САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ МЕТОДОВ МЕХАНИКИ ДЕФОРМИРУЕМОГО ТЕЛА

Верховцев Андрей Николаевич

Магистерская диссертация

Разработка программы трёхмерного отображения истинной тактической обстановки

Направление 01.04.02

Прикладная математика и информатика

Магистерская программа надежность и безопасность сложных систем

Научный руководитель,
доктор физ.-мат. наук,
профессор
Павловский В. А.

Санкт-Петербург

2016

Содержание

Введение

Трёхмерная графика (3D графика) – раздел компьютерной графики, в котором изображение на экране монитора, получается в результате проецирования на плоскость трёхмерного пространства. В современном понимании, трёхмерная компьютерная графика появилась в 80х годах 20го столетия, и с тех пор бурно развивается вместе с ростом вычислительных мощностей компьютеров. Трёхмерная графика получила столь широкое распространение, что на сегодняшний день практически в любом компьютере установлен дополнительный процессор, предназначенный исключительно для вычислений связанных с её выводом. Наиболее активно трёхмерная графика применяется в сфере мультимедиа: для спецэффектов в киноиндустрии и для создания компьютерных игр. 3D графика используется в инженерных пакетах черчения и проектирования (AutoCAD и Компас). Также трёхмерная графика часто применяется для визуализации физических моделей деформации твёрдых тел, поведения текучих сред. В рамках настоящей выпускной работы трёхмерная компьютерная графика будет применена для создания видеопотока, наглядно отражающего результат работы испытываемых аппаратно-программных комплексов.

При проектировании сложных систем таких как системы навигации, наблюдения или управления, разрабатываются специальные устройства, называемые стендами моделирования. Стенд моделирования – функционально воссозданный на подобии реального оборудования программно-аппаратный комплекс. В зависимости от назначения, стенд моделирования может служить инструментом для подготовки технического персонала, а также платформой для испытания и отладки программного обеспечения изделия. При применении такого рода устройств на этапе моделирования, проработка технических решений наиболее удобна и практична по причине низкой себестоимости в сравнении с проведением опытов на реальном оборудовании. Стенды моделирования успешно

используются для отладки алгоритмов обработки информации в вышеперечисленных системах. Таким образом, магистерская диссертация посвящена разработке модуля визуализации для стенда моделирования системы освещения обстановки широкого назначения. Целью рассматриваемого стенда была отладка как систем наблюдения обитаемых судов, так и автоматических систем наблюдения и управления научно-исследовательских автономных необитаемых аппаратов.

Одной из важных частей стенда моделирования систем наблюдения является моделирование тактической обстановки, в результате которого на основе начальных данных формируется движение объекта-носителя системы и наблюдаемых объектов. Тактической обстановкой, в данном контексте, будем называть совокупность сведений о местоположении и параметрах движения объектов в заданном регионе. Имитатором тактической обстановки (ИТО) является программа, моделирующая тактическую обстановку на некотором временном интервале по заданным начальным данным.

Трёхмерное графическое отображение движения моделируемых объектов в рамках создания таких стендов имеет два основных применения. Во-первых, это наглядное представление информации на экране, доступное для восприятия не только разработчику, но и специалистам других областей. Во-вторых, подобная программа, формирующая трёхмерное изображение всех объектов с учётом их движения и взаимного расположения может быть использована в качестве имитатора выхода канала видеонаблюдения (оптоэлектронного канала наблюдения), расположенного на одном из объектов. Изменяя параметры модели, можно получить наглядную графическую интерпретацию моделируемой ситуации. Разработка программы трёхмерной визуализации тактической обстановки, а также аспекты её применения рассмотрены далее.

1. Постановка задачи

Основной целью работы была разработка компонента стенда моделирования, обеспечивающего трёхмерное графическое отображение тактической обстановки для наглядной демонстрации результатов моделирования и обеспечения выполнения стендом ряда специальных задач.

Рассмотрим подробнее вышеописанные области применения программ. Наглядность трёхмерного представления информации о движении и взаимном расположении объектов в трёхмерном пространстве не вызывает сомнений: человеку проще воспринимать информации в как можно более естественном представлении. Так как это информация о взаимном перемещении объектов, то воспринимать её в виде генерируемой в реальном времени трехмерной сцены будет значительно проще, чем в имеющихся альтернативах: табличном или двумерном графическом представлении информации. Особенно, если речь идёт об анализе данных не программистом разработчиком системы, а сторонним экспертом, не имеющим опыта работы с подобными системами.

Одним из применений, в котором трёхмерная визуализация особенно актуальна, является моделирование автоматических систем управления. На рисунке 1 приведена краткая схема ПО стенда моделирования для отладки таких систем применительно к автономным необитаемым аппаратам.

В ИТО задаётся начальная тактическая обстановка и устанавливается соединение с разрабатываемой программой. Текущая тактическая обстановка из ИТО передаётся в программу имитации средств наблюдения и навигации объекта-носителя, которые на её основе формируют массив наблюдаемых объектов (в том числе препятствий) и оценку координат и параметров движения носителя.

Рисунок 1 – Структура ПО стенда моделирования систем наблюдения и управления автономным необитаемым аппаратом

Эти данные отличаются от истинных, так как формируются с учётом ошибок и погрешностей измерения. Результаты работы систем наблюдения и навигации передаются в систему управления, которая на их основе принимает те или иные решения по управлению движения объектом. Эти решения передаются в виде управляющего воздействия обратно в ИТО, который будет менять изначально заложенное движение в соответствии с поступившими командами. Таким образом, мы получаем замкнутую систему с обратной связью. С периодичностью один раз в секунду, ИТО отправляет данные о положении объектов в трёхмерном пространстве, и на их основе изменяется положение объектов и носителя в разрабатываемой программе отображения истинной тактической обстановки (ОИТО).

Очевидно, основным результатом моделирования работы системы управления в данном случае является поведение объекта, наблюдение за которым значительно удобнее производить при трёхмерном его отображении, нежели путём анализа таблиц его параметров и проекции траектории и движения на разные плоскости.

Рисунок 2 – Структура ПО стенда моделирования оптоэлектронного канала наблюдения

На рисунке 2 представлена упрощённая структура ПО, моделирующего работу оптоэлектронного канала наблюдения. Здесь в ИТО также задаётся начальная тактическая обстановка и устанавливается соединение с разрабатываемой программой ОИТО. Роль ОИТО в данной системе – имитация выходного видеопотока, аналогичного получаемому от аппаратно-программных средств из оптоэлектронного канала наблюдения, который будет в дальнейшем обработан программой анализа видеопотока, и вместе с результатами работы других каналов наблюдения будет выведен в системе отображения. Здесь программа ОИТО используется не столько для отображения результатов моделирования, сколько в качестве используемого при моделировании инструмента. Вывод результатов здесь осуществляют штатные программы системы отображения с графическим интерфейсом пользователя. Целью такого моделирования является анализ работы, во-первых, алгоритмов обработки видеопотока, а во-вторых, проработка технических решений по организации отображения самого видеопотока в системе управления и интеграции его с остальными результатами наблюдения для оптимального представления оператору.

Подводя итог сказанному выше, задачей магистерской диссертации, является разработка модуля визуализации, основная функция которого, построение тактической обстановки в трёхмерном пространстве, непрерывно меняющейся в реальном времени в соответствии с поступающими данными от программы ИТО.

2. Обзор литературы

Кратко изложим обзор использованной в ходе выполнения выпускной работы литературы и основных публикаций по теме.

Основной справочной литературой, являются книги о специфике и принципах разработки программного обеспечения, с использованием языка C# [11], [12]. Книга [15] внесла вклад в расширение алгоритмической базы, изучении концепций и парадигм объектно-ориентированного программирования. Она заслуженно считается классической книгой по объектной технологии и не зря является первой в списке рекомендуемых книг по этой теме.

В изучении функционала графической программной платформы Unity3D, её сильных и слабых сторон помог справочник [13]. Также полезно было ознакомиться с [14], в качестве вводной части в программирование с использованием графической платформы.

Концепции и принцип работы современной трёхмерной компьютерной графики (в частности реального времени) рассмотрены в [1], [8], [16]. Шейдерное программирование (написание программ исполняемых на графическом ускорителе) изучалось по книге [17].

Основные математические понятия различных разделов математики, а также программная реализация алгоритмов (в частности быстрого преобразования Фурье) подробно и доходчиво изложены справочнике [7], который также использовался в ходе выполнения работы.

3. Ход работы

Исходя из постановки задачи, последовательно пройдем по этапам реализации.

