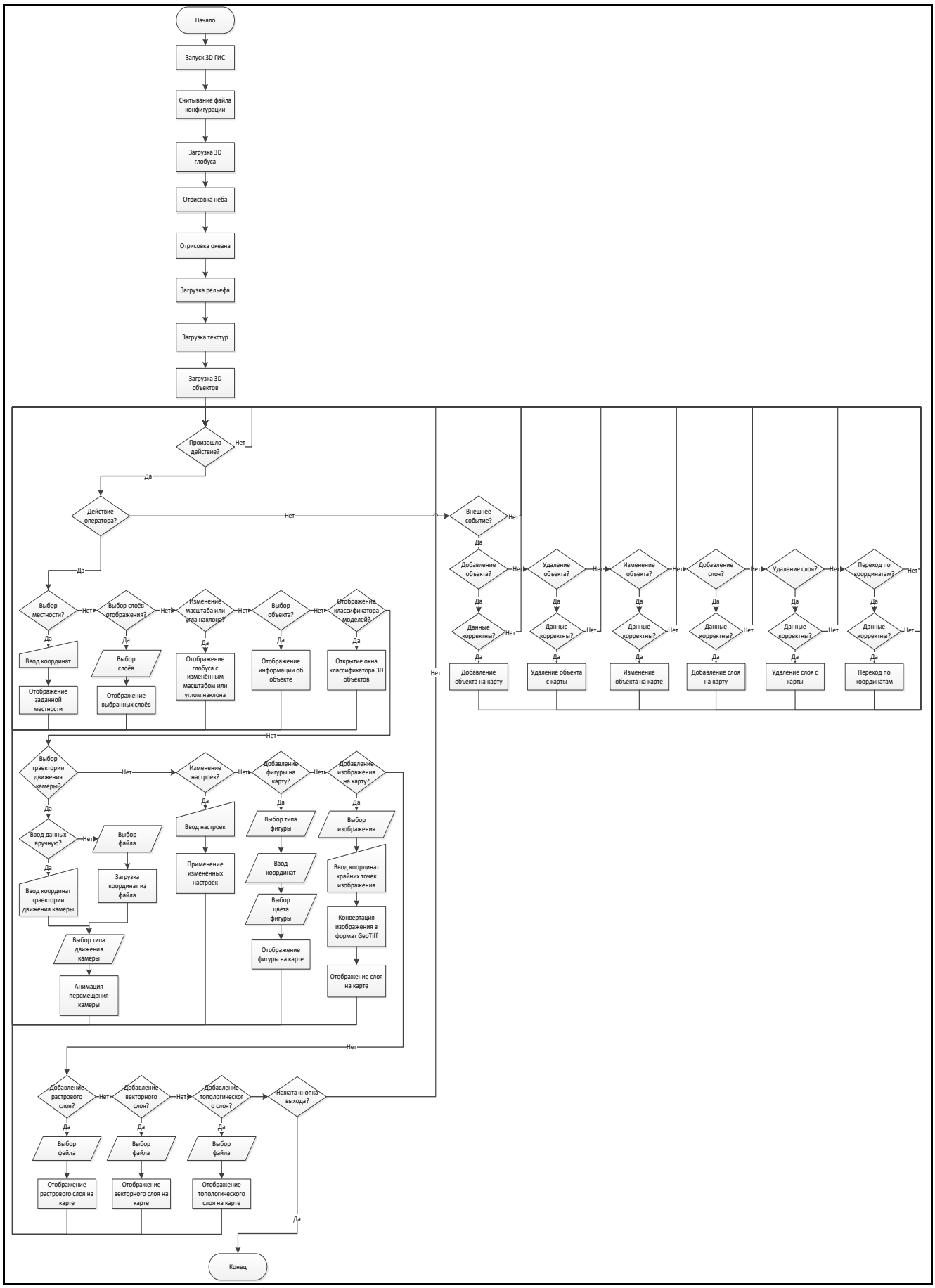

Рисунок 4.9. Общий алгоритм функционирования 3D ГИС

Далее представлена последовательность действий общего алгоритма функционирования 3D ГИС.

1. Выполняется запуск 3D ГИС. Запуск осуществляется путём вызова исполняемого файла из директории, где установлена система, либо с помощью ярлыка на этот исполняемый файл.
2. Производится считывание файла конфигурации 3D ГИС (по умолчанию: config.ini).
3. Является загрузка 3D глобуса, которая представляет собой вызов библиотек, выполняющих построение трехмерной модели поверхности Земли. Такими библиотеками являются OpenSceneGraph и osgEarth.
4. Выполняется отображение модели неба, Солнца, Луны и звёзд при помощи вызова функций библиотеки osgEarth.
5. После отображения неба и звёзд происходит отображение модели океана и водной поверхности с помощью библиотеки osgEarth.
6. Указанные в конфигурационном файле модели рельефа загружаются на модель 3D глобуса с учётом их порядка в конфигурационном файле.
7. Выполняется загрузка векторных и растровых текстур, прописанных в конфигурационном файле, и их отображение на 3D глобусе.
8. После загрузки текстур производится загрузка моделей 3D объектов из конфигурационного файла и их визуализация на 3D модели глобуса.
9. Основные компоненты системы загружены. 3D ГИС находится в состоянии ожидания команд. Данные команды могут передаваться вручную оператором системы либо автоматически путём изменения XML файла или от внешних систем по UDP порту.
10. При поступлении команды система определяет тип команды.
11. Если команда поступила от оператора 3D ГИС, то определяются действия, какие должна выполнить система. Для каждой команды определен свой набор действий:
    a. *выбор местности:*
        i. ввод координат местности, на которую необходимо перейти;
        ii. отображение местности по введённым координатам;
    b. выбор слоёв отображения:
        i. выбор в открывшемся диалоговом окне слоёв, которые необходимо отобразить;
        ii. поиск на глобусе указанных слоёв и их отображение;
    c. изменение масштаба при помощи указателя мыши:
        i. отображение местности с изменённым масштабом;
    d. *выбор объекта:*
        i. выбор объекта нажатием левой клавиши мыши по нему и открытие диалогового окна с информацией об объекте;
    e. *открытие классификатор моделей:*
        i. выбор левой клавишей мыши пункта «Классификатор объектов» на панели инструментов и открытие диалогового окна классификатора объектов;
    f. выбор траектории движения камеры:
        i. ввод координат траектории движения камеры вручную оператором либо из файла;
        ii. выбор типа движения камеры (по рельефу, на заданной высоте);
        iii. анимация движения камеры;

g. *изменение настроек:*
    i. ввод новых настроек через клавиатуру;
    ii. сохранение настроек в конфигурационный файл;
h. *добавление фигуры на карту:*
    i. выбор типа фигуры в открывшемся диалоговом окне;
    ii. ввод координат местоположения фигуры вручную либо из файла;
    iii. выбор цвета фигуры из выпадающего списка;
    iv. отображение фигуры на карте;
i. *добавление изображения на карту:*
    i. выбор изображения на файловой системе;
    ii. ввод координат крайних точек изображения;
    iii. конвертация изображения в формате GeoTiff при помощи библиотеки GDAL;
    iv. загрузка сконвертируемого растра на 3D глобус;
j. *добавление растрового слоя:*
    i. выбор растра на файловой системе;
    ii. отображение растра на глобусе при помощи библиотеки GDAL;
k. *добавление векторного слоя:*
    i. выбор файла с векторными данными на файловой системе;
    ii. отображение векторного слоя на глобусе при помощи библиотеки OGR;
l. *добавление топологического слоя:*
    i. выбор файла с топологическими данными на файловой системе;
    ii. отображение топологического слоя на глобусе при помощи библиотеки OpenSceneGraph;
m. *команда закрытия системы:*
    i. выход.

Рассмотрим более подробно каждый алгоритм из базы алгоритмов функционирования.

Алгоритм идентификация ситуации (Рисунок 4.10) решает задачи отображения ситуации на местности, а также объектов, участвующих в данной ситуации, моделей текстур и рельефа. Дополнительной возможностью данного алгоритма является поиск ситуации по заданным пользователем критериям, например, по координатам местности, участвующим в ситуации объектам.

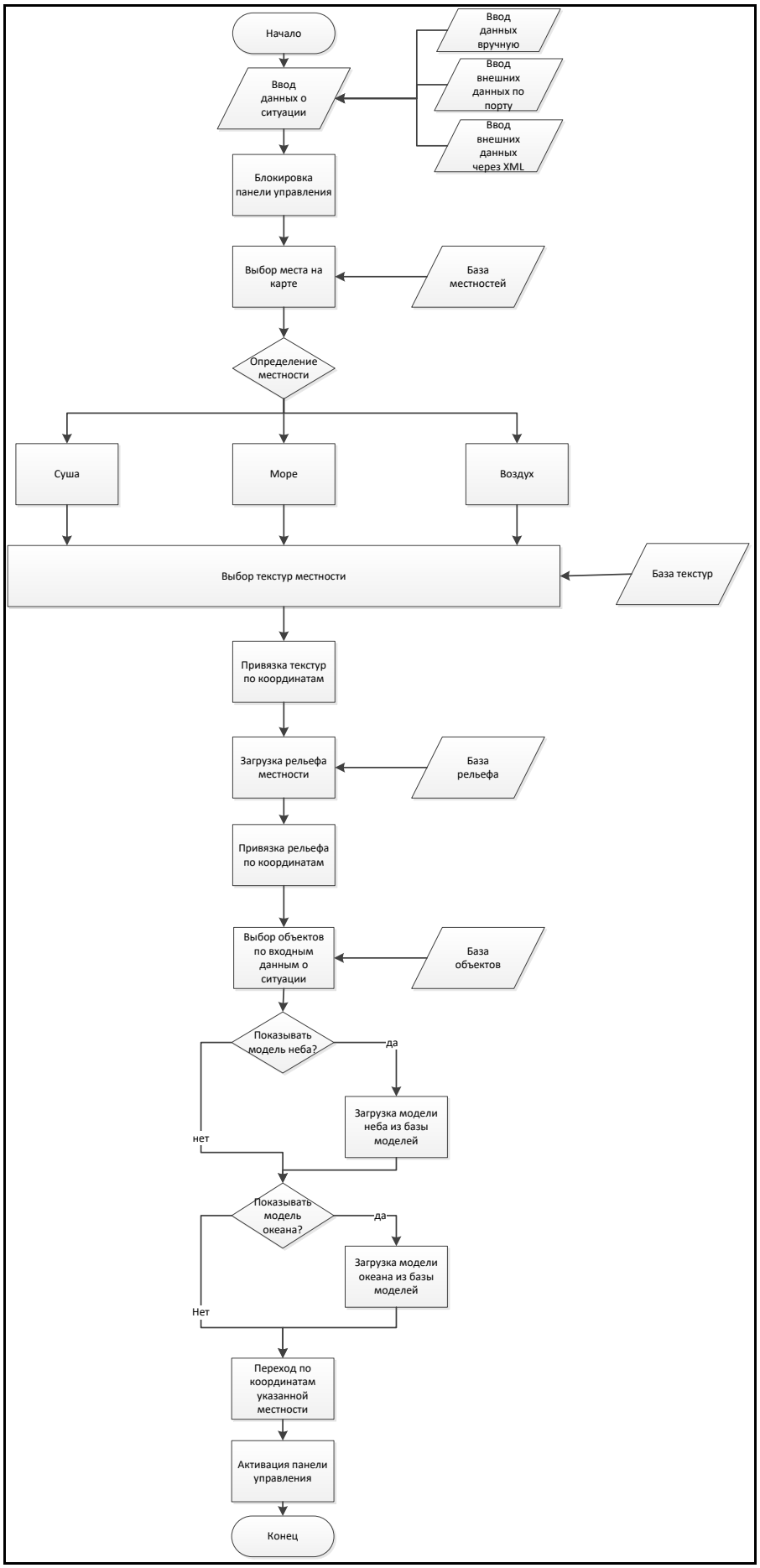

Рисунок 4.10. Алгоритм идентификации ситуации

Алгоритм идентификации ситуации входит в компонент работы с источниками обстановки базового ядра 3D ГИС и взаимодействует с такими ресурсами как: растровые и векторные карты местности, матрицы высот и глубин, 3D модели объектов, принимающих участие в ситуации.

Основной характеристикой ситуации являются координаты местности, на которой она происходит. Для каждой ситуации присваивается уникальный идентификатор и множество объектов, которые в ней участвуют. По всем этим параметрам 3D ГИС имеет возможность поиска и выбора необходимой пользователю ситуации.

Режимы обстановки, в которых функционирует алгоритм идентификации ситуации: морская, наземная, воздушная, смешанная.

Примеры свободно-распространяемых библиотек, с которыми взаимодействует алгоритм идентификации ситуации:

- osgEarth (отображение местности, 3D глобуса);
- GDAL (отображение растровых карт);
- OGR (отображение векторных карт);
- OpenSceneGraph (отображение 3D моделей объектов).

Алгоритм распознавания объекта (Рисунок 4.11) осуществляет поиск объекта в системе по идентификатору, создание нового объекта или изменение существующего, а также его отображение на местности. Отображение объекта производится специальной библиотекой, которая проверяет наличие 3D модели объекта на файловой системе или в локальной сети, загружает данную модель в оперативную память и в последствие отображает её на 3D глобусе по указанным оператором координатам и с указанным масштабом.

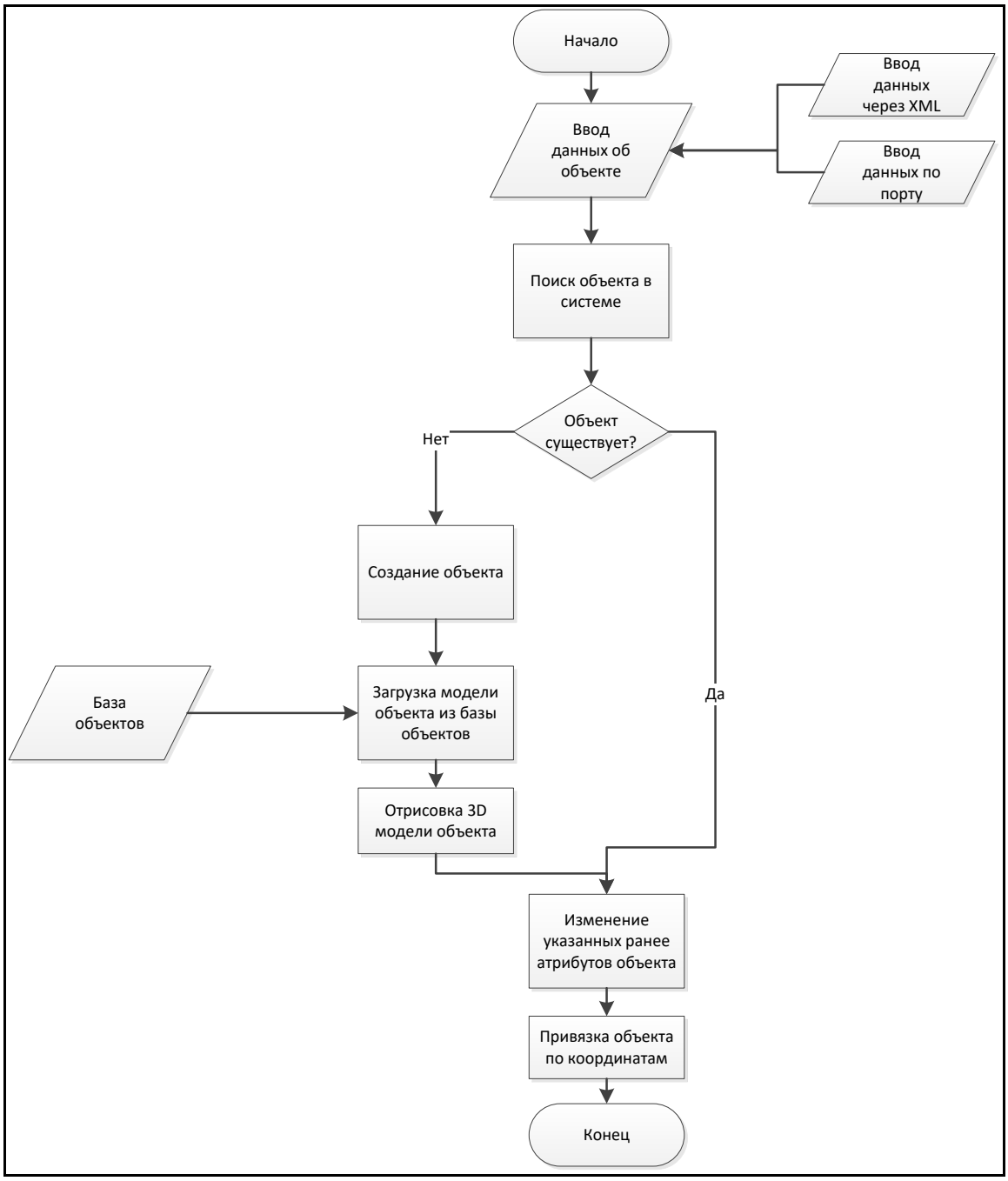

Рисунок 4.11. Алгоритм распознавания объекта

Алгоритм распознавания объекта входит в компонент работы с источниками обстановки базового ядра 3D ГИС и взаимодействует с такими ресурсами как: 3D модели объектов, принимающих участие в ситуации.

Характеристиками объектов являются: уникальный идентификатор объекта, координаты объекта на местности, путь к 3D модели объекта, масштаб 3D модели, дополнительные семантические атрибуты.

Режимы обстановки, в которых функционирует алгоритм распознавания объекта: морская, наземная, воздушная, смешанная.

Примеры свободно-распространяемых библиотек, с которыми взаимодействует алгоритм распознавания объекта:

- osgEarth (поиск и отображение объектов по координатам);
- OpenSceneGraph (загрузка 3D моделей объектов в оперативную память).

Алгоритм распознавания объекта функционирует в двух режимах считывания данных:

- считывание XML файла (в данном случае алгоритм с определенной периодичностью загружает содержимое файла в оперативную память, проверяет содержимое файла на изменения и при наличии изменений выполняет обновление обстановки);
- считывание UDP/TCP порта (алгоритм имеет постоянно доступный для передачи порт, при поступлении сообщения на который 3D ГИС в реальном времени отображает полученные изменения на местности).

Алгоритм привязки растрового изображения по координатам (Рисунок 4.12) описывает создание растрового слоя в формате GeoTiff из обычного изображения и его размещение на 3D глобусе. Алгоритм позволяет задавать координаты космическим снимкам форматов JPG, PNG, BMP и формировать на их основе карту формата GeoTiff с геопривязкой.

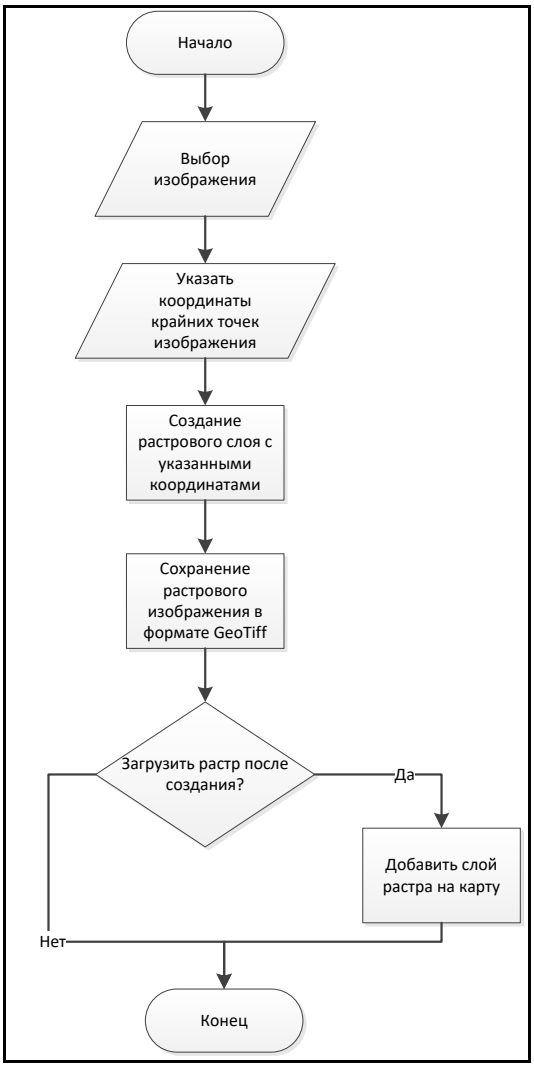

Рисунок 4.12. Алгоритм привязки растрового изображения по координатам

Алгоритм привязки растрового изображения по координатам входит в компонент работы с источниками обстановки базового ядра 3D ГИС и взаимодействует с такими ресурсами как: растровые карты формата GeoTiff и растровые изображения форматов PNG, JPG, BMP. [123]

Характеристиками выходной растровой карты являются: уникальный идентификатор карты, координаты левого верхнего края карты и правого нижнего края карты.

Режимы обстановки, в которых функционирует алгоритм привязки растрового изображения по координатам: морская, наземная, воздушная, смешанная.

Примеры свободно-распространяемых библиотек, с которыми взаимодействует алгоритм привязки растрового изображения по координатам:

- OpenSceneGraph (загрузка растровых изображений форматов JPG, PNG, BMP);
- osgEarth (привязка растровых изображений по координатам и генерация карты формата GeoTiff).

Алгоритм рисования фигуры на карте (Рисунок 4.13) описывает последовательность действий при рисовании фигур (точек, линий, полигонов) на 3D глобусе. Пользователь выбирает тип фигуры и при помощи курсора мыши последовательно указывает точки на глобусе, по которым в дальнейшем «строится» сама фигура. Дополнительным возможностями алгоритма являются: выбор цвета линий, выбор цвета заливки, выбор высоты (фиксированной или с учётом рельефа местности).

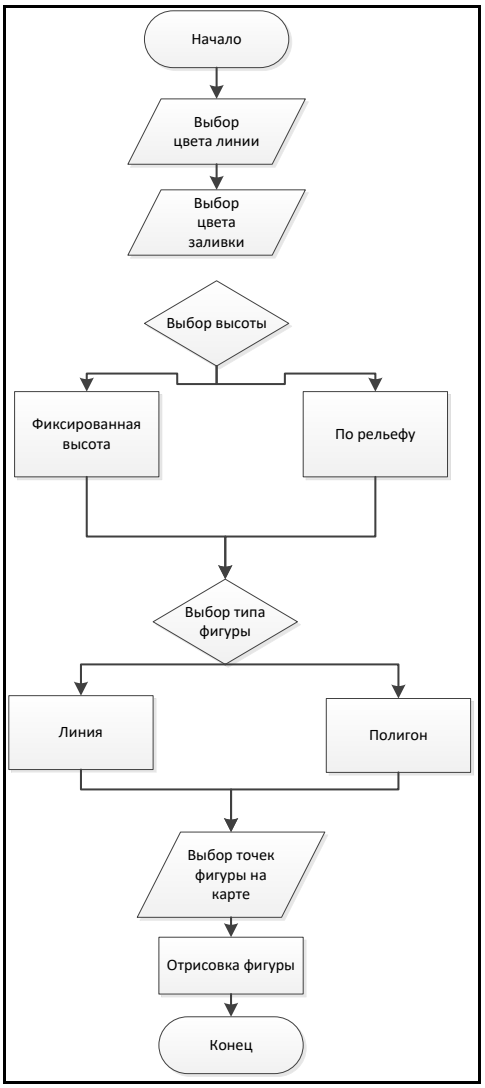

Рисунок 4.13. Алгоритм рисования фигуры на карте

Алгоритм рисования фигуры на карте входит в компонент 3D визуализации базового ядра 3D ГИС и взаимодействует с такими объектами как: точки, линии, полигоны.

Характеристиками фигуры являются: уникальный идентификатор фигуры, тип фигуры, последовательность точек с координатами.

Режимы обстановки, в которых функционирует алгоритм рисования фигуры на карте по координатам: морская, наземная, воздушная, смешанная.

Примеры свободно-распространяемых библиотек, с которыми взаимодействует алгоритм привязки растрового изображения по координатам:

- Qt (выбор точек и сохранение фигуры);
- osgEarth (отображение фигуры на 3D глобусе с учётом матрицы высот и глубин).

Алгоритм полёта камеры (Рисунок 4.13) осуществляет выбор координат для движения камеры и моделирует изменение камеры смотрящего по введённым координатам с апроксимацией перехода по ним для более плавного отображения.

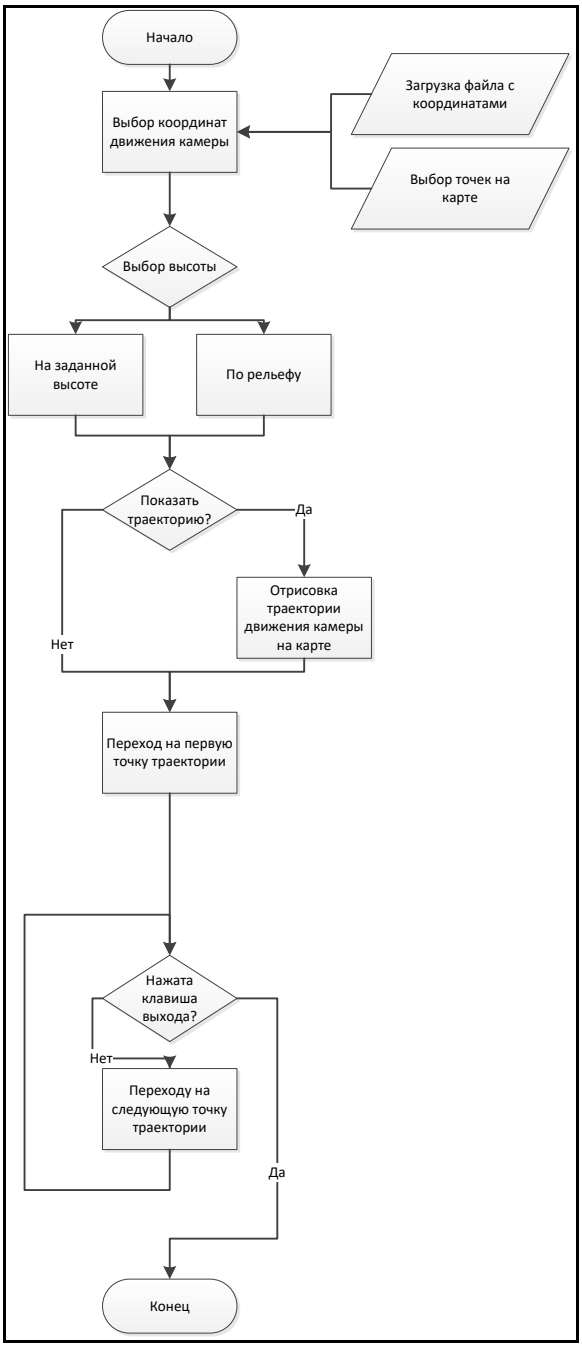

Рисунок 4.14. Алгоритм полёта камеры

Алгоритм полёта камеры входит в компонент имитации движения базового ядра 3D ГИС и взаимодействует с такими объектами как: матрица высот и глубин.

Характеристиками алгоритма являются: координаты точек для облёта, скорость перехода между точками с учётом апроксимации, отображение траектории облёта.

Режимы обстановки, в которых функционирует алгоритм полёта камеры: морская, наземная, воздушная, смешанная.

Примеры свободно-распространяемых библиотек, с которыми взаимодействует алгоритм полёта камеры:

- Qt (выбор точек или загрузка файла с настройками полёта камеры);
- osgEarth (плавный переход камеры между точками на 3D глобусе).

Алгоритм полёта камеры получает требуемые для функционирования данные двумя способами:

- из файла на файловой системе или в локальной сети (система загружает все характеристики, необходимые для полёта камеры, из выбранного файла и производит плавный переход между заданными точками);
- вручную (пользователь сам задаёт точки на 3D глобусе курсором мыши и активирует полёт камеры).

Алгоритм обмена информацией по UDP-порту (*Рисунок 4.15*) выполняет считывание входящих сообщений на заданный в системе UDP порт, производит идентификацию команды и отображает необходимые изменения в обстановке.

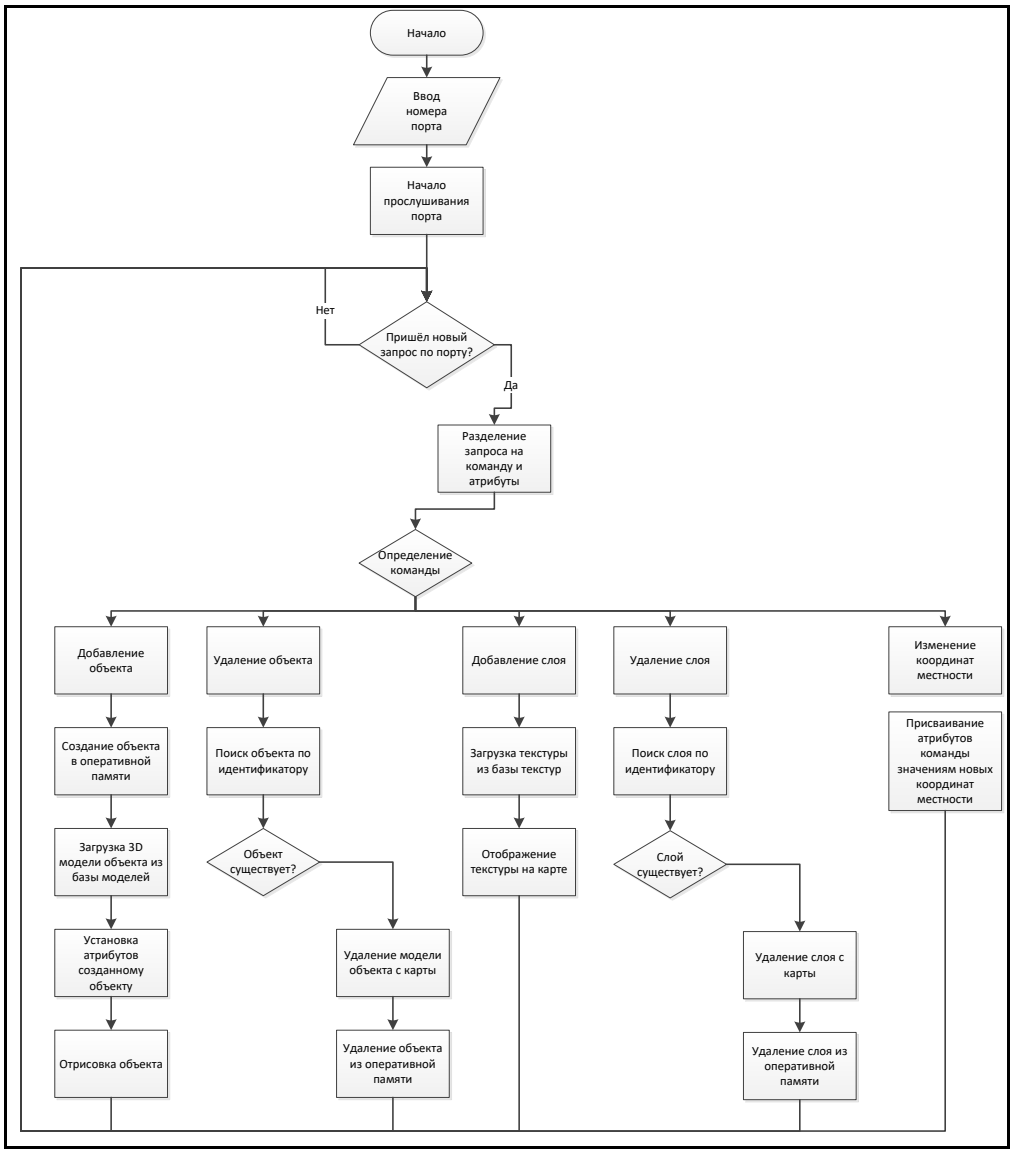

Рисунок 4.15. Алгоритм обмена информацией по UDP-порту

Алгоритм обмена информацией по UDP-порту входит в компонент взаимодействия с внешними системами базового ядра 3D ГИС.

Характеристиками алгоритма являются: входящая команда, её атрибуты.

Режимы обстановки, в которых функционирует алгоритм обмена информацией по UDP-порту: морская, наземная, воздушная, смешанная.

Примеры свободно-распространяемых библиотек, с которыми взаимодействует алгоритм обмена информацией по UDP-порту:

- Qt («прослушивание» порта, распознавание команды);
- OpenSceneGraph, osgEarth (отображение изменений на 3D глобусе).

Алгоритм обмена информацией по UDP-порту имеет следующий набор основных команд:

- добавление объекта (атрибуты: идентификатор объекта, 3D модель объекта, координаты объекта, имя объекта, семантические характеристики объекта);
- удаление объекта (атрибуты: идентификатор объекта);
- добавление слоя (атрибуты: идентификатор слоя, растровая или векторная карта, координаты карты, уровень вложенности, отображаемые слои);
- удаление слоя (атрибуты: идентификатор слоя);
- изменение координат местности (атрибуты: координаты местности, угол наклона камеры, угол поворота камеры).

