`

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет _____Компьютерных Технологий и Управления_____
(название факультета)
Кафедра _____Информатики и Прикладной математики_____
(название кафедры)
Направление подготовки (специальность) _____Программная инженерия_____

О Т Ч Е Т

о практике

Тема задания: ___*Симуляция взаимодействия твёрдых тел (rigid body simulation)*_____

Студент ___Широков О.И.____ _____гр. 2120____
(Фамилия И.О.) (номер группы)

Руководитель практики ____Перминов И.В_____
(Фамилия И.О., должность и место работы)

Оценка, рекомендуемая руководителем _____

Практика пройдена с оценкой _____

Подписи членов комиссии

_____(_____)
(Фамилия И.О.)
_____(_____)
(Фамилия И.О.)
_____(_____)
(Фамилия И.О.)

Дата _____

Санкт-Петербург
2014г.

Оглавление

Введение

Цели

Знакомство с физическими движками и написание приложения, демонстрирующего взаимодействие твердых тел.

Использованные технологии

В процессе разработки приложения были использованы следующие технологии: программный интерфейс OpenGL[1] и физический движок Bullet[2]. Разработка велась на языке C++. Выбор OpenGL обусловлен плаформо-независимостью и наличием обширной документации. В качестве аналогов можно привести DirectX API[3], но его использование не позволяет написать платформо-независимый код.

При выборе физического движка рассматривались следующие критерии:

- Открытость
- Наличие документации
- Доступность для различных платформ

В качестве альтернатив рассматривались Newton[4] и Tokamak[5]. Newton был отброшен из-за отсутствия документации, а Tokamak из-за отсутствия поддержки платформы разработки (Mac OS X).

Кроме того Bullet поддерживает SIMD расширения, многопоточность, и кроме симуляции поведения твердых тел (rigid bodies) позволяет также симулировать поведение деформируемых тел (soft bodies).

Теоретические сведения

Матрицы преобразований

Тот факт, что преобразования в линейной алгебре можно представлять с помощью матриц, широко используется в компьютерной графике.

В OpenGL зачастую используются следующие:

- Модельная матрица (Model Matrix) эта матрица, используемая для задания положения объекта в пространстве.
- Проекционная матрица (Projection Matrix) матрица, которая используется для задания параметров проекции
- Видовая матрица (View Matrix) матрица, определяющая положение и направление камеры.

Результатом перемножения всех этих матриц на вектор координаты точки станет вектор определяющий положение точки на экране.

Физические движки

Физическим движком называют специализированное ПО предназначенное для моделирования физических процессов таких как: симуляция столкновения твердых тел(rigid bodies), симуляция поведения деформируемых тел (soft bodies), динамика жидкостей, газов.

Симуляция столкновения твердых тел (Rigid body simulation)

Под твердым телом понимается такое тело, в котором дистанция между любыми двумя точками всегда остается прежней, другими словами, деформация такого тела невозможна. Твердое тело описывается множеством параметров, включая: положение центра масс, импульс, угловой момент, масса, кинетическая энергия, и т.п.

Симуляция столкновения делится на две составляющие: Обнаружение столкновений (Collision Detection) и Реакция на столкновение (Collision Reaction). Первая стадия будет подробнее описана ниже. Во время второй стадии вычисляются новые параметры тел в соответствии с физическими законами.

Обнаружение столкновений

Под обнаружением столкновений обычно понимают задачу о нахождении пересечения физических объектов во время физической симуляции. Это одна из самых сложных и затратных частей симуляции. Зачастую обнаружение столкновений разбивается на две фазы:

- Широкая фаза (Broad phase) определяет объекты, которые потенциально могут пересекаться, для уменьшения количества обрабатываемых объектов в узкой фазе. Для определения потенциально пересекающихся объектов объекты «ограничиваются» более простыми геометрическими фигурами (Bounding volume) как, например, AABB(Axis-Aligned Bounding Box) или BS (Bounding Sphere), пример на Рисунке 1.