3.1 Базовые концепции трёхмерной компьютерной графики

Из установленного ранее определения, трёхмерная графика есть проецирование трёхмерного пространства на плоскость. Чтобы получить в результате этого процесса изображение на экране, в первую очередь необходимо «населить» некоторое пространство трёхмерными объектами. В самом простом случае, объемную фигуру можно описать, перечислив координаты точек и правила по которым эти точки будут соединены многоугольниками.

На рисунке 3 приведен пример описания пирамиды в трёхмерном пространстве (в локальной системе координат объекта, также известной как local space), на котором перечислены точки (вершины) углов пирамиды, а также тройки вершин, определяющие соответствующие грани. Файл, содержащий информацию о геометрии трёхмерного объекта, называют трёхмерной моделью. О том, какая информация может быть записана в трёхмерной модели, а также о существующих форматах, более подробно описано в [1].

Обозначив форму, в которой будет задаваться объект трёхмерного пространства, необходимо определить операции, при помощи которых можно изменять его положение в глобальной системе координат (в пространстве именуемом world space). Выделяют три основных операции: перемещение (translation), вращение (rotation) и масштабирование (scaling), каждая из них осуществляется при помощи соответственно матриц перемещения, вращения и масштабирования. Применить операцию к объекту означает применить эту операцию к каждой вершине объекта.

Машинное представление:

Вершины:
1: (-1,0,1)
2: (1,0,1)
3: (1,0,-1)
4: (-1,0,-1)
5: (0,2,0)

Треугольники:
512 523 534 541 214 243

Интуитивное представление:

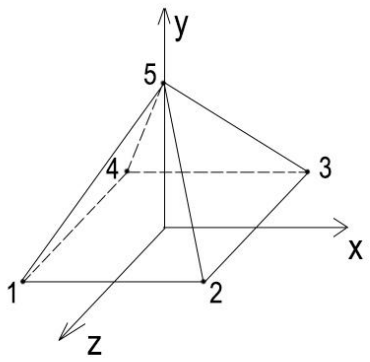


Рисунок 3 – Описание геометрии объекта: данные, которыми оперирует компьютер (слева) и наглядная их интерпретация (справа)

Положение вершины a в пространстве может быть изменено путём добавления к ней вектора перемещения t, однако, операцию перемещения удобнее применять при помощи умножения на матрицу перемещения T:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

тогда:

$$a' = a \cdot T$$

$$[a'_x \quad a'_y \quad a'_z \quad 1] = [a_x \quad a_y \quad a_z \quad 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

$$= [a_x + t_x \quad a_y + t_y \quad a_z + t_z \quad 1]$$

Это делается для того, чтобы несколько операций, перемещения, вращения и масштабирования, путём взаимного перемножения можно было объединить в одну матрицу, называемую матрицей преобразования.

Операция масштабирования применяется при помощи умножения координат вершин объекта на матрицу S следующего вида:

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

тогда:

$$a' = a \cdot S$$

$$[a'_x \quad a'_y \quad a'_z \quad 1] = [a_x \quad a_y \quad a_z \quad 1] \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [a_x \cdot S_x \quad a_y \cdot S_y \quad a_z \cdot S_z \quad 1]$$

Операция вращения имеет свои особенности – необходимо указать ось вокруг которой будем производить вращение. В самом простом случае, в трёхмерном пространстве можно производить вращение вокруг осей X, Y и Z. Тогда матрица поворота на угол α записывается следующим образом с учётом оси вращения:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$R_y = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$R_z = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Чтобы задать ориентацию объекта при последовательном вращении по нескольким осям, необходимо перемножить матрицы вращения по соответствующим осям справа налево, тогда общая матрица поворота будет иметь следующий вид:

$$R = R_z(\gamma)R_y(\beta)R_x(\alpha)$$

где α, β, γ углы поворота по осям z, y и x соответственно.

Стоит отметить, что использование углов Эйлера, для задания ориентации объекта в пространстве, не всегда удобно. Бывают ситуации, когда в следствие поворота, теряется одна степень свободы, что приводит к неожиданному поведению объекта при вращении. Эта проблема известна как складывание рамок (gimbal lock) [2]. Также, если перед нами стоит задача постепенного перехода объекта из одной ориентации в другую, траектория поворота может принять непредсказуемую форму. Обе эти проблемы, решаются использованием специализированных гиперкомплексных чисел называемых кватернионами, которые имеют следующий вид:

$$q = a + bi + cj + dk,$$

где a, b, c, d – вещественные числа, i, j, k – мнимые единицы, для которых выполняется следующее свойство:

$$i^2 = j^2 = k^2 = ijk = -1$$

Числа b, c *и* d определяют ось вращения в трёхмерном пространстве, a – угол на который осуществляется поворот. Более подробно о теории кватернионов можно посмотреть в [3], [4].

Определив способ задания геометрии объекта и элементарные операции, позволяющие задать для него произвольное положение в

пространстве (world space), можно перейти непосредственно к проецированию пространства на экран монитора. Этот процесс происходит в два шага: сначала мы должны перейти от пространства, которое “населяют” все объекты, в видимое пространство (view space или camera space), а затем применить непосредственно трансформацию из видимого пространства в пространство экрана (screen space), при помощи матрицы проецирования. Изложим эти операции последовательно.

Прежде всего, необходимо определить точку обзора, или положение камеры, которое и будет определять видимое пространство. Это вспомогательное подпространство, центром которого является положение камеры, используется для того, чтобы впоследствии отсечь все вершины объектов, которые не попадают в область видимости, и производить операцию проецирования и перспективные преобразования только для “видимых” объектов. Матрица преобразования в видимое пространство, определяемая как инверсия матрицы преобразования необходимой для задания положения и ориентирования камеры в пространстве world space, применённая ко всем объектам, переместит весь “мир” в видимое пространство (рисунок 4).

Находясь в пространстве камеры, перед тем как «сплющить» изображение, необходимо отсечь всё, что не входит в область обзора камеры. Это осуществляется при помощи куба, который иногда называют проецируемым пространством (projection space). Это пространство имеет размерности от -1 до 1 по всем осям, следовательно, вершины которые не лежат в этом интервале, находятся вне зоны видимости и тем самым отсекаются.

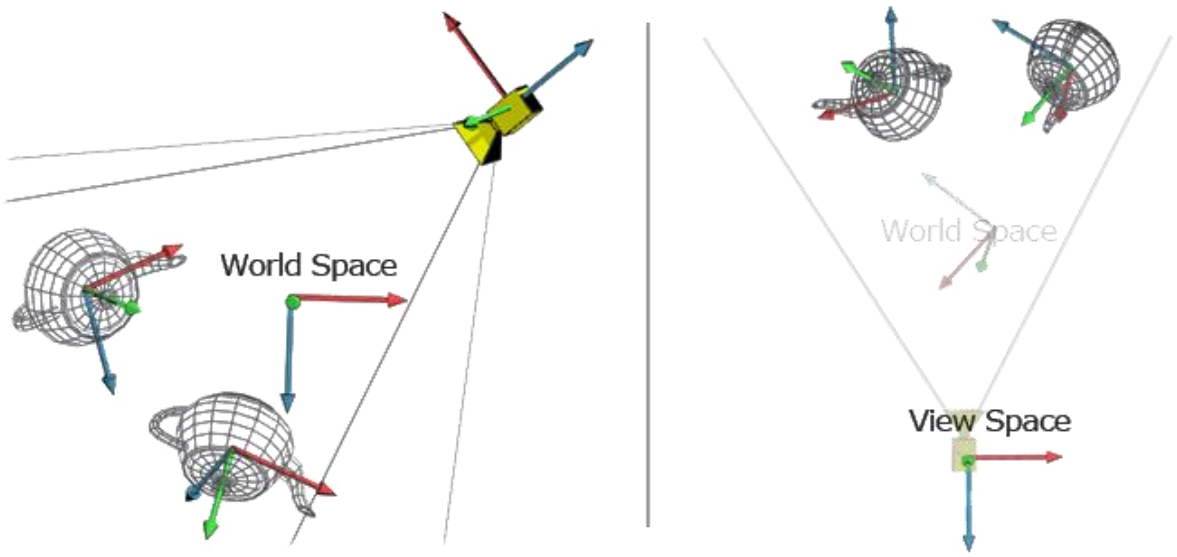


Рисунок 4 – Два чайника в world space (слева) и в видимом пространстве (справа)

Чтобы перейти от видимого пространства к проецируемому, необходима ещё одна матрица, значения которой зависят от типа проецирования. Наиболее часто используемые – ортографическое (без учёта перспективы) и, соответственно, перспективное проецирование.

Для того чтобы осуществить ортографическое проецирование, необходимо определить размеры области видимости камеры: ширину и высоту, а также ближнюю и дальнюю плоскости отсечения (рисунок 5).