Все алгоритмы функционирования 3D ГИС, включая общий алгоритм функционирования 3D ГИС, были программно реализованы на языке программирования C++ в среде разработки Qt и проверены на тестовых данных на корректное выполнение поставленных перед ними задач.

§ 4.4. Дополнительные возможности системы 3D моделирования проектных решений

Система 3D моделирования проектных решений предусматривает не только загрузку режимов отображения реального мира, но и ряд других полезных для визуального моделирования сервисов. К ним относятся:

1. Возможность визуализации семантических данных;
2. Возможность масштабирования местности;
3. Возможность интерактивного искажения;
4. Возможность окрашивания объектов.

Наборы визуализируемых семантических данных о трехмерном объекте представляют собой матрицы, в которых ряды являются данными, а колонки — атрибутами данных. При этом данные могут характеризоваться одним или несколькими атрибутами. Кроме того, сами данные

могут иметь более сложную структуру: иерархическую, текстовую, графическую и т.п. Таким образом, выделяют следующие виды данных, с которыми могут работать средства визуализации системы 3D моделирования проектных решений:

— одномерные данные — одномерные массивы, временные ряды и т. п.;
— двумерные данные — точки двумерных графиков, географические координаты и т. п.;
— многомерные данные — финансовые показатели, результаты экспериментов и т. п.;
— тексты и гипертексты —Web-документы и т. п.;
— иерархические и связанные — гиперссылки документов и т. п.

Одномерные данные обычно характеризуются одним атрибутом. Типичным примером одномерных данных являются хронологические данные (временные ряды), где единственным атрибутом является время. Необходимо обратить внимание, что с одной отметкой времени может быть связано одно или несколько значений данных. [6]

Двумерные данные имеют два отдельных измерения. Типичным примером могут служить географические данные, характеризующиеся двумя атрибутами: долгота и широта. Обычно такие данные отображаются в виде точек в двумерной системе координат. [6]

Кажущаяся простота визуализации одномерных и двумерных данных может быть обманчива. При большом объеме данных их визуализация может не дать желаемого эффекта. Например, при визуализации большого количества данных на длинном временном интервале отображение их на экран приведет к необходимости значительного уменьшения масштаба и, как следствие, потери наглядности. [6]

Многие наборы данных характеризуются более чем тремя измерениями, из-за чего просто отобразить их в виде двумерных или даже трехмерных точек невозможно. Примерами многомерных данных являются реляционные таблицы баз данных, где атрибутами являются колонки. Для отображения таких данных на экране монитора требуются специальные методы визуализации, например, параллельные координаты. [6]

Типы методов визуализации

Пользователь системы 3D моделирования проектных решений имеет возможность работы с моделями объектов (образами): видеть их с разных сторон, в разном масштабе и т. п. Для этого у него должны быть соответствующие возможности взаимодействия с образами:

— масштабирование образов;
— интерактивное искажение.

Масштабирование — это хорошо известный метод взаимодействия, используемый во многих приложениях. При работе с большим объемом данных этот метод хорош для представ-

ления данных в сжатом общем виде, и, в то же время, он предоставляет возможность отображения любой их части в более детальном виде. Масштабирование может заключаться не только в простом увеличении объектов, но в изменении их представления на разных уровнях. Так, например, на нижнем уровне объект может быть представлен пикселом, на более высоком уровне — неким визуальным образом, а на следующем — текстовой меткой. [6]

Метод интерактивного искажения поддерживает процесс исследования данных с помощью искажения масштаба данных при частичной детализации. Основная идея этого метода заключается в том, что часть данных отображается с высокой степенью детализации, а одновременно с этим остальные данные показываются с низким уровнем детализации. Наиболее известные методы — это гиперболическое и сферическое искажения, которые часто используются на иерархиях и графах, но могут применяться и в других визуальных образах. В системе 3D моделирования проектных решений метод интерактивного искажения позволяет уменьшить нагрузки на аппаратные средства при визуализации 3D моделей объектов за счёт изменения их уровней детализации. При визуализации объекта в малом масштабе – объект отображается в качестве метки, при увеличении масштаба у объекта загружаются текстуры низкого разрешения, и при более детальном рассмотрении – высокого разрешения. [6]

Окрашивание объектов

Окрашивание объектов в системе 3D моделирования проектных решений позволяет увеличить скорость распознавания ситуации и объектов, в ней участвующих, за счёт использования заданных цветовых характеристик как отдельных объектов, так и групп объектов с поясняющей легендой. Использование различных цветов помогает оператору быстрее оценить обстановку и определить, какие объекты являются гражданскими, какие военными, а какие представляют опасность. Дополнительной возможностью является окрашивание выбранного оператором объекта для последующего слежения за ним. [6]

Таким образом, в системе 3D моделирования кроме основных режимов отображения имеются дополнительные возможности, такие как: масштабирование объектов, интерактивное искажение, окрашивание объектов, позволяющие упростить взаимодействие оператора с системой и улучшить наглядность отображения обстановки для более быстрого принятия решения по ней.

Выводы

1. С помощью разработанного в диссертации CASE-средства создана система 3D моделирования проектных решений, которая позволяет визуально оценивать полученные проектные решения в большинстве режимов обстановки, встречаемых в реальном мире.
2. Предложены основные режимы функционирования системы 3D моделирования проектных решений, а также входящие в них трехмерные объекты, текстуры, матрицы высот и глубин, позволяющие наиболее адекватно отображать реальную обстановку и проводить визуальную оценку проектных решений.
3. Разработан комплекс алгоритмов функционирования 3D ГИС отображения ситуационной обстановки, включающий 13 алгоритмов, которые реализуют многорежимный приём и обработку данных, обеспечивают отображение морской, наземной и воздушной ситуационных обстановок с заданной точностью и могут быть использованы на практике разработчиками.

Глава 5. Тестирование, оценка, проведение испытаний системы 3D моделирования проектных решений

§ 5.1. Функциональные компоненты системы 3D моделирования проектных решений

Система 3D моделирования проектных решений морской, наземной и воздушной обстановки состоит из следующих элементов:

1) компонент 3D визуализации с пользовательским интерфейсом;
2) компонент работы с источниками обстановки;
3) компонент поддержки интерфейсов взаимодействия с внешними системами;
4) компонент имитации движения объектов в 3D пространстве. [43,44]

5.1.1. Компонент 3D визуализации с пользовательским интерфейсом

Компонент 3D визуализации с пользовательским интерфейсом имеет следующие возможности:

- построение трехмерной модели поверхности Земли в каркасном или текстурированном виде;
- регулирование пользователем скорости визуализации с использованием механизма уровней детализации объектов и текстурных элементов местности;
- масштабирование трехмерной модели с автоматической генерализацией картографических и трехмерных объектов;
- возможность навигации по трехмерной сцене в режиме реального времени по всем степеням свободы с возможностью масштабирования в точку интереса и изменения уровня горизонта;
- выбор режима перемещения по трёхмерной модели (облет, движение по рельефу, движение на заданной высоте/глубине);
- отображение в качестве слоя картографической подложки космических снимков высокого разрешения на всю территорию, трансформированных с учетом рельефа;
- отображение линейных, точечных и площадных объектов на поверхности;
- наложение на модель рельефа векторных и топографических основ, выполненных по соответствующей спецификации;
- поиск и выделение объектов по различным критериям отбора с центрированием камеры на результате поиска;
- фиксация участка местности в заданном масштабе для наблюдения изменения его обстанов-

ки в отдельном окне;

- отображение пути по рельефу местности и перемещение виртуальной камеры по нему;
- фильтрацию отображения слоев информации;
- реализацию всех функций с помощью пользовательского меню и удобных элементов управления;
- реализацию полноэкранного режима отображения обстановки (без границ окна);
- управление всеми параметрами отображения через программный интерфейс;
- запуск компонента с параметрами (центрирование камеры наблюдателя в определенной точке местности, заданной геокоординатами, высотой/глубиной и уровнем горизонта, с определенным набором слоев обстановки);
- настройка параметров приложения через интерфейс пользователя в пользовательском меню;
- реализация классификатора 3D моделей в виде панели со значками с учетом возможности разделения значков по категориям;
- нанесение 3D моделей перетаскиванием (выбором) на панели 3D моделей;
- нанесение вспомогательных линий и фигур с помощью редактора обстановки.

5.1.2. Компонент работы с источниками обстановки

Компонент работы с источниками обстановки имеет следующие возможности:

- открытие карт, текстур, моделей поверхности наиболее распространённых форматов: SXF, S-57(63), KML, OSM, GeoTiff, JPG, 3DS;
- привязка карт, текстур, моделей поверхности по координатам;
- формирование библиотеки 3D объектов на основе распространённых форматов 3D моделей (3DS и т.п.) с автоматическим формированием панели редактора;
- хранение в упорядоченном виде библиотек, импортированных карт, текстур, моделей поверхности, 3D объектов в базе данных или на файловой системе с возможностью пакетного переноса на другой компьютер;
- кеширование данных от интернет-источников (OGC-сервисы) для использования в автономном режиме.

В состав компонента входят:

- база данных 3D объектов;
- набор карт, текстур, моделей поверхности различных масштабов и степеней детализации.

5.1.3. Компонент поддержки интерфейсов взаимодействия с внешними системами

Компонент поддержки интерфейсов взаимодействия с внешними системами имеет следующие возможности:

- автоматизированное нанесение 3D обстановки через программные интерфейсы;
- автоматизированное изменение параметров 3D обстановки (координат объектов, направления и др. свойств);
- открытие, редактирование слоев обстановки автоматизированным способом через программные интерфейсы;
- получение информации по объектам 3D обстановки из внешних источников (БД, внешние системы);
- вывод справки об интересуемом объекте, включающей:
    - наименование объекта;
    - фотографии объекта;
    - просмотр пространственно-координатного описания объекта;
    - просмотр семантических характеристик объекта;
    - просмотр информации о слое, к которому принадлежит объект;
    - просмотр геометрии объекта в виде каркасной модели;
- центрирование камеры наблюдателя в определенную точку местности, заданную геокоординатами, высотой/глубиной и уровнем горизонта.

5.1.4. Компонент имитации движения объектов в 3D пространстве

Компонент имитации движения объектов имеет следующие возможности:

- задание траекторий, скоростей движения объектов в 3D пространстве с помощью редактора через пользовательский интерфейс;
- создание, сохранение, загрузка, воспроизведение сценариев движения объектов в 3D пространстве;
- моделирование динамики перемещения объектов по трехмерной модели.

§ 5.2. Форматы карт в системе 3D моделирования проектных решений

Разработанная система 3D моделирования позволяет работать с форматами KML, OSM, GeoTiff, JPG, 3DS, SXF, S57. [56]

Формат SXF - открытый формат цифровой информации о местности, который предназначен для применения в геоинформационных системах и хранения цифровой информации о местности, обмена данными между различными системами, создания цифровых и электронных карт и решения прикладных задач.

Форма записи в формате:

- двоичная;
- текстовая.

Основной инструмент создания электронных карт в формате SXF — ГИС «Карта 2008» и ГИС «Карта 2011» (ЗАО КБ «Панорама»).

В формате SXF осуществляется создание и хранение цифровой картографической продукции подразделениями Росреестра, в том числе цифровых навигационных карт по ФЦП «ГЛОНАСС».

Формат S57 используется при стандартизации обмена гидрографическими данными между Гидрографическими Службами, Агентствами, производителями картографической продукции и ECDIS-систем.

Согласно S-57, гидрографическая информация структурируется в наборы данных, которые, в свою очередь, могут объединяться в наборы обмена. Набор данных S-57 может рассматриваться как объектно-ориентированная база данных, подчиняющаяся перечисленным в стандарте семантическим правилам (объекты, атрибуты, связи между ними и т.д.) и записанная (закодированная) в соответствии с описанным в стандарте синтаксисом. [101,102,103]

Для конвертации карт из формата S57 в другие распространенные форматы используется библиотека GDAL.

Семантика стандарта опирается на то, что любой картографический объект обладает, как пространственно-геометрическими, так и функционально-описательными свойствами. В соответствии с этим карта S-57 состоит из двух типов объектов: пространственных (spatial) и описательных (feature). Spatial объекты (например, node - узел, edge - сегмент, face - площадь), характеризуются координатами, задающими их местоположение на поверхности Земли. Feature объекты, обладают определенным набором атрибутов и описывают некий естественный или искусственный предмет, например: LNDARE - область суши, DEPARE - область глубин, BOYCAR - кардинальный буй и т.д. Между объектами могут существовать связи различного типа, позволяющие смоделировать сколь угодно сложную сущность реального мира. Т.к. количество нестандартных объектов достаточно велико, то их семантика и описание должны иметь программную поддержку для отображения в разрабатываемой системе. Отсюда, для открытия карт в формате S57 была необходимость разработки и ввода в состав системы специализированные средства программной поддержки. [54,55,66]

§ 5.3. Оценка адекватности проектного решения на основе его визуального отображения

Большую часть информации об окружающем мире человек получает с помощью зрения, поэтому представление данных в виде визуальных объектов, а не атрибутивных значений способно существенно увеличить скорость их понимания.

В ходе работы с полученной информацией возникают трудности, обусловленные следующими факторами:

- поступающая информация не всегда является достаточной для однозначной оценки ситуации;
- сложность данных требует увеличения времени для их обработки;
- необходимо выявлять и анализировать противоречивые, неточные или ошибочные данные.

В течение длительного времени основными направлениями при решении указанных проблем являлись совершенствование вычислительных методов и увеличение мощности компьютерной и измерительной техники. Это привело к тому, что наиболее медленным звеном в цепи «исходные данные - решение» стало взаимодействие «компьютер - человек». Одним из способов преодоления сложностей, возникающих на этом этапе, является представление данных в визуальной форме.

Примером обработки больших объемов сложной числовой информации является моделирование геологических пространств по имеющимся геолого-геофизическим данным. Современные методы визуализации позволяют создавать модели пространственных объектов, максимально точно отражающие большое количество разнородной исходной информации. Таким образом, существует возможность построения визуальных моделей сложных реальных трехмерных пространственных объектов, включающих в качестве характеристик большой набор данных, необходимых для оценки интересующих наблюдателя свойств и состояний.

Важной особенностью трехмерной модели является возможность ее детального исследования. С одной стороны, такая модель содержит огромный объем информации, доступной для сравнительного наблюдения в едином визуальном пространстве, что позволяет в несколько раз увеличить скорость и качество анализа данных. С другой стороны, имеется возможность изучения пространственных характеристик объекта моделирования под любым углом зрения, что является очевидным преимуществом по сравнению с традиционными способами построения картографических проекций.

Такие модели являются статическими. Однако при создании сложных визуальных образов имеется возможность включать в их описание изменяющиеся характеристики. Получаемые таким образом динамические модели являются эффективным инструментом анализа динамики и диапазонов изменения значительного количества величин.

Визуальный анализ является эффективным и привычным методом, широко применяемым для исследования экспериментальных, расчетных и даже абстрактных данных. Уровень развития технических средств визуализации в настоящее время позволяет создавать искусственные изображения высокой степени сложности, приближенные, при необходимости, к фотореалистичному восприятию. Эти обстоятельства являются условиями, необходимыми для построения визуальных моделей данных, интерпретация которых использует и учитывает когнитивные процессы, свойственные зрительному восприятию человека. [48]

Перечисленные выше условия построения визуальных моделей данных позволяют значительно повысить эффективность моделирования трехмерных ситуационных обстановок и принятия на их основе качественных решений.

Система 3D моделирования проектных решений выполняет отображение (F) множества проектных решений ($PR = \{p_1, p_2, \ldots, p_n\}$) во множество отображаемых объектов и ситуаций ($SO = \{s_1, s_2, \ldots, s_n\}$). Это отображение является инъективным, т.е. для любых $p_1, p_2 \in PR$ из неравенства $p_1 \neq p_2$ следует неравенство $F(p_1) \neq F(p_2)$.

После отображения ситуаций и объектов обстановки система 3D моделирования проектных решений оценивает адекватность отображения, т.е. соответствие проектных решений множеству отображаемых объектов и ситуаций (Рисунок 5.1).

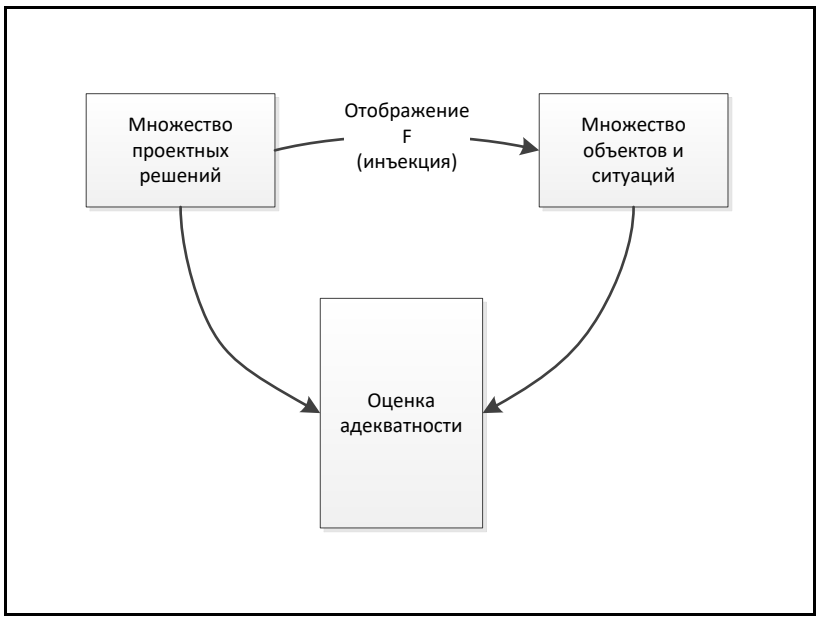

Рисунок 5.1. Система 3D моделирования проектных решений

У каждого проектного решения в системе 3D моделирования есть набор критериев, по которым производится оценка его визуализации. Для расчёта адекватности проектного решения предлагается использовать интервальную оценку по каждому из критериев. Для этого экспертным путём задаются минимальные и максимальные значения каждого критерия и определяется, входит ли значение критерия в заданный интервал или имеет отклонение (∆).

В реальных условиях изменяющейся обстановки необходимо принять качественное решение по тому или иному вопросу. Для получения качественного решения требуется устранить все неопределенности в обстановке, которые возникают по причинам:

- неточного определения геокоординат места события;
- неточного определения размеров объекта;
- неточного определения местоположения объекта;
- неточного определения расстояния между двумя объектами;
- неточного определения высоты (глубины) расположения объекта;
- неточного определения скорости движения объекта;
- плохого качества текстур и рельефа;
- работы со старыми картами;
- плохой освещённости;
- низкой цветопередачи. [20]

Все неточности влияют на качество принимаемых решений в различных ситуационных обстановках:

- военной обстановки;
- гражданской обстановки;
- чрезвычайной ситуации;
- экологической обстановки;
- геологоразведывательной обстановки;
- метеорологической обстановки;
- и др.

Неопределенность в той или иной ситуации отрицательно влияет на скорость и качество принимаемых решений. Уменьшение неопределенности для принятия решения позволяет сократить:

- количество чрезвычайных ситуаций;
- урон от природных катаклизмов;
- количество автомобильных, железнодорожных и авиационных катастроф.

Из всего этого возникает необходимость формирования критериев принятия решений по трёхмерной обстановке, которые одновременно являются критериями оценки адекватности формируемого CASE-средством проектного решения:

- скорость отображения новых объектов и изменения существующих (K_1), эффективное принятие решений при боевых действиях, чрезвычайных ситуациях, возможных авиационных столкновениях;
- масштаб объекта, т.е. отношение реальных размеров объекта к размерам модели этого объекта на местности (K_2), эффективное принятие решений при тушении пожаров, определении траекторий пересечения объектов, геологоразведке;
- скорость интерпретации - величина обратная времени, необходимому наблюдателю для понимания смысла образа исследуемых данных с точностью, определяемой условиями задачи визуализации (K_3);
- группирование объектов (радиус, в пределах которого объекты группируются) (K_4), определение местности и объектов, подверженных оползням, распространению пожаров, наводнений;
- время идентификации объекта (K_5), скорость обработки поступающей информации и принятия решения по ней;
- время идентификации ситуации (K_6), скорость обработки поступающей информации и принятия решения по ней;
- разрешение моделей объектов (K_7), распознавание типов и моделей самолетов, танков, автомобилей, морских судов;
- глубина цвета моделей объектов, текстур (K_8), скорость определения ситуации, объектов, участвующих в ней. [20]

Указанные критерии взаимодействуют друг с другом и в итоге влияют на скорость интерпретации обстановки пользователем в целом.

Граф взаимодействия критериев представлен на рисунке 5.2. Стрелками обозначены направления влияния критериев друг на друга.

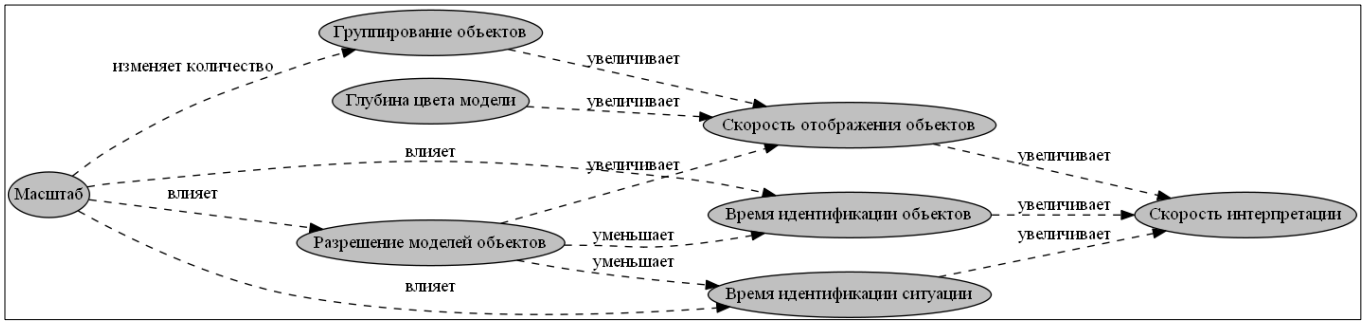

Рисунок 5.2. Граф взаимодействия критериев для оценки адекватности проектного решения

На основе анализа существующих 3D ГИС (ArcGIS, Quantum GIS, GRASS, ГИС «Панорама»), а также собственных разработок, моделирования и тестирования таких систем, как: 3D ГИС отображения морской, наземной и воздушной обстановок, для каждого критерия предложены следующие интервалы соответствия:

K_1 = [0; 5] сек.;

K_2 = [0,1; 10] раз;

K_3 = [0; 4] сек.;

K_4 = [0; 50] пикселей;

K_5 = [0; 3] сек.;

K_6 = [0; 3] сек.;

K_7 = [600x400; 1920x1080] пикселей;

K_8 = [8; 32] бит.

По всем критериям проектного решения строится вектор несоответствий, в который для каждого критерия записывается 0, если значение критерия входит в заданный интервал, либо разность по модулю между текущим значением критерия и ближайшим минимальным или максимальным значением интервала (1).

$$\Delta\alpha_i = \begin{cases} 0, \text{если } \alpha_i \in K_i \\ |\alpha_i - \max(K_i)|, \text{если } \alpha_i > \max(K_i) \\ |\min(K_i) - \alpha_i|, \text{если } \alpha_i < \min(K_i) \end{cases} \quad (5.1)$$

В системе 3D моделирования имеется блок оценки адекватности проектных решений (Рисунок 5.3), в котором реализована оценка адекватности по представленным выше критериям. В случае необходимости доработки проектного решения полученный вектор несоответствий вместе с самим проектным решением поступают на вход CASE-средства, в котором с учётом полученных данных модифицируются выбранные в проектном решении библиотеки, либо подбираются другие библиотеки, более подходящие под заданные требования. В случае отсутствия подходящей библиотеки CASE-средство предлагает замену на собственную разработку. В зависимости от входных данных, поступающих из системы 3D моделирования проект-

ных решений, CASE-средство дополнительно изменяет содержимое базы объектов и ситуаций. После модификации всех необходимых элементов CASE-средство формирует новые проектные решения, которые повторно оцениваются в системе 3D моделирования.

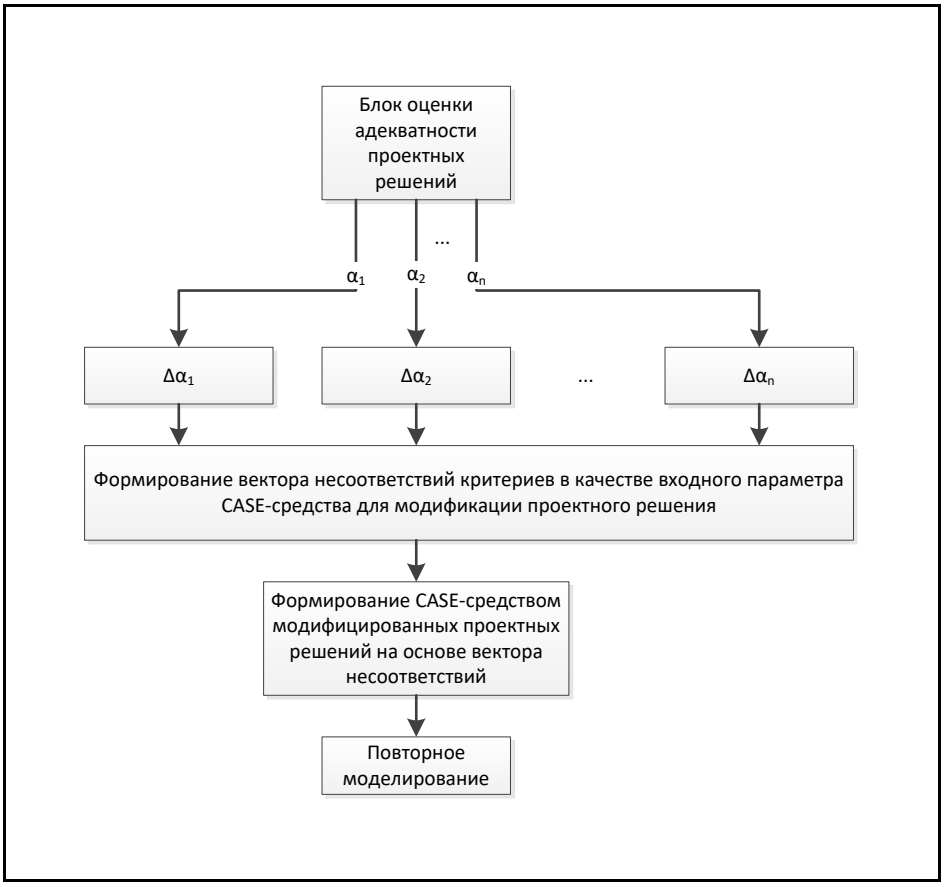

Рисунок 5.3. Блок оценки адекватности проектного решения

После формирования вектора несоответствий рассчитывается общий коэффициент несоответствия проектного решения по формуле (2).

$$Q = \sum w_i * \Delta\alpha_i, \quad (5.2)$$

где w_i – вес i-ого критерия, заданный экспертным путём на этапе проектирования.

Рассмотрим более подробно алгоритм функционирования блока оценки адекватности проектного решения (Рисунок 5.4). На первом этапе проводится последовательная проверка вхождения значений каждого критерия в заданный для него интервал соответствия.

Интервал соответствия – это интервал конкретного критерия, в пределах которого значение этого критерия для проектного решения означает его адекватность.

В случае вхождения значения в интервал алгоритм переходит к проверке следующего критерия. В противном случае вычисляется значение несоответствия (Δ), которое записывается в оперативную память. Далее происходит переход к оценке следующего критерия.

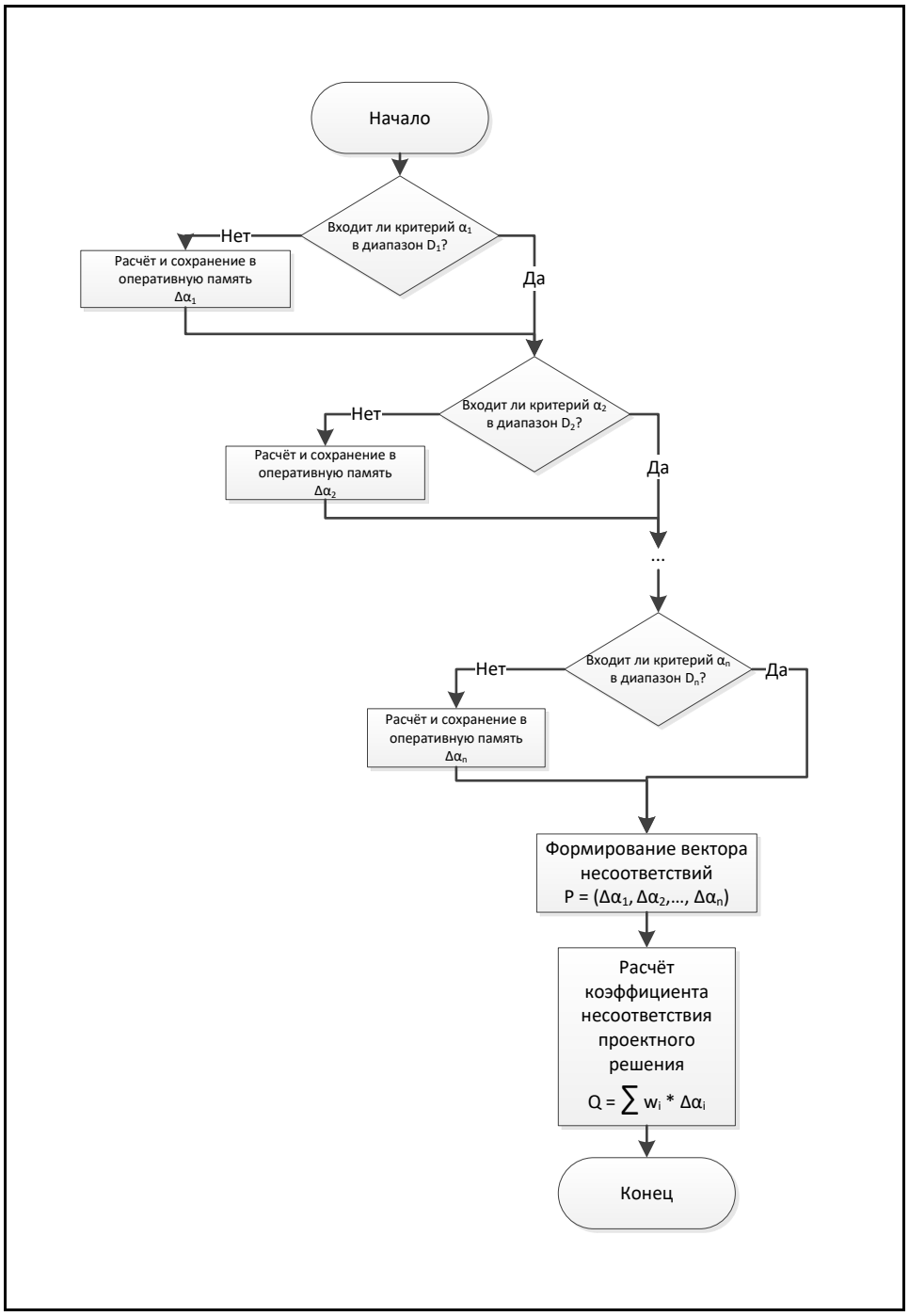

Рисунок 5.4. Алгоритм функционирования блока оценки адекватности проектного решения

После проверки всех критериев алгоритм формирует вектор несоответствий из сохранённых в оперативную память значений Δ критериев. На основе полученного вектора вычисляется общий коэффициент несоответствия проектного решения. Если этот коэффициент не входит в заданный заранее экспертами интервал, то проектное решение не прошло проверку и рекомендуется его дальнейшая доработка. В данном случае, вектор несоответствий и само проектное решение (в которое входят: функциональная, структурная, брокерная, композиционная модели, схема описания обстановки, проекций библиотек, диаграммы классов библиотек, классов объектов и ситуаций, ERD диаграммы, а также алгоритмы функционирования, исходные

коды файлов заголовков, библиотеки и собственные разработки) передаются в качестве входных параметров в блок модификаций CASE-средства для дальнейшей доработки. [20]

Отклонение каждого критерия в векторе несоответствий определяет, в каких именно библиотеках необходимо внести уточнения или полностью их заменить. Например, отклонение критерия «Группирование объектов» указывает на то, что библиотека отображения 3D моделей объектов не производит автоматическую группировку этих моделей, либо имеет малый радиус группировки. Примером такой библиотеки может быть библиотека OpenSceneGraph, в которой имеется возможность задания диапазона группируемых значений в метрах. В случае большого отклонения критерия CASE-средство произведёт замену предложенной библиотеки на библиотеку, решающую аналогичную задачу, либо предложит создать собственную разработку.

После формирования скорректированного проектного решения производится его оценка в системе 3D моделирования. Если проектное решение снова не удовлетворяет качеству, то возможна повторная его доработка, в противном случае процесс моделирования завершается и можно приступать к программной реализации корректного проектного решения.

