Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Разработка программного обеспечения для моделирования упругих столкновений объектов в пространстве.

Студент ИУ7-54Б _____ К. А. Рунов
(Группа) (Подпись, дата) (И. О. Фамилия)

Руководитель курсовой работы _____ А. А. Павельев
(Подпись, дата) (И. О. Фамилия)

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На сегодняшний день компьютерная графика является неотъемлемой частью нашей жизни и используется повсеместно. Ей находится применение в самых разных областях человеческой деятельности: она используется в науке, в бизнесе, в кино, играх и везде, где нужно визуальное представление информации на электронном дисплее.

Перед разработчиками графического программного обеспечения часто стоит задача синтеза реалистического изображения. Для решения этой задачи существует множество алгоритмов, но, как правило, алгоритмы, дающие наилучшие результаты являются наиболее трудозатратными как по объёму требуемой памяти, так и по количеству требуемых вычислений, потому что учитывают множество световых явлений (дифракция, интерференция, преломление, поглощение, множественное отражение). Поэтому, в зависимости от задачи и от имеющихся вычислительных мощностей, программистам приходится выбирать наиболее целесообразные в их случае алгоритмы и жертвовать либо временем генерации кадра, либо его реалистичностью.

Но генерация реалистического изображения — это лишь одна из многих задач, которые стоят перед программистами графических приложений. При разработке игр или приложений для моделирования физики твёрдого тела возникает задача обнаружения коллизий (столкновений) между объектами виртуального пространства и реагирования на них, например, разрушение объектов в результате столкновения, деформация объектов, или их упругое соударение.

В данной курсовой работе было решено создать программу — песочницу, в которой пользователь сможет размещать объекты в виртуальном пространстве, изменять их свойства (такие как размер, местоположение, цвет, масса), задавать им начальные скорости; после чего наблюдать их перемещение и столкновения.

Цель работы — разработка программы для моделирования упругих столкновений объектов в пространстве.

Для достижения поставленной цели требуется решить следующие задачи:

— описать свойства объекта, которыми он должен обладать для моделирования его движения и столкновения с другими объектами;

— проанализировать существующие способы представления объектов и обосновать выбор наиболее подходящего для решения поставленной задачи;

— проанализировать существующие алгоритмы удаления невидимых линий и поверхностей и обосновать выбор тех из них, которые в наибольшей степени подходят для решения поставленной задачи;

— проанализировать существующие алгоритмы обнаружения коллизий и обосновать выбор тех из них, которые в наибольшей степени подходят для решения поставленной задачи;

— проанализировать существующие модели освещения и обосновать выбор модели, наиболее подходящей для решения поставленной задачи;

— реализовать выбранные алгоритмы;

— разработать программное обеспечение для решения поставленной задачи;

— провести анализ производительности работы программы в зависимости от количества объектов на сцене и их типов.

1 Аналитическая часть

В данной части под «сценой» понимается виртуальное пространство, в котором расположены объекты, предназначенные для визуализации.

1.1 Описание объектов сцены

В данном разделе будут описаны свойства объектов, которыми он должен обладать для моделирования его движения и столкновения с другими объектами.

Для визуализации объекта, он должен содержать следующую информацию:

— геометрическая информация, какую форму имеет объект;

— информация о местоположении в пространстве (с учётом поворота и масштабирования);

— информация о цвете и/или текстуре объекта.

Для моделирования движения и столкновения объектов, объекты должны содержать следующую информацию:

— информация о «коллайдере» (как правило, упрощённая форма исходного объекта, которая используется при обнаружении столкновений);

— информация о физических свойствах объекта (скорость, ускорение, масса).

1.2 Выбор представления объектов сцены

Далее будут рассмотрены возможные способы представления объектов сцены.

1.2.1 Каркасная модель

Объект представляется с помощью его вершин и рёбер, без каких-либо поверхностей.

Преимущества:

— простота;

— позволяет получить базовое представление о форме объекта;

— быстрая визуализация.

Недостатки:

— не позволяет производить реалистическое освещение, для которого требуется информация о гранях объекта;

— не содержит информации, нужной для обнаружения столкновений.

1.2.2 Поверхностная модель

Объект представляется аппроксимированно, в виде набора поверхностей. Набор поверхностей можно задать как аналитически (уравнением или системой уравнений), так и в виде полигональной сетки. Часто бывает проще задать поверхности в виде полигональной сетки: набора граней и набора вершин, из которых грани состоят.

Преимущества:

— простота;

— возможность учёта освещения;

— возможность достижения высокого уровня реализма;

— содержит информацию о поверхностях, которая нужна при обнаружении столкновений объектов;

Недостатки:

— не математически точное представление, аппроксимация.