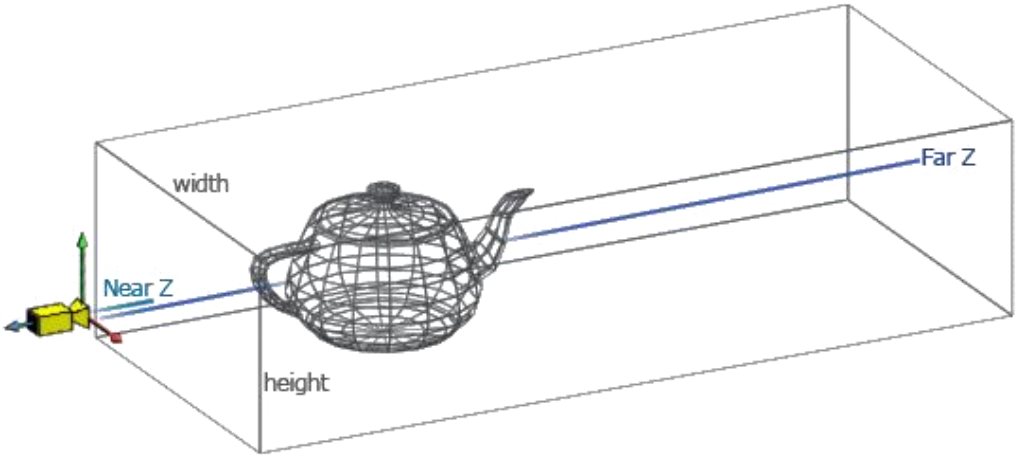


Рисунок 5 – Область видимости камеры без учёта перспективы

Зная эти величины, мы можем задать матрицу трансформации, которая переведёт координаты параллелепипеда, в координаты куба. Эта матрица имеет следующий вид:

$$\begin{bmatrix} \frac{1}{width} & 0 & 0 & 0 \\ 0 & \frac{1}{height} & 0 & 0 \\ 0 & 0 & -\frac{2}{z_{far} - z_{near}} & -\frac{z_{far} + z_{near}}{z_{far} - z_{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

На рисунке 6 представлен результат применения матрицы:

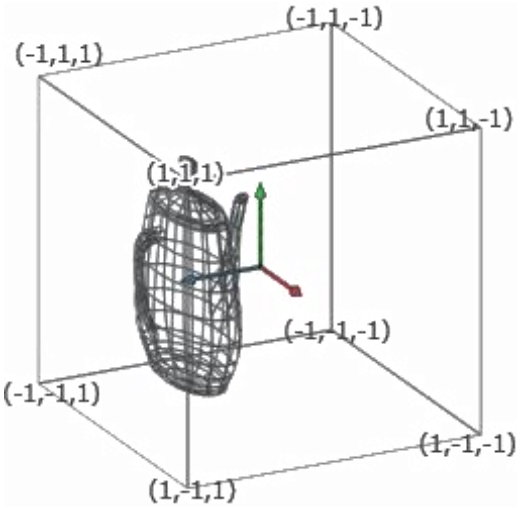


Рисунок 6 – Полученное проецируемое пространство без учёта перспективы

При перспективном проецировании область видимости определяется двумя дополнительными параметрами: горизонтальным и вертикальным углами обзора (рисунок 7).

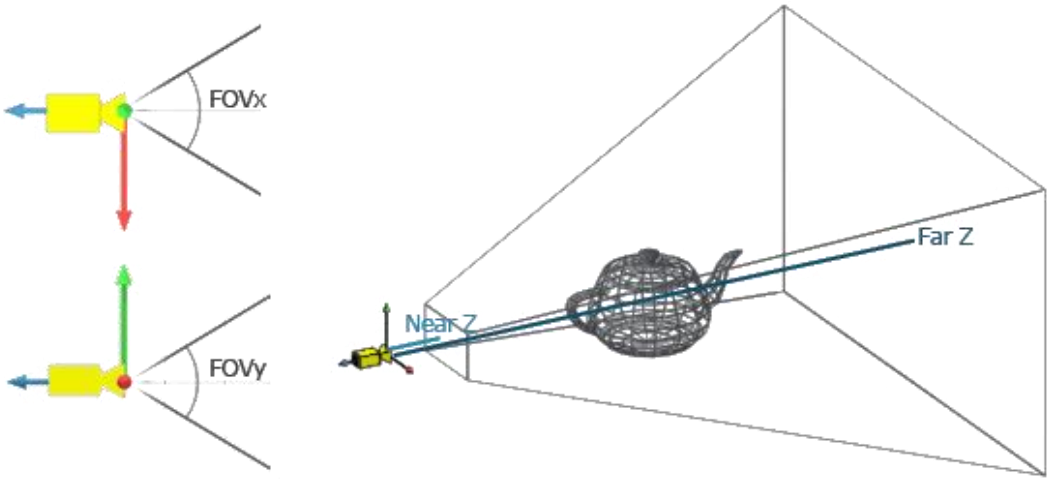


Рисунок 7 – Область видимости камеры с учётом перспективы

Матрица трансформирования приобретает следующий вид:

$$\begin{bmatrix} \tan^{-1}\left(\frac{FOV_x}{2}\right) & 0 & 0 & 0 \\ 0 & \tan^{-1}\left(\frac{FOV_y}{2}\right) & 0 & 0 \\ 0 & 0 & -\frac{z_{far}+z_{near}}{z_{far}-z_{near}} & -\frac{2(z_{far}z_{near})}{z_{far}-z_{near}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

где FOV_x и FOV_y горизонтальный и вертикальный углы обзора соответственно. Результат применения матрицы проиллюстрирован на рисунке 8.

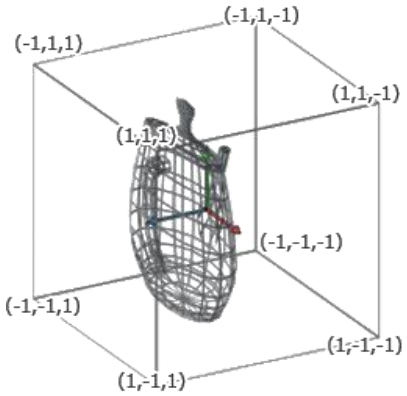


Рисунок 8 – Полученное проецируемое пространство с учётом перспективы

В результате выполненных преобразований, получены данные необходимые для выведения изображения на экран. Каждый пиксель экрана окрашивается в цвет проецируемого объекта, в который он попадает. На рисунке 9 в упрощенной форме проиллюстрирован процесс заполнения экрана цветом.

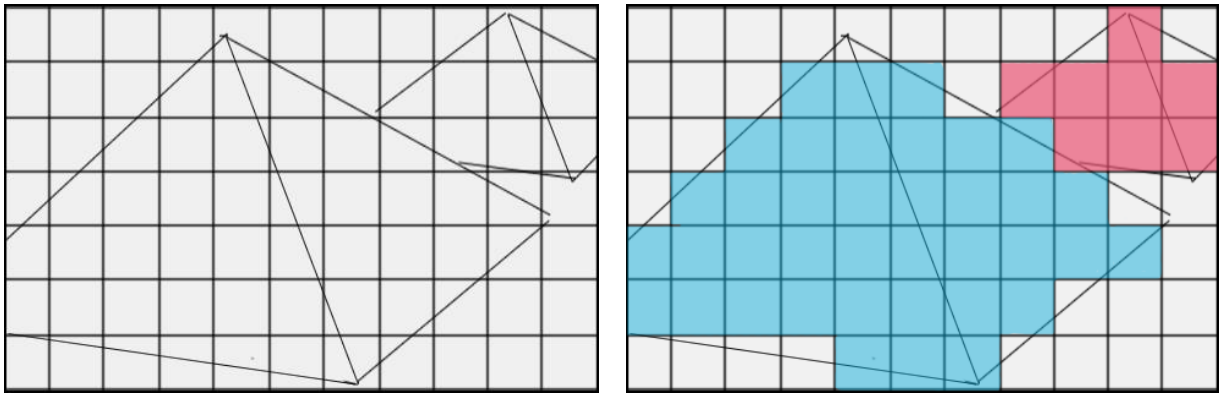


Рисунок 9 – Окрашивание пикселей экрана монитора

Цвет трёхмерного объекта в общем случае определяется его текстурой. Текстура – плоская обёртка для объемного объекта, определяющая его цвет. Правила наложения текстуры определяются, так называемой, картой текстуры. При помощи текстуры также можно охарактеризовать некоторые свойства поверхности объекта, например его гладкость/шероховатость или реакцию на освещение. В рамках базовых концепций трёхмерной графики этот вопрос не представляет большого интереса, более подробно об этом можно узнать в [5].