На основании заданных интервалов критериев, а также рассчитанных значений критериев рассматриваемого проектного решения можно построить лепестковую диаграмму критериев оценки адекватности проектного решения (Рисунок 5.5). Если при построении значений критериев проектного решения его пространство не выходит за границы пространства критериев данной диаграммы, то проектное решение будет считаться адекватным. В противном случае проектное решение рекомендуется модифицировать.

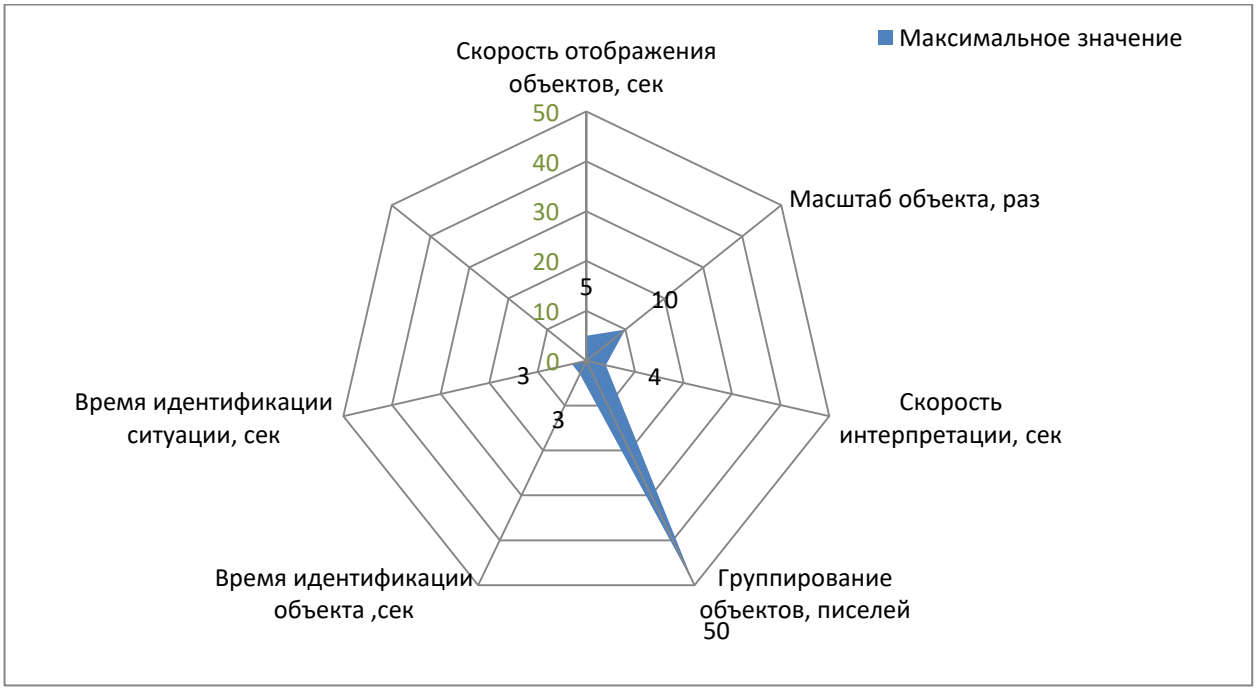

Рисунок 5.5. Диаграмма отображения пространства критериев адекватного проектного решения

В качестве примера оценки адекватности проектного решения рассмотрим оценку проектного решения 3D ГИС отображения ситуационной обстановки.

В проектное решение входят следующие библиотеки:

- OpenSceneGraph;
- osgEarth;
- OpenGL;
- GDAL;
- OGR;
- Qt;
- MySQL;
- CURL;
- CMake;
- GEOS;
- Minizip.

Режимы отображения обстановки:

- морской режим (объекты: морские суда, подводные лодки, морские препятствия, береговая линия);
- воздушный режим (объекты: самолеты, вертолеты, ракеты, птицы);
- наземный режим (объекты: деревья, дома, дороги, мосты, горы, реки, пустыни, транспорт, люди, трубопроводы, ЛЭП);
- смешанный режим (все объекты из морского, воздушного, наземного режимов).

В ходе оценки проектного решения были определены значения всех критериев:

K_1 = 2 сек.;

K_2 = 5 раз;

K_3 = 2 сек.;

K_4 = 30 пикселей;

K_5 = 1,5 сек.;

K_6 = 2,5 сек.;

K_7 = 1024x768 пикселей;

K_8 = 32 бита.

Построим диаграмму отображения пространства критериев проектного решения 3D ГИС и пространства критериев адекватного проектного решения (Рисунок 5.6). На диаграмме видно,

что пространство критериев проектного решения полностью входит в пространство максимальных значений, из чего следует, что проектное решение адекватное и не нуждается в доработке.

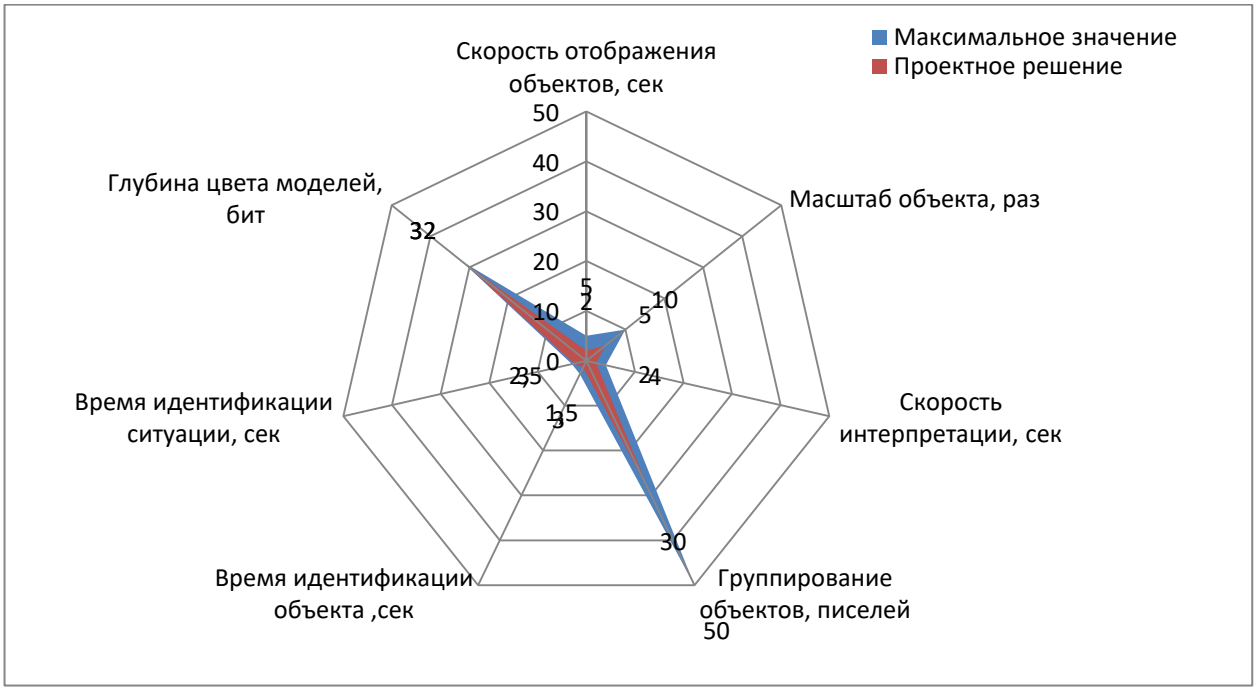

Рисунок 5.6. Диаграмма пространства критериев проектрого решения 3D ГИС

В связи с тем, что проектное решение полностью удовлетворяет заданным критериям, его коэффициент несоответствия равен нулю.

После проведения всех оценок проектное решение передаётся разработчикам для его программной реализации.

§ 5.4. Методы испытаний системы 3D моделирования проектных решений

Компонент 3D визуализации оценивается по возможности выборки и отображения района на 3D глобусе Земли с данными высот, глубин, 3D объектов, текстур местности.

Компонент работы с источниками обстановки оценивается по базам текстур, высот и глубин, 3D объектов.

База текстур оценивается по набору текстур различных масштабов и степеней детализации, используемых при визуализации выбранного района.

База высот и глубин оценивается по набору файлов высот и глубин, обеспечивающих возможность оценки высоты суши и глубины морского дна в метрах.

База 3D объектов оценивается по наличию 3D объектов различных типов: наземных, надводных, подводных, воздушных в количестве, достаточном для проведения экспериментов. Каждый объект должен сопровождаться атрибутикой, включающей в себя описание, метрику, идентификаторы, тип и др. [35]

Последовательность проверки компонентов изложена в таблице 5.1.

Таблица 5.1. Проверка компонента 3D – визуализации с пользовательским интерфейсом и компонента работы с источниками обстановки

Наименование этапов проверки	**Порядок действий**	**Результат выполнения**
Компонент 3D-визуализации с пользовательским интерфейсом		
построение трехмерной модели поверхности Земли в каркасном или текстурированном виде	Переключение вида выполняется нажатием на панели инструментов на кнопку «Каркасный/текстурированный вид»	На экране происходит построение трехмерной модели поверхности Земли в выбранном (каркасном или текстурированном) виде
регулирование пользователем скорости визуализации с использованием механизма уровней детализации объектов и текстурных элементов местности	В подпункте «Настройки» пункта «Файл» верхнего меню в выпадающем списке выбрать скорость детализации «низкая/высокая»	На экране происходит обновлении текстур в соответствии с указанной скоростью визуализации
масштабирование трехмерной модели с автоматической генерализацией картографических и трехмерных объектов	Масштабирование трехмерной модели происходит вращением колесиком мыши	На экране происходит масштабирование трехмерной модели
возможность навигации по трехмерной сцене в режиме реального времени по всем степеням свободы с возможностью масштабирования в точку интереса и изменения уровня горизонта	Для навигации по карте необходимо воспользоваться соответствующей кнопочной панелью в правом нижнем углу экрана либо манипулятором «мышки».	На экране осуществляется навигация по трехмерной сцене в режиме реального времени
возможность отображения в качестве слоя картографической подложки космические снимки высокого разрешения на всю территорию, трансформированные с учетом рельефа	Для отображения в подпункте «Добавить слой» пункта «Правка» верхнего меню выбрать добавляемый слой	На экране появляется выбранной слой
возможность отображения линейных, точечных и площадных объектов на поверхности;	Для отображения линейных, точечных и площадных объектов необходимо в XML файле прописать новый объект	На экране появляется новый линейный, точечный или площадной объект
возможность наложения на модель рельефа векторных и топографических основ, выполненных по соответствующей спецификации	Для отображения в подпункте «Добавить векторный слой» пункта «Правка» верхнего меню выбрать добавляемый слой	На экране появляется выбранной слой
возможность поиска и выделения объектов по различным критериям отбора с центрированием камеры на результате поиска	Для поиска и выделения объектов необходимо воспользоваться поисковой строкой, расположенной на панели инструментов.	Центрирование камеры в объект реализовано плавным отъездом камеры от текущей позиции до обзорного уровня местности и перемещением ее в точку с найденным объектом с последующим приближением до уровня четкой узнаваемости объекта

возможность фиксации участка местности в заданном масштабе для наблюдения изменения его обстановки в отдельном окне	Для фиксации местности необходимо воспользоваться кнопкой «В отдельном окне» на верхней панели инструментов	Открывается отдельное окно
фильтрация отображения слоев информации	Для фильтрации слоёв информации необходимо нажать на панели инструментов на кнопку «Скрыть/Показать» для необходимых слоёв	На экране скрывается/отображается выбранный слой
реализация всех функций с помощью пользовательского меню и удобных элементов управления	Для управления системой необходимо воспользоваться функциями, размещенными на панели инструментов	На экране происходит действие по выбранному пункту
реализация полноэкранного режима отображения обстановки (без границ окна)	Для перехода в полноэкранный режим необходимо воспользоваться кнопкой «На весь экран» на верхней панели инструментов	Система переходит в полноэкранный режим
возможность управления всеми параметрами отображения через программный интерфейс	Для управления всеми параметрами отображения необходимо воспользоваться XML файлом	На экране применяются указанные параметры отображения
возможность запуска компонента с параметрами (центрирование камеры наблюдателя в определенной точке местности, заданной геокоординатами, высотой/глубиной и уровнем горизонта, с определенным набором слоев обстановки)	Для запуска компонента с параметрами необходимо указать дополнительные параметры в командной строке после бинарного файла системы	Запускается компонент с указанными параметрами
настройка параметров приложения через интерфейс пользователя в пользовательском меню	Настройка параметров происходит в диалоговом окне «Настройки», вызываемом в подменю «Настройки» меню «Файл» верхнего меню	На экране происходит применение измененных настроек
реализация классификатора 3D моделей в виде панели со значками с учетом возможности разделения значков по категориям	Вызов классификатора происходит через подменю «Классификатор» меню «Файл» верхнего меню	На экране открывается диалоговое окно классификатора
Компонент работы с источниками обстановки		
возможность открытия карт, текстур, моделей поверхности наиболее распространённых форматов: SXF, S-57(63), KML, OSM, GeoTiff, JPG, 3DS;	Для отображения в подпункте «Добавить слой» пункта «Правка» верхнего меню выбрать добавляемый слой в формате KML, OSM, GeoTiff, JPG, 3DS	На выбранный район добавляется указанный в диалоговом окне слой.
возможность привязки карт, текстур, моделей поверхности по координатам	Процедура привязки карт, текстур и моделей поверхности по координатам реализована посредством использования библиотеки GDAL, используя команду gdal_translate.	Результатом команды будет привязанная текстура, карта или модель поверхности в указанной директории.

возможность формирования библиотеки 3D объектов на основе распространённых форматов 3D моделей (3DS и т.п.) с автоматическим формированием панели редактора	Формирование библиотеки 3D объектов на основе распространённых форматов реализовано средствами файловой системы.	Панель управления автоматически формируется считыванием конкретной директории, указанной в настройках 3D ГИС системы. В данной директории осуществляется поиск файлов по маске “*.3DS “.
хранение в упорядоченном виде библиотек импортированных карт, текстур, моделей поверхности, 3D объектов в базе данных или на файловой системе с возможностью пакетного переноса на другой компьютер (например, за счет использование индексных файлов, файлов настроек);	Указать директории библиотек в меню настроек 3D ГИС системы, и поместить файлы карт, текстур, моделей поверхности, 3D объектов в указанные директории	Библиотеки импортированных карт, текстур, моделей поверхности, 3D объектов хранятся в соответствующих директориях на файловой системе.
возможность кеширования данных от интернет-источников (OGC-сервисы) для использования в автономном режиме	Указать директорию для кеширования в меню настроек 3D ГИС системы	3D ГИС система будет кэшировать данные от интернет-источников в директорию, указанную в настройках системы. При последующем запуске система автоматически загружает сохранённые cache-файлы.
база данных 3D объектов (не менее 50 значков основных моделей, состоящей на вооружении в настоящее время надводной, подводной и воздушной техники);	Обратиться в директорию хранения 3D объектов указанной в настройках системы.	Убедиться в наличии в указанной директории не менее 50 файлов в формате *.3ds
набор карт, текстур, моделей поверхности для выбранного района различных масштабов и степеней детализации	Обратиться в директорию хранения набор карт, текстур, моделей поверхности указанной в настройках системы.	Убедиться в наличии в указанной директории набора карт, текстур и моделей поверхности
3D-модели должны создаваться на основе фотографий с таким качеством, которое должно обеспечивать однозначную узнаваемость модели. Готовые трехмерные модели объектов должны быть покрыты фотореалистичными текстурами.	Обратиться в директорию хранения 3D объектов указанной в настройках системы и открыть файлы в формате *.3ds в просмоторщике 3D моделей	Убедиться в фотореалистичности используемых моделей
База текстур должна содержать космические снимки различного разрешения для разных уровней детализации	Обратиться в директорию хранения набор карт, текстур, моделей поверхности указанной в настройках системы.	Убедиться в наличии в указанной директории набора космических снимков различного разрешения для разных уровней детализации.
Формат используемых в макете текстур и высотных/глубинных моделей поверхности должен быть совместим с форматами используемого 3D –движка.	Загрузить текстуры и высотно-глубинные модели поверхности в систему.	Убедиться в том, что система поддерживает загруженные текстуры и высотно-глубинные модели поверхности

Компонент поддержки интерфейсов взаимодействия с внешними системами		
Возможность автоматизированного нанесения 3D обстановки через программные интерфейсы.	Добавить запись о новом слое 3D обстановки в XML файл	В системе автоматически появляется новый слой 3D обстановки
Возможность автоматизированного изменения параметров 3D обстановки (координат объектов, направления и др. свойств)	Внести изменения в параметры редактируемых слоёв 3D обстановки в XML файле	В системе автоматически отображаются внесенные изменения
Возможность открытия, редактирования слоев обстановки автоматизированным способом через программные интерфейсы	Добавить запись о новом слое или внести изменения в существующем в XML файле	Система автоматически обновит 3D обстановку
Возможность получения информации по объектам 3D обстановки из внешних источников (БД, внешние системы)	Вся информация об объектах извлекается из внешней БД и сохраняется в XML файл.	Система автоматически реагирует на изменения в XML файле
Возможность вывода справки об интересуемом объекте	Вся атрибутика интересуемого объекта хранится в XML файле	При нажатии на объект открывается окно с выведенной атрибутикой
Возможность центрирования камеры наблюдателя в определенную точку местности, заданную геокоординатами, высотой/глубиной и уровнем горизонта	Точка местности, заданная геокоординатами, высотой, глубиной и уровнем горизонта хранится в XML файле	При запуске системы автоматически происходит переход на данную точку местности
Компонент имитации движения объектов в 3D пространстве		
Возможность задания траекторий, скоростей движения объектов в 3D пространстве с помощью редактора через пользовательский интерфейс	Выбрать на карте 3D объект, открыть окно информации кликом по нему и в появившемся окне нажать на кнопку «Траектория». В новом окне в таблице задать траектории движения	В системе для выбранного объекта сохраняются указанные траектории
Возможность создания, сохранения, загрузки, воспроизведения сценариев движения объектов в 3D пространстве	В диалоговом окне «Траектория» окна информации об объекте есть возможность загрузки и сохранения в файл сценариев движения объекта	В системе автоматически добавляется новый сценарий движения объекта
Возможность моделирования динамики перемещения объектов по трехмерной модели	Задать в диалоговом окне «Траектория» окна информации об объекте сценарий движения объекта	В системе автоматически начинается имитация движения объекта на карте

Входные данные для проведения испытания заданы в формате XML.

По результатам проводимых испытаний составляется заключение о функционировании проектных решений и верификации программного обеспечения.

Система 3D моделирования проектных решений функционирует в Linux-подобных операционных системах (с версией ядра не ниже 2.4) на вычислительных средствах архитектуры

Intel с возможностью полной перекомпиляции для функционирования в операционной системе Windows.

Программное обеспечение включает в себя язык программирования C++ с использованием библиотек Qt 4 и OpenGL.

Испытания проводились на технических средствах отладочно-моделирующего испытательного комплекса (ОМИК).

Минимальный состав необходимых технических средств:

1. Компьютер на базе архитектуры Intel.
2. Видеокарта с памятью не менее 2 ГБ
3. Оперативная память размером не менее 6 ГБ

Для нормального функционирования системы 3D моделирования проектных решений было установлено и настроено следующее программное обеспечение:

- операционная система семейства Linux;
- язык программирования C++ с использованием библиотек Qt 4 и OpenGL;
- база текстур;
- база высот и глубин;
- база 3D объектов;
- библиотека OpenSceneGraph
- библиотека OpenGL
- библиотека GDAL
- библотека osgEarth.

§ 5.5. Разработка системы 3D моделирования проектных решений

5.5.1. Разработка компонента 3D визуализации с пользовательским интерфейсом

Компонент (подсистема) 3D визуализации содержит основные функции по отображению трехмерной обстановки Земли. В частности, выполняются такие функции как:

- построение трехмерной модели поверхности Земли (3D глобуса);
- отображение векторных и растровых карт на поверхности Земли;
- отображение 3D моделей объектов обстановки;
- настройка качества отображаемых текстур и моделей;

– управление слоями геоинформации;
– управление движением камерой;
– поиск и идентификация объекта. [21,104]

В основе подсистемы 3D-визуализации лежит алгоритм построения модели обстановки, описанный в §3.2 настоящей диссертации.

Подсистема 3D-визуализации запускается из терминала, при этом имеется возможность задать в параметрах запускаемой команды географические координаты, с которых начнется отображение обстановки при запуске компонента, а также угол наклона камеры к горизонту и набор слоев отображения обстановки. Пример:

user:$ mygis –coords x y z –angle a -ontexture -offdeep,

где x, y, z – координаты отображения, a – угол наглона.

Настройка параметров подсистемы 3D-визуализации осуществляется с помощью графического интерфейса пользователя во всплывающем меню (Рисунок 5.7) по нажатию пункта меню «настройки» на верхней части боковой панели.

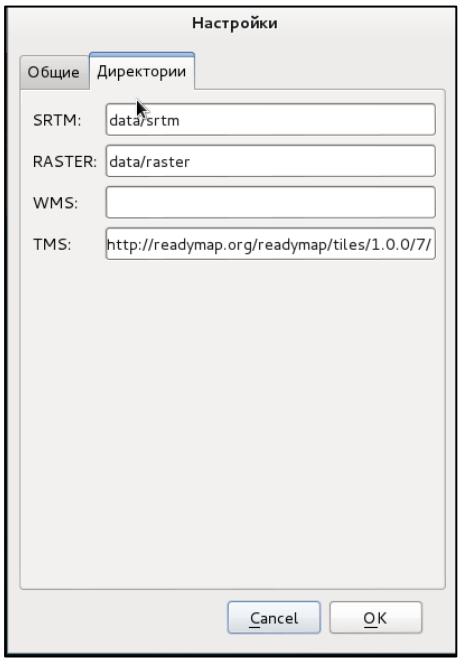

Рисунок 5.7. Меню настроек

Построение трехмерной модели поверхности Земли осуществляется в текстурированном виде по умолчанию при запуске системы (Рисунок 5.8).

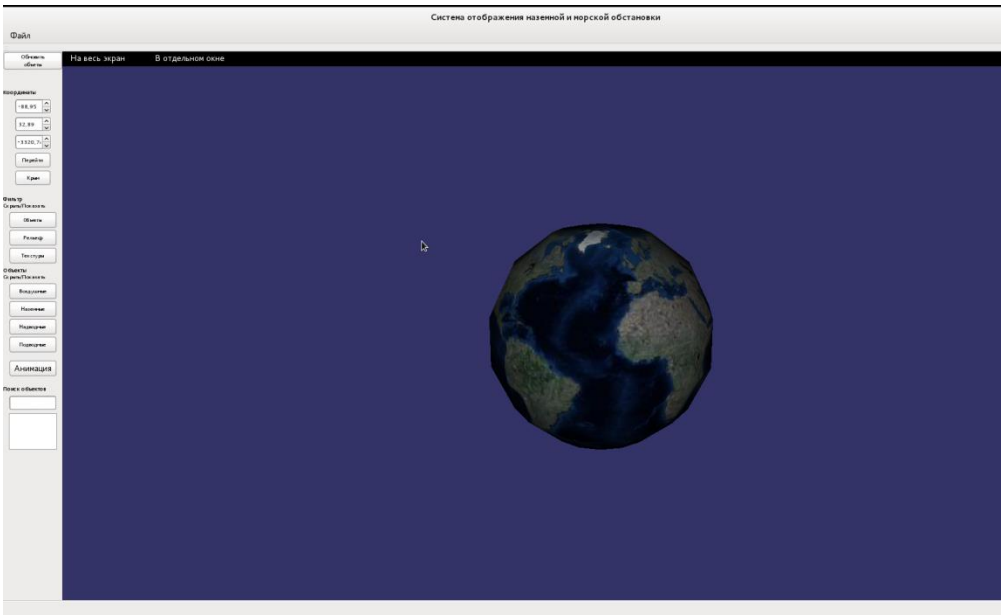

Рисунок 5.8. Основное окно подсистемы 3D–визуализации

Выбор качества отображаемых текстур, а именно скорости визуализации осуществляется пользователем с помощью пунктов меню «high», «low», расположенных в настройках системы 3D моделирования проектных решений (Рисунок 5.9). Таким образом, механизм уровней детализации объектов и текстурных элементов местности реализуется путём переключения между двумя степенями качества текстур и моделей. При выборе пользователем скорости визуализации «high», подсистема 3D-визуализации производит загрузку текстур и моделей низкого разрешения, что позволяет снизить аппаратные затраты как по объёму памяти, так и по объёму графических вычислений, в результате чего и повышается скорость визуализации. Режим скорости «low» реализован аналогичным способом только с более высококачественными текстурами и 3D моделями. [21,122]

Также реализована функция регулирования пользователем скорости визуализации. Для каждого уровня детализации загружаются текстуры только соответствующего разрешения. При изменении уровня детализации загруженные ранее текстуры очищаются (при этом сохраняясь в cache) и загружаются текстуры другого разрешения. Текстуры загружаются только на текущую область отображения.

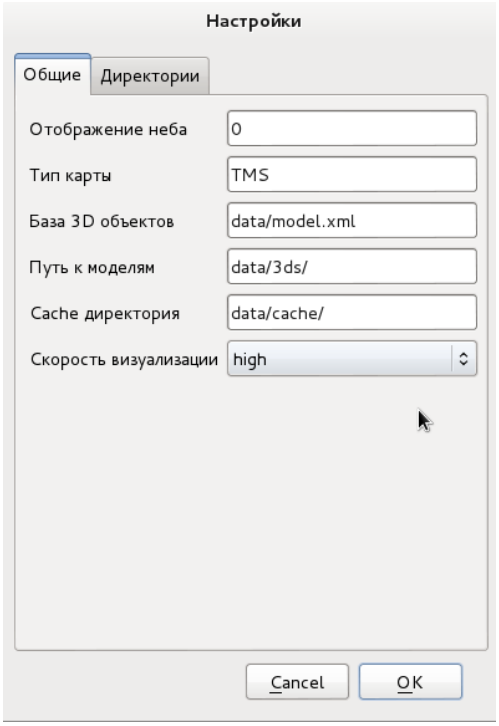

Рисунок 5.9. Меню настроек подсистемы 3D-визуализации

Масштабирование трехмерных моделей в подсистеме 3D-визуализации производится с автоматической картографической генерализацией, путем изменения степени детализации объектов в зависимости от расстояния до камеры. С увеличением расстояния до камеры трехмерная модель становится более «грубой», за счет снижения числа отображаемых полигонов модели, оставляя тем самым лишь наиболее заметные черты, по которым её можно распознать. Например, модель Земли при достаточном отдалении сменяет округлую форму на форму выпуклого многогранника (Рисунок 5.8).

Навигация по трехмерной сцене в режиме реального времени по всем степеням свободы осуществляется при помощи движения мыши с зажатой правой клавишей, с зажатой левой и правой клавишей – изменение уровня горизонта, масштабирование в точку нахождения курсора производится поворотом колесика мыши.

Космические снимки высокого разрешения подгружаются в подсистему 3D–визуализации из базы текстур и трансформируются на всю территорию с учетом рельефа благодаря информации из базы высот и глубин. База текстур и база высот и глубин могут находиться как на локальной машине, так и на удаленной (Рисунок 5.10).

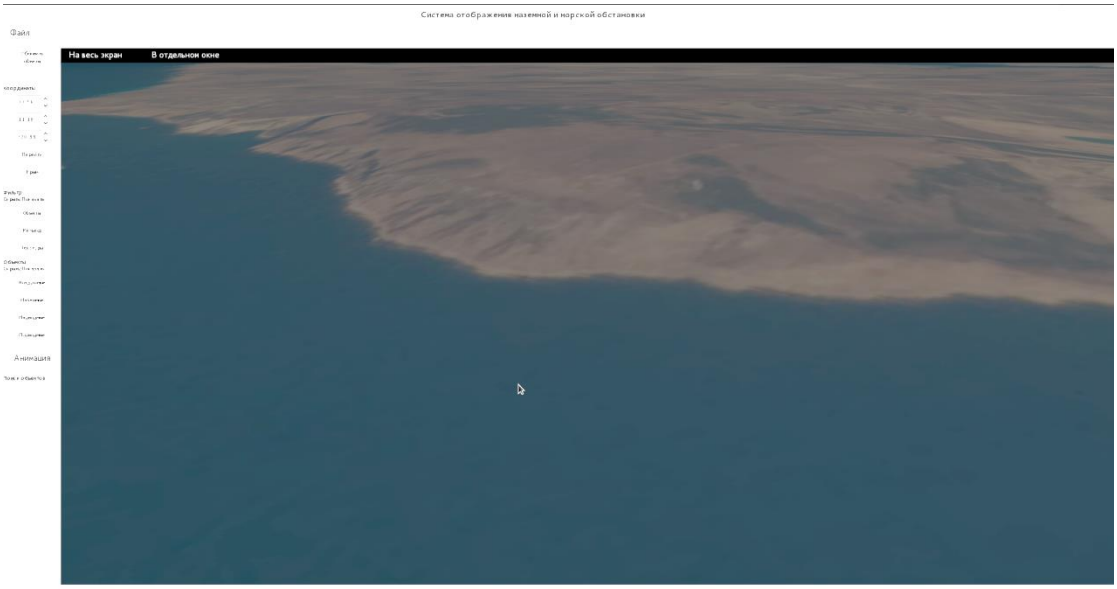

Рисунок 5.10. Участок местности с наложенными текстурами, высотами и глубинами

Подсистема 3D–визуализации позволяет отображать произвольные линейные, точечные и площадные объекты, для их создания необходимо задать координаты опорных точек в конфигурационном XML-файле (Рисунок 5.11). Алгоритм рисования фигуры на карте представлен в §4.2 диссертации.

```
Объект Слой = "РПЛСН" Номер = "4" Тип="Линия" Ключ = "Линия1" Имя = "Линия 1" Код = "004">

Семантика>
/Семантика>

Метрика>
Запись Значение = "33.46" Номер = "1" Название = "x0" Код = "13"/>
Запись Значение = "44.6" Номер = "2" Название = "y0" Код = "14"/>
Запись Значение = "750" Номер = "3" Название = "z0" Код = "16"/>
Запись Значение = "33.36" Номер = "4" Название = "x1" Код = "13"/>
Запись Значение = "44.6" Номер = "5" Название = "y1" Код = "14"/>
Запись Значение = "600" Номер = "6" Название = "z1" Код = "16"/>
/Метрика>

/Объект>

Объект Слой = "РПЛСН" Номер = "4" Тип="Полигон" Ключ = "Полигон1" Имя = "Полигон 1" Код = "004">

Семантика>
/Семантика>

Метрика>
Запись Значение = "" x="33.43" y="44.5" z="600" Номер = "1" Название = "Точка" Код = "13"/>
Запись Значение = "" x="33.46" y="44.5" z="500" Номер = "1" Название = "Точка" Код = "13"/>
Запись Значение = "" x="33.59" y="44.62" z="900" Номер = "1" Название = "Точка" Код = "13"/>
/Метрика>

/Объект>
/Объекты>
```

Рисунок 5.11. XML-файл с линейными, точечными и площадными объектами

Подсистема 3D–визуализации позволяет наложить на поверхность и визуализировать растры GeoTIFF, векторные данные, например, shape-файлы ESRI, OGC-совместимые веб-

сервисы (например, WMS), слои ГИС, опубликованные при помощи MapServer или ESRI ArcGIS Server, а также карты OpenStreetMap, ArcGIS Online или NASA OnEarth (Рисунок 5.12). [75]

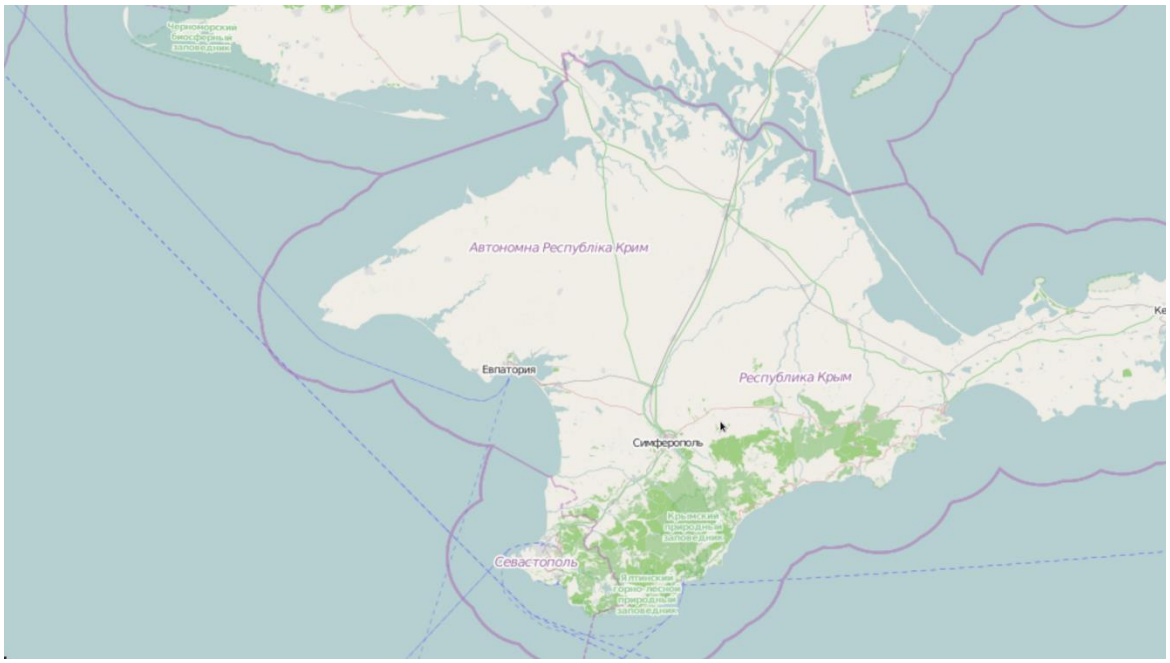

Рисунок 5.12. Участок местности в OpenStreetMap

Поиск и выделение объекта осуществляется с помощью поисковой строки, расположенной на боковой панели графического интерфейса подсистемы 3D–визуализации (Рисунок 5.13). После набора пользователем поисковой строки и нажатию клавиши «Enter» осуществляется центрирование камеры на результатах поиска. Алгоритм распознавания объекта описан в §4.2 настоящей диссертации.