1.2.3 Твердотельная модель

Существует несколько методов представления твердотельных моделей: метод констркутивного представления (англ. Constructive representation, сокращённо C – rep) и метод граничного представления (англ. Boundary representation,

сокращённо B – rep). Оба метода предоставляют наиболее полное описание объекта, включая его внешнюю форму и внутреннюю структуру.

Преимущества:

— математически точное представление;

— наиболее полное описание структуры объекта.

Недостатки:

— сложность;

— требовательность к памяти;

— содержит излишнюю информацию, которая не будет использована при обнаружении столкновений объектов.

Вывод

На основе проведённого анализа различных способов представления объектов сцены, была выбрана поверхностная модель, так как она обладает всей информацией, нужной для обнаружения столкновений объектов и учёта освещения, а также является менее требовательной к памяти по сравнению с твёрдотельной моделью.

1.3 Выбор алгоритма удаления невидимых линий и поверхностей

1.3.1 Алгоритм Робертса

1.3.2 Алгоритм обратной трассировки лучей

1.3.3 Алгоритм Варнока

1.3.4 Алгоритм, использующий z-буфер

Вывод

1.4 Выбор алгоритмов обнаружения коллизий

В данном разделе будут проанализированы алгоритмы обнаружения коллизий и выбраны те из них, которые будут использоваться в разработанной программе.

1.4.1 Алгоритм обнаружения коллизий сферы относительно сферы

Алгоритм обнаружения коллизий сферы относительно сферы очень прост: две сферы пересекаются, если длина вектора, проведённого из центра одной сферы к центру другой, будет меньше суммы радиусов этих сфер.

Два объекта считаются «столкнувшимися», если пересекаются их сферические оболочки.

Таким образом, алгоритм будет достаточно точно обнаруживать коллизии сферообразных объектов, но менее точно — для объектов, сильно отличающихся от сфер, например, для длинных тонких объектов.

1.4.2 Алгоритм AABB

Согласно алгоритму ограничивающего прямоугольного параллелепипеда, выровненного по осям (англ. Axis Aligned Bounding Box, сокращённо AABB), объекты заключаются в оболочки из прямоугольных параллелепипедов, чьи

рёбра параллельны осям координат, после чего применяется несколько последовательных тестов для определения, пересекаются эти оболочки или нет.

1.4.3 Алгоритм OBB

Согласно алгоритму ориентированного ограничивающего параллелепипеда (англ. Oriented Bounding Box, сокращённо OBB), объекты заключаются в оболочки из прямоугольных параллелепипедов, чья ориентация совпадает с ориентацией самих объектов, после чего проверяется, пересекаются эти оболочки или нет.

Его преимущество относительно алгоритма AABB заключается в том, что при повороте объекта в пространстве, поворачивается также и его OBB — оболочка, в то время как AABB — оболочку надо пересчитывать заново.

Однако проверку пересечения AABB — оболочек выполнять гораздо быстрее и проще, чем проверку пересечения OBB — оболочек.

1.4.4 Алгоритм GJK

Алгоритм Гильберта — Джонсона — Кирти (англ. Gilbert — Johnson — Keerthi, сокращённо GJK) позволяет обнаруживать пересечения любых выпуклых многогранников. В алгоритме используется геометрическая операция под названием «сумма Минковского» (иногда ошибочно называемая разностью Минковского). Для двух множеств точек $A, B \subset \mathbb{R}^3$ сумма Минковского определяется как:

$$A \oplus B = \{\boldsymbol{a} + \boldsymbol{b} : \boldsymbol{a} \in A, \boldsymbol{b} \in B\},$$

где, для векторов $\boldsymbol{a} = (a_x, a_y, a_z)$ и $\boldsymbol{b} = (b_x, b_y, b_z)$, сумма определена, как

$$\boldsymbol{a} + \boldsymbol{b} := (a_x + b_x, a_y + b_y, a_z + b_z).$$

У суммы Минковского есть несколько полезных свойств, которые используются в алгоритме.

1) Сумма Минковского двух выпуклых многогранников есть выпуклый многогранник.

2) Если два выпуклых многогранника P и Q пересекаются, то центр координат находится внутри выпуклой оболочки $P \oplus \{-\boldsymbol{q} : \boldsymbol{q} \in Q\}$.

3) Если найдётся хотя бы одно множество точек $S \subseteq P \oplus \{-\boldsymbol{q} : \boldsymbol{q} \in Q\}$, которое заключает в себе центр координат, то и $P \oplus \{-\boldsymbol{q} : \boldsymbol{q} \in Q\}$ заключает в себе центр координат.