Процесс преобразования данных о положении объектов в трёхмерном пространстве (трёхмерной сцены) в изображение на экране называется рендеринг. В зависимости от сложности сцены, например при визуализации фотореалистичного изображения, с использованием сложных алгоритмов распространения света, этот процесс может занимать достаточно длительный промежуток времени. Предположим следующее: если мы будем постепенно изменять положение некоторого объекта сцены, в реальном времени (и непрерывно наблюдать за ним при помощи камеры), то при условии того что процесс рендеринга будет происходить с достаточной частотой мы сможем

добиться так называемой “иллюзии движения”, то есть бесшовного потока изображений, достоверно отражающего перемещение объекта. Более того, мы сможем воздействовать на объект произвольно, и изображение на экране будет создаваться “на лету”. В этом заключается концепция рендеринга в реальном времени (real-time rendering) – основы интерактивных (т. е. предполагающих пользовательский ввод) 3D приложений, например компьютерных игр. Естественно такой подход предполагает применение “упрощенных” алгоритмов визуализации, хотя стоит отметить, что современный персональный компьютер, может визуализировать довольно сложные трёхмерные сцены с частотой 30 кадров в секунду и выше.

3.2 Выбор графической платформы

Так как в настоящей работе идет речь о разработке интерактивного трёхмерного программного продукта, необходимо написать или использовать готовую программную платформу, объединяющую в себе следующий функционал: визуализацию трёхмерной графики (renderer) и обработку пользовательского ввода. Это базовый функционал программной платформы, которую принято называть Game (3D) Engine (Игровая Платформа). Выделение игровых платформ в отдельный класс программных продуктов, обуславливается тем, что это программы, существенно упрощающие разработку игровых приложений путём решения типовых задач с ней связанных. Разработка игровой платформы является сложной инженерной задачей, однако на сегодняшний день, различными компаниями создано множество коммерчески распространяемых графических платформ, каждая из которых имеет некоторую специализацию, притом базовый функционал остается одним и тем же. При разработке программы, которой посвящена данная работа, выбор пал на игровую платформу Unity3D. Эта коммерческая программная система, нашла применение, в разработке множества успешных программных продуктов, чем хорошо себя зарекомендовала на мировом

уровне. Пример трехмерной сцены, построенной с использованием графической платформы Unity3D, представлен на рисунке 10.

Рисунок 10 – Пример трехмерной сцены визуализированной в реальном времени с использованием графической платформы Unity3D

В разрабатываемой программе реалистичность изображения будет являться далеко не первостепенной задачей. Можно выделить основные причины, по которым была выбрана именно эта платформа:

1. Мультиплатформенность – проекты, написанные с использованием данной платформы, можно без проблем компилировать под все основные операционные системы.
2. Модель распространения – существуют две версии Unity3D: индивидуальная (бесплатная) и профессиональная, причём ограничения бесплатной версии данного программного продукта не являются существенными.
3. Полноценная среда разработки – свобода в написании кода компонентов программы, возможность привлечения сторонних библиотек, написанных при помощи любых языков программирования, для расширения встроенного функционала.

4. Возможность использования языка C# для написания компонентов программы.

3.3 Генерация волн на поверхности воды

Основой динамики жидкости являются уравнения Навье-Стокса, однако, их использование для расчёта движения несжимаемой жидкости в реальном времени не практично по причине колоссальной вычислительной сложности. В рамках решаемой задачи наибольший интерес вызывает характер движения воды на океанической поверхности, следовательно, необходимо разработать такую математическую модель, которая подходит для вычисления в реальном времени и обеспечивает правдоподобное поведение водной поверхности. Моделированием водной поверхности занимались многие разработчики, как результат, существует достаточно подходов к моделированию и анимации поверхности воды. Большая часть из них основана на аналитической модели суперпозиции волн, и решение здесь либо задается сразу в виде линейной комбинации тригонометрических функций со специально подобранными коэффициентами, либо получается в результате применения обратного преобразования Фурье со специально заданным спектром. Кратко изложим наиболее успешную модель, предложенную Джерри Тессендорфом (Jerry Tessendorf) [6]. Его подход нашел применение для создания спецэффектов в таких фильмах как “Титаник” (Titanic 1997) и “Водный мир” (Waterworld 1995).

Реализация волнения водной поверхности подразумевает создание поля высот $h(x, t)$, где x – некоторая точка на плоскости, t – время. Поле высот состоит из набора синусоид, с различными амплитудами и фазами. В методе Тессендорфа, предлагается генерировать эти амплитуды и фазы, основываясь на эмпирических данных океанографии, а затем при помощи быстрого преобразования Фурье, получить сумму соответствующих синусоид. Быстрое

преобразование Фурье – алгоритм быстрого вычисления дискретного преобразования Фурье (ДПФ), он подробно изложен в [7].

Рассмотрим уравнение, в котором необходимо применить быстрое преобразование Фурье для получения суммы:

$$h(\bar{x},t) = \sum_{k} \tilde{h}(\bar{k},t)\exp(i\bar{k}\cdot x)$$

Здесь $h(\bar{x},t)$ – высота в позиции $\bar{x} = (x,z)$ во времени t. $\tilde{h}(\bar{k},t)$ – комплексное число, содержащее амплитуду и фазу волны во время t, а вектор $\bar{k}$ указывает направление движения волны. Длина вектора $\bar{k}$ зависит от длины волны λ следующим образом:

$$k = 2\pi/\lambda$$

Обозначим за S размер поля высот и за N размерность сетки для быстрого преобразования Фурье (N должно быть степенью двойки). Определим вектор $\bar{k}$ следующим образом:

$$\bar{k} = (k_x, k_z) = \left(\frac{2\pi m}{S}, \frac{2\pi n}{S}\right)$$

$$-\frac{N}{2} \le m,n < \frac{N}{2}$$

$$m,n \in Z$$

Далее необходимо сгенерировать амплитуды и фазы для каждого $\tilde{h}(\bar{k},t)$. Океанографами выведен спектр соответствующий ветряным волнам на море, называемый спектр Филлипса, который определяется следующим уравнением:

$$P_h(\bar{k}) = A\frac{\exp(-\frac{1}{(kL)^2})}{k^4}\left|\hat{k}\cdot\hat{w}\right|^2$$

$L = \frac{V^2}{g}$, где V – скорость ветра и g – ускорение свободного падения;

$\hat{w}$ – направление ветра (шляпка обозначает нормализацию);

A – численное значение амплитуды.

В первую очередь создаётся начальный набор амплитуд и фаз:

$$\tilde{h}_0(\bar{k}) = \frac{1}{\sqrt{2}}(\zeta_r + i\zeta_i)\sqrt{P_h(\bar{k})}$$

Здесь, ζ_r и ζ_i две независимые случайные величины нормального распределения с математическим ожиданием 0 и средним квадратичным отклонением 1. Далее вычисляется набор частотных амплитуд в зависимости от $\bar{k}$ и t:

$$\tilde{h}(\bar{k}, t) = \tilde{h}_0(\bar{k})\exp(i\omega(k)t) + \tilde{h}_0^*(-\bar{k})\exp(-i\omega(k)t)$$

где, $\omega(k)$ – частота волны, которая устанавливается в зависимости от соответствующей длины волны следующим соотношением:

$$\omega^2(k) = gk$$

После применения данного алгоритма, получается набор амплитуд и частот изменяющийся во времени. Далее, применяя быстрое преобразование Фурье, мы получим необходимую карту вершин $h(\bar{x}, t)$.

3.4 Написание необходимых компонентов

Процесс разработки включал в себя решение нескольких основных задач. Необходимо было установить соединение с программой ИТО и принять от неё сигнал, реализовать возможность динамического чтения из базы данных программного комплекса для построения трёхмерных моделей объектов, разработать алгоритм реалистичного волнения водной поверхности, визуализировать рельеф океанического дна. Более подробно решение этих и других задач рассмотрено далее.

3.4.1 Установка соединения с ИТО

После задания тактической обстановки в ИТО, запускается разработанная программа. В ней указывается IP адрес и порт сокета, через который будет происходить соединение. Сокет соединение реализовано при

помощи стандартных библиотек C# .Net. В таблице 1 представлен протокол обмена данными между программами.