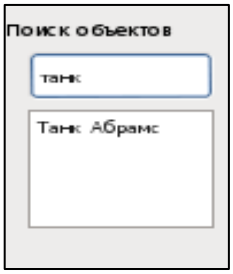

Рисунок 5.13. Панель поиска объектов

В системе 3D моделирования проектных решений имеется возможность фиксации участка местности в заданном масштабе для последующего его наблюдения в отдельном окне.

Данная функция осуществляется в подсистеме 3D-визуализации по нажатию пункта меню «в отдельном окне», в пользовательском меню (Рисунок 5.14), с последующим перетаскиванием отделившегося окна на второй экран при его наличии.

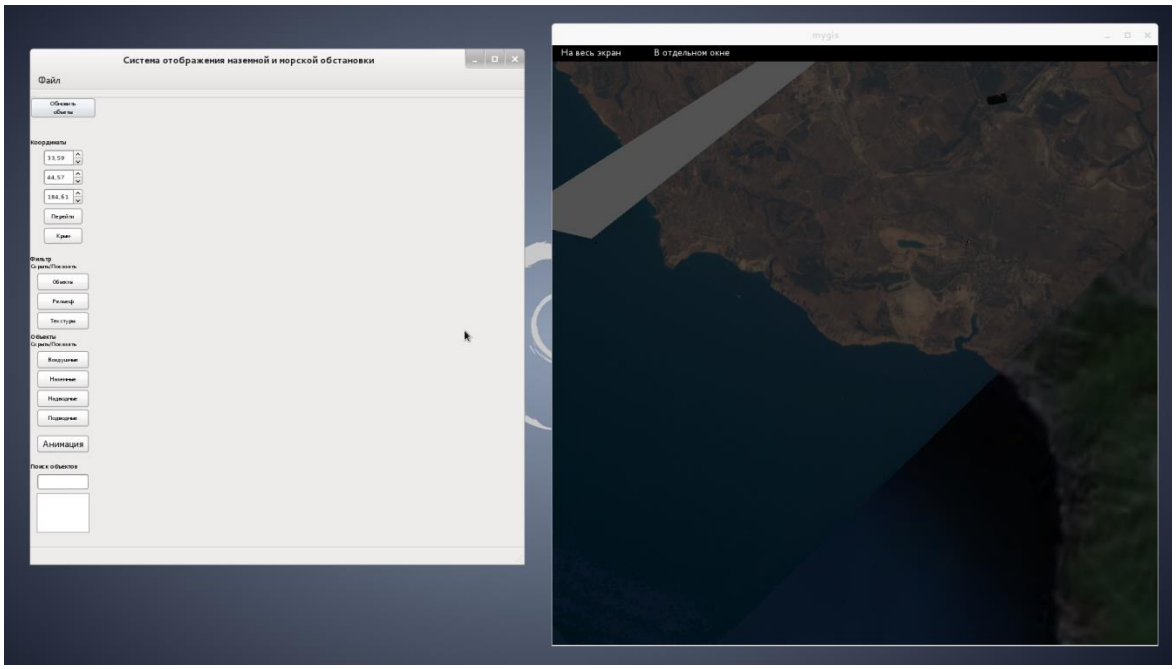

Рисунок 5.14. Наблюдение за участком местности в отдельном окне

Алгоритм полёта камеры, описанный в §4.2 диссертации, позволяет осуществлять отображение пути по рельефу местности и облёта камеры по нему. Поскольку путь по рельефу местности представляет собой совокупность линейных и точечных объектов, то возможность его отображения гарантирована наличием возможности отображения линейных и точечных объектов. Перемещение виртуальной камеры по пути производится за счет последовательной смены координат камеры по соответствующим координатам точек пути со смещением на 10 метров вверх по координате z для удобства восприятия перемещения, а также поворотом её фокуса в направлении движения. [99]

Фильтрация отображения слоев информации в подсистеме 3D-визуализации осуществляется с помощью пунктов меню, расположенных на боковой панели графического интерфейса (Рисунок 5.15). Названия пунктов меню соответствуют слоям отображаемой информации. Одно нажатие выключает тот или иной слой информации, например, текстуры, повторное нажатие снова включает его в основной состав отображаемых слоев. Алгоритм фильтрации отображения слоёв информации описан в §4.2 диссертации.

Рисунок 5.15. Панель фильтрации отображения слоев информации

Все функции подсистемы 3D-визуализации реализованы с помощью интуитивно понятного интерфейса, состоящего из боковой и верхней панелей навигации (Рисунок 5.16).

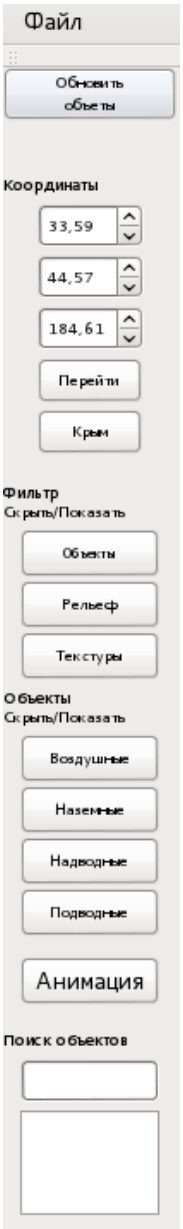

Рисунок 5.16. Пользовательское меню с удобными элементами управления

Возможен переход в полноэкранный режим отображения обстановки в подсистеме 3D-визуализации, который осуществляется по нажатию пункта меню «на весь экран», расположенной на верхней панели навигации графического интерфейса (Рисунок 5.17).

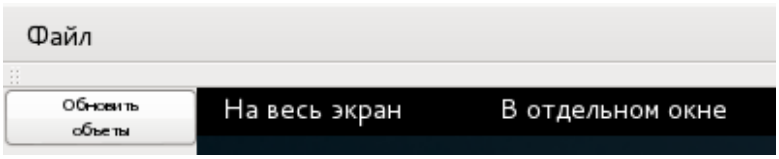

Рисунок 5.17. Меню выбора режима отображения обстановки

Возможность управления всеми параметрами отображения в подсистеме 3D-визуализации реализована при помощи программного взаимодействия с библиотекой osgEarth, а так же с конфигурационным XML–файлом (Рисунок 5.18), структура которого позволяет хранить все объекты обстановки. Путем программного внесения изменений в этот файл, иными словами, используя программный интерфейс, можно управлять всеми параметрами отображения подсистемы 3D-визуализации.

Рисунок 5.18. Конфигурационный XML–файл взаимодействия системы 3D моделирвоания с внешними приложениями и базами данных

Имеется возможность переключения режима отображения обстановки между каркасным и текстурированным видами поверхности Земли с учетом рельефа при помощи соответствующего пункта меню на панели управления. Пример отображения каркасного вида поверхности Земли представлен на рисунке 5.19.

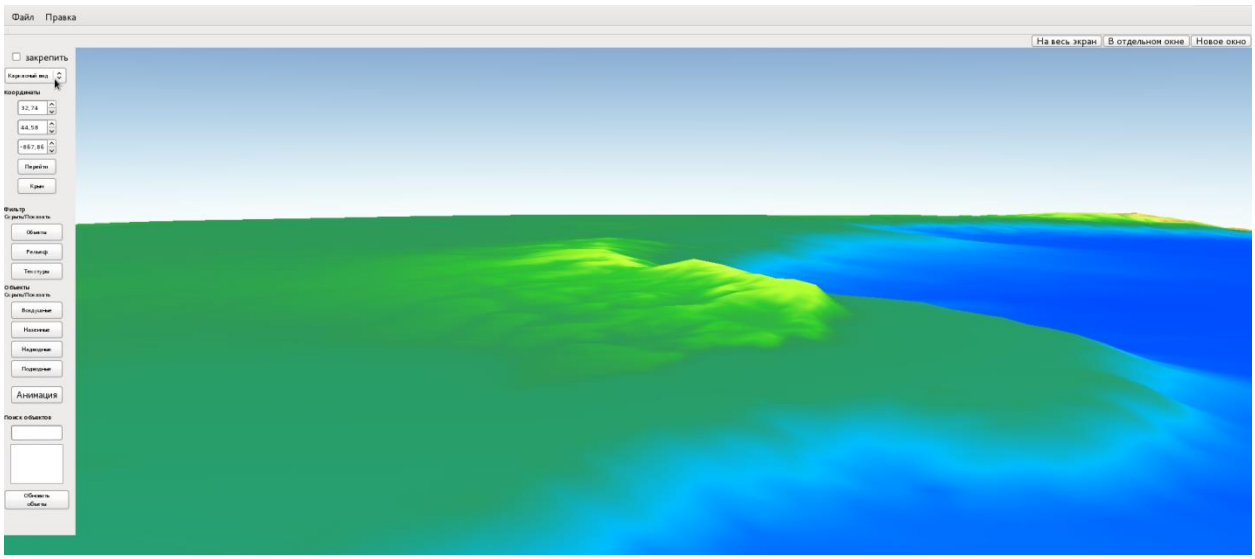

Рисунок 5.19. Каркасный вид морского побережья

Реализован диалог выбора и загрузки растровых и векторных карт с файловой системы в виде модального окна с последующим их отображением.

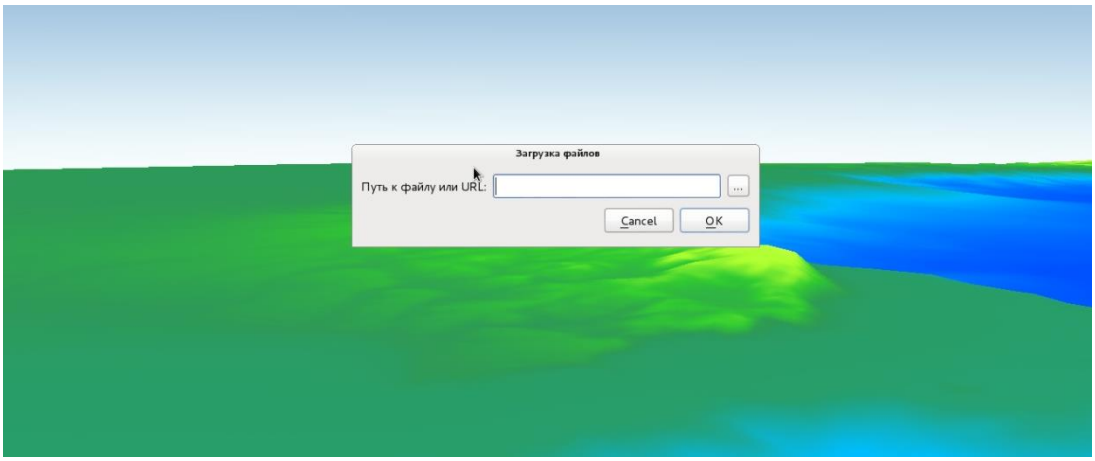

Рисунок 5.20. Модальное окно диалога выбора и загрузки растровых и векторных карт

Также в подсистеме 3D-визуализации реализованы следующие возможности:

- реализована возможность учитывать рельеф при отображении линейных, точечных и площадных объектов на поверхности за счёт дополнительной конфигурации библиотеки osgEarth;
- реализован диалог выбора и загрузки с файловой системы (или из другого источника) векторных и топографических основ в виде модального окна;
- реализовано выделение области в отдельном немодальном окне;

- реализован программный интерфейс в части управления положением камеры наблюдения, выделения конкретного объекта обстановки с использованием манипулятора мыши;
- реализована загрузка растровых, высотных/глубинных, топографических данных по местности в виде диалогового окна;
- реализована возможность запуска системы 3D моделирования проектных решений с параметрами в конфигурационном XML-файле, а также с помощью командной строки;
- реализован классификатор 3D моделей, распределенных по категориям в виде плавающей панели с предварительным просмотром общего вида модели, а также с динамическим просмотром по средствам запуска «просмоторщика» 3D моделей;
- реализовано отключение и выключение конкретного слоя 3D обстановки, которое производится в меню «слои» с помощью «чекбоксов».

В меню отображения информации об объектах имеется возможность просмотра координат, выраженных как широта, долгота и высота рассматриваемого объекта (Рисунок 5.21).

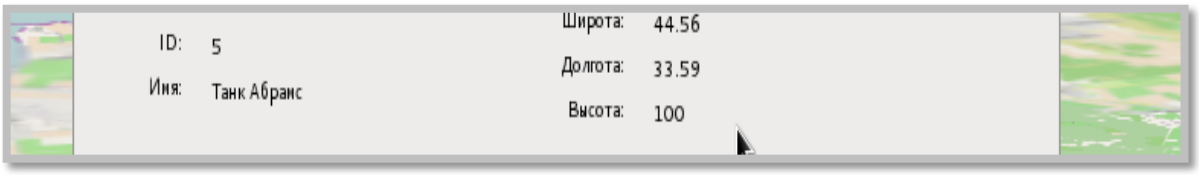

Рисунок 5.21. Окно редактирования слоев

Для отображения 3D обстановки с разной степенью точности в системе используются файлы формата GeoTiff, состоящие из нескольких слоев геоинформации.

Выбор порядка наложения слоев геоинформации на местность производится в меню «слои» с помощью перетаскивания курсором мыши.

Отображение включенных в данный момент слоев геоинформации осуществляется в меню «слои» с помощью «чекбоксов» (Рисунок 5.22).

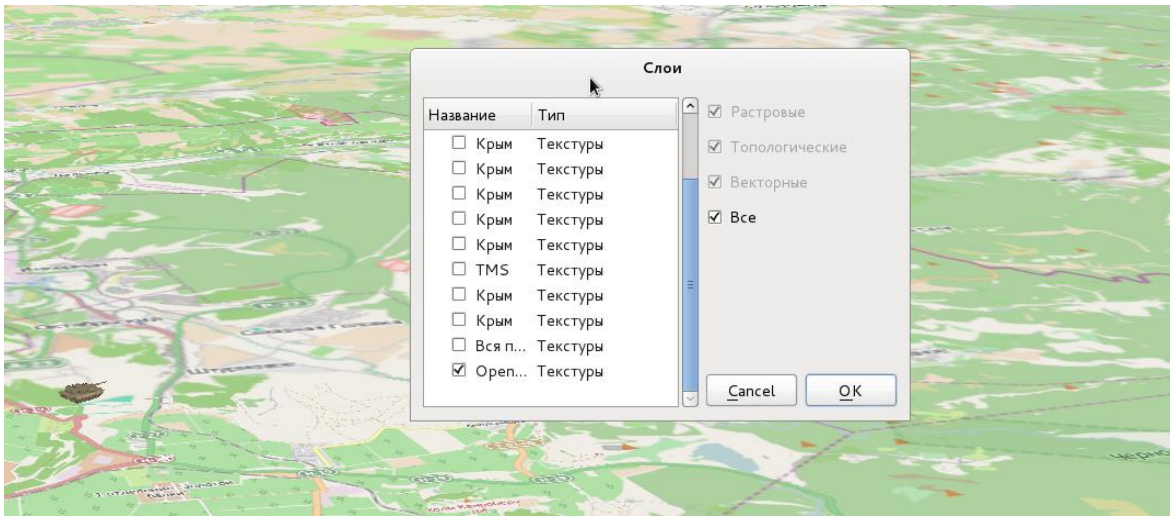

Рисунок 5.22. Окно редактирования слоев

5.5.2. Разработка компонента работы с источниками обстановки

Компонент (подсистема) работы с источниками обеспечивает взаимодействие (чтение, запись) с растровыми и векторными форматами карт, расположенными как локально на файловой системе, так и в локальной сети, а также загрузку матриц высот и глубин с последующей их визуализацией. В частности, выполняются такие функции как:

- открытие карт, текстур, моделей поверхности наиболее распространённых форматов: SXF, S-57(63), KML, OSM, GeoTiff, JPG, 3DS;
- привязка карт, текстур, моделей поверхности по координатам;
- формирование библиотеки 3D объектов на основе распространённых форматов 3D моделей (3DS и т.п.);
- хранение в упорядоченном виде библиотек импортированных карт, текстур, моделей поверхности, 3D объектов в базе данных или на файловой системе с возможностью пакетного переноса на другой компьютер;
- кеширование данных от интернет-источников (OGC-сервисы) для использования в автономном режиме. [24,30]

Средствами файловой системы реализовано формирование библиотеки 3D объектов на основе распространённых форматов. Панель управления автоматически формируется считыванием конкретной директории, указанной в настройках системы 3D моделирования проектных решений. В данной директории осуществляется поиск файлов по маске “*.3DS “ и их загрузка в оперативную память.

Классификатор 3D моделей реализован в виде панели, которая появляется по нажатию на пункт меню «классификатор моделей» расположенный в верхней части графического интер-

фейса пользователя. На панели расположены четыре элемента, соответствующие категориям объектов: воздушные, наземные, надводные, подводные. По нажатию на тот или иной элемент появляется меню выбора объекта, которое представляет собой выпадающий список с названиями объектов и окно предпросмотра изображения выбранной модели (Рисунок 5.23).

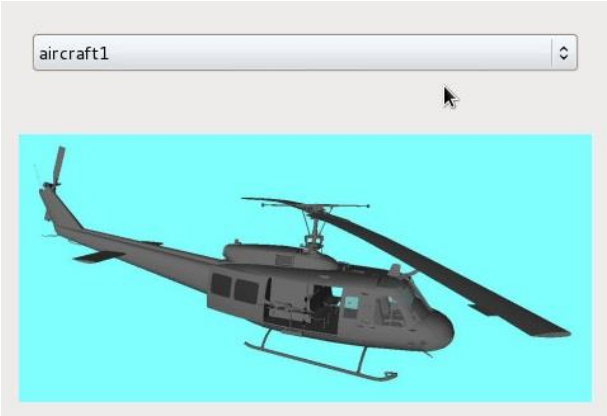

Рисунок 5.23. Классификатор 3D моделей

Система 3D моделирования проектных решений имеет возможность кэширования данных от интернет-источников в директорию, указанную в настройках системы. При последующем запуске система автоматически загружает сохранённые cache-файлы. Схема кеширования данных реализованная в библиотеке osgEarth представлена на рисунке 5.24.

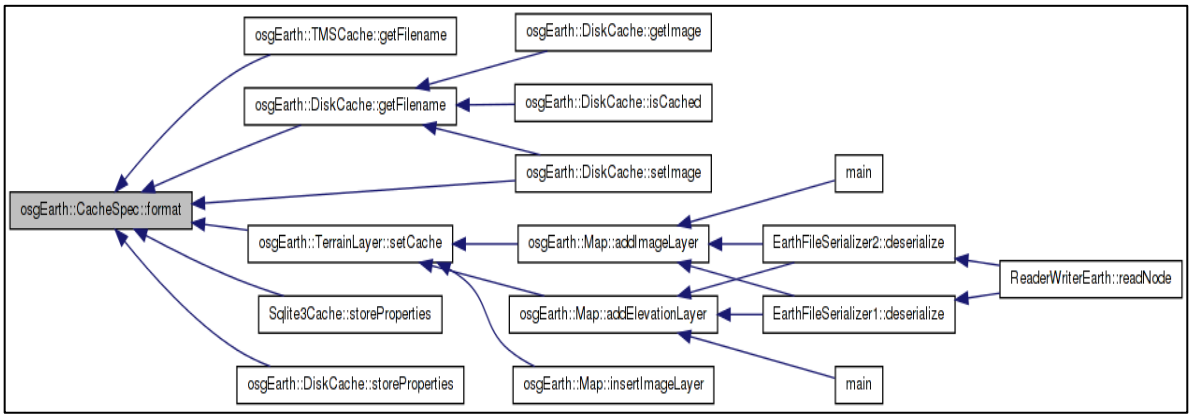

Рисунок 5.24. Схема кеширования данных, реализованная в библиотеке osgEarth

В базу данных системы 3D моделирования проектных решений введено более 50 низкополигональных моделей объектов надводной, подводной и воздушной техники:

- танки;
- корабли;
- самолёты;
- вертолёты;
- подводные лодки;

- автомобили.

Пример модели корабля представлен на рисунке 5.25.

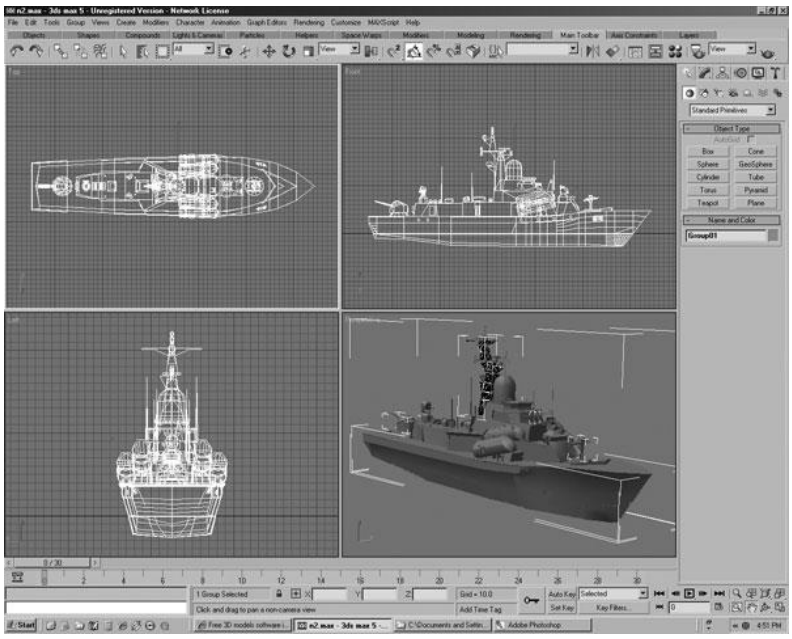

Рисунок 5.25. Модель корабля в окне редактора 3Ds MAX

В системе 3D моделирования проектных решений собраны карты, текстуры, модели поверхности Земли из следующих источников:

- Openstreetmap.org - веб-карты на основе данных OSM со встроенным «валидатором» этих данных и возможностью отправки пользователями сообщений об ошибках;
- maps.google.com – карты от компании Google;
- NASA.gov – снимки высокого разрешения, а также матрицы высот/глубин (Рисунок 5.26).

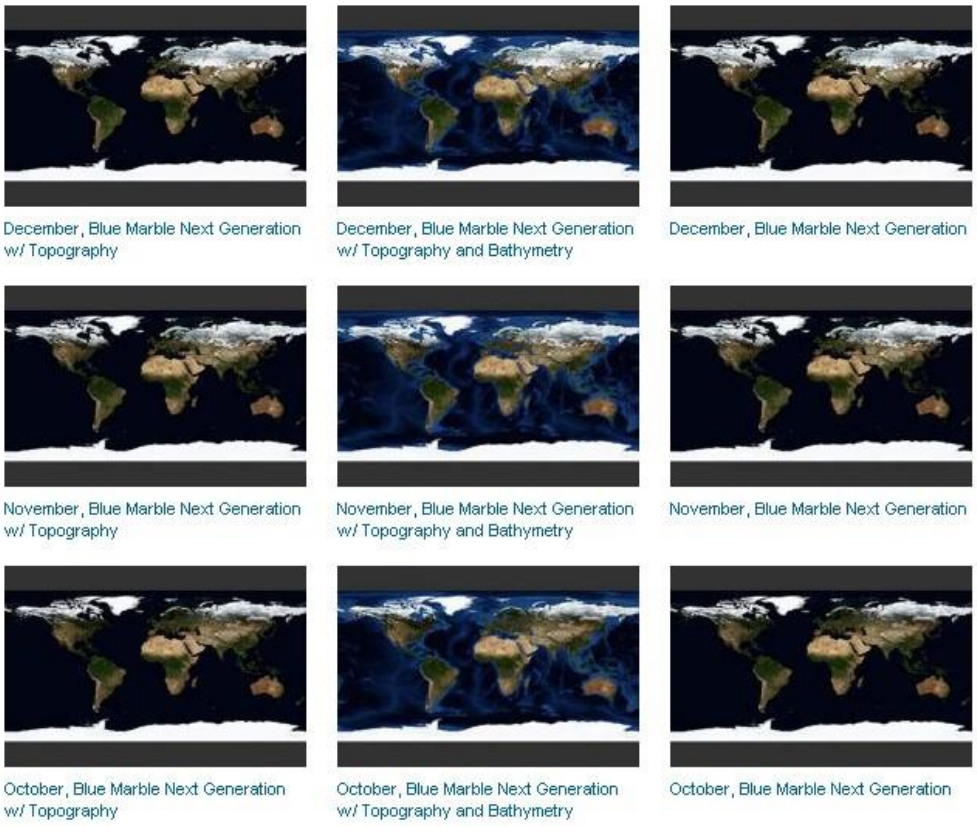

Рисунок 5.26. Снимки поверхности Земли по временам года, представленные на сайте nasa.gov

Все трехмерные модели объектов, используемых в данной системе, покрыты текстурами, созданными на основе их фотографий.

Создана база для хранения текстур и космических снимков различного разрешения. Например, формат GeoTiff имеет возможность хранения нескольких слоёв в одном файле для различных уровней детализации.

5.5.3. Разработка компонента взаимодействия с внешними системами

Компонент (подсистема) поддержки интерфейсов взаимодействия с внешними системами реализован при помощи работы с XML-файлом или по UDP-порту, и позволяет системе 3D моделирования проектных решений взаимодействовать с внешними системами. Функционирование подсистемы взаимодействия с внешними системами реализовано на базе алгоритма обмена информацией с внешними системами и алгоритма обмена информацией по UDP-порту, которые описаны в §4.2 настоящей диссертации.

Структура компонента поддержки интерфейсов взаимодействия с внешними системами представлена на рисунке 5.27.

Рисунок 5.27. Структура компонента поддержки интерфейсов взаимодействия с внешними системами

В данной подсистеме разработан модуль взаимодействия с базами геоинформационных данных и координат объектов, расположенных как в локальной сети, так и в сети Интернет, с целью конвертации формата геоданных от внешних источников во внутренний формат XML-файла. При редактировании параметров XML-файла система 3D моделирования проектных решений автоматически реагирует на внесенные изменения, отображая актуальную 3D обстановку в подсистеме визуализации.

Автоматизированное нанесение 3D обстановки через программные интерфейсы происходит путём добавления записи о новом слое 3D обстановки в XML файл. В системе автоматически появляется новый слой 3D обстановки.

Обстановка до внесения изменений в XML-файл представлена на рисунке 5.28.

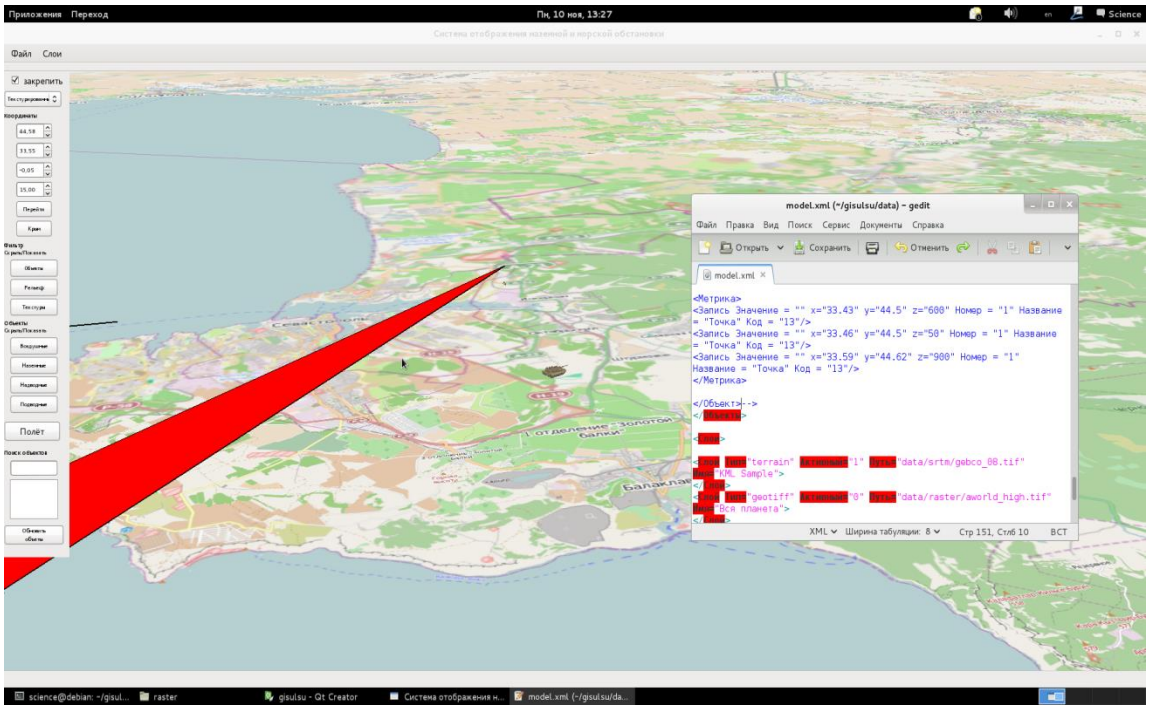

Рисунок 5.28. 3D обстановка до внесения изменений в XML-файл

Автоматизированное изменение параметров 3D обстановки происходит внесением изменений в параметры редактируемых слоёв 3D обстановки в XML файле. В системе автоматически отображаются внесенные изменения (Рисунок 5.29).

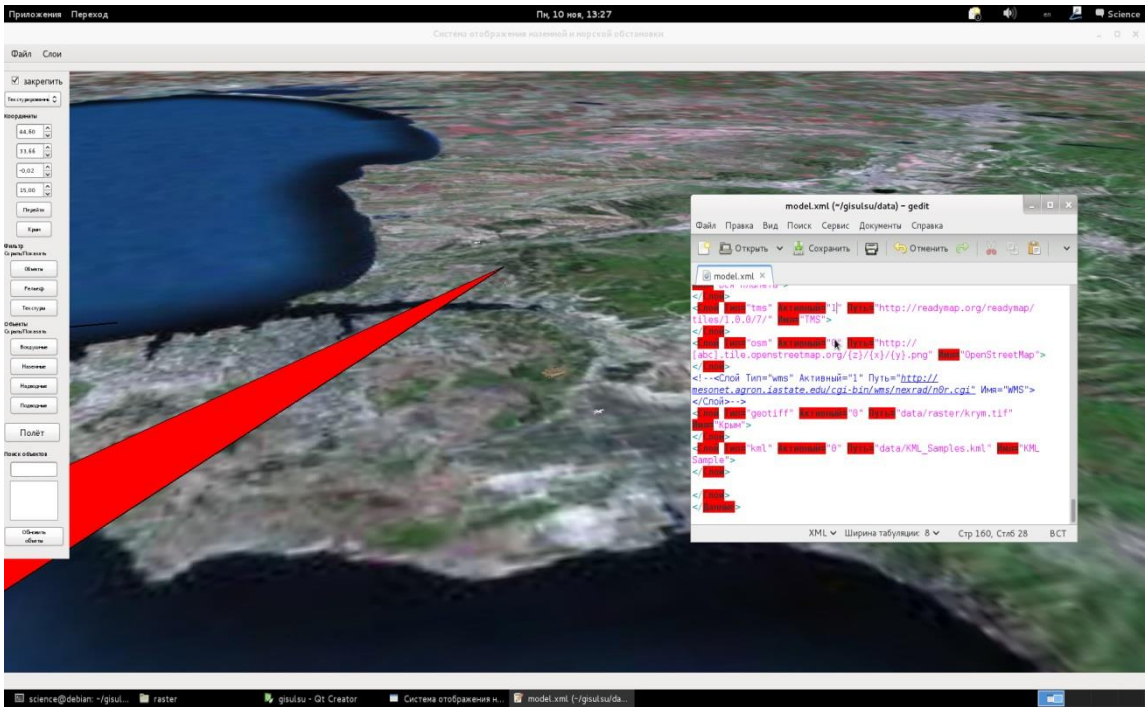

Рисунок 5.29. 3D обстановка после внесения изменений в XML-файл

Открытие и редактирование слоев обстановки автоматизированным способом через программные интерфейсы происходит за счёт добавления записи о новом слое или внесения изменений в существующем в XML файле. Система автоматически обновит 3D обстановку.