Алгоритм заключается в поиске многогранника, составленного из точек суммы Минковского, который содержит в себе центр координат. Также в алгоритме используются несколько хитростей, позволяющих считать не всю сумму Минковского целиком, а только её часть.

Вывод

	Алгоритм обнаружения коллизий сферы относительно сферы	Алгоритм AABB	Алгоритм OBB	Алгоритм GJK
Вычислительная нагрузка	Низкая	Низкая	Средняя	Высокая
Точность обнаружения коллизий у сложных объектов	Низкая	Низкая	Средняя	Высокая
Сложность реализации	Низкая	Низкая	Средняя	Высокая

На основе проведённого анализа алгоритмов обнаружения коллизий, для реализации в программе были выбраны алгоритмы AABB (для обнаружения коллизий на ранней стадии) и GJK (для обнаружения коллизий на поздней стадии).

Выбор сразу нескольких алгоритмов обнаружения коллизий обусловлен тем, что в программе объекты могут быть часто расположены на больших, относительно своих размеров, расстояниях друг от друга, и эффективнее будет на ранней стадии убрать из рассмотрения те объекты, чьи менее точные выпуклые оболочки не пересекаются, ведь если не пересекаются и они, то тем более не пересекаются и сами объекты. Такой подход часто применяется в физических движках и носит название — обнаружение коллизий на ранней стадии (англ. Narrow Phase Collision Detection).

1.5 Выбор модели освещения

Вывод

2 Конструкторская часть

В данной части будут приведены требования к программному обеспечению, на формальном языке будут описаны алгоритмы, которые будут реализованы при разработке программного обеспечения, а также будет обоснован выбор типов и структур, которые будут использованы при разработке, и приведена общая архитектура разрабатываемой программы.

2.1 Требования к программному обеспечению

2.2 Разработка алгоритмов

2.2.1 Общий алгоритм решения поставленной задачи

2.2.2 Алгоритм, использующий z-буфер

2.2.3 Алгоритм AABB

2.2.4 Алгоритм GJK

2.2.5 Алгоритм EPA

2.2.6 Модель освещения Фонга

2.3 Выбор типов и структур данных

2.4 Общая архитектура разрабатываемой программы

Вывод

3 Технологическая часть

В данной части будет обоснован выбор графического API, языка программирования и среды разработки, которые будут использоваться при разработке программного обеспечения. Также будет приведена UML-диаграмма классов, описывающая структуру программы. Будет продемонстрирован интерфейс, и будут приведены примеры работы программы.

3.1 Средства реализации

Далее будет обоснован выбор языка программирования и средств разработки, использованных при разработке программы.

3.1.1 Выбор графического API

В качестве используемого графического API был выбран OpenGL по следующим причинам:

— кроссплатформенность (в отличие от DirectX);

— простота инициализации (в отличие от Vulkan);

— использование вычислительных мощностей графического ускорителя (в отличие от рендеринга на процессоре);

— спецификация OpenGL реализована в драйверах большинства массовых видеокарт (NVIDIA, AMD, Intel).

3.1.2 Выбор языка программирования

В качестве используемого языка программирования был выбран C++ по следующим причинам:

— язык широко используется при разработке графических приложений;

— наличие комплияторов, генерирующих высокопроизводительный исполняемый код;

— язык типизирован, в связи с чем в процессе разработки возникает меньше ошибок времени выполнения;

— наличие библиотек для обеспечения доступа к функциям OpenGL (glad), а также для создания окон и управления вводом (GLFW);

— наличие математических библиотек (glm);

— язык поддерживается отладчиками gdb и RenderDoc;

— наличие кроссплатформенной утилиты для автоматической сборки программы cmake.

3.1.3 Выбор среды разработки

В качестве среды разработки был выбран текстовый редактор Neovim по следующим причинам:

— высокая отзывчивость в отличие от графических сред разработки, таких как Visual Studio, Visual Studio Code, Clion, QtCreator;

— полная поддержка vim-движений, что ускорит навигацию по исходному коду проекта;

— возможность использования протокола Language Server, что позволит определять наличие ошибок времени компиляции в коде без необходимости компиляции проекта;

— наличие расширений, ещё больше ускоряющих процесс разработки и навигации по исходному коду проекта (harpoon, vim-fugitive, nvim-tree, nvim-cmp, nvim-treesitter, nvim-lspconfig, telescope-nvim, luasnip, vim-surround, vim-commentary, friendly-snippets).

3.2 Структура программы

3.3 Интерфейс

3.4 Работа программы

Далее будут приведены замечания относительно работы программы.

3.5 Демонстрация работы программы

4 Исследовательская часть

4.1 Технические характеристики

4.2 Влияние количества объектов на скорость генерации кадра

4.3 Влияние типов объектов на скорость генерации кадра

Вывод

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