№ поля	Описание поля	Тип данных, размер (в байтах)	Допустимые значения
1	Кодовое слово начала нового сообщения	unsigned int, 4	0xFACEFACE
2	Число объектов	unsigned int, 4	[0; 50]
3	Массив данных по объектам	массив, -	-
3.1	Идентификационный номер объекта	unsigned int, 4	от 1
3.2	Идентификатор класса объекта	unsigned short, 2	0 – абстрактный объект; 1 – торпеда (Т); 2 – подводная лодка (ПЛ); 3 – автономный необитаемый подводный аппарат (АНПА); 41 – автономная гидроаккустическая станция (АГС); 42 – автономный гидроаккустический излучатель (АГС-И).
3.3	Абсолютная географическая широта объекта, °	double, 8	[-90; 90]
3.4	Абсолютная географическая долгота объекта, °	double, 8	[-180; 180]
3.5	Глубина объекта, м	float, 4	[0; 500]
3.6	Курс объекта, рад.	float, 4	[0; 2π)
3.7	Крен объекта, рад.	float, 4	(-π; π]
3.8	Дифферент объекта, рад.	float, 4	(-π; π]
3.9	Скорость объекта, м/с	float, 4	[0;80)

Таблица 1 – Состав сообщения от «Шлюза ИТО-АНПА».

На основе обработанной информации формируется список данных, длина которого зависит от количества объектов участвующих в тактической обстановке.

В момент получения первого пакета данных у программы будут все необходимые данные для построения трехмерной сцены, а именно, количество объектов, их тип и координаты. При получении второго и последующих пакетов, объекты сцены начинают реагировать на изменения, и в течение одной секунды, трехмерная сцена будет переходить в состояние соответствующее последнему полученному пакету данных. Таким образом отображаемая тактическая обстановка будет отличаться от данных, полученных от ИТО ровно на одну секунду.

3.4.2 Интеграция с базой данных

Каждому типу объекта в базе данных комплекса соответствует файл формата .3ds, содержащий трехмерную модель объекта. Этот формат был выбран в связи с тем, что он хранит в себе не только информацию о геометрии объекта, но также, в случае необходимости, можно хранить в нем и данные об анимации объекта [8]. Изначально было предложено использовать формат .fbx, однако его спецификация (структура записи данных) является закрытой. При появлении каждого нового объекта в сцене, данные о его модели считываются из базы данных комплекса и представляются в формате графической платформы для вывода в трехмерной сцене. Чтение файлов .3ds реализовано при помощи сторонней библиотеки, под названием Assimp (Open Asset Import Library) [9]. База моделей объектов может изменяться вне зависимости от разработанного программного продукта.

3.4.3 Генерация волнения водной поверхности

На момент написания выпускной магистерской диссертации, описанная ранее модель, не до конца отработана для включения в программу. Её реализация сопряжена с некоторыми дополнительными сложностями, а именно с выбором способа применения полученной карты вершин. Учитывая

то, что сама по себе модель позволяет достичь впечатляющих визуальных эффектов, вычисление спектра и расчёт поворотных множителей, даже при применении быстрого преобразования Фурье, является довольно сложной вычислительной задачей, поэтому требуется определить уровни детализации. Напомним, речь идет о визуализации океанической поверхности большого масштаба и необходимо принять во внимание тот факт что точка обзора, может меняться пользователем произвольно, следовательно отображать с максимальной детализацией требуется только те участки воды, которые находятся в непосредственной близости к камере, и упрощать уровень детализации по мере удаления.

Одним из способов направленных на решение проблемы уровней детализации, является метод проекционной сетки (Projected Grid) изложенный в [10]. Суть метода заключается в построении равномерной сетки в плоскости экрана и проецировании её на плоскость океанической поверхности (Рисунок 11). Такой подход заставляет узлы сетки кучнее располагаться по мере приближения к камере, после чего следует применять к ним полученную по методу Тессендорфа карту вершин для симуляции волнений.

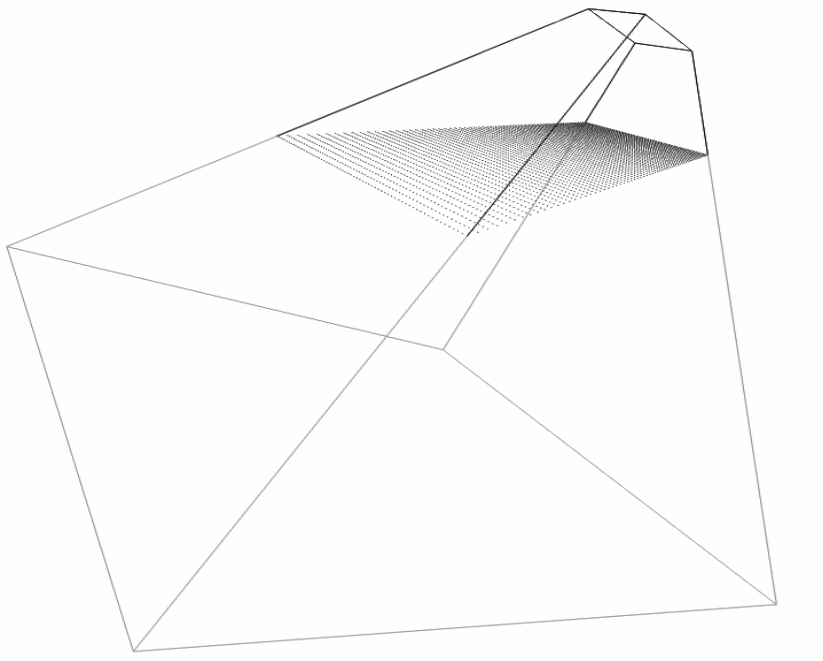


Рисунок 11 – Проекционная сетка

При своей простоте, метод все же обладает существенными недостатками. В некоторых случаях сетка не проецируется на плоскость воды (рисунок 12) или при её проецировании, полученная трапеция имеет слишком большую площадь. Сильно растянутая сетка будет иметь недостаточную плотность вблизи камеры – появятся графические артефакты.

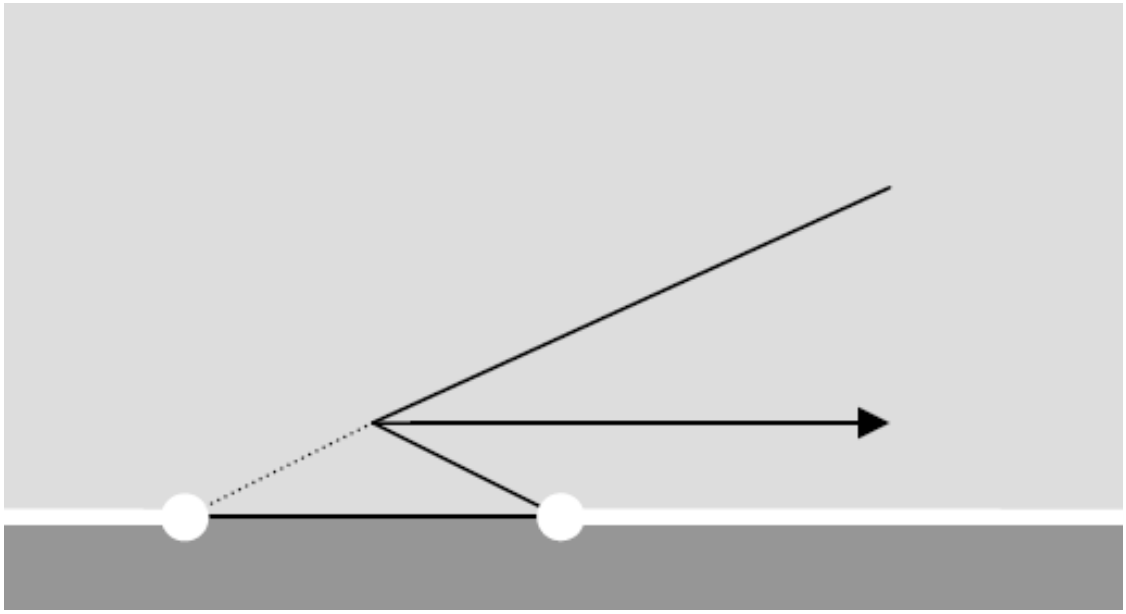


Рисунок 12 – Случай некорректного проецирования

Учитывая необходимость свободного перемещения камеры, возможно придется рассмотреть другой способ задания плоскости для визуализации, например, в виде радиальной сетки.

3.4.4 Построение рельефа дна

Данная задача имеет некоторые общие черты с симуляцией волнения водной поверхности. Аналогично то, что необходимо строить поверхность, на основании данных записанных в некоторую карту вершин, и снова необходимо учитывать дистанцию удаления рельефа от камеры, на основании чего, принимать решение, строить ли рельеф поверхности и, если строить, то с каким уровнем детализации. Однако здесь не стоит вопроса о

непосредственной генерации карты вершин, она передается из программы ИТО в зависимости от положения камеры.