Информация по объектам 3D обстановки из внешних источников извлекается из внешней базы данных и сохраняется в XML файл. Система автоматически реагирует на изменения в XML файле.

Вся атрибутика интересуемого объекта хранится в XML файле. При нажатии на объект открывается окно справки с выведенной атрибутикой (Рисунок 5.30).

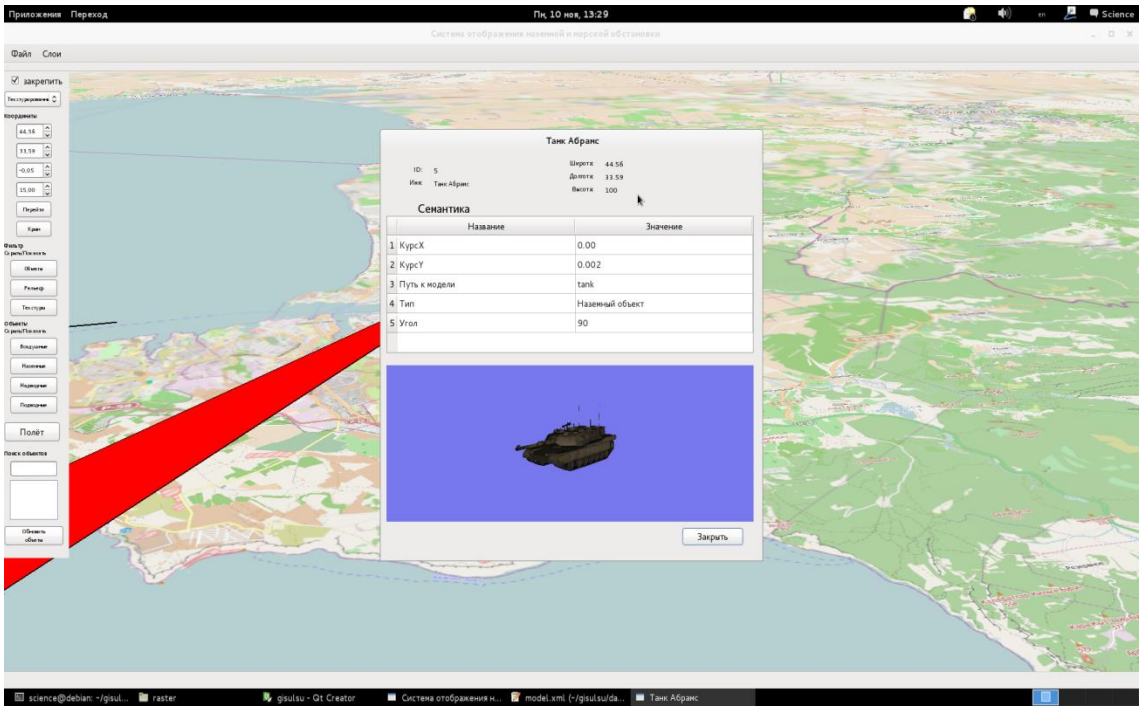

Рисунок 5.30. Окно справки об объекте

Точка местности, заданная геокоординатами, высотой, глубиной и уровнем горизонта хранится в XML файле. При запуске системы автоматически происходит переход на данную точку местности.

5.5.4. Разработка компонента имитации движения объектов

Компонент (подсистема) имитации движения объектов реализован при помощи заданных «троек» координат (<x, y, z>), по которым происходит последовательное перемещение с использованием методов апроксимации. Функционирование подсистемы имитации движения объектов реализовано на базе алгоритма идентификации ситуации и алгоритма распознавания объекта, которые описаны в §4.2 настоящей диссертации.

Для задания траектории движения объекта необходимо выбрать на карте 3D объект, открыть окно информации кликом мыши по нему и в появившемся окне нажать на пункт меню «Траектория». В новом окне в таблице задать траектории движения (Рисунок 5.31).

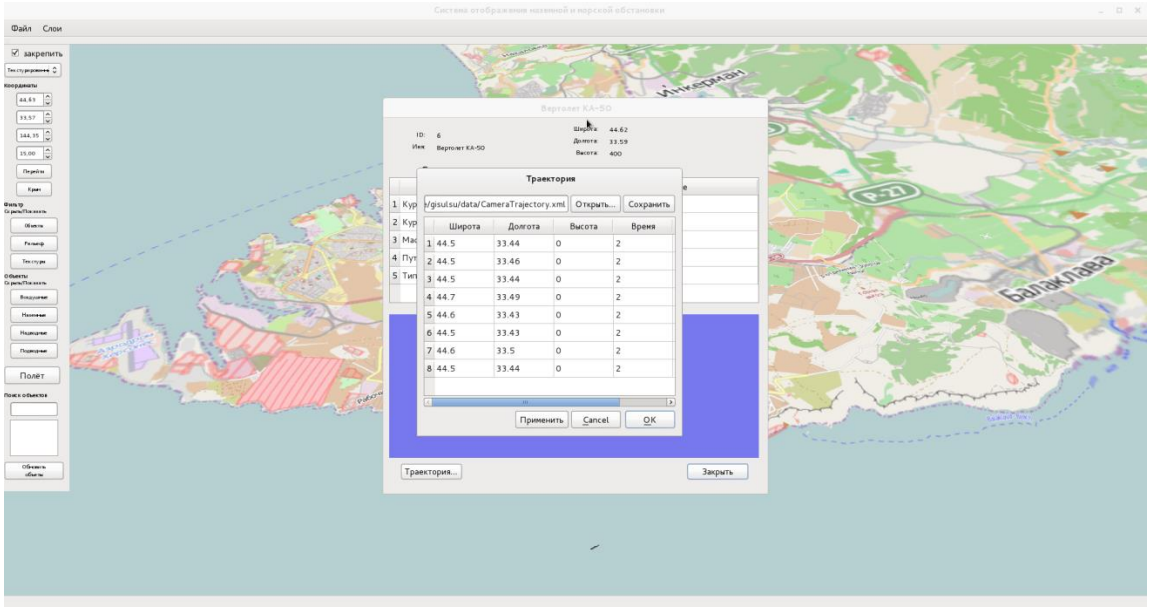

Рисунок 5.31. Задание траектории движения объекта

В диалоговом окне «Траектория» окна информации об объекте есть возможность загрузки и сохранения в файл сценариев движения объекта (Рисунок 5.32).

CameraTrajectory.xml

```
<Траектория>
 <Точка Широта="44.5" Долгота="33.44" Время="2" Высота="0"/>
 <Точка Широта="44.5" Долгота="33.46" Время="2" Высота="0"/>
 <Точка Широта="44.5" Долгота="33.44" Время="2" Высота="0"/>
 <Точка Широта="44.7" Долгота="33.49" Время="2" Высота="0"/>
 <Точка Широта="44.6" Долгота="33.43" Время="2" Высота="0"/>
 <Точка Широта="44.5" Долгота="33.43" Время="2" Высота="0"/>
 <Точка Широта="44.6" Долгота="33.5" Время="2" Высота="0"/>
 <Точка Широта="44.5" Долгота="33.44" Время="2" Высота="0"/>
</Траектория>
```

Рисунок 5.32. Загрузка и сохранение сценариев

Для моделирования динамики перемещения объектов необходимо задать сценарий движения объекта в диалоговом окне «Траектория» окна информации об объекте.

§ 5.6. Программно-аппаратный комплекс отображения морской, наземной, воздушной ситуационной обстановок

CASE-средство проектирования 3D ГИС отображения ситуационной обстановки формирует проектные решения для разработки на их основе 3D ГИС, которые включают как программные средства (свободно-распространяемые ресурсы, библиотеки, текстуры, 3D модели), так и аппаратные средства, которые обеспечивают их функционирование.

Проектные решения хранят информацию о библиотеках, с использованием которых ведётся разработка. У каждой библиотеки заданы свои минимальные системные требования, при

которых она функционирует. 3D модели объектов имеют качество детализации, которое требует наличия видеопроцессоров определенной мощности и скорости обработки графики.

На основе библиотек, текстур, 3D моделей объектов местности формируется база данных, описывающая их аппаратные требования. С использованием этой базы данных проектируется аппаратно-программный комплекс, включающий как наборы библиотек, текстур и моделей местности, так и аппаратную часть: процессор, оперативную память, видеоконтроллеры, шину данных, мониторы, устройства ввода и т.д.

Системные требования библиотек включают: частоту процессора (ГГц); видеокарту (GeForce, RADEON); объём оперативной памяти (ГБ); место на жёстком диске (ГБ); наличие сети; GPS-модуль; CD/DVD-ROM; клавиатуру; мышь; джойстик; тачпад; принтер; звуковую карту; операционную систему; разрешение экрана (пикселей).

На основе сформированных системных требований построен граф требований 3D ГИС (Рисунок).

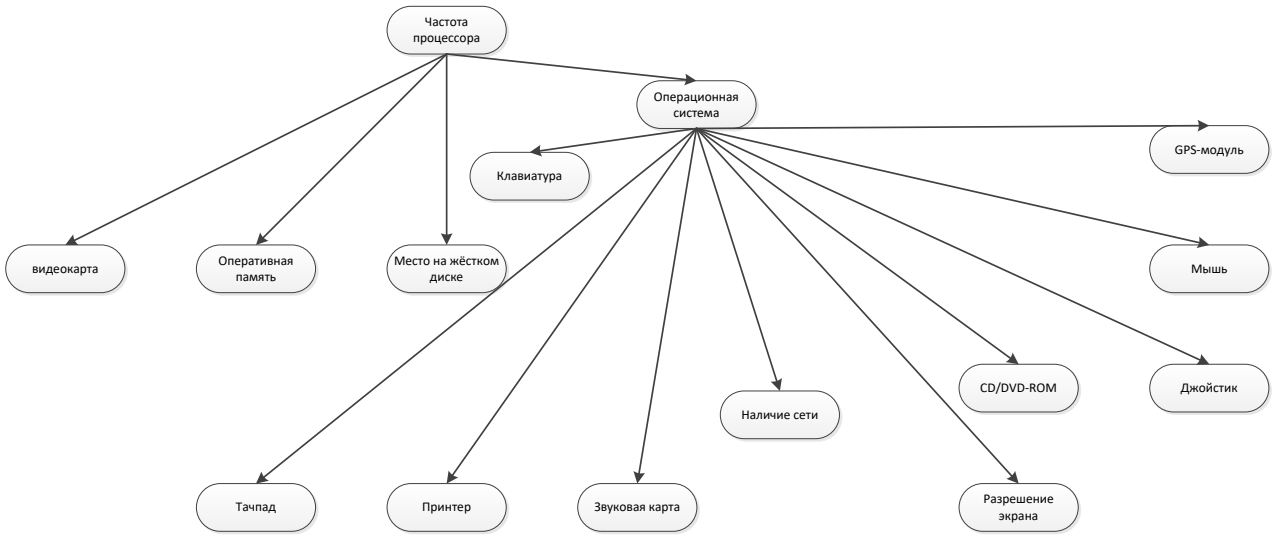

Рисунок 5.33. Граф системных требований 3D ГИС

Некоторые требования, такие как 3, 4, являются «суммируемыми», т.е. для каждой библиотеки значение этого требования складывается со значениями требований других библиотек.

Требования 5, 6, 7, 8, 9, 10, 11, 12, 13 являются «присутствующими», т.е. такими, которые добавляются к системе, если имеются хотя бы у одной из библиотек и содержат значение 1 (присутствуют) или 0 (отсутствуют).

Требования 1, 2, 15 являются «наращиваемыми», т.е. их значения увеличиваются, если каждая последующая библиотека требует большее значение этого требования.

Требование 14 содержит особенности и «присутствующих», и «наращиваемых» требований. 3D ГИС должно функционировать в одной из операционных систем и операционная система имеет минимальную версию.

Структура программно-аппаратного комплекса с учётом описанных выше системных требований представлена на рисунке 5.34.

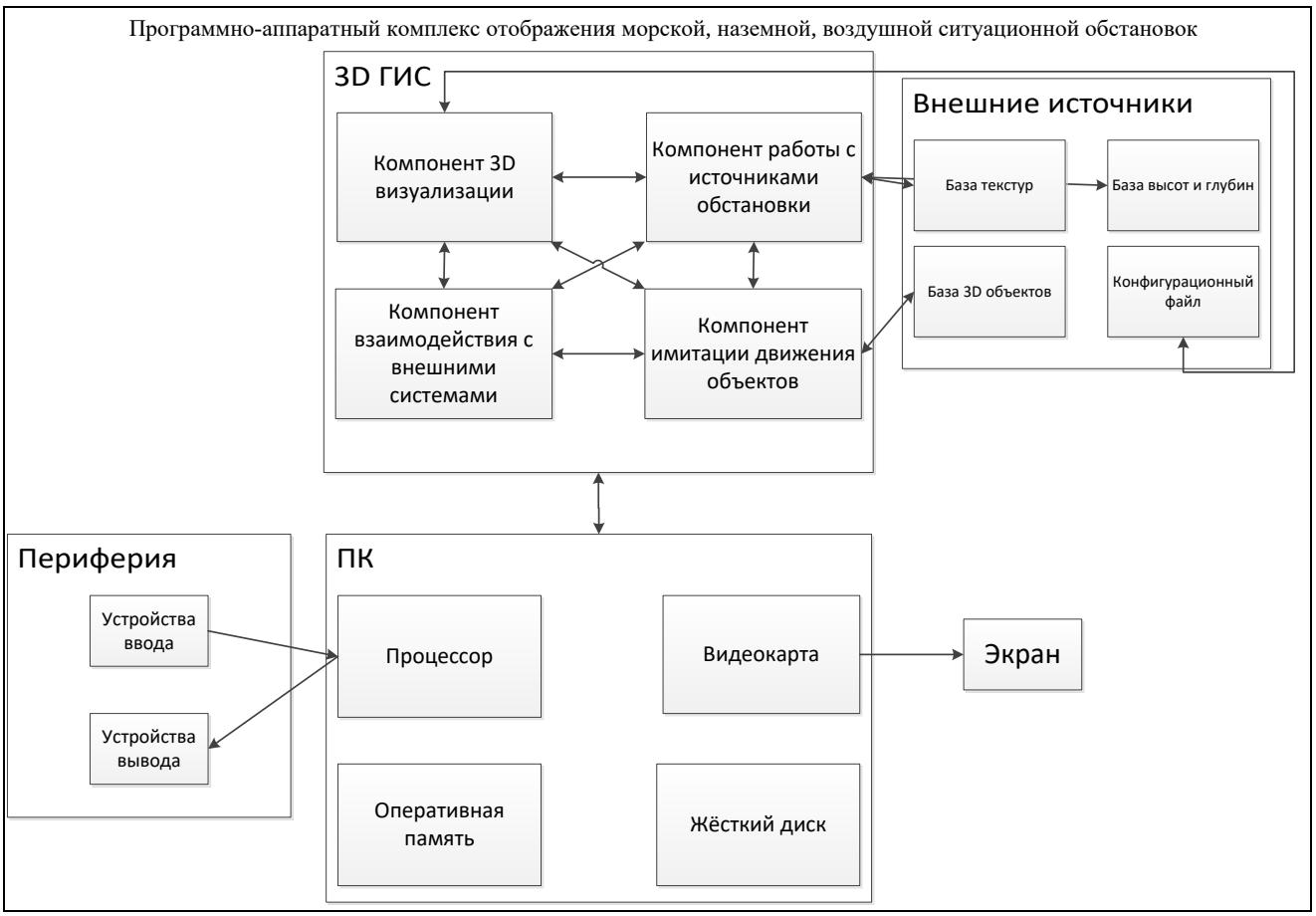

Рисунок 5.34. Структура программно-аппаратного комплекса 3D ГИС

§ 5.7. Оценка эффективности проектирования 3D ГИС на основе CASE-средства

Для оценки эффективности проектирования 3D ГИС с использованием свободно распространяемых библиотек и расчёта финансовых затрат необходимо рассчитать сложность проектирования 3D ГИС с использованием предложенного CASE-средства и сложности разработки 3D ГИС другими способами и в дальнейшем сравнить их.

Предлагается оценивать сложность всей 3D ГИС как сумму оценок сложности каждой из используемых при разработке библиотек и оценок сложности алгоритмов, реализующих недостающую функциональность. В свою очередь, для оценки сложности библиотеки или алгоритма предлагается проводить декомпозицию модулей до функциональных выражений, таких эле-

ментов, которые образуют минимальный функциональный набор команд, сложность которого может однозначно рассчитаться на конкретном языке программирования.

Примером такого функционального выражения может быть расчёт значения косинуса какой-либо величины.

Также к функциональным выражениям относятся управляющие конструкции – операторы. Все управляющие конструкции можно разделить на 3 основных типа:

- действие или линейный оператор – последовательность операций;
- условие, которое на выходе имеет два действия в зависимости от результата;
- цикл, который определенное количество раз выполняет одну и ту же последовательность действий.

Для большинства стандартных функциональных выражений на выбранном языке программирования сложность можно определить однозначно и она не будет изменяться. Рассчитанные значения рекомендуется занести в специальную базу значений сложности и использовать в дальнейшем. Для других функциональных выражений оценка сложности может понадобиться только один раз, но расчёт также необходим. И в первом, и во втором случае для оценки сложности необходимо использовать специальные метрики.

Можно выделить несколько наиболее подходящих метрик для оценки сложности функционального выражения.

1. Метрика Холстеда:

   $n_{оператор}$ – число уникальных операторов функционального выражения;

   $n_{операнд}$ – число уникальных операндов функционального выражения;

   $N_{оператор}$ – общее число операторов функционального выражения;

   $N_{операнд}$ – общее число операндов функционального выражения;

   $n = n_{оператор} + n_{операндов}$ – словарь функционального выражения;

   $N = N_{оператор} + N_{операндов}$ – длина функционального выражения;

   $V = N*\log_2 n$ – объём функционального выражения.

   Сложность функционального выражения в количестве операторов:

   $$S = \frac{n_{оператор}}{2} * \frac{N_{операнд}}{n_{операнд}}. \quad (5.3)$$

2. Метрика Чепина:

   {P} – множество вводимых переменных для расчётов и обеспечения вывода;

   {M} – множество модифицируемых или создаваемых внутри функционального выражения переменных;

   {C} – множество управляющих переменных;

{T} – множество неиспользуемых переменных;

Сложность функционального выражения в количестве переменных:

S = P + 2M + 3C + 0.5T. (5.4)

Для корректного расчёта сложности 3D ГИС все библиотеки и алгоритмы должны быть оценены с помощью одной и той же метрики.

После оценки сложности всех функциональных выражений библиотеки или алгоритма необходимо рассчитать общую сложность этой библиотеки или алгоритма.

$$S_{\text{библ (алг)}} = \sum_{i=1}^{n} S_{i\ \text{ф.в.}}\ , \quad (5.5)$$

где n – количество функциональных выражений библиотеки или алгоритма.

Для оценки эффективности использования готовой библиотеки вместо собственной разработки на основе предложенного алгоритма используется:

$$S_{\text{эфф.библ}} = S_{\text{алг}} - S_{\text{библ}} + S_{\text{ад}}\ , \quad (5.6)$$

где $S_{\text{ад}}$ – сложность создания адаптера для подключения библиотеки к разрабатываемой 3D ГИС.

Сложность адаптера рассчитывается аналогично сложности библиотеки или алгоритма путём декомпозиции до функциональных выражений.

Сложность всей 3D ГИС вычисляется по формуле:

$$S_{\text{ГИС}} = \sum_{i=1}^{n} S_{i\ \text{алг}} + \sum_{j=1}^{m} S_{j\ \text{ад}}, \quad (5.7)$$

где n – количество собственных алгоритмов, m – количество адаптеров.

Эффективность проектирования 3D ГИС с использованием свободно распространяемых библиотек вместо разработки всех компонентов системы «с нуля» вычисляется по формуле:

$$S_{\text{эфф}} = \sum_{i=1}^{k} S_{i\ \text{алг}} - S_{\text{ГИС}}\ , \quad (5.8)$$

где k = n+m общее число алгоритмов и библиотек, используемых в 3D ГИС.

При декомпозиции алгоритмов и библиотек на функциональные выражения рекомендуется использовать специализированные программы для статистического анализа исходного кода.

Для различных языков программирования существуют свои средства:

C/C++:

- Blast;
- Parasoft;
- Cppcheck;
- PVS-Studio.

Java:

- FindBugs;
- Parasoft Jtest.

.NET:

- FxCop;
- PVS-Studio;
- Resharper.

Python:

- Pychecker;
- Pylint.

Таким образом, оценка сложности проектируемой 3D ГИС с использованием свободно распространяемых библиотек позволяет вычислить финансовые и временные «выигрыши» относительно создания систем, разрабатываемых с «нуля».

На основе представленных выше метрик проведена оценка эффективности проектирования 3D ГИС с использованием предложенного во второй главе CASE-средства относительно разработки системы «с нуля» (Таблица 5.2).

Таблица 5.2. Оценка эффективности проектирования 3D ГИС на основе CASE-средства

Характеристика	**Разработка «с нуля»**	**Разработка на основе CASE-средства**
Количество операторов	34 765 операторов	37 547 операторов
Количество готовых библиотек	0 библиотек	10 библиотек
Количество собственных операторов	34 765 операторов	17 865 операторов
Количество адаптеров	0 адаптеров	6 адаптеров (2 130 операторов)
Сложность разработки	34 765 операторов	19995 операторов

Эффективность разработки 3D ГИС на основе CASE-средства – 14 770 операторов, что составляет 42% от сложности разработки 3D ГИС «с нуля».

Выводы

1. Представлены требования к функционированию системы 3D моделирования проектных решений, на основе которых система наиболее полно может визуализировать и оценивать полученные CASE-средством проектирования 3D ГИС проектные решения.
2. Предложена оценка адекватности проектного решения на основе критериев принятия решений по трёхмерной обстановке, которая позволяет определить необходимость дальнейшей доработки проектного решения или возможность его программной реализации.
3. Предложена методика испытаний системы 3D моделирования проектных решений по основным функциям, выполняемым системой для наиболее полной оценки проверяемых проектных решений.
4. Разработаны компоненты системы 3D моделирования и визуальной оценки проектных решений, обеспечивающие такие функции как: визуализацию трехмерной обстановки, взаимодействие с внешними системами, работу с источниками обстановки, имитацию движения трехмерных объектов на поверхности Земли, которые задают основную функциональность ядра разрабатываемой 3D ГИС.

Заключение

На основе проведенного анализа существующих геоинформационных систем и особенностей их проектирования, а также существующих CASE-средств проектирования информационных систем делается вывод о том, что для повышения качества и уменьшения сложности проектирования современных трехмерных геоинформационных систем возникает необходимость построения специализированного CASE-средства проектирования 3D ГИС отображения ситуационной обстановки на основе свободно-распространяемых ресурсов.

Задачи, поставленные в рамках диссертационной работы, решены и к основным результатам можно отнести следующее.

1. Разработана двухкомпонентная функционально-ресурсная модель проектирования 3D ГИС отображения ситуационной обстановки с использованием свободно распространяемых ресурсов (библиотек), включающая – функциональную модель, которая используется для формирования покрытий заданной функциональности 3D ГИС на основе матрицы соответствия доступных для использования функций и реализующих их библиотек, и – геометрическую модель – для оценки затрат на его разработку в виде круговой диаграммы.
2. Разработан алгоритм проектирования 3D ГИС на базе свободно-распространяемых ресурсов и собственных разработок с использованием функциональной и геометрической моделей, обеспечивающих формирование функциональных покрытий, их оценку и выбор среди них оптимального покрытия с последующей его программной реализацией.
3. Разработана модель и действующая программная реализация специализированного CASE-средства проектирования 3D ГИС отображения ситуационной обстановки, позволяющего использовать современные информационные технологии в виде свободно-распространяемых ресурсов: библиотек, текстур, моделей высот и глубин, 3D моделей объектов, а также формировать готовые проектные решения, отвечающие требованиям на разработку 3D ГИС и делать их оценку качества для последующей реализации.
4. Разработана база моделей, диаграмм 3D ГИС, алгоритмов проектирования и функционирования 3D ГИС отображения ситуационной обстановки, которые позволяют генерировать проектные решения 3D ГИС и реализовывать их с использованием свободно распространяемых библиотек и собственных разработок.

5. Создана система 3D моделирования проектных решений, которая позволяет проводить их визуальную оценку в большинстве режимов обстановки, встречаемых в реальном мире.
6. Разработана методика проведения 3D моделирования проектных решений.

Полученные результаты исследования могут найти практическое применение при разработке трехмерных геоинформационных систем и в смежных предметных областях, использоваться на предприятиях, разрабатывающих средства автоматизации процессов мониторинга местности с учётом рельефа в режиме, близком к реальному времени.

Список литературы

1. Абламейко С.В., Апарин Г.П., Крючков А.Н., Географические системы. Создание цифровых карт. Минск: Институт технической кибернетики НАН Беларуси, 2000. -276 с.
2. Андерсон Дж. Дискретная математика и комбинаторика. – М. : Вильямс, 2006. – 960 с.
3. Андреев А.М., Березкин Д.В., Куликов Ю.В. и др. Объектно-ориентированный подход к проектированию ГИС // Геодезия и картография. 1995. – №9. – С.41-44.
4. Андрианов Д.Е. Геоинформационные системы: исследование, анализ и разработка. М.: Государственный научный центр Российской Федерации -ВНИИгеосистем, 2004. - 184 с.
5. Андрианов Д.Е., Еремеев С.В. Некоторые методы описания геоинформационных систем // Данные, информация и их обработка: Сборник научных статей М.: Горячая линия – Телеком, 2002, С. 87-97.
6. Барсегян А.А. Технологии анализа данных: Data Mining, Text Mining, Visual Mining, OLAP. 2 изд. // А.А. Барсегян. Санкт-Петербург
7. Беляков С.Л. Изменение сложности картографических изображений в сетевой геоинформационной справочной системе // Информационные технологии. – 2001. – №6. – С. 31-35.
8. Беляков С.Л. Картографические образы в информационно-управляющих системах // Приборы и системы. Управление, контроль, диагностика. – 2000. – № 5.
9. Беляков С.Л. Нечеткие знания и вывод в геоинформационной системе // Информационные технологии. – 2002. – №1.
10. Беляков С.Л. Об интеллектуальной обработке запросов сетевой геоинформационной справочной системой // Геоинформатика. – 2001. – №2.
11. Беляков С.Л. Обработка запросов сервером геоинформационной справочной системы // Программные продукты и системы. – 2001. – № 1. – С.30-33.
12. Беляков С.Л. Репликация электронной карты в геоинформационной справочной системе // Программные продукты и системы. – 2002. – № 1.
13. Беляков С.Л. Сетевое представление картографической основы геоинформационной системы для повышения динамичности диалогового взаимодействия // Геоинформатика. 2000. – № 1. – С.24-29.
14. Берлянт, А.М. Картографический словарь Текст. / А.М. Берлянт. М.: Научный мир. – 2005. – 424 с.
15. Берлянт, А.М. Взаимодействие картографии и геоинформатики / А.М. Берлянт. – М.: Научный мир. – 2000. – 189 с.
16. Берлянт, А.М. Картографический словарь / А.М. Берлянт. – М.: Научный мир. – 2005. – 424 с.

17. Бершадский А.М. Геоинформационные технологии и системы: Учеб. пос. / А.М. Бершадский, А.С. Бождай. Пенза: ПГУ,. – 2001. - 41 с.
18. Берштейн Л.С., Беляков С.Л. Геоинформационные справочные системы. Научное издание. Таганрог: Изд-во ТРТУ, 2001. 160 с.
19. Бугаевский, Л.М. Геоинформационные системы. – М.: Златоуст, 2000. – 221 с.
20. Булаев А.А. Оценка адекватности проектных решений 3D ГИС отображения ситуационной обстановки. // Наука и бизнес: пути развития. – М. : ТМБпринт. – 2017. – № 8 (74)
21. Булаев А.А., Кукин Е.С., Леонтьев М.Ю., Смагин А.А. Система отображения морской, наземной и воздушной обстановки на трехмерной модели Земли // Учёные записки УлГУ. – 2014. – № 1(6) . – С. 5-11.
22. Булаев А.А., Липатова С.В., Мерзляков Д.А., Смагин А.А. CASE-средство проектирования 3D-ГИС на основе свободно распространяемых библиотек // Автоматизация процессов управления. – 2016. – № 2 (44). – С. 35-44.
23. Булаев А.А., Липатова С.В., Смагин А.А. Система автоматизированного проектирования и моделирования 3D ГИС // Вестник НГИЭИ. – 2017. - №4.
24. Булаев А.А., Смагин А.А. Проектирование системы 3D-ГИС визуализации на базе свободно распространяемых ресурсов // Информационные системы и технологии 2015: матер. III Международной науч.-техн. интернет-конф. – URL: http://youconf.ru/isit2015/members/view/401
25. Булаев А.А., Смагин А.А., Липатова С.В. Система автоматизированного проектирования и моделирования 3D ГИС // Перспективные информационные технологии. – 2017. – С. 51-54
26. Варламов, А.А. Земельный кадастр. Географические и земельные информационные системы / А.А. Варламов, С.А. Гальченко. Т. 6. – М.: Колос, 2005. – 480 с.
27. Василейский А.С. Железнов М.М., Зиман Я.Л. Алгоритмы координатной привязки космических видеоданных по навигационным измерениям.// Изв. Вузов. Приборостроение. – 2003. – Т. 46. – №4. – С. 37-43.
28. Васильев К.К., Павлыгин Э.Д., Гуторов А.С. Многомодельные алгоритмы обработки данных системы мобильных РЛС // Автоматизация процессов управления. – 2014. – № 4 (38). – С. 4–13.
29. Васильева Е.Е., Марков Н.Г. Методы, алгоритмы и информационные системы для управления геолого-техническими мероприятиями на фонде скважин нефтегазодобывающего предприятия // Известия Волгоградского государственного технического университета. – 2016. – № 11 (190). – С. 31-37.
30. Востокова, А.В. Оформление карт. Компьютерный дизайн Текст.: учебник / А.В. Востокова, С.М. Кошель, Л.А. Ушакова. М.: Аспект Пресс. – 2002. – 288 с.