Предполагается строить рельеф размера достаточного, для того чтобы покрывать область видимости камеры, используя для этого карту вершин, записанную в текстуру фиксированного разрешения. Таким образом, чем ближе камера будет находиться ко дну, тем меньше будет размер строимой области рельефа, и тем выше его детализация.

На настоящий момент решение этой задачи имеет меньший приоритет по сравнению с остальными, поэтому построение рельефа дна планируется к реализации в будущем, после визуализации волнения водной поверхности.

3.4.5 Прочее

К прочим задачам можно отнести создание интерфейса взаимодействия с пользователем, создание формы, в которую при установке соединения вписывается IP адрес и порт сокета, инструменты для управления положением камеры в пространстве, список объектов с возможностью переключения камеры в режим наблюдения.

Вывод

Разработан компонент стенда моделирования, обеспечивающий трёхмерное графическое отображение тактической обстановки для наглядной демонстрации результатов моделирования работы системы управления, позволяющий проводить наблюдение за объектом или создавать изображения подаваемые на вход программе анализа видеопотока.

Для иллюстрации результатов работы программы приведём тестовый эпизод. На рисунке 13 представлен пример задания тактической обстановки в программе ИТО. В этом эпизоде мы задаём пять тестовых объектов, устанавливаем скорость с которой они будут плыть и направление движения. В целях эксперимента, предпишем каждому из них плыть держа курс на север, с различной скоростью. По нажатию кнопки “пуск” объекты начнут перемещение. Затем запустим программу отображения истинной тактической обстановки, и установим соединение с программой ИТО. После установки соединения, заданная тактическая ситуация визуализируется с применением трёхмерной графики в программе отображения истинной тактической обстановки. На рисунке 14 можно увидеть результат работы разрабатываемой программы. В процессе работы программы ИТО можно свободно менять параметры движения объектов, удалять существующие и добавлять новые.

Непосредственно в самой программе отображения истинной тактической обстановки, можно осуществлять привязку камеры к объектам или же свободно изменять её положение в трёхмерном пространстве.

Таким образом, при запуске разработанной программы, пользователю представлен результат работы ИТО в наглядной форме. Для удобства наблюдения за объектами в сцене положение камеры не фиксировано и может быть задано произвольно при помощи соответствующих команд, или управляться непосредственно через интерфейс программного комплекса.

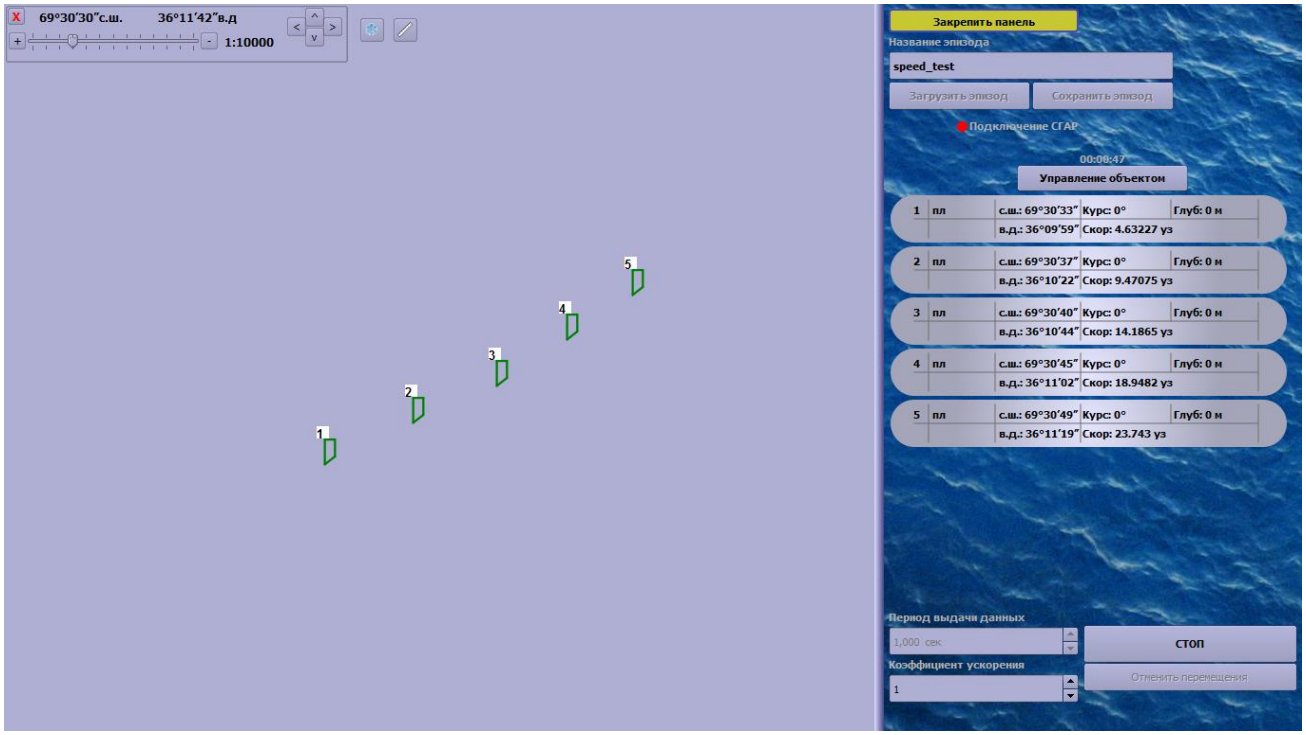


Рисунок 13 – Интерфейс программы ИТО. Создание тактической обстановки

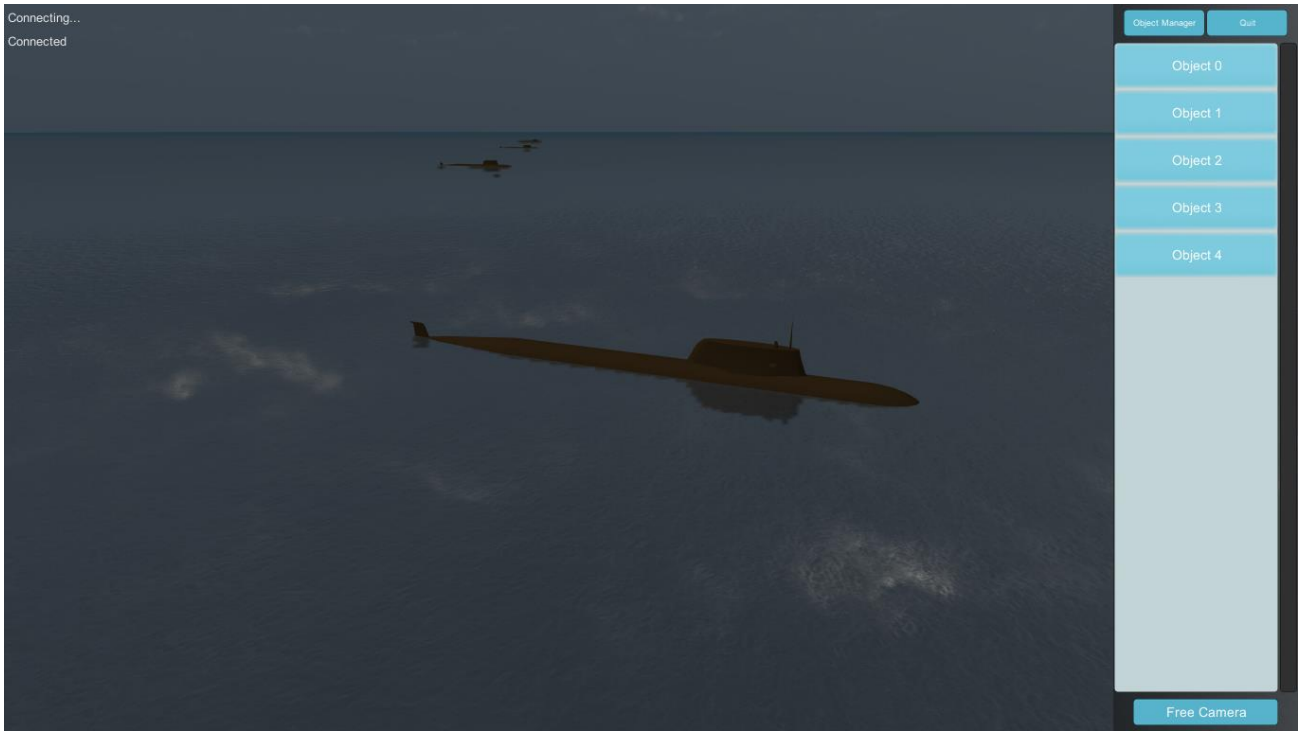


Рисунок 14 – Полученная трехмерная сцена

Другие примеры изображений, построенных с применением разработанной программы, можно найти в приложении.