31. Географический атлас России Атлас. М.: Роскартография. – 2005. – 186 с.
32. Глазунов, В.В. Геоинформационные системы / В.В. Глазунов. – СПб.: ВИРГ-рудгеофизика. – 2002. – 82 с.
33. Глушков В.В., Насретдинов К.К., Шаравин А.А. Космическая геодезия: методы и перспективы развития. / Москва: Институт политического и военного анализа. – 2002. – 448 с.
34. Горбачев И.В., Похилько А.Ф. Представление процессов проектирования в функционально адаптируемой форме для хранения классов проектных решений // Программные продукты и системы. – 2013. – № 1. – С. 17.
35. ГОСТ 19.201-78 Единая система программной документации. Техническое задание. Требования к содержанию и оформлению. – М. : ИПК Издательство стандартов, 2010. – 3 с.
36. ГОСТ 22487-77. Проектирование автоматизированное. Термины и определения.
37. ГОСТ 28441–99. Картография цифровая. – М.: Госстандарт России, 2000. – 10 с.
38. ГОСТ 34.602-89 Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. – М. : ИПК Издательство стандартов, 2002. – 12 с.
39. ГОСТ Р 51353-99. Геоинформационное картографирование. Метаданные электронных карт. Состав и содержание Текст. М:: Госстандарт России, 1999.
40. ГОСТ Р 51608-2000. Карты цифровые топографические. Требования к качеству Текст. М.: Госстандарт России, 2000.
41. ГОСТ Р 52055-2003. Геоинформационное картографирование. Пространственные модели местности. Общие требования Текст. — М.: Госстандарт России, 2003.
42. ГОСТ Р 52155–2003. Географические информационные системы: федеральные, региональные, муниципальные. Общие технические требования. – М.: Госстандарт России, 2004. – 11 с.
43. ГОСТ Р ИСО/МЭК 15408-2-2013 Информационная технология (ИТ). Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 2. Функциональные компоненты безопасности – М. : Стандартинформ, 2014. – 161 с.
44. ГОСТ Р ИСО/МЭК 27033-1-2011 Информационная технология (ИТ). Методы и средства обеспечения безопасности. Безопасность сетей. Часть 1. Обзор и концепции. – М. : Стандартинформ, 2012. – 73 с.
45. Гурвиц Г.А. Microsoft Access 2010. Разработка приложений на реальном примере // БХВ-Петербург. – 2010.
46. Данные для построения 3D-моделей местности // ГИС ПАНОРАМА - Технологии. – URL: https://gisinfo.ru/3d/databuild.htm

47. ДеМерс, Майкл Н. Географические информационные системы / Майкл Н. ДеМерс. – М., 1999. – 262 с.
48. Диденко Д.А. Разработка модели оценки качества информации в ГИС // Известия ЮФУ. Технические науки. – 2010. – №4.
49. Дискретная математика и математические вопросы кибернетики / Ю.Л. Васильев, Ф.Я. Ветухновский, В.В. Глаголев, Ю.И. Журавлев, В.И. Левенштейн, С.В. Яблонский. – М. : Наука. – 1974.
50. Добровольский А. Интеграция приложений: методы взаимодействия, топология, инструменты // Открытые системы. – 2006. – № 9. – С. 30–34.
51. Дышленко С.Г. Использование адаптивного механизма «ГИС Конструктор» проектирования ГИС в области навигации // Землеустройство, кадастр и мониторинг земель. – 2010. – № 6. – С. 77–82.
52. Дышленко С.Г. Трехмерное моделирование в ГИС // Перспективы науки и образования. – 2014. – №2 (8). – С. 28-34.
53. Дышленко С.Г., Цветков В.Я. Особенности проектирования ГИС пользователя на основе базового комплекта ГИС «Карта 2011» // Землеустройство, кадастр и мониторинг земель . – 2010. – № 8. – С. 79–84.
54. Еремеев С.В. Алгоритм размещения слоев на цифровой карте в ГИС // Геоинформатика. 2005. №2. С 22-27.
55. Еремеев С.В. Моделирование геоинформации с использованием топологических отношений и структур. // Проблемы передачи и обработки информации в сетях и системах телекоммуникаций: Материалы 13-й Международной науч.-техн. конф. Рязань: РГРТА, 2004, С. 199.
56. Еремеев С.В. Способы представления геоинформации. // Проблемы передачи и обработки информации в сетях и системах телекоммуникаций: Материалы 11-й Международной науч.-техн. конф. Рязань: РГРТА, 2002, С. 79-81.
57. Еремеев С.В., Епифанов Д.В. Расширение информационной модели данных в ГИС технологиях // Обработка информации: методы и системы: Сборник научных статей М.: Горячая линия - Телеком, 2003, С. 67-72.
58. Еремеев С.В., Епифанов Д.В. Создание модели пространственных данных // Гагаринские чтения. Москва, 2004. С.
59. Ефимов Г. Жизненный цикл информационных систем // Сетевой журнал. – 2001. – № 2. – URL: http://www.setevoi.ru/cgi-bin/text.pl/magazines/2001/2/44
60. Жалковский, Е.А. Цифровая картография и геоинформатика. Краткий терминологический словарь / Е.А. Жалковский. – М.: Картгеоцентр – Геодезиздат, 1999, 46 с.

61. Закревский А.Д., Торопов Н.Р. Минимизация булевых функций многих переменных в классе ДНФ – итеративный метод и программная реализация // ПДМ. – 2009. – № 1 (3). – С. 5–14.
62. Зеркаль О.В., Антипина И.С., Терешкова Н.Ю. Основные подходы к применению ГИС-технологий при проведении мониторинга экзогенных геологических процессов. // Проблемы современной инженерной геологии. – 2003. – Т.153. – С.64.
63. Иванов К.А., Кампанья М., Кудинов А.В., Марков Н.Г. Волонтерские геоинформационные системы в управлении муниципалитетами и регионами // Информационное общество. – 2014. – № 3. – С. 10-19.
64. Иванов К.А., Кудинов А.В., Марков Н.Г., Кампанья М., Масса П. Взаимодействие и интеграция геоинформационных веб-сервисов для систем поддержки принятия решений в пространственном планировании территорий // Фундаментальные исследования. – 2015. – № 2-15. – С. 3267-3271.
65. Инструкция по фотограмметрическим работам при создании цифровых топографических карт и планов. (под ред. С.С. Нехина). М.: ЦНИИГАиК, 2002, 100 с.
66. Калантаев П.А. Семантическая организация пространственных данных // Институт Вычислительной Математики и Математической Геофизики СО РАН. – URL: http://loi.sscc.ru/BDM/Katrina/attr/htm/geosemantic_org.htm
67. Карманов А.Г. Учебное пособие по курсу геоинформатика / А.Г. Карманов. – Санкт-Петербург. – 2012. – 116 с.
68. Карпик, А.П. Методологические и технологические основы геоинформационного обеспечения территорий: монография / А.П. Карпик. – Новосибирск: СГГА, 2004. – 260 с.
69. Клюшниченко, В.Н. Об одном из возможных путей создания автоматизированных информационных систем / В.Н. Клюшниченко, Н.В. Клюшниченко; Сиб. гос. геодез. акад. – Новосибирск, 2001. – 6 с. – Деп. в ОНИПР ЦНИИГАиК 19.07.2001, № 740-гд 2001.
70. Клюшниченко, Н.В. Применение информационных технологий для обслуживания заявителей. Методы дистанционного зондирования и ГИС-технологии для оценки состояния окружающей среды, инвентаризация земель и объектов недвижимости / Н.В. Клюшниченко, В.А. Середович // Регистрация прав недвижимости имущества и сделок по нему: 6-я Междунар. науч. конф., 11–18 мая 2002 года: тез. докл. – Испания, 2002. – С. 33–35.
71. Ковин Р.В. Геоинформационные системы / Р.В. Ковин, Н.Г. Марков // Федеральное агентство по образованию, Гос. образовательное учреждение высш. проф. образования «Томский политехнический ун-т». – Томск. – 2008.

72. Козловский С.В. Методические аспекты, принципы и последовательность организации геоинформационной системы в инженерной геологии // Инженерная геология // ОАО «ПНИИИС». – М.:2010. – №1. – С.18-22.
73. Косяков С.В. Геоинформационные системы в управлении и производстве: Учеб. пос. С.В. Косяков. Иваново: Иван. гос. энергет. ун-т, 2001.-99 с.
74. Краткое введение в ГИС. Часть 2: Векторные данные // Gis-Lab. – URL: http://gis-lab.info/qa/gentle-intro-gis-2.html
75. Кулагин В.П., Цветков В.Я. Геоинформационные и информационные технологии // Геодезия и картография. – 2002. – № 3. – С. 41–43.
76. Лёвин Б.А., Круглов В.М., Матвеев С.И., Цветков В.Я., Коугия В.А. Геоинформатика транспорта. М.: ВИНИТИ РАН, 336 с.
77. Ловцов Д. А., Черных А. М. Геоинформационные системы: Учеб. пособие // Под ред. Д. А. Ловцова. – М.: РГУП, – 2012. – 208 с.
78. Маклаков С.В. Bpwin и Erwin. CASE средства разработки информационных систем. М., Диалог-МИФИ, 2000.
79. Марков Н.Г. Геоинформационные системы предприятий нефтегазовой отрасли: функциональность, архитектура и перспективы развития [Электронный ресурс] / Н. Г. Марков // Известия Томского политехнического университета. Инжиниринг георесурсов / Национальный исследовательский Томский политехнический университет (ТПУ). – 2017. – Т. 328, № 9. – С. 16-32.
80. Марков Н.Г., Сонькин Д.М., Газизов Т.Т., Лещик Ю.В., Фадеев А.С., Шемяков А.О. Комбинированный алгоритм прогнозирования дорожной обстановки на основе методов нечеткого поиска в региональной навигационно-информационной системе мониторинга и управления транспортом // Доклады Томского государственного университета систем управления и радиоэлектроники. – 2013. – № 4 (30). – С. 182-190.
81. Матвеев С.И, Коугия В.А. Высокоточные цифровые модели пути и спутниковая навигация железнодорожного транспорта. М.: Маршрут. – 2005. – 288 с.
82. Матвеев С.И., Коугия В.А., Цветков В.А. Геоинформационные системы и технологии на железнодорожном транспорте. М.: Маршрут. – 2002. – 208 с.
83. Матвеев С.И., Михайлов С.В., Розенберг И.Н. и др. Концепция и программа атласа железных дорог России (170 лет железным дорогам России). М.:ИПЦ ДИК. – 2007. – 58 с.
84. Мусин О.Р. Цифровые модели для ГИС // Информационный бюллетень ГИС-Ассоциации. -1999. – №2. – С.9-10.

85. О. Л. Кузнецов, А. А. Никитин, Е. Н. Черемисина Геоинформатика и геоинформационные системы. Учебник для вузов. // Государственный научный центр Российской Федерации ВНИИгеосистем. – 2005. – 453 с.
86. Основы геоинформатики: В 2 кн. Кн. 1: Учеб. пос./ Е.Г. Капралов, А.В. Кошкарев, В.С. Тикунов и др.; Под ред. В.С. Тикунова. М.: Издательский центр «Академия». – 2004. - 352с.
87. Описание и получение данных SRTM // Gis-Lab. – URL: http://gis-lab.info/qa/srtm.html
88. Писарев, В.С. Создание электронного атласа Новосибирской области «Люби и знай свой край родной» для средней школы Текст. / В.С. Писарев, Е.В. Комисарова // ГЕО-Сибирь-2005. Т. 4, Геоинформатика.
89. Писарев, В.С. Справочно-картографические ГИС: назначение, сущность, технология и опыт реализации Текст. /В.С. Писарев // Геодезия и картография. 2008. – № 2. – С. 31-35.
90. Питенко А.А. Нейросетевой анализ в геоинформационных системах // диссертация на соискание ученой степени кандидата технических наук / Красноярск. – 2000.
91. Похилько А.Ф., Горбачев И.В., Рябов С.В. Моделирование процессов и данных с использованием CASE-технологий // Ульяновск. – 2014.
92. Похилько А.Ф., Цыганков Д.Э., Горбачев И.В. Структурно-логическое обобщение класса проектных решений с использованием функционально адаптированного представления проектных процедур // Автоматизация процессов управления. – 2016. – № 3 (45). – С. 71-78.
93. Прикладная геоинформатика / А.Д. Иванников, В.П. Кулагин, А.Н. Тихонов, В.Я. Цветков. – М.: МаксПресс. – 2005. – 336 с.
94. Пространственные типы данных / Бобов П. // ГИС-ресурс. – URL: http://loi.sscc.ru/gis/data_model/typs.htm
95. Работа с WMS и WFS в QGIS // Gis-Lab. – URL: http://gis-lab.info/qa/qgis-wms-wfs.html
96. Раклов В.П. Географические информационные системы в тематической картографии: учеб. пособие / В.П. Раклов. – М.: ГУЗ. – 2003.
97. Родионов О.В. Геоинформационные системы: Учеб. пособие О.В. Родионов, Е.Н. Коровин, А.И. Воронин. Воронеж: Воронежский государственный техн. ун-т. – 2002. - 173 с.
98. Розенберг И.Н., Соловьев И.В., Цветков В.Я. Комплексные инновации в управлении сложными организационно-техническими системами // М.: Феория. – 2010. – 248 с.
99. Розенберг И.Н., Старостина Т.А, Решение задач размещения с нечёткими данными с использованием геоинформационных систем. М.: Научный мир. – 2006. – 208 с.
100. Савиных В.П., Цветков В.Я. Геоинформационный анализ данных дистанционного зондирования. М.: Картгеоцентр – Геодезиздат. – 2001. – 228 с.

101. Садыков С.С., Андрианов Д.Е., Еремеев С.В. Формальное определение топологических отношений между картографическими объектами в ГИС // Обработка информации: методы и системы: Сборник научных статей М.: Горячая линия – Телеком. – 2003. – С. 49-58.
102. Садыков С.С., Еремеев С.В. Анализ информации в ГИС, распределенной по разным слоям // Современные управляющие и информационные системы: Сборник научных статей. Ташкент. – 2003. – С. 120125.
103. Садыков С.С., Симаков Р.А. Автоматизированный способ идентификации объектов карты//Геоинформатика. – 2004. – №2. – С. 46-51.
104. Середович, В.А. Разработка системы автоматизированного управления нефтегазодобывающим предприятием на основе интеграции ГИС и СУБД / В.А. Середович, В.А. Калюжин, А.В. Дубровский // Тез. Междунар. пром. форума Geoform+. – М.: Проспект. – 2005. – С. 21-24.
105. Смагин А.А., Булаев А.А. Профессионально ориентированная информационная сеть кафедры вуза // Учёные записки УлГУ. – 2014. – №1(6). – С. 171-179
106. Смагин А.А., Булаев А.А. Профессионально-ориентированная информационная сеть кафедры вуза // XIII Международная научно-методическая конференция образовательных организаций реального направления подготовки «Инфокоммуникационные технологии и системы связи». – 2014. – С. 153-155
107. Смагин А.А., Булаев А.А., Липатова С.В. Модель покрытия структуры программного комплекса с использованием библиотек // Автоматизация процессов управления. – 2017. – № 4 (50).
108. Смагин А.А., Липатова С.В., Смикун П.И., Мельниченко А.С. Методика построения специализированных АС на базе SOA // Автоматизация процессов управления. – 2009. – № 3. – С. 44-50.
109. Технологии передачи данных: файл-сервер, клиент-сервер, терминал-сервер // URL: http://www.itsaturn.ru/articles/article14.html
110. Технология построения инфраструктуры пространственных данных // ГИС ПАНОРАМА - Технологии. – 2017. – URL: http://gisinfo.ru/support/tehno.htm
111. Цветков В.Я. Автоматизированные земельные информационные системы. -М.: Минпромнауки. – 2001. – 140 с.
112. Цветков В.Я. Модель геоданных для управления транспортом // Успехи современного естествознания. – 2009. – №4. – С.50-51.
113. Цветков В.Я. Пространственные отношения в геоинформатике // Международный научно-технический и производственный журнал «Науки о Земле». Выпуск 01-2012. – С.59-61.
114. Цветков, В.Я. Модели в информационных технологиях. М. – 2006. – 85 с.

115. Шабалкин Д.Ю., Липатова С.В. Построение интегрированной поливендорной цифровой среды, обеспечивающей поддержку жизненного цикла воздушного судна на основе сервис-ориентированного подхода // Известия Самарского научного центра Российской академии наук. – 2013. – Т. 15. – № 4–3. – С. 653–661.
116. Шаши Шекхар, Санжей Чаула. Основы пространственных баз данных. / Пер. с англ.- М.: КУДИЦ-ОБРАЗ. – 2004. – 336 с.
117. Шелухин О.И. Моделирование информационных систем: учеб. пособие для вузов. – 2-е изд., перераб. и доп. – М. : Горячая линия-Телеком, 2012. – 516 с.
118. Эддоус М., Стэнсфилд Р. Методы принятия решений / пер. с англ. под ред. член-корр. РАН И.И. Елисеевой. – М. : Аудит, ЮНИТИ, 1997. – 590 с.
119. ЮЗ.Конон Н.И. Концепция математической модели геоинформационных систем // Геодезия и картография. – 2001. – № 6. – С. 48-54.
120. ArcGIS 9: Что такое ArcGIS / ESRI. Электронный ресурс.
121. Computational neural netwoks for geophisical data processing Ed. by Mary M. Poulton Amsterdam: Pergamon, 2001 XIII, 335 p.
122. Dunn M., Hiclcey R., The effect of slope algorithms on slope estimates within a GIS. Cartography (Austral.). - 1998. - 27. - №1. - P. 9-15.
123. Flacke W., Kraus B.: Koordinatensysteme in ArcGIS: Praxis der Transformation und Projektionen. 1 .Auflage. Norden Halmstadt: Points Verlag, 2003.
124. Goodchild Michel F. GIS and transportation: status and challenges. // Geonformatica, 2004, №2, P. 127-139.
125. ISO OSI/TC 211 Geographicslnformation/ Geomatics, International Draft Standart
126. Markov N.G., Vasilyeva E.E., Evsyutkin I.V. The intellectual information system for management of geological and technical arrangements during oil field exploitation // Journal of Physics: Conference Series. –2017. – Т. 803. – № 1. – С. 012093.
127. Stojic M. 3D Geographic imaging // GIM International. March 2000. V. 14. N3.P 70-73.
128. Winter Stephan, Frank AndreW U. Topology in raster and vector representation. Geoinformatica. 2000. 4 №1 p. 35-65.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЕ КОДЫ CASE-СРЕДСТВА ПРОЕКТИРОВАНИЯ 3D ГИС ОТОБРАЖЕНИЯ СИТУАЦИОННОЙ ОБСТАНОВКИ

Файл конфигурации CASE-средства “config.ini”

```
[main]
database=main.db
[stat]
custom_count=10
line_count=0.01
time_execute=0.02
input_param_avg=1
broker_count=1
```

Пример файла сохраненного проектного решения “case.csg”

{"lang": ["Python"], "platform": ["Mac OS"], "models": ["pr_model", "func_model", "obst_model", "brock_model", "libr_model", "str_model"], "func": ["\u0414\u043e\u0431\u0430\u0432\u043b\u0435\u043d\u0438\u0435 \u0442\u0435\u043a\u0441\u0442\u0443\u0440"]}

Функция построения дерева библиотек функциональных покрытий

```
def showPrTree(self):
    matrix = "<table cellspacing=\"2\" border=\"1\"><tr><td></td>"
    funcs = []
    for f in self.buffer['flmap']:
        funcs.append(f)
    funcs.sort()
    for f in funcs:
        matrix += "<td>%s</td>" % f
    matrix += "</tr>"
    for l, fs in self.buffer['lfmap'].items():
        matrix += "<tr><td>%s</td>" % l
        for f in funcs:
            if f in fs:
                matrix += "<td>1</td>"
            else:
                matrix += "<td>0</td>"
        matrix += "</tr>"
    matrix += "</table>"
    graph = """<!DOCTYPE html>
```

```
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Collapsible Tree Example</title>
    <style>
       .node circle {
         fill: #fff;
         stroke: steelblue;
         stroke-width: 3px;
       }
       .node text { font: 12px sans-serif; }
       .link {
         fill: none;
         stroke: #ccc;
         stroke-width: 2px;
       }
       #my-div {
         margin: 0 auto;
       }
       .link.active {
          stroke: #e45959;
       }
       circle.cactive {
          stroke: #f00;
                }
    </style>
    <script type="text/javascript">
    </script>
  </head>
  <body>
    <div>%s</div>
    <canvas id="canvas" style="float: left;"></canvas>
    <div id="my-div"></div>
    <!-- load the d3.js library -->
    <script src="http://d3js.org/d3.v3.min.js"></script>
    <script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
    <script type="text/javascript" >
       var canvas = document.getElementById('canvas');
                                          canvas.width  = 500;
                                          canvas.height = 500;
                                          var context = canvas.getContext("2d");
                                          var centerX = Math.floor(canvas.width / 2);
```

```
var centerY = Math.floor(canvas.height / 2);
var radius = Math.floor(canvas.width / 2);
var apiradius = 50;
var startAngle = 0;
/*var pattern;
var imageObj = new Image();
imageObj.onload = function() {
        pattern = context.createPattern(imageObj, 'repeat');
};
imageObj.src =
'http://www.html5canvastutorials.com/demos/assets/wood-pattern.png';*/
function clear() {
    context.clearRect(0, 0, canvas.width, canvas.height);
}
        function draw(libs) {
    startAngle = 0;
    context.clearRect(0, 0, canvas.width, canvas.height);
            context.beginPath();
            context.arc(centerX, centerY, radius,
                        0, 2*Math.PI);
            context.closePath();
            context.strokeStyle = "#000000";
            context.stroke();
            for (var i = 0; i < libs.length; i++) {
                drawSegment(canvas, context, libs[i]);
            }
            context.beginPath();
            var endAngle = 2*Math.PI - libs[libs.length-
1].fcount*2*Math.PI / libs[libs.length-1].ftotal;
            context.arc(centerX, centerY, apiradius,
                        0, endAngle);
            context.closePath();
            context.fillStyle = "#fff";
            context.fill();
            context.strokeStyle = "#000000";
            context.stroke();
            startAngle = 0;
            for (var i = 0; i < libs.length; i++) {
                drawAPI(canvas, context, libs[i]);
            }
        }
        function drawSegment(canvas, context, lib) {
```

```
		context.save();
		var arcSize = lib.fcount*2*Math.PI / lib.ftotal;
		var endingAngle = startAngle + arcSize;
		context.beginPath();
		context.moveTo(centerX, centerY);
		context.arc(centerX, centerY, radius,
					startAngle, endingAngle, false);
		context.closePath();
		if (lib.type == 'custom')
			context.fillStyle = 'lightgray';
		else
			context.fillStyle = tag2color(lib.name);
		context.strokeStyle = "#000000";
		context.fill();
		context.stroke();
		context.restore();
		drawSegmentLabel(canvas, context, lib);
		startAngle = endingAngle;
	}
	function drawAPI(canvas, context, lib) {
		if (lib.type == 'custom')
			return;
		context.save();
		var arcSize = lib.fcount*2*Math.PI / lib.ftotal;
		var endingAngle = startAngle + arcSize;
		context.beginPath();
		context.moveTo(centerX, centerY);
		var c = Math.cos(Math.PI/2 + startAngle);
		var s = Math.sin(Math.PI/2 + startAngle);
		var x1 = centerX + apiradius * s;
		var y1 = centerY - apiradius * c;
		c = Math.cos(Math.PI/2 + endingAngle);
		s = Math.sin(Math.PI/2 + endingAngle);
		var x2 = centerX + apiradius * s;
		var y2 = centerY - apiradius * c;
		context.lineTo(x1, y1);
		context.lineTo(x2, y2);
		//context.arc(centerX, centerY, apiradius,
					//startAngle, endingAngle, false);
		context.closePath();
		context.fillStyle = 'lightgray';
		context.strokeStyle = "#000000";
```

```
                context.fill();
                context.stroke();
                context.restore();
                startAngle = endingAngle;
        }
        function decimalToHex(d, padding) {
                var hex;
                hex = Number(d).toString(16);
                padding = typeof padding === "undefined" || padding === null ? padding = 2 : padding;
                while (hex.length < padding) {
                        hex = "0" + hex;
                }
                return hex;
        };
        function tag2color(t) {
                var baseColor, c, color, i, len, x;
                c = 0;
                t = t != null ? t : 'undefined';
                baseColor = [0x66, 0x77, 0x66];
                for (i = 0, len = t.length; i < len; i++) {
                        x = t[i];
                        c = (c + 1217 * x.charCodeAt(0)) %% 87911;
                }
                baseColor = [(baseColor[0] + (c >> 16) %% 0xa0) %% 0x100, (baseColor[1] + (c >> 8) %% 0xb0) %% 0x100, (baseColor[2] + c %% 0xc0) %% 0x100];
                color = "#" + (decimalToHex(baseColor[0])) + (decimalToHex(baseColor[1])) + (decimalToHex(baseColor[2]));
                return color;
        };
        function drawSegmentLabel(canvas, context, lib) {
                context.save();
                var x = Math.floor(canvas.width / 2);
                var y = Math.floor(canvas.height / 2);
                var angle = startAngle;
                //var endingAngle = startAngle + arcSize;
                //var angle = degreesToRadians(sumTo(data, i));
          context.translate(x, y);
          context.rotate(angle);
          var dx = Math.floor(canvas.width * 0.5) - 10;
          var dy = Math.floor(canvas.height * 0.05);
          context.textAlign = "right";
```

```
                                    var fontSize = Math.floor(canvas.height / 25);
                                    context.font = fontSize + "pt Helvetica";
                                    context.fillText(lib.name, dx, dy);
                                    context.restore();
                                }
</script>
<script>
    var treeData = JSON.parse('%s');
    // ************** Generate the tree diagram      *****************
    var margin = {top: 40, right: 120, bottom: 20, left: 120},
        width = 960 - margin.right - margin.left,
        height = 500 - margin.top - margin.bottom;
    var i = 0;
        var mpath = [0, 1, 2, 3, 4, 9];
    var tree = d3.layout.tree()
        .size([height, width]);
    var diagonal = d3.svg.diagonal()
        .projection(function(d) { return [d.x, d.y]; });
    var svg = d3.select("#my-div").append("svg")
        .attr("id", "my-svg")
        .attr("width", width + margin.right + margin.left)
        .attr("height", height + margin.top + margin.bottom)
        .style("overflow", "visible")
      .append("g")
        .attr("transform", "translate(" + margin.left + "," + margin.top + ")");
    root = treeData[0];
    showType = "short";
    function changeType(selectObject) {
        showType = selectObject.value;
        svg.selectAll("text.node-text").text(function(d) {
            //if (showType == "short")
            if (d.type != "end")
                return d.name+" Количество функций: "+d.fcount+" Коэффициент затрат: "+d.koef;
            else
                return d.name;
            /*if (showType == "lang")
                return d.name+" / "+(d.lang || "");
            if (showType == "platform")
                return d.name+" / "+(d.platform || "");*/
          });
        updateSize();
    }
```

```
function update(source) {
 // Compute the new tree layout.
 var nodes = tree.nodes(root).reverse(),
       links = tree.links(nodes);
 // Normalize for fixed-depth.
 nodes.forEach(function(d) { d.y = d.depth * 100; });
 // Declare the nodes
 var node = svg.selectAll("g.node")
       .data(nodes, function(d) { return d.id || (d.id = ++i); });
 // Enter the nodes.
 var nodeEnter = node.enter().append("g")
       .attr("class", "node")
       .attr("transform", function(d) {
             return "translate(" + d.x + "," + d.y + ")"; });
 nodeEnter.append("circle")
       .attr("r", 10)
      .on("click", function (d) {
                  var isActive = $(this).hasClass('cactive');
            $('.cactive').removeClass('cactive');
                  if (!isActive) {
                        $(this).addClass('cactive');
                        var libs = [];
                        libs.push(d);
                        var ftotal = d.ftotal;
                        var fcount = d.fcount;
                        while (d.parent != "null") {
                              d = d.parent;
                              if (d.parent != "null") {
                                    libs.push(d);
                                    fcount += d.fcount;
                              }
                        }
                        var cus = {'name': 'Соб. разработка', 'ftotal': ftotal, 'fcount': ftotal-fcount,
'type': 'custom'};
                        libs.push(cus);
                        draw(libs);
                  } else {
                     clear();
                  }
       })
       .style("fill", function(d) { return (d.type == "end" ? "steelblue" : "#fff"); });
 nodeEnter.append("text")
```

```
        .attr("y", function(d) {
              return -18; })
        .attr("dy", ".35em")
        .attr("class", "node-text")
        .attr("text-anchor", "middle")
        .text(function(d) {
           return d.name;
        })
        .style("fill-opacity", 1);
nodeEnter.append("text")
        .attr("y", function(d) {
              return 18; })
        .attr("dy", ".35em")
        .attr("class", "node-text")
        .attr("text-anchor", "middle")
        .text(function(d) {
           if (d.type != "end")
              return d.fcount+" функций";
           else
              return "";
        })
        .style("fill-opacity", 1);
nodeEnter.append("text")
        .attr("y", function(d) {
              return 36; })
        .attr("dy", ".35em")
        .attr("class", "node-text")
        .attr("text-anchor", "middle")
        .text(function(d) {
           if (d.type != "end")
              return "K = "+d.koef;
           else
              return "";
        })
        .style("fill-opacity", 1);
// Declare the links
var link = svg.selectAll("path.link")
        .data(links, function(d) { return d.target.id; });
// Enter the links.
link.enter().insert("path", "g")
        .attr("class", function(d) {
           if ((io = mpath.indexOf(d.source.cid)) >= 0 && mpath[io+1] == d.target.cid)
```

```
                        return "link active";
                    return "link";
                })
                .attr("d", diagonal);
        }
        update(root);
        updateSize();
        function updateSize() {
            var svgEl = document.getElementById("my-svg"),
                bb = svgEl.getBBox();
            svgEl.style.height = bb.y + bb.height;
            svgEl.style.width = bb.x + bb.width;
            //alert(svgEl.style.width );
            document.getElementById("my-div").style.width = svgEl.style.width;
        }
    </script>
  </body>
</html>
""" % (matrix, json.dumps(self.data['tree']))
        print graph
        graph = fromUtf8(graph)
        self.prtree_widget.ui.webView.setHtml(graph)
```

Функция генерации проектных решений 3D ГИС

```
    def generate(self):
        msgBox = QtGui.QMessageBox()
        msgBox.setWindowTitle(fromUtf8("Внимание!"))
        if len(self.data['objects']) == 0:
            msgBox.setText(fromUtf8("Необходимо выбрать объекты!"))
            msgBox.exec_()
            #return
        if len(self.data['func']) == 0:
            msgBox.setText(fromUtf8("Необходимо выбрать функции!"))
            msgBox.exec_()
            return
        c = self.conn.cursor()
        ids = ','.join(str(x) for x in self.data['func'])
        q = """SELECT l.id,l.name,l.db_tex,l.db_obj,l.db_elev,fl.id_func,l.language,fl.line_count,fl.time_execute,
fl.input_param_count FROM libraries as l
                INNER JOIN funcs_libs as fl ON l.id = fl.id_lib
                LEFT JOIN functions as f ON f.id = fl.id_func
                WHERE fl.id_func in (%s)""" % ids
```

```
c.execute(q)
self.data['projects'] = []
funcs = {}
for row in c:
    key = row['id_func']
    if not (key in funcs):
        funcs[key] = []
    funcs[key].append(row)
#print funcs
def get_projects(keys):
    out = []
    data = []
    if len(keys) > 1:
        data = get_projects(keys[1:])
    func = funcs[keys[0]]
    for f in func:
        if len(data):
            for d in data:
                o = []
                o.append(f)
                o.extend(d)
                out.append(o)
        else:
            o = []
            o.append(f)
            out.append(o)
    return out
def get_exp_value():
    c = self.conn.cursor()
    q = """SELECT * FROM functions"""
    c.execute(q)
    funcs = set()
    for row in c:
        funcs.add(row['id'])
    flen = len(funcs)
    self.data['flen'] = flen
    ls = dict()
    for project in self.data['projects']:
        pr = dict()
        for p in project:
            if p['id'] is 0:
                continue
```

```
            if not (p['id'] in pr):
                pr[p['id']] = 0
            pr[p['id']] += 1
        for p in pr:
            if not (pr[p] in ls):
                ls[pr[p]] = 0
            ls[pr[p]] += 1
    prob = dict()
    total = sum(ls.values())
    for p in ls:
        x = round(math.radians(360 / flen) * p, 2)
        prob[x] = round(ls[p] * 1.0 / total, 2)
    #print ls
    #print prob
    m = 0
    for [k, v] in prob.items():
        m += round(k * v, 2)
    self.data['exp_value'] = m
def get_complexity(project):
    libs = []
    stat = {
        'custom_count': 0,
        'line_count': 0,
        'time_execute': 0,
        'input_param_count': 0,
        'libs_count': 0,
        'broker_count': 0
    }
    for p in project:
        if p['id'] == 0:
            stat['custom_count'] += 1
        else:
            stat['libs_count'] += 1
            stat['time_execute'] += p['time_execute']
            stat['line_count'] += p['line_count']
            stat['input_param_count'] += p['input_param_count']
            if self.data['mainLang'] != p['language']:
                stat['broker_count'] += 1
            libs.append(p['name'])
    stat['input_param_avg'] = stat['input_param_count']/stat['libs_count']
    stat['custom_count_per'] = stat['custom_count']*100/len(project)
    libs = list(set(libs))
```

```
    weight = dict(self.config.items('stat'))
    complexity = 0
    for k in weight:
        if k in stat:
            complexity += stat[k] * float(weight[k])
    return complexity
self.data['projects'] = get_projects(funcs.keys())
i = 1
for key, pr in enumerate(self.data['projects']):
    fs = []
    funcs = []
    changed = False
    for p in pr:
        funcs.append(p['id_func'])
    #print funcs
    for f in self.data['func']:
        if not (f in funcs):
            pr.append({'id': 0, 'name': u"Собственная разработка", 'id_func': f})
            changed = True
    if changed:
        self.data['projects'][key] = pr
complexities = {}
for key, pr in enumerate(self.data['projects']):
    complexities[key] = get_complexity(pr)
complexities = sorted(complexities.items(), key=operator.itemgetter(1))
colors = {}
for k, cl in enumerate(complexities):
    if k < len(complexities) / 3:
        colors[cl[0]] = '#3CB371'
    elif k < 2 * len(complexities) / 3:
        colors[cl[0]] = '#DAA520'
    else:
        colors[cl[0]] = '#CD5C5C'
    #print c
self.data['classes'] = {}
for key, pr in enumerate(self.data['projects']):
    libs = []
    self.data['classes'][key] = {}
    for p in pr:
        if p['id'] != 0:
            lclass = {'id': p['id'], 'name': p['name'], 'vars': [], 'funcs': []}
            lclass['vars'].append({'name':'id','scope':'public','type':'int'})
```

```
                lclass['vars'].append({'name':'name','scope':'public','type':'string'})
                lclass['funcs'].append({'name':'create()','scope':'public','type':'void'})
                q = """SELECT * FROM lib_attrs WHERE id_lib = %s""" % p['id']
                c.execute(q)
                for r in c:
                    if r['atype'] == 'variable':
                        lclass['vars'].append({'name':r['name'],'scope':r['scope'],'type':r['type']})
                    elif r['atype'] == 'function':
                        lclass['funcs'].append({'name':r['name'],'scope':r['scope'],'type':r['type']})
                self.data['classes'][key][p['id']] = lclass
            libs.append(p['name'])
        libs = list(set(libs))
        item = QtGui.QListWidgetItem("%s %d (%s)" % (fromUtf8("Проектное решение"), i, ", ".join(libs)))
        item.setForeground(QtGui.QColor(colors[key]))
        self.project_widget.ui.listWidget.addItem(item)
        i+=1
    get_exp_value()
    self.project_widget.ui.prModelButton.setEnabled(False)
    self.project_widget.ui.prLibrButton.setEnabled(False)
    self.project_widget.ui.prHFileButton.setEnabled(False)
    self.project_widget.ui.prSaveButton.setEnabled(False)
    self.project_widget.ui.strButton.setEnabled(False)
    self.project_widget.ui.funcButton.setEnabled(False)
    self.project_widget.ui.classButton.setEnabled(False)
    self.project_widget.ui.geoModelButton.setEnabled(False)
    self.central_widget.setCurrentWidget(self.project_widget)
```