Заключение

Разработана программа, выполняющая подключение и выводящая на экран изображение, соответствующее истинной трехмерной графической интерпретации тактической ситуации, аналогичное состоянию моделируемой системы, основываясь на полученных от ИТО данных. Имеется возможность использовать программу для генерации изображения, подаваемого на вход программе для анализа видеопотока.

Программа опирается на общую базу данных имитационно-моделирующего комплекса, которая может быть модифицирована вне зависимости от настоящего программного продукта.

Разработанная программа нашла применение в проведении моделирования на стендах и успешно внедрена в работающие имитационно-моделирующие комплексы. Программа наглядно отображает информацию и позволяет организовывать видеозахват экрана с трансляцией его в сеть для обеспечения моделирования оптоэлектронного канала.

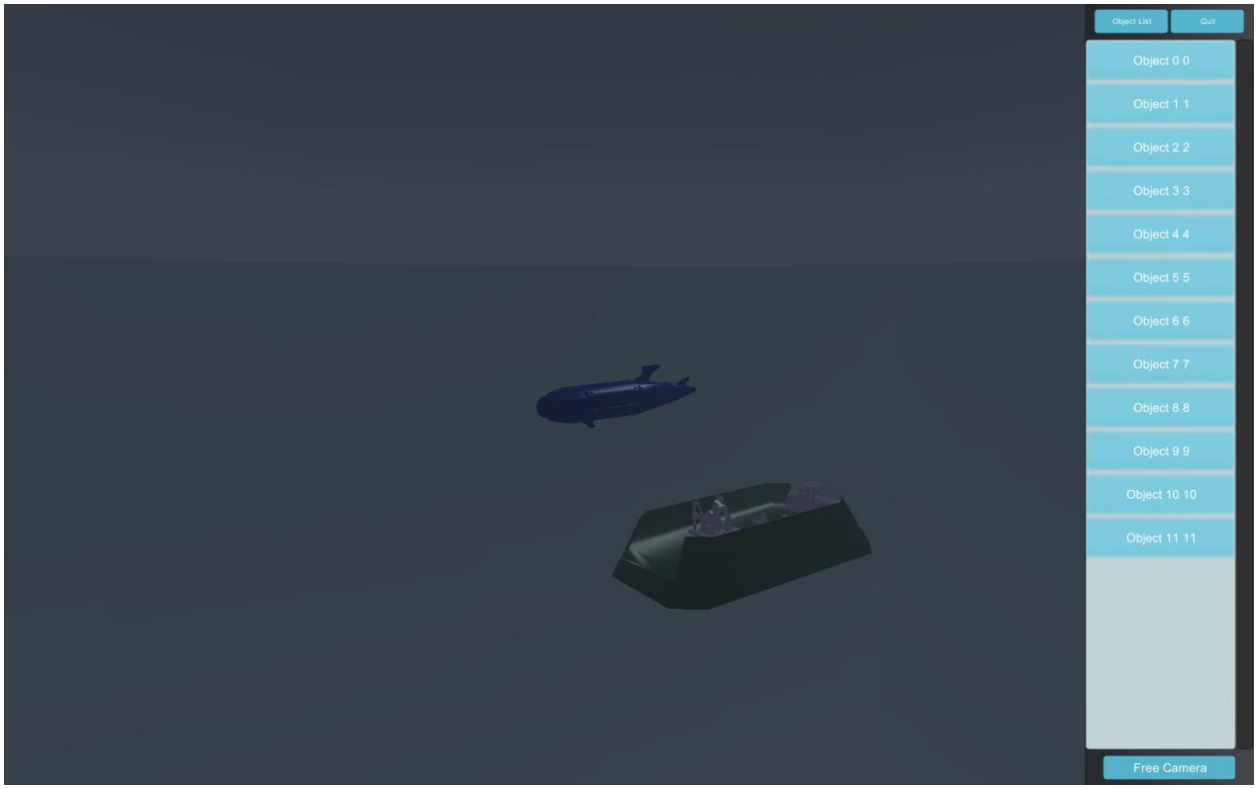
В будущем, планируется применить описанную модель для генерации волнения водной поверхности, а также решить задачу построения и отображения океанического рельефа дна.