Функция построения графика зависимости коэффициентов площадей от угла сектора

```
def graphButtonClicked(self):
    data = list()
    slibs = dict()
    sfuncs = dict()
    for project in self.data['projects']:
        custom_count = 0
        libs = dict()
        for p in project:
            if p['id'] == 0:
                continue
            if not (p['id'] in libs):
                libs[p['id']] = 0
```

```
                libs[p['id']] += 1
            ll = len(libs)
            if not (ll in slibs):
                slibs[ll] = 0
            slibs[ll] += 1
            for l in libs:
                if not (libs[l] in sfuncs):
                    sfuncs[libs[l]] = 0
                sfuncs[libs[l]] += 1
                data.append("[%s, %s, null]" % (len(libs), libs[l]))
        #print slibs
        #print sfuncs
        #print data
        sllen = sum(slibs.values())
        for s in slibs:
            slibs[s] = round(slibs[s] * 1.0 / sllen, 2)
        sflen = sum(sfuncs.values())
        for s in sfuncs:
            sfuncs[s] = round(sfuncs[s] * 1.0 / sflen, 2)
        fm = 0
        for s in slibs:
            fm += s * slibs[s]
        ff = 0
        for s in sfuncs:
            ff += s * sfuncs[s]
        data.append("[%s, null, %s]" % (round(fm, 2), round(ff, 2)))
        graph = """<html>
    <head>
      <script type="text/javascript" src="https://www.google.com/jsapi"></script>
      <script type="text/javascript">
            google.load("visualization", "1", {packages:["corechart"]});
            google.setOnLoadCallback(drawChart);
            function drawChart() {
                var data = new google.visualization.DataTable();
                data.addColumn('number', 'Количество библиотек');
                data.addColumn('number', 'Количество функций');
                data.addColumn('number', 'Оптимальное значение');
                        data.addRows([%s]);
                var view = new google.visualization.DataView(data);
                var options = {
                    title: 'График отношения количества библиотек в покрытии к количеству функций, покрыва-
емых каждой библиотекой этого покрытия',
```

```
                is3D: true,
                vAxis: {
                  minValue:0,
                  viewWindow: {
                    min: 0
                  }
                },
                hAxis: {
                  minValue:0,
                  viewWindow: {
                    min: 0
                  }
                },
                        isStacked: true,
                series: {
                    0: { type: 'point', pointSize: 2 },
                    1: { type: 'point', pointSize: 4, color: "#f00" }
                }
              };
              var chart = new google.visualization.ScatterChart(document.getElementById('chart'));
              chart.draw(view, options);
            }
  </script>
 </head>
 <body>
  <div id="chart" style="width: 900px; height: 500px"></div>
  <div>Оптимальная точка: (%.2f, %.2f)</div>
 </body>
</html>
""" % (", ".join(data), fm, ff)
        print graph
        graph = fromUtf8(graph)
        self.graph_widget.ui.webView.setHtml(graph)
        self.central_widget.setCurrentWidget(self.graph_widget)
```

ПРИЛОЖЕНИЕ Б. ИСХОДНЫЕ КОДЫ СИСТЕМЫ 3D МОДЕЛИРОВАНИЯ ПРОЕКТНЫХ РЕШЕНИЙ

Модуль рисования векторных фигур на 3D глобусе

```
#include "dialogdrawfigure.h"
#include "ui_dialogdrawfigure.h"
#include <QColorDialog>
#include <QColor>
#include <QPainter>
#include <QPixmap>
#include <QPicture>
DialogDrawFigure::DialogDrawFigure(CoordsHandler *ch, QWidget *parent) :
    QDialog(parent),
    ch(ch),
    ui(new Ui::DialogDrawFigure),
    lineColor(QColor(0,0,0)),
    zColor(QColor(130,255,130))
{
    ui->setupUi(this);
    drawLine();
    drawPolygone();
}
DialogDrawFigure::~DialogDrawFigure()
{
    delete ui;
}
void DialogDrawFigure::on_pushButton_clicked()
{
    QColorDialog *cd = new QColorDialog(this);
    cd->open(this,SLOT(on_line_color_selected(QColor)));
}
void DialogDrawFigure::on_line_color_selected(const QColor & color)
{
    /*QPicture img;
    lineColor=color;
    QPainter p(&img);
    p.setRenderHint(QPainter::Antialiasing);
    p.setPen(color);
    p.drawLine(ui->label->width()*0.1,ui->label->height()*0.5,ui->label->width()*0.4,ui->label->height()*0.1);
    p.drawLine(ui->label->width()*0.4,ui->label->height()*0.1,ui->label->width()*0.6,ui->label->height()*0.6);
```

```
    p.drawLine(ui->label->width()*0.6,ui->label->height()*0.6,ui->label->width()*0.9,ui->label->height()*0.2);
    ui->label->setPicture(img);*/
    lineColor=color;
    drawLine();
    drawPolygone();
}
void DialogDrawFigure::drawLine()
{
    QPicture img;
    QPainter p(&img);
    p.setRenderHint(QPainter::Antialiasing);
    p.setPen(lineColor);
    p.drawLine(ui->label->width()*0.1,ui->label->height()*0.5,ui->label->width()*0.4,ui->label->height()*0.1);
    p.drawLine(ui->label->width()*0.4,ui->label->height()*0.1,ui->label->width()*0.6,ui->label->height()*0.6);
    p.drawLine(ui->label->width()*0.6,ui->label->height()*0.6,ui->label->width()*0.9,ui->label->height()*0.2);
    ui->label->setPicture(img);
}
void DialogDrawFigure::drawPolygone()
{
    QPicture img;
    QPainter p(&img);
    QPainterPath path;
    p.setRenderHint(QPainter::Antialiasing);
    p.setPen(lineColor);
    p.setBrush(QBrush(zColor));
    path.moveTo(ui->label_2->width()*0.2,ui->label_2->height()*0.8);
    path.lineTo(ui->label_2->width()*0.4,ui->label_2->height()*0.2);
    path.lineTo(ui->label_2->width()*0.7,ui->label_2->height()*0.3);
    path.lineTo(ui->label_2->width()*0.9,ui->label_2->height()*0.9);
    path.closeSubpath();
    p.drawPath(path);
    ui->label_2->setPicture(img);
}
void DialogDrawFigure::addPoint(double x, double y, double z)
{
    points << osg::Vec3d(x,y,z+100);
    qDebug() << "x:" << x <<" y:"<<y<<" z:"<<z;
    if(points.count() <= 0)
        return;
    if(ui->tableWidget->rowCount() < points.count())
        ui->tableWidget->insertRow(ui->tableWidget->rowCount());
    ui->tableWidget->setItem(points.count()-1,0,new QTableWidgetItem(QString::number(x)));
```

```
        ui->tableWidget->setItem(points.count()-1,1,new QTableWidgetItem(QString::number(y)));
        ui->tableWidget->setItem(points.count()-1,2,new QTableWidgetItem(QString::number(z)));
    }
    void DialogDrawFigure::on_pushButton_2_clicked()
    {
        QColorDialog *cd = new QColorDialog(this);
        cd->open(this,SLOT(on_polygone_color_selected(QColor)));
    }
    void DialogDrawFigure::on_polygone_color_selected(const QColor & color)
    {
        zColor = color;
        drawPolygone();
    }
    void DialogDrawFigure::addLine(QList<osg::Vec3d> points) {
        using namespace osgEarth;
        using namespace osgEarth::Features;
        using namespace osgEarth::Drivers;
        using namespace osgEarth::Symbology;
        using namespace osgEarth::Util;
        using namespace osgEarth::Annotation;
        using namespace osgEarth::Util::Controls;
        const osgEarth::SpatialReference* geoSRS = mainMapNode->getMapSRS()->getGeographicSRS();
        osgEarth::Symbology::Style geomStyle;
        geomStyle.getOrCreate<LineSymbol>()->stroke()->color() = Col-
or(lineColor.redF(),lineColor.greenF(),lineColor.blueF());
        //geomStyle.getOrCreate<PolygonSymbol>()->fill()->color() = Col-
or(lineColor.redF(),lineColor.greenF(),lineColor.blueF());
        geomStyle.getOrCreate<LineSymbol>()->stroke()->width() = 2.0f;
        geomStyle.getOrCreate<AltitudeSymbol>()->technique() = AltitudeSymbol::TECHNIQUE_GPU;
        osgEarth::Symbology::Geometry* path = new osgEarth::Features::LineString();
        if(!ui->checkBox->isChecked())
            foreach(osg::Vec3d v,points)
            {
                path->push_back(v);
            }
        else
        {
            foreach(osg::Vec3d v,points)
            {
                path->push_back(v.x(),v.y(),ui->doubleSpinBox_height->value());
            }
        }
```

```
        osgEarth::Annotation::FeatureNode* pnode = new FeatureNode(mainMapNode, new Feature(path, geoSRS,
geomStyle));
        rootNode->addChild( pnode );
    }
    void DialogDrawFigure::addPolygone(QList<osg::Vec3d> points) {
        using namespace osgEarth;
        using namespace osgEarth::Features;
        using namespace osgEarth::Drivers;
        using namespace osgEarth::Symbology;
        using namespace osgEarth::Util;
        using namespace osgEarth::Annotation;
        using namespace osgEarth::Util::Controls;
        const osgEarth::SpatialReference* geoSRS = mainMapNode->getMapSRS()->getGeographicSRS();
        osgEarth::Symbology::Style geomStyle;
        geomStyle.getOrCreate<LineSymbol>()->stroke()->color() = Col-
or(lineColor.redF(),lineColor.greenF(),lineColor.blueF());
        geomStyle.getOrCreate<PolygonSymbol>()->fill()->color() = Col-
or(zColor.redF(),zColor.greenF(),zColor.blueF());
        geomStyle.getOrCreate<LineSymbol>()->stroke()->width() = 2.0f;
        geomStyle.getOrCreate<AltitudeSymbol>()->technique() = AltitudeSymbol::TECHNIQUE_GPU;
        osgEarth::Symbology::Geometry* path = new osgEarth::Features::LineString();
        if(!ui->checkBox->isChecked())
          foreach(osg::Vec3d v,points)
          {
            path->push_back(v);
          }
        else
        {
          foreach(osg::Vec3d v,points)
          {
            path->push_back(v.x(),v.y(),ui->doubleSpinBox_height->value());
          }
        }
        //P•P°PjC‹PєP°PµPj PïPsP»PëPiPsPS
        if(!points.isEmpty())
          path->push_back(points[0]);
        osgEarth::Annotation::FeatureNode* pnode = new FeatureNode(mainMapNode, new Feature(path, geoSRS,
geomStyle));
        rootNode->addChild( pnode );
    }
    void DialogDrawFigure::on_pushButton_3_clicked()
    {
```

```
        /*if(ui->pushButton_draw->isChecked())
        {
            con-
nect(ch,SIGNAL(leftMouseButtonPressed(double,double,double)),this,SLOT(addPoint(double,double,double)));
            points.clear();
            ui->tableWidget->clear();
        }
        else
        {
            discon-
nect(ch,SIGNAL(leftMouseButtonPressed(double,double,double)),this,SLOT(addPoint(double,double,double)));
            if(ui->radioButton_line->isChecked())
                addLine(points);
            else if(ui->radioButton_poly->isChecked())
                addPolygone(points);
        }*/
    }
    void DialogDrawFigure::on_pushButton_draw_clicked()
    {
        if(ui->pushButton_draw->isChecked())
        {
            con-
nect(ch,SIGNAL(leftMouseButtonPressed(double,double,double)),this,SLOT(addPoint(double,double,double)));
            points.clear();
            ui->tableWidget->clear();
            ui->pushButton_draw->setText(QString::fromUtf8("Нарисовать"));
        }
        else
        {
            discon-
nect(ch,SIGNAL(leftMouseButtonPressed(double,double,double)),this,SLOT(addPoint(double,double,double)));
            if(ui->radioButton_line->isChecked())
                addLine(points);
            else if(ui->radioButton_poly->isChecked())
                addPolygone(points);
            ui->pushButton_draw->setText(QString::fromUtf8("Указать точки"));
        }
    }
    void DialogDrawFigure::on_checkBox_toggled(bool checked)
    {
        ui->doubleSpinBox_height->setEnabled(checked);
        if(checked)
```

```
            ui->tableWidget->hideColumn(2);
        else
            ui->tableWidget->showColumn(2);
    }
    bool CoordsHandler::handle( const osgGA::GUIEventAdapter& ea, osgGA::GUIActionAdapter& aa )
    {
    //    if ( (ea.getEventType() != osgGA::GUIEventAdapter::LEFT_MOUSE_BUTTON) && (ea.getEventType() !=
osgGA::GUIEventAdapter::RIGHT_MOUSE_BUTTON))
    //        return false;
        if((ea.getEventType() != osgGA::GUIEventAdapter::RELEASE) && (ea.getEventType() !=
osgGA::GUIEventAdapter::KEYUP))
            return false;
        osgUtil::LineSegmentIntersector::Intersections results;
        if ( view->computeIntersections( ea.getX(), ea.getY(), results, 0x01 ) )
        {
            // find the first hit under the mouse:
            osgUtil::LineSegmentIntersector::Intersection first = *(results.begin());
            osg::Vec3d point = first.getWorldIntersectPoint();
            double lat_rad, lon_rad, height;
            _mapSRS->getEllipsoid()->convertXYZToLatLongHeight( point.x(), point.y(), point.z(), lat_rad, lon_rad,
height );
            qDebug() << height;
            // query the elevation at the map point:
            double lat_deg = osg::RadiansToDegrees( lat_rad );
            double lon_deg = osg::RadiansToDegrees( lon_rad );
            if(ea.getButton() == osgGA::GUIEventAdapter::LEFT_MOUSE_BUTTON )
            {
                emit leftMouseButtonPressed(lon_deg,lat_deg,height);
                qDebug() << "emited leftMouseButtonPressed";
            }
            else if(ea.getKey() == osgGA::GUIEventAdapter::KEY_M)
            {
                emit rightMouseButtonPressed(lon_deg,lat_deg,height);
                qDebug() << "emited rightMouseButtonPressed";
            }
        }
        return false;
    }
```

Функции запуска системы 3D моделирования проектных решений

```
int main(int argc, char *argv[])
{
    QTextCodec::setCodecForCStrings(QTextCodec::codecForName("UTF-8"));
    QTextCodec::setCodecForTr(QTextCodec::codecForName("UTF-8"));
    QApplication::setAttribute(Qt::AA_X11InitThreads);
    QApplication a(argc, argv);
    //обрабатываем аргументы командной строки
    double x_com(0),y_com(0),z_com(0),angle(0),range(0);
    QStringList args = QCoreApplication::arguments();
    int args_index=0;
    bool isGoToCoords = false;
    configFile = "config.ini";
    //находим индекс аргумента, который начинается с "-"
    while((args_index=args.indexOf(QRegExp("[-][a-zA-Z]+"),args_index)) != -1){
        if(args[args_index]=="-coords"){
            if((args.length()-args_index-1) < 3){
                x_com=y_com=z_com=0;
            }
            else{
                bool ok_x,ok_y,ok_z;
                x_com=args[args_index+1].toDouble(&ok_x);
                y_com=args[args_index+2].toDouble(&ok_y);
                z_com=args[args_index+3].toDouble(&ok_z);
                if (args.length()-args_index-1 >= 5) {
                    angle=args[args_index+4].toDouble();
                    range=args[args_index+5].toDouble();
                }
                //если не получилось - сбрасываем координаты в дефолтное значение
                if(!(ok_x && ok_y && ok_z)){
                    ok_x=ok_y=ok_z=0;
                }
                //проверяем значение координат
                bool coords_ok=false;
                if(x_com >= -90 && x_com <= 90)
                    if(y_com >= -180 && y_com <= 180)
                        coords_ok=true;
                //если координаты вне диапазона
                if(!coords_ok){
                    ok_x=ok_y=ok_z=0;
                } else {
                    isGoToCoords = true;
```

```
            }
        }
    }
    if(args[args_index]=="-config"){
     //обрабатываем его
        if((args.length()-args_index-1) < 1) {
            configFile = "config.ini";
        } else {
            configFile = args[args_index+1];
        }
    }
    args_index++;
  }
  qDebug() << configFile;
  gConfig = new QSettings(configFile, QSettings::IniFormat);
  MainWindow w;
  preLoad(&w);
  w.show();

  if (isGoToCoords)
    myEarth::gotoCoordExt(x_com, y_com, z_com, range, angle, z_com);


  w.modelsWatcher->on_fileChanged();
  return a.exec();
}

void preLoad(MainWindow *w) {
  // включение кэширования
  FileSystemCacheOptions cacheOptions;
  cacheOptions.rootPath() = gConfig->value("common/cachePath").toString().toStdString();
  MapOptions mapOptions;
  mapOptions.cache() = cacheOptions;

  config["show_objects"] = "1";
  config["show_height"] = "1";
  config["show_texture"] = "1";

  oTypes[16] = 2;
  oTypes[17] = 4;
  oTypes[18] = 8;
  oTypes[19] = 16;
```

```
*objects_type = 2 | 4 | 8 | 16;

osgEarth::Util::EarthManipulator* manip = new osgEarth::Util::EarthManipulator();

// корневой слой
rootNode = new osg::Group();
osgEarth::TerrainOptions terrianOptions;
terrianOptions.verticalScale() = 3.0;
MapNodeOptions nodeOptions(terrianOptions);
//mapOptions.elevationTileSize()= 8;
/*if (gConfig->value("common/definition").toString() == "low") {
   mapOptions.elevationInterpolation() = INTERP_NEAREST;
} else {
   mapOptions.elevationInterpolation() = INTERP_BILINEAR;
}*/
//mapOptions.
map = new osgEarth::Map( mapOptions );
mainMapNode = new osgEarth::MapNode( map, nodeOptions );
// добавление высот и глубин
//addHeights();

ElevationLayer* layer = 0;

TMSOptions options;
options.url() = "http://readymap.org/readymap/tiles/1.0.0/9/";
layer = new osgEarth::ElevationLayer( "http://readymap.org/readymap/tiles/1.0.0/9/", options );
map->addElevationLayer( layer );

// добавление текстур
addTextures();

// слой моделей
modelsNode = new osg::Group();
showObjects(w);
rootNode->addChild(modelsNode);

// установка текущего времени
if (gConfig->value("common/sky").toString() == "1") {
   QDate nowDate = QDate::currentDate();
   QTime nowTime = QTime::currentTime();
   sky = osgEarth::Util::SkyNode::create( osgEarth::MapNode::findMapNode( mainMapNode ) );
```

```
            sky->setMoonVisible(true);
            //sky->setDateTime( nowDate.year(), nowDate.month(), nowDate.day(), nowTime.hour() );
            sky->attach( w->qosgwidget->view );
            rootNode->addChild( sky );
        }

        if (gConfig->value("common/ocean").toString() == "1") {
            SimpleOcean::SimpleOceanOptions ocean_opts;
            ocean_opts.textureURI() = "data/watersurface1.png";
            //ocean_opts.baseColor() = osg::Vec4(33,4f,7f,bf);
            ocean = osgEarth::Util::OceanNode::create( ocean_opts, osgEarth::MapNode::findMapNode( mainMapNode
) );
            rootNode->addChild(ocean);

        }

        rootNode->addChild(mainMapNode);
        // вывод координат по текущему местоположению указателя мыши
        MouseCoordsTool* tool = new MouseCoordsTool(mainMapNode);
        tool->addCallback( new PrintCoordsToStatusBar(w->spinCoordinatesX, w->spinCoordinatesY, w-
>spinCoordinatesZ) );
        w->qosgwidget->view->addEventHandler( new QueryElevationHandler( mainMapNode->getMap()-
>getProfile()->getSRS() ) );
        w->qosgwidget->view->addEventHandler( tool );
        w->qosgwidget->view->setCameraManipulator( manip );
        w->qosgwidget->view->setSceneData(rootNode);
        w->_viewer.addView( w->qosgwidget->view );
        w->setCoordsHandler(new CoordsHandler(mainMapNode->getMap()->getProfile()->getSRS(),w->qosgwidget-
>view, w));
    }
```

Функция загрузки конфигурационного файла

```
    /*
    void setupControls()
    {
      osgEarth::Util::EarthManipulator* manip = dynamic_cast<osgEarth::Util::EarthManipulator*>( mOsgViewer-
>getCameraManipulator() );
    }*/

    void showObjects(MainWindow* w) {
        isAnimate = 0;
        modelsNode->removeChildren(0, modelsNode->getNumChildren());
```

```
allModels.clear();
//getModelsForSearching();
QFile* file = new QFile(gConfig->value("common/objectsFile").toString());
if (!file->open(QIODevice::ReadOnly | QIODevice::Text))
{
   return;
}

QDomDocument doc;
doc.setContent(file);

QDomElement docElem = doc.documentElement();

QDomNode n = docElem.firstChildElement("Объекты").firstChild();
while(!n.isNull()) {
   QDomElement e = n.toElement();
   if(!e.isNull() && e.tagName() == "Объект") {
      QDomNode sem = n.firstChildElement("Семантика");
      if (sem.isNull())
            continue;
      double x = 0;
      double y = 0;
      double z = 0;
      int id = e.attribute("Код").toInt();
      double mScale = 1;
      double mAngle = 0;
      double courseX = 0;
      double courseY = 0;
      int mType = 0;
      QString name = e.attribute("Имя");
      QString type = e.attribute("Тип");
      QString modelFile = "";
      QDomNode record = sem.firstChild();

      QMap <QString,QString> model;
      model["name"]=e.attribute("Имя");
      model["type"]=e.attribute("Тип");

      if (type == "Модель") {
         while(!record.isNull()) {
            int code = record.toElement().attribute("Код").toInt();
            switch (code) {
```

```
        case 31:
            modelFile = record.toElement().attribute("Значение");
            break;
    case 32:
        mScale = record.toElement().attribute("Значение").toDouble();
        break;
    case 33:
        mAngle = record.toElement().attribute("Значение").toDouble();
        break;
    case 20:
        courseX = record.toElement().attribute("Значение").toDouble();
        break;
    case 21:
        courseY = record.toElement().attribute("Значение").toDouble();
        break;
    case 16:
    case 17:
    case 18:
    case 19:
        mType = record.toElement().attribute("Код").toInt();
        break;
    }

    record = record.nextSibling();
}

QDomNode metr = n.firstChildElement("Метрика");
record = metr.firstChild();

while(!record.isNull()) {
    int code = record.toElement().attribute("Код").toInt();
    switch (code) {
        case 13:
            model["x"]=record.toElement().attribute("Значение");
            break;
        case 14:
            model["y"]=record.toElement().attribute("Значение");
            break;
        case 16:
            model["z"]=record.toElement().attribute("Значение");
            break;
    }
```

```
                record = record.nextSibling();
            }
            models << model;

            if (!(*objects_type & oTypes[mType])) {
                n = n.nextSibling();
                continue;
            }

            if (metr.isNull())
                continue;

            record = metr.firstChild();

            while(!record.isNull()) {
                int code = record.toElement().attribute("Код").toInt();
                switch (code) {
                    case 13:
                        x = record.toElement().attribute("Значение").toDouble();
                        break;
                    case 14:
                        y = record.toElement().attribute("Значение").toDouble();
                        break;
                    case 16:
                        z = record.toElement().attribute("Значение").toDouble();
                        break;
                }

                record = record.nextSibling();
            }
            x += aniStep * courseX * 1;
            y += aniStep * courseY * 1;
            //osgEarth::Annotation::ModelNode* mNode;
            //mNode=myEarth::addObjectLayer(id, modelFile, gConfig-
>value("common/modelPath").toString()+modelFile+"/model.3ds", name, x, y, z, mScale, mAngle);
            //w->modelsWatcher->addModel(QString::number(id),mNode);

        } else if (type == "Линия") {
            double x0,y0,z0,x1,y1,z1;
            QDomNode metr = n.firstChildElement("Метрика");
            QDomNode record = metr.firstChild();
```

```
            while(!record.isNull()) {
                QString cname = record.toElement().attribute("Название");
                if (cname == "x0") {
                    x0 = record.toElement().attribute("Значение").toDouble();
                } else if (cname == "y0") {
                    y0 = record.toElement().attribute("Значение").toDouble();
                } else if (cname == "z0") {
                    z0 = record.toElement().attribute("Значение").toDouble();
                } else if (cname == "x1") {
                    x1 = record.toElement().attribute("Значение").toDouble();
                } else if (cname == "y1") {
                    y1 = record.toElement().attribute("Значение").toDouble();
                } else if (cname == "z1") {
                    z1 = record.toElement().attribute("Значение").toDouble();
                }

                record = record.nextSibling();
            }
            myEarth::addLine(x0, y0, z0, x1, y1, z1);
        } else if (type == "Полигон") {
            QList< QMap<QString, double> > points;
            QDomNode metr = n.firstChildElement("Метрика");
            QDomNode record = metr.firstChild();
            while(!record.isNull()) {
                QString cname = record.toElement().attribute("Название");
                if (cname == "Точка") {
                    QMap<QString, double> point;
                    point["x"] = record.toElement().attribute("x").toDouble();
                    point["y"] = record.toElement().attribute("y").toDouble();
                    if (record.toElement().hasAttribute("z"))
                        point["z"] = record.toElement().attribute("z").toDouble();
                    points.append(point);
                }

                record = record.nextSibling();
            }
            qDebug() << points;
            myEarth::addPolygon(points);
        }
    }
    n = n.nextSibling();
}
```

```
}
```

Модуль имитации движения объектов

```
#include "movableobject.h"
#include "movableobjectsmanager.h"
#include "osgearth.h"
#include <QString>
#include <QVector3D>
#include <QTimer>
#include <QDebug>
#include <math.h>
#include <osg/Vec3f>

MovableObject::MovableObject(osgEarth::Annotation::MapNode *mapNode, const osgEarth::Annotation::Style &style, const osgDB::Options *dbOptions)
    :ModelNode(mapNode,style,dbOptions),QObject(0)
{
    id = -1;
    name = "unnamed object";
    isFirstUpdate =true;
    interval = 0;
    lastMoveTime = QTime::currentTime();
    lastUpdate   = QTime::currentTime();
    manager = 0;
    legend = 0;
//    QTimer *t = new QTimer(this);

//    connect(t,SIGNAL(timeout()),this,SLOT(nextPoint()));
//    t->start(3000);
}

void MovableObject::setLegend(PlaceNode *legend)
{
    this->legend = legend;
    this->addChild(legend);
    checkDistance();
}

void MovableObject::setID(int id)
{
    this->id = id;
```

```
}

void MovableObject::setName(std::string name)
{
   this->name = name;
}

void MovableObject::setNextPoint(double x, double y, double z)
{
   const SpatialReference* geoSRS = this->getMapNode()->getMapSRS()->getGeographicSRS();
   if(isFirstUpdate)
   {
      //если получили первые координаты, то сразу перемещаем объект на них
      GeoPoint point = GeoPoint(geoSRS,x,y,z);
      this->setPosition(point);
      isFirstUpdate = false;
      lastUpdate = QTime::currentTime();
      lastMoveTime = lastUpdate;
      interval = 0;
      QTimer::singleShot(9000,this,SLOT(deleteIn9Second()));
      return;
   }

   GeoPoint nextPoint = GeoPoint(geoSRS,x,y,z);
   //qDebug() << x << y << z;
   interval = lastUpdate.msecsTo(QTime::currentTime());
   direction = QVector3D(nextPoint.x() - this->getPosition().x(),nextPoint.y() - this-
>getPosition().y(),nextPoint.z() - this->getPosition().z());
   lastUpdate = QTime::currentTime();

   qDebug() << "to node" << myEarth::getDistanceToNode(this);


}

void MovableObject::traverse(osg::NodeVisitor &nv)
{
   ModelNode::traverse(nv);
   legend->setText(QString("ID:%1\nX:%2\nY:%3").arg(id).arg(this->getPosition().x()).arg(this-
>getPosition().y()).toStdString());
   legend->setPosition(this->getPosition());
```

```
    if(interval == 0)
        return;

    //dp - приращение координат
    dp = direction*(double)lastMoveTime.msecsTo(QTime::currentTime())/(double)interval;
    //qDebug() << dp << lastMoveTime.msecsTo(QTime::currentTime()) << interval;
    p = this->getPosition();
    p.x() +=dp.x();
    p.y() +=dp.y();
    p.z() +=dp.z();

    //float course = 0;
    v = direction.normalized();
    if(v.x() > 0)
    {
        course = asin(v.y())-M_PI_2;
    }
    else
    {
        course = -asin(v.y())-M_PI_2 - M_PI;
    }
    this->setLocalRotation( osg::Quat(course+M_PI, osg::Vec3d(0.0,0.0,1.0)));

    lastMoveTime = QTime::currentTime();

    this->setPosition(p);
}

void MovableObject::nextPoint()
{
    setNextPoint(this->getPosition().x()+.01,this->getPosition().y()+.01,this->getPosition().z()+.01);
}

void MovableObject::deleteIn9Second()
{
    if(lastUpdate.msecsTo(QTime::currentTime()) >= 9000)
    {
        //delete
        if(manager)
            manager->removeObject(this->id);
        modelsNode->removeChild(this);
        return;
```

```
    }
    else
    {
        QTimer::singleShot(9000,this,SLOT(deleteIn9Second()));

    }
}

void MovableObject::checkDistance()
{
    //qDebug() << myEarth::getDistanceToNode(this)/this->getBound().radius();
    if(myEarth::getDistanceToNode(this) < 50000)
        legend->setNodeMask(0xffffffff);
    else
        legend->setNodeMask(0x0);
    QTimer::singleShot(1000,this,SLOT(checkDistance()));
}
```

Модуль взаимодействия с osgEarth

```
#include "mainwindow.h"
#include "movableobject.h"
#include "ui_mainwindow.h"
#include <osgGA/CameraManipulator>
/*
 * void addRasterLayer(QString name, QString path);
 * Добавление растрового слоя на карту
 */
void myEarth::addRasterLayer(QString name, QString path, bool isActive) {
//    qDebug() << name;
    osgEarth::Drivers::GDALOptions rast;
    rast.url() = path.toStdString();
    rast.setDriver("gdal");
    ImageLayer *iL = new ImageLayer( name.toStdString(), rast ) ;
    iL->setVisible(isActive);
    map->addImageLayer( iL );
}
osgEarth::ImageLayer* myEarth::addVectorLayer(QString name, QString path, bool isActive) {
    qDebug() << path.contains(".osm");
    if (path.contains(".osm") || path.contains("openstreetmap")) {
        TMSOptions driverOpt;
        driverOpt.url() = "http://tile.openstreetmap.org/";
```

```
            driverOpt.tmsType() = "google";
            ImageLayerOptions layerOpt( "OSM", driverOpt );
        //      layerOpt.profile() = ProfileOptions( "global-mercator" );
            ImageLayer* osmLayer = new ImageLayer( layerOpt );
            osmLayer->setVisible(isActive);
            mainMapNode->getMap()->addImageLayer( osmLayer );
            return osmLayer;
          } else if (path.contains(".kml")) {
            osg::ref_ptr<osgDB::Options> options = new osgDB::Options();
            options->setPluginData( "osgEarth::MapNode", mainMapNode );
            osg::Node* kml = osgDB::readNodeFile( path.toStdString(), options.get() );
            if ( kml ) {
              kml->setName(name.toStdString());
              mainMapNode->addChild( kml );
              qDebug() << "CHILDREN OF KML:"<<kml->asGroup()->getNumChildren();
               qDebug() << "CHILDREN OF CHILD KML:"<<kml->asGroup()->getChild(0)->asGroup()-
>getNumChildren();
            }
          } else if (path.contains(".000")) {
            qDebug() << "s57";
            myEarth::addS57Layer(name, path);
          }
          return 0;
        }
        /*
         * void addElevationLayer(QString name, QString path);
         * Добавление слоя высот и глубин на карту
         *
         */
        void myEarth::addElevationLayer(QString name, QString path, bool isActive) {
          osgEarth::Drivers::GDALOptions gdal;
          gdal.url() =  path.toStdString();
          gdal.setDriver("gdal");
          osgEarth::ElevationLayer* layerTerrain = new osgEarth::ElevationLayer(name.toStdString(), gdal );
          layerTerrain->setVisible(isActive);
          map->addElevationLayer( layerTerrain );
        }
        /*
         * void addS57Layer(QString name, QString path);
         * Добавление слоя S57
         *
         */
```

```
void myEarth::addS57Layer(QString name, QString path) {
   QString lpath = path;
   lpath.replace(".000",".csv");
   QFile file(lpath);
   if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {
      qDebug() << "Opening file error!";
      return;
   }
   QByteArray bytes = file.readAll();
   QStringList layers = QString::fromUtf8(bytes.data(),
bytes.size()).split(QRegExp("[\r\n]"),QString::SkipEmptyParts);
   for (int i = 0; i < layers.size(); i++) {
   //foreach (QString layer, layers) {
      QString layer = layers.at(i);
      qDebug() << layer;
      OGRFeatureOptions opts;
      opts.layer() = layer.toStdString();
      opts.url() = path.toStdString();
      opts.ogrDriver() = "S57";
      osgEarth::Symbology::StyleSheet* style = new osgEarth::Symbology::StyleSheet;
      Style geomStyle;
      PointSymbol* pts = geomStyle.getOrCreate<PointSymbol>();
      //pts->fill()->color() = Color("#00ff00");
      pts->fill()->color() = osg::Vec4f(static_cast <float> (rand()) / static_cast <float> (RAND_MAX), stat-ic_cast
<float> (rand()) / static_cast <float> (RAND_MAX), static_cast <float> (rand()) / static_cast <float> (RAND_MAX), 1);
      pts->size() = 1.0f;
      LineSymbol* ls = geomStyle.getOrCreate<LineSymbol>();
      //ls->stroke()->color() = Color("#0000ff");
      ls->stroke()->color() = osg::Vec4f(static_cast <float> (rand()) / static_cast <float> (RAND_MAX), stat-
ic_cast <float> (rand()) / static_cast <float> (RAND_MAX), static_cast <float> (rand()) / static_cast <float>
(RAND_MAX), 1);
      ls->stroke()->width() = 1.0f;
      PolygonSymbol* ps = geomStyle.getOrCreate<PolygonSymbol>();
      ps->fill()->color() = osg::Vec4f(static_cast <float> (rand()) / static_cast <float> (RAND_MAX), static_cast
<float> (rand()) / static_cast <float> (RAND_MAX), static_cast <float> (rand()) / static_cast <float> (RAND_MAX), 1);
   //    ps->fill()->color() = Color("#ff0000");
      //RenderSymbol* rs = geomStyle.getOrCreate<RenderSymbol>();
      //rs->depthOffset().defaultValue().enabled();
      ExtrusionSymbol* es = geomStyle.getOrCreate<ExtrusionSymbol>();
      es->height() = 15.0f;
      es->flatten() = true;
      AltitudeSymbol *as = geomStyle.getOrCreate<AltitudeSymbol>();
```

```
            //as->verticalOffset() = i*1;
            as->clamping() = AltitudeSymbol::CLAMP_RELATIVE_TO_TERRAIN;
            as->technique() = AltitudeSymbol::TECHNIQUE_DRAPE;
            style->addStyle(geomStyle);
            FeatureDisplayLayout layout;
            osgEarth::Drivers::FeatureGeomModelOptions opt;
            opt.featureOptions() = opts;
            opt.styles() = style;
            opt.layout() = layout;
            osgEarth::ModelLayer* ml = new osgEarth::ModelLayer("S57", opt);
            map->addModelLayer( ml );
        }
    }
    /*
     * void addWireframeLayer(QString name, QString path);
     * Добавление слоя высот и глубин на карту с каркасным видом
     *
     */
    void myEarth::addWireframeLayer(QString name, QString path) {
        osgEarth::Drivers::GDALOptions gdal;
        gdal.url() =  path.toStdString();
        gdal.setDriver("gdal");
        osgEarth::ElevationLayer* layerTerrain = new osgEarth::ElevationLayer(name.toStdString(), gdal );
        map->addElevationLayer( layerTerrain );
        osgEarth::Drivers::ColorRampOptions opts;
        opts.elevationLayer() = gdal;
        opts.ramp() = gConfig->value("common/elColor").toString().toStdString();
        ImageLayer *iL = new ImageLayer( "ColorRamp", opts );
        map->addImageLayer( iL );
    }
    /*
     * void addObjectLayer(int id, QString mFile, QString name, double x, double y, double z, double mScale, double
mAngle);
     * Добавление 3D объекта
     *
     */
    MovableObject* myEarth::addObjectLayer(int id, QString modelFile, QString mFile, QString name, double x,
double y, double z, double mScale = 1, double mAngle = 0) {
        MapOptions mapOptions;
        osgEarth::Map *map = new osgEarth::Map( mapOptions );
        MapNodeOptions nodeOptions;
        osgEarth::MapNode *mapNode = new osgEarth::MapNode( map, nodeOptions );
```

```
            osg::Node* modelNode = osgDB::readNodeFile(mFile.toStdString());
            osgEarth::Style style;
            style.getOrCreate<ModelSymbol>()->setModel( modelNode );
            //altitude-resolution
            style.getOrCreate<AltitudeSymbol>()->technique() = AltitudeSymbol::TECHNIQUE_MAP;
            style.getOrCreate<AltitudeSymbol>()->clampingResolution() = 0.000001;
            MovableObject* model = new MovableObject( mapNode, style );
            qDebug() << "!@#!";
            const SpatialReference* geoSRS = mapNode->getMapSRS()->getGeographicSRS();
            model->setPosition( GeoPoint(geoSRS, y, x, z, ALTMODE_ABSOLUTE) );
            model->setLocalRotation( osg::Quat(mAngle, osg::Vec3d(0.0,0.0,1.0)));
            model->setScale(osg::Vec3(mScale, mScale, mScale));
            model->setName(name.toStdString());
            osgEarth::Util::AnnotationEventCallback* cb = new osgEarth::Util::AnnotationEventCallback();
            cb->addHandler(new ObjectAnnotationEventHandler(modelFile, id, x, y, z));
            model->addEventCallback(cb);
        /*    aDmodels *dmodel = new aDmodels();
            dmodel->courseX = courseX;
            dmodel->courseY = courseY;
            dmodel->x = x;
            dmodel->y = y;
            dmodel->z = z;
            dmodel->node = model;
            allModels.push_back(dmodel);*/
            modelsNode->addChild(model);
            osgEarth::Util::Annotation::PlaceNode *pn = new
osgEarth::Util::Annotation::PlaceNode(mapNode,GeoPoint(mainMapNode->getMapSRS(), y, x, z, ALT-
MODE_ABSOLUTE) ,QString("ID:%1\nX:%2  Y:%3").arg(name).arg(x).arg(y).toStdString());
            pn->setDynamic(true);
            model->setLegend(pn);
            model->setID(id);
            return model;
        }
        /*
         * void gotoCoord(double x, double y, double z);
         * Функция перехода камеры по указанным координатам
         *
         */
        void myEarth::gotoCoord(double x, double y, double z) {
            osgEarth::Util::EarthManipulator* manip = dynamic_cast<EarthManipulator*>(mWindow->qosgwidget-
>view->getCameraManipulator());
```

```
        manip->setViewpoint(osgEarth::Viewpoint(osg::Vec3d(x, y, z), manip->getViewpoint().getHeading(), manip->getViewpoint().getPitch(), z), 4);
    }
    /*
     * void gotoCoordExt(double x, double y, double z, double heading, double pitch, double range);
     * Расширенная функция перехода камеры по указанным координатам
     * heading - поворот относительно центра
     * pitch - сдвиг по горизонтали относительно экватора
     *
     */
    void myEarth::gotoCoordExt(double x, double y, double z, double heading, double pitch, double range) {
        osgEarth::Util::EarthManipulator* manip = dynamic_cast<EarthManipulator*>(mWindow->qosgwidget->view->getCameraManipulator());
        manip->setViewpoint(osgEarth::Viewpoint(osg::Vec3d(x, y, z), heading, pitch, range), 4);
    }
    void myEarth::addLine(double x0, double y0, double z0, double x1, double y1, double z1) {
        const SpatialReference* geoSRS = mainMapNode->getMapSRS()->getGeographicSRS();
        Style geomStyle;
        geomStyle.getOrCreate<LineSymbol>()->stroke()->color() = osgEarth::Symbology::Color::Black;
        geomStyle.getOrCreate<PolygonSymbol>()->fill()->color() = Color::Red;
        geomStyle.getOrCreate<LineSymbol>()->stroke()->width() = 2.0f;
        //geomStyle.getOrCreate<AltitudeSymbol>()->clamping() = AltitudeSymbol::CLAMP_TO_TERRAIN;
        //geomStyle.getOrCreate<AltitudeSymbol>()->technique() = AltitudeSymbol::TECHNIQUE_GPU;
        Geometry* path = new LineString();
        path->push_back( osg::Vec3d(x0,y0,z0) );
        path->push_back( osg::Vec3d(x1,y1,z1) );
        FeatureNode* pnode = new FeatureNode(mainMapNode, new Feature(path, geoSRS, geomStyle));
        rootNode->addChild( pnode );
    }
    osgEarth::Annotation::FeatureNode* myEarth::createLine(std::vector<osg::Vec3> points) {
        using namespace osgEarth;
        using namespace osgEarth::Features;
        using namespace osgEarth::Drivers;
        using namespace osgEarth::Symbology;
        using namespace osgEarth::Util;
        using namespace osgEarth::Annotation;
        using namespace osgEarth::Util::Controls;
        const osgEarth::SpatialReference* geoSRS = mainMapNode->getMapSRS()->getGeographicSRS();
        osgEarth::Symbology::Style geomStyle;
        geomStyle.getOrCreate<LineSymbol>()->stroke()->color() = Color(0,0,0);
        //geomStyle.getOrCreate<PolygonSymbol>()->fill()->color() = Color(lineColor.redF(),lineColor.greenF(),lineColor.blueF());
```

```
        geomStyle.getOrCreate<LineSymbol>()->stroke()->width() = 1.0f;
        geomStyle.getOrCreate<AltitudeSymbol>()->technique() = AltitudeSymbol::TECHNIQUE_GPU;
        osgEarth::Symbology::Geometry* path = new osgEarth::Features::LineString();
        for(int i=0;i<points.size();i++)
        {
          path->push_back(points[i]);
        }
        qDebug() << "POINTS SIZE" << points.size();
        osgEarth::Annotation::FeatureNode* pnode = new FeatureNode(mainMapNode, new Feature(path, geoSRS,
geomStyle));
        //modelsNode->addChild( pnode );
        return pnode;
      }
      void myEarth::addPolygon(QList< QMap<QString, double> > points) {
        const SpatialReference* geoSRS = mainMapNode->getMapSRS()->getGeographicSRS();
        Polygon *poly = new Polygon();
        for (int i = 0; i < points.size(); ++i) {
          QMap<QString, double> point = points.at(i);
          if (point.contains("z"))
            poly->push_back( osg::Vec3d(point["x"], point["y"], point["z"]) );
          else
            poly->push_back( point["x"], point["y"] );
        }
        Style geomStyle;
        geomStyle.getOrCreate<LineSymbol>()->stroke()->color() = osgEarth::Symbology::Color::Black;
        geomStyle.getOrCreate<PolygonSymbol>()->fill()->color() = Color::Red;
        geomStyle.getOrCreate<LineSymbol>()->stroke()->width() = 2.0f;
        geomStyle.getOrCreate<AltitudeSymbol>()->clamping() = AltitudeSymbol::CLAMP_TO_TERRAIN;
        geomStyle.getOrCreate<AltitudeSymbol>()->technique() = AltitudeSymbol::TECHNIQUE_DRAPE;
        //geomStyle.getOrCreate<AltitudeSymbol>()->binding() = AltitudeSymbol::BINDING_VERTEX;
        //geomStyle.getOrCreate<AltitudeSymbol>()->verticalOffset() = 10900.0f;
        FeatureNode* fnode = new FeatureNode(mainMapNode, new Feature(poly, geoSRS, geomStyle));
        rootNode->addChild( fnode );
      }
      void myEarth::addFlagPlacemark(double x, double y)
      {
        using namespace osgEarth;
        using namespace osgEarth::Annotation;
        osg::ref_ptr<osgDB::Options> options = new osgDB::Options();
        options->setPluginData( "osgEarth::MapNode", mainMapNode );
        osg::Node* kml = osgDB::readNodeFile( "./data/flag.kml", options.get() );
        if ( kml ) {
```

```
            kml->setName("flag");
            mainMapNode->addChild( kml );
            qDebug() << "CHILDREN OF KML:"<<kml->asGroup()->getNumChildren();
             qDebug() << "CHILDREN OF CHILD KML:"<<kml->asGroup()->getChild(0)->asGroup()->getNumChildren();
        }
    //    osgEarth::MapOptions mapOptions;
    //    osgEarth::Map *map = new osgEarth::Map( mapOptions );
    //    osgEarth::MapNodeOptions nodeOptions;
    //    osgEarth::MapNode *mapNode = new osgEarth::MapNode( map, nodeOptions );
    //    osg::Node* modelNode = new osg::Node();//osgDB::readNodeFile(mFile.toStdString());
    //    osgEarth::Style style;
    //    style.getOrCreate<ModelSymbol>()->setModel( modelNode );
    //    osgEarth::Annotation::ModelNode* model = new osgEarth::Annotation::ModelNode( mapNode, style );
    //    mapNode->addChild(kml);
    //    modelsNode->addChild(model);
    //    //mainMapNode->addChild(model);
    //    model->setPosition(GeoPoint(mainMapNode->getMapSRS()->getGeographicSRS(),x,y));
        ////////////////////
    }
    double myEarth::getDistanceToNode(osgEarth::Annotation::LocalizedNode *node)
    {
        static osgEarth::Util::EarthManipulator* manip = dynamic_cast<EarthManipulator*>(mWindow->qosgwidget->view->getCameraManipulator());
        static const SpatialReference* geoSRS = mainMapNode->getMapSRS()->getGeographicSRS();
        osg::Vec3d eye;
        osg::Vec3d temp;
        mWindow->qosgwidget->view->getCamera()->getViewMatrixAsLookAt(eye,temp,temp);
        if(manip)
        {
            osg::Vec3d world;
            node->getPosition().toWorld(world);
            QVector3D v3d = QVector3D(eye.x(),eye.y(),eye.z())-QVector3D(world.x(),world.y(),world.z());
            //qDebug() << "distnce" << v3d.length();
            return v3d.length();
        }
        else
            return 0;
    }
```

Модуль взаимодействия с внешними системами по UDP порту

```
#include "udplistener.h"
```

```
#include "mainwindow.h"
UdpListener::UdpListener(QString modelsFilePath,QObject *parent) :
    QObject(parent),port(5838),modelsFilePath(modelsFilePath)
{
    socket=new QUdpSocket(this);
    connect(socket, SIGNAL(readyRead()), SLOT(recieveMessage()));
    port=5838;
    socket->bind(QHostAddress::LocalHost, port);
}
UdpListener::UdpListener(QObject *parent):QObject(parent)
{
    socket=new QUdpSocket(this);
    connect(socket, SIGNAL(readyRead()), SLOT(recieveMessage()));
    port=5838;
    socket->bind(QHostAddress::LocalHost, port);
}
void UdpListener::recieveMessage()
{
    qDebug() << "reading message...";
    QByteArray datagram;
    datagram.resize(socket->pendingDatagramSize());
    QHostAddress *address = new QHostAddress();
    socket->readDatagram(datagram.data(), datagram.size(), address);
    QDataStream in(&datagram, QIODevice::ReadOnly);
    qint64 size = -1;
    if(in.device()->size() > sizeof(qint64)) {
        in >> size;
    } else return;
    if (in.device()->size() - sizeof(qint64) < size) return;
    qint8 type = 0;
    in >> type;
    QString str;
    in >> str;
    qDebug()<<"message:" << str;
    //добавление объекта
    if (type == 0) {
        qDebug()<<str;
        addObject(str);
    }
    //удаление объекта
    else if (type == 1)
    {
```

```
        deleteObject(str);
    }
    else if (type == 2)
    {
        addLayer(str);
    }
    else if(type == 3)
    {
        deleteLayer(str);
    }
    else if (type == 4)
    {
        setCoords(str);
    }
}
void UdpListener::addObject(/*QDomNode newObject*/ QString str)
{
    QFile file;
    file.setFileName(modelsFilePath);
    if (!file.open(QIODevice::ReadWrite | QIODevice::Text))
    {
        return;
    }
    doc.setContent(&file);
    file.close();
    QDomDocument temp;
    QString error;
    int line;
    qDebug() << temp.setContent(str,&error,&line);
    //QDomElement newObject = temp.documentElement().firstChildElement("Объект");
    QDomNode newObject = temp.documentElement();//.firstChild();
    QDomElement n = doc.documentElement().firstChildElement("Объекты");
    qDebug() <<"node"<< newObject.toDocument().toString(2);
    qDebug() <<"document"<< temp.toString(2);
    n.appendChild(newObject);
    saveModelsFile();
}
void UdpListener::deleteObject(QString numOrName)
{
    QFile file;
    file.setFileName(modelsFilePath);
    if (!file.open(QIODevice::ReadWrite | QIODevice::Text))
```

```
    {
        return;
    }
    doc.setContent(&file);
    file.close();
    QDomNode n = doc.documentElement().firstChildElement("Объекты").firstChild();
    QDomNode objNode = doc.documentElement().firstChildElement("Объекты");
    QDomElement e;
    while(!n.isNull()) {
        e = n.toElement();
        if(e.attribute("Код")==numOrName || e.attribute("Имя")==numOrName)
        {
            objNode.removeChild(n);
            saveModelsFile();
            return;
        }
        n = n.nextSibling();
    }
}
void UdpListener::addLayer(QString str)
{
    QFile file;
    file.setFileName(modelsFilePath);
    if (!file.open(QIODevice::ReadWrite | QIODevice::Text))
    {
        return;
    }
    doc.setContent(&file);
    file.close();
    QDomDocument temp;
    QString error;
    int line;
    qDebug() << temp.setContent(str,&error,&line);
    QDomNode newObject = temp.documentElement();//.firstChild();
    QDomElement n = doc.documentElement().firstChildElement("Слои");
    qDebug() <<"node"<< newObject.toDocument().toString(2);
    qDebug() <<"document"<< temp.toString(2);
    n.appendChild(newObject);
    saveModelsFile();
}
void UdpListener::deleteLayer(QString name)
{
```

```
            QFile file;
            file.setFileName(modelsFilePath);
            if (!file.open(QIODevice::ReadWrite | QIODevice::Text))
            {
                return;
            }
            doc.setContent(&file);
            file.close();
            QDomNode n = doc.documentElement().firstChildElement("Слои").firstChild();
            QDomNode layersNode = doc.documentElement().firstChildElement("Слои");
            QDomElement e;
            while(!n.isNull()) {
                e = n.toElement();
                if(e.attribute("Имя")==name)
                {
                    layersNode.removeChild(n);
                    saveModelsFile();
                    return;
                }
                n = n.nextSibling();
            }
        }
        void UdpListener::setCoords(QString str)
        {
            QStringList coords = str.split(":");
            if(coords.size())
            {
                myEarth::gotoCoordExt(coords[0].toDouble(), coords[1].toDouble(), coords[2].toDouble(),
coords[4].toDouble(), coords[3].toDouble(), coords[4].toDouble());
        /*      QFile file;
                file.setFileName(modelsFilePath);
                if (!file.open(QIODevice::ReadWrite | QIODevice::Text))
                {
                    return;
                }
                doc.setContent(&file);
                file.close();
                QDomNode n = doc.documentElement().firstChildElement("Основное").firstChild();
                QDomElement e;
                while(!n.isNull()) {
                    e = n.toElement();
                    if(e.attribute("Название")=="Ycoord")
```

```
            {
               n.toElement().setAttribute("Значение",coords[0]);
            }
            else if(e.attribute("Название")=="Xcoord")
            {
               n.toElement().setAttribute("Значение",coords[1]);
            }
            else if(e.attribute("Название")=="horizont")
            {
               n.toElement().setAttribute("Значение",coords[2]);
            }
            n = n.nextSibling();
         }
         saveModelsFile();*/
      }
   }
   void UdpListener::saveModelsFile()
   {
      QFile file;
      file.setFileName(modelsFilePath);
      if (!file.open(QIODevice::ReadWrite | QIODevice::Text))
      {
         return;
      }
      QTextStream stream( &file );
      stream << doc.toString();
      file.close();
   }
   void UdpListener::setModelsFileName(QString name)
   {
      modelsFilePath=name;
   }
```

ПРИЛОЖЕНИЕ В. XML ФАЙЛЫ КОНФИГУРАЦИИ СИСТЕМЫ 3D МОДЕЛИРОВАНИЯ ПРОЕКТНЫХ РЕШЕНИЙ

Конфигурационный файл для отображения обстановки “model.xml”

```
<?xml version='1.0' encoding='UTF-8'?>
<!--
Тип объекта:
- Линия
- Точка
- Полигон
- Модель
-->
<Данные>
 <Основное>
<Запись Название="Xcoord" Значение="-66" />
<Запись Название="Ycoord" Значение="18" />
<Запись Название="horizont" Значение= "2500" />
 </Основное>
 <Объекты>
<Объект Код="1" Слой="РПЛСН" Номер="1" Ключ="ААК" Тип="Модель" Имя="Стерегущий">
  <Семантика>
   <Запись Значение="ship" Код="31" Название="Путь к модели" Номер="1"/>
   <Запись Значение="Надводный объект" Код="17" Название="Тип" Номер="3"/>
   <Запись Значение="-0.001" Код="20" Название="КурсX" Номер="2"/>
   <Запись Значение="0.002" Код="21" Название="КурсY" Номер="3"/>
   <Запись Значение="290" Код="33" Название="Угол" Номер="3"/>
   <Запись Значение="1" Код="32" Название="Масштаб" Номер="4"/>
  </Семантика>
  <Метрика>
   <Запись Значение="44.5" Код="13" Название="Широта" Номер="1"/>
   <Запись Значение="33.43" Код="14" Название="Долгота" Номер="2"/>
   <Запись Значение="0" Код="16" Название="Высота" Номер="4"/>
   <Траектория>
    <Точка Широта="44.5" Долгота="33.44" Время="2" Высота="0"/>
    <Точка Широта="44.5" Долгота="33.46" Время="1" Высота="0"/>
    <Точка Широта="44.7" Долгота="33.49" Время="2" Высота="0"/>
    <Точка Широта="44.6" Долгота="33.43" Время="3" Высота="0"/>
    <Точка Широта="44.5" Долгота="33.44" Время="2" Высота="0"/>
   </Траектория>
```

```
 </Метрика>
</Объект>
<Объект Код="2" Слой="РПЛСН" Номер="2" Ключ="ПЛЗ" Тип="Модель" Имя="Подводная лодка За-
ря">
 <Семантика>
  <Запись Значение="submarine" Код="31" Название="Путь к модели" Номер="1"/>
  <Запись Значение="Подводный объект" Код="18" Название="Тип" Номер="5"/>
  <Запись Значение="0.001" Код="20" Название="КурсX" Номер="2"/>
  <Запись Значение="-0.004" Код="21" Название="КурсY" Номер="3"/>
  <Запись Значение="0" Код="33" Название="Угол" Номер="3"/>
  <Запись Значение="0.12" Код="32" Название="Масштаб" Номер="6"/>
 </Семантика>
 <Метрика>
  <Запись Значение="44.5" Код="13" Название="Широта" Номер="1"/>
  <Запись Значение="33.46" Код="14" Название="Долгота" Номер="2"/>
  <Запись Значение="0" Код="16" Название="Высота" Номер="4"/>
  <Траектория>
   <Точка Широта="44.5" Долгота="33.43" Время="2" Высота="10"/>
   <Точка Широта="44.3" Долгота="33.46" Время="2" Высота="1100"/>
   <Точка Широта="44.6" Долгота="33.46" Время="4" Высота="10"/>
   <Точка Широта="44.6" Долгота="33.43" Время="1" Высота="10"/>
   <Точка Широта="44.5" Долгота="33.43" Время="1" Высота="10"/>
  </Траектория>
 </Метрика>
</Объект>
<Объект Код="3" Слой="РПЛСН" Номер="2" Ключ="ПЛЗ" Тип="Модель" Имя="Вертолёт">
 <Семантика>
  <Запись Значение="aircraft" Код="31" Название="Путь к модели" Номер="1"/>
  <Запись Значение="Воздушный объект" Код="19" Название="Тип" Номер="5"/>
  <Запись Значение="0.001" Код="20" Название="КурсX" Номер="2"/>
  <Запись Значение="-0.004" Код="21" Название="КурсY" Номер="3"/>
  <Запись Значение="-90" Код="33" Название="Угол" Номер="3"/>
  <Запись Значение="0.1" Код="32" Название="Масштаб" Номер="4"/>
 </Семантика>
 <Метрика>
  <Запись Значение="44.67" Код="13" Название="Широта" Номер="1"/>
  <Запись Значение="33.59" Код="14" Название="Долгота" Номер="2"/>
  <Запись Значение="300" Код="16" Название="Высота" Номер="4"/>
 </Метрика>
</Объект>
<Объект Код="4" Слой="РПЛСН" Номер="2" Ключ="ПЛЗ" Тип="Модель" Имя="Самолёт Air France">
 <Семантика>
```

```
 <Запись Значение="air" Код="31" Название="Путь к модели" Номер="1"/>
 <Запись Значение="Воздушный объект" Код="19" Название="Тип" Номер="5"/>
 <Запись Значение="0.001" Код="20" Название="КурсX" Номер="2"/>
 <Запись Значение="0.004" Код="21" Название="КурсY" Номер="3"/>
 <Запись Значение="90" Код="33" Название="Угол" Номер="3"/>
 <Запись Значение="0.04" Код="32" Название="Масштаб" Номер="4"/>
</Семантика>
<Метрика>
 <Запись Значение="44.52" Код="13" Название="Широта" Номер="1"/>
 <Запись Значение="33.59" Код="14" Название="Долгота" Номер="2"/>
 <Запись Значение="1000" Код="16" Название="Высота" Номер="4"/>
</Метрика>
</Объект>
<Объект Код="5" Слой="РПЛСН" Номер="3" Ключ="TA" Тип="Модель" Имя="Танк Абрамс">
<Семантика>
 <Запись Значение="tank" Код="31" Название="Путь к модели" Номер="1"/>
 <Запись Значение="Наземный объект" Код="16" Название="Тип" Номер="5"/>
 <Запись Значение="0.00" Код="20" Название="КурсX" Номер="2"/>
 <Запись Значение="0.002" Код="21" Название="КурсY" Номер="3"/>
 <Запись Значение="90" Код="33" Название="Угол" Номер="6"/>
</Семантика>
<Метрика>
 <Запись Значение="44.56" Код="13" Название="Широта" Номер="1"/>
 <Запись Значение="33.59" Код="14" Название="Долгота" Номер="2"/>
 <Запись Значение="100" Код="16" Название="Высота" Номер="4"/>
</Метрика>
</Объект>
<Объект Код="6" Слой="РПЛСН" Номер="4" Ключ="ВКА50" Тип="Модель" Имя="Вертолет КА-50">
<Семантика>
 <Запись Значение="UH60" Код="31" Название="Путь к модели" Номер="1"/>
 <Запись Значение="Воздушный объект" Код="19" Название="Тип" Номер="5"/>
 <Запись Значение="0.001" Код="20" Название="КурсX" Номер="2"/>
 <Запись Значение="0.001" Код="21" Название="КурсY" Номер="3"/>
 <Запись Значение="10" Код="32" Название="Масштаб" Номер="4"/>
</Семантика>
<Метрика>
 <Запись Значение="44.62" Код="13" Название="Широта" Номер="1"/>
 <Запись Значение="33.59" Код="14" Название="Долгота" Номер="2"/>
 <Запись Значение="400" Код="16" Название="Высота" Номер="4"/>
</Метрика>
</Объект>
<Объект Код="20" Ключ="" Тип="Модель" Имя="Марс объект">
```

```
 <Семантика>
  <Запись Значение="tank" Код="31" Название="Путь к модели" Номер="1"/>
 </Семантика>
 <Метрика>
  <Запись Значение="40" Код="13" Название="Широта" Номер="1"/>
  <Запись Значение="30" Код="14" Название="Долгота" Номер="2"/>
  <Запись Значение="1000" Код="16" Название="Высота" Номер="4"/>
 </Метрика>
</Объект>
<Объект Код="1001" Ключ="13002;16;bdo_l" Тип="Модель" Имя="Марс объект">
 <Семантика>
  <Запись Значение="tank" Код="31" Название="Путь к модели" Номер="1"/>
 </Семантика>
 <Метрика>
  <Запись Значение="30" Код="13" Название="Широта" Номер="1"/>
  <Запись Значение="30" Код="14" Название="Долгота" Номер="2"/>
  <Запись Значение="1000" Код="16" Название="Высота" Номер="4"/>
 </Метрика>
</Объект>
<Объект Код="1001" Ключ="13002;17;bdo_l" Тип="Модель" Имя="Марс объект">
 <Семантика>
  <Запись Значение="air" Код="31" Название="Путь к модели" Номер="1"/>
 </Семантика>
 <Метрика>
  <Запись Значение="30" Код="13" Название="Широта" Номер="1"/>
  <Запись Значение="30" Код="14" Название="Долгота" Номер="2"/>
  <Запись Значение="1000" Код="16" Название="Высота" Номер="4"/>
 </Метрика>
</Объект>
<Объект Код="1002" Ключ="13002;17;bdo_l" Тип="Модель" Имя="Марс объект1">
 <Семантика>
  <Запись Значение="air" Код="31" Название="Путь к модели" Номер="1"/>
 </Семантика>
 <Метрика>
  <Запись Значение="32" Код="13" Название="Широта" Номер="1"/>
  <Запись Значение="32" Код="14" Название="Долгота" Номер="2"/>
  <Запись Значение="1000" Код="16" Название="Высота" Номер="4"/>
 </Метрика>
</Объект>
<Объект Слой = "РПЛСН" Номер = "4" Тип="Полигон" Ключ = "Полигон1" Имя = "Полигон 1" Код = "8">
<Семантика>
</Семантика>
```

```
<Метрика>
<Запись Значение = "" x="33.43" y="44.5" Номер = "1" Название = "Точка" Код = "13"/>
<Запись Значение = "" x="33.46" y="44.5" Номер = "1" Название = "Точка" Код = "13"/>
<Запись Значение = "" x="33.59" y="44.62" Номер = "1" Название = "Точка" Код = "13"/>
</Метрика>
</Объект>
 </Объекты>
 <Слои>
  <Слой Тип="terrain" Путь="data/srtm/gebco_08.tif" Имя="gebco_08.tif" Активный="0"/>
  <Слой Тип="geotiff" Путь="data/raster/aworld_high.tif" Имя="Вся планета" Активный="0"/>
  <Слой Тип="tms" Путь="http://readymap.org/readymap/tiles/1.0.0/7/" Имя="TMS" Активный="1"/>
  <Слой Тип="osm" Путь="http://[abc].tile.openstreetmap.org/{z}/{x}/{y}.png" Имя="OpenStreetMap" Активный="0"/>
  <Слой Тип="s57" Путь="/home/user/gisulsu/data/S57/US5PR43M.000" Имя="S57" Активный="1"/>
  <Слой Тип="s57" Путь="/home/user/gisulsu/data/S57/US1AK90M.000" Имя="S571" Активный="1"/>
  <Слой Тип="s57" Путь="/home/user/gisulsu/data/S57/1W5DEK00.000" Имя="S572" Активный="1"/>
  <Слой Тип="geotiff" Путь="data/raster/ul.tif" Имя="Ульяновск" Активный="0"/>
  <Слой Тип="kml" Путь="data/KML_Samples.kml" Имя="KML Sample" Активный="0"/>
 </Слои>
</Данные>
```

XML файл траектории движения камеры

```
<Траектория>
 <Точка Широта="44.5" Долгота="33.44" Время="2" Высота="0"/>
 <Точка Широта="44.5" Долгота="33.46" Время="2" Высота="0"/>
 <Точка Широта="44.5" Долгота="33.44" Время="2" Высота="0"/>
 <Точка Широта="44.7" Долгота="33.49" Время="2" Высота="0"/>
 <Точка Широта="44.6" Долгота="33.43" Время="2" Высота="0"/>
 <Точка Широта="44.5" Долгота="33.43" Время="2" Высота="0"/>
 <Точка Широта="44.6" Долгота="33.5" Время="2" Высота="0"/>
 <Точка Широта="44.5" Долгота="33.44" Время="2" Высота="0"/>
</Траектория>
```