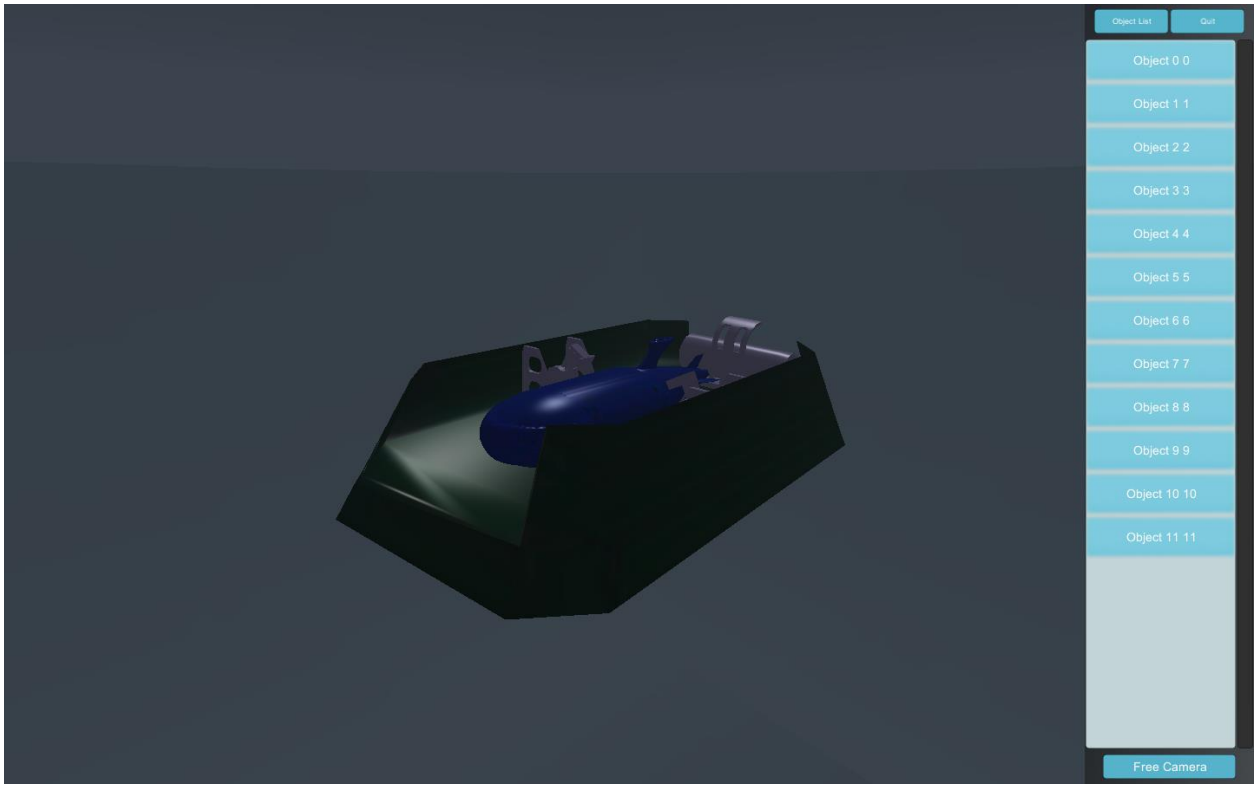
Список литературы

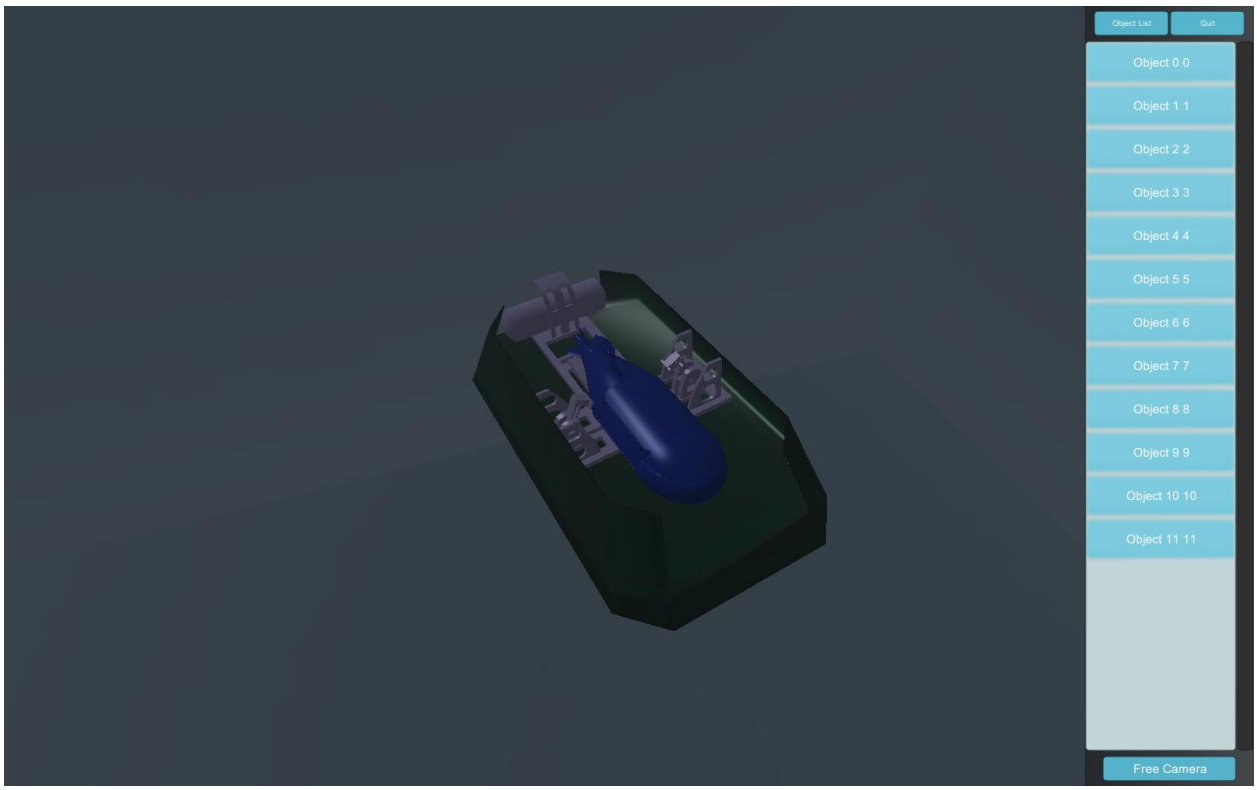
1. Kenton McHenry, Peter Bajcsy, An Overview of 3D Data Content, File Formats and Viewers. – University of Illinois, 2008
2. New perspective on the gimbal lock problem / Danail S. Brezov, Clementina D. Mladenova, Ivailo M. Mladenov – AIP Publishing
3. Гордеев В.Н. Кватернионы и Трёхмерная Геометрия – Киев, 2012
4. Andrew J. Hanson, Visualizing Quaternions, 1st Edition – Computer Science Department, Indiana University
5. Mark Masters, Essential 3D Texturing Terms, http://blog.digitaltutors.com/cover-bases-common-3d-texturing-terminology/
6. Jerry Tessendorf, Simulating Ocean Water – ACM SIGGRAPH, 2001
7. Numerical Recipes in C / William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery – Cambridge University Press
8. 3D-Studio File Format / Martin van Velsen, Robin Fercoq, Jim Pitts, Albert Szilvasy
9. Assimp Documentation and C/C++ Reference, http://assimp.sourceforge.net
10. Claes Johanson, Real-Time Water Rendering – Lund University
11. Джозеф Албахари, Бен Албахари, C# 6.0. Справочник. Полное описание языка. – O'Reilly, 2015
12. Эндрю Стиллмен, Дженнифер Грин, Изучаем C#. – O'Reilly, 2012
13. Joe Hocking, Unity in Action: Multiplatform Game Development in C# with Unity 5, 1st Edition – Manning, 2015
14. Alex Okita, Learning C# Programming with Unity3D – CRC Press Book, 2014
15. Бертран Мейер, Объектно-ориентированное конструирование программных систем – Интернет Университет, 2005
16. Tomas Akenine-Moller, Eric Haines, Naty Hoffman, Real-Time Rendering, Third Edition – CRC Press Book, 2008

17. David Wolff, OpenGL 4.0 Shading Language Cookbook – PACKT Publishing, 2011

Приложение

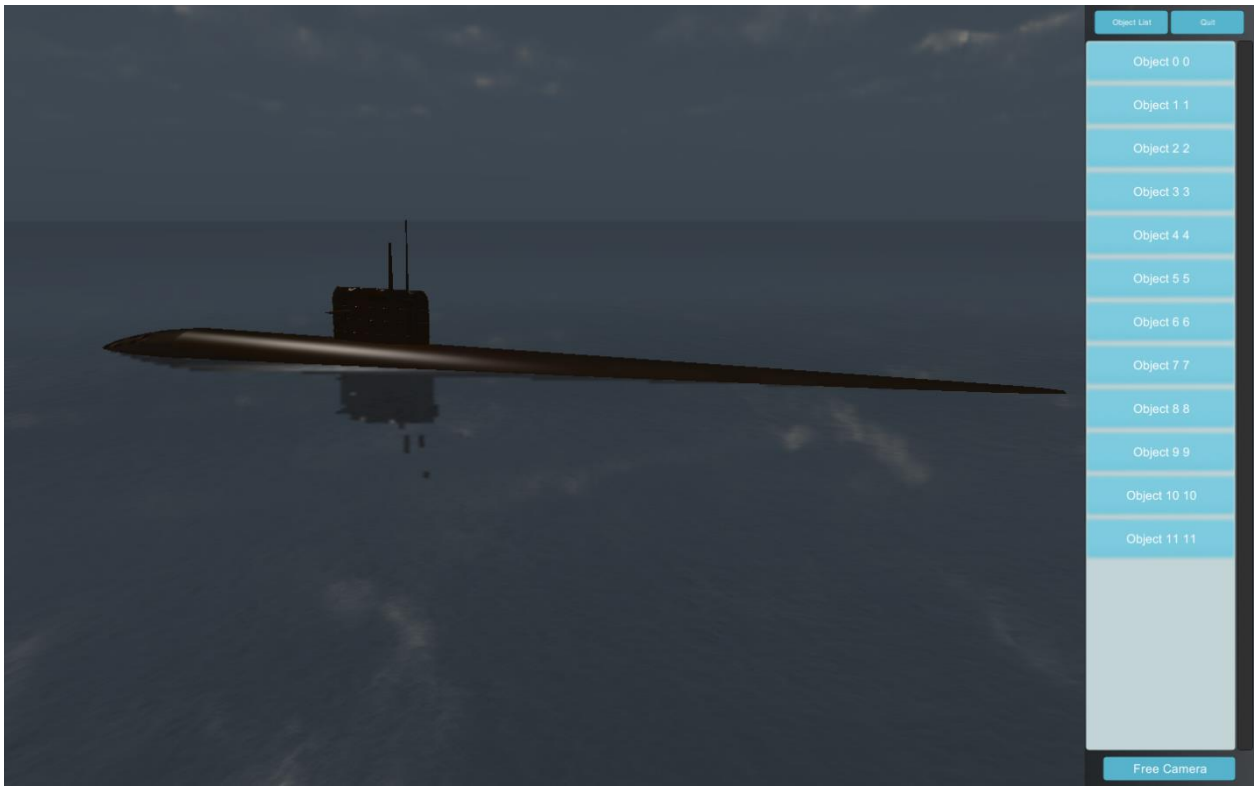

Object List
Quit
Object 0 0
Object 1 1
Object 2 2
Object 3 3
Object 4 4
Object 5 5
Object 6 6
Object 7 7
Object 8 8
Object 9 9
Object 10 10
Object 11 11
Free Camera


Object List
Quit
Object 0 0
Object 1 1
Object 2 2
Object 3 3
Object 4 4
Object 5 5
Object 6 6
Object 7 7
Object 8 8
Object 9 9
Object 10 10
Object 11 11
Free Camera


Object List
Quit
Object 0 0
Object 1 1
Object 2 2
Object 3 3
Object 4 4
Object 5 5
Object 6 6
Object 7 7
Object 8 8
Object 9 9
Object 10 10
Object 11 11
Free Camera


Object List
Quit
Object 0 0
Object 1 1
Object 2 2
Object 3 3
Object 4 4
Object 5 5
Object 6 6
Object 7 7
Object 8 8
Object 9 9
Object 10 10
Object 11 11
Free Camera


Object List
Quit
Object 0 0
Object 1 1
Object 2 2
Object 3 3
Object 4 4
Object 5 5
Object 6 6
Object 7 7
Object 8 8
Object 9 9
Object 10 10
Object 11 11
Free Camera


Object List
Quit
Object 0 0
Object 1 1
Object 2 2
Object 3 3
Object 4 4
Object 5 5
Object 6 6
Object 7 7
Object 8 8
Object 9 9
Object 10 10
Object 11 11
Free Camera