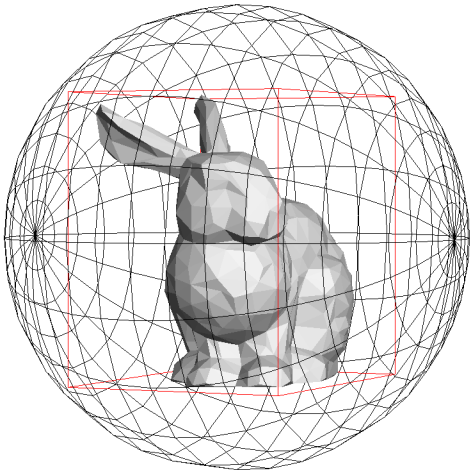

Рисунок 1: Ограничивающий цилиндр (AABB) и ограничивающая сфера (BS)

Часто используют один из следующих алгоритмов:

  - Метод грубой силы(Brute force) проверка пересечения ограничивающих фигур для каждого объекта с каждым другим. Сложность этого алгоритма получается равной $O(n^2)$

    - Spatial Hashing. В этом алгоритме все пространство разбивается на ячейки содержащие объекты и на столкновение проверяются только объекты из соседних ячеек.
    - Sort And Sweep. Суть этого объекта состоит в сортировке координат ограничивающих элементов вдоль какой-либо оси, и объекты помечаются для обработки в следующей фазе, если их координаты пересекаются.

- Узкая фаза (Narrow phase) стадия в которой определяются столкновения объектов, найденных в предыдущей Широкой фазе, в данной фазе используются такие алгоритмы как V-Clip и GJK-EPA[6].

Реализация

Структура программы

Программа симуляции написана с использованием прадигмы ООП. Программа содержит следующие классы:

- Box — класс, который описывает поведение и состояние объектов симуляции
- Camera — класс, хранит состояние камеры
- Scene — класс, содержащий информацию о объектах участвующих в симуляции

И файлы:

- inputUtils.cpp — в нем определены функции ввода с клавиатуры и мыши
- MyShadersCompile.cpp — в этом файле определяются функции для компиляции шейдеров (OpenGL)Shaders)[7]

Использование OpenGL и Bullet

Для хранения ссылок на буферы и уменьшения количества обращений к API используются VAO (Vertex Array Object). Инициализация буферов и VAO происходит в конструкторе класса Box.

Создание OpenGL контекста и обработка ввода осуществляется с помощью библиотеки GLFW[8].

Также в конструкторе класса Box создаются объекты типа btRigidBody, используемые физическим движком для расчетов, и их добавление в симуляцию.

После расчета следующего шага симуляции, можно получить из объекта btRigidBody соответствующую его положению матрицу преобразований, и передать ее во вершинный шейдер (Vertex Shader)(Листинг 1)

Видовая и проекционная матрицы хранятся и вычисляются в классе Camera.

```
 void Box::draw()
{
   GLint uniform_location;

   setModel();

   glUseProgram(program);

   uniform_location = glGetUniformLocation(program, "ModelMatrix");

   if(uniform_location == -1)
   {
      fprintf(stderr, "Could not bind to uniform\n");
      return;
   }

   glUniformMatrix4fv(uniform_location, 1, GL_FALSE, glm::value_ptr(Model));

   glBindVertexArray(vao);
   glDrawElements(GL_TRIANGLES, NumOfPrimitives, GL_UNSIGNED_INT, 0);
   glBindVertexArray(0);
}

void Box::setModel()
{
   btTransform trans;
   btScalar    m[16];
   body -> getMotionState() -> getWorldTransform(trans);

   trans.getOpenGLMatrix(m);

   Model = glm::make_mat4((float *)m);
}
```

Листинг 1. Передача Model Matrix в вершинный шейдер.

Результат

В результате была разработана программа симулирующая поведение твердых тел управление камерой осуществляется с помощью мыши и кнопок 'w','a','s','d'. Кнопка 'f' запускает симуляцию, а кнопка 'r' переводит все объекты в начальное состояние. На рисунках 2 приведены снимки экрана во время работы программы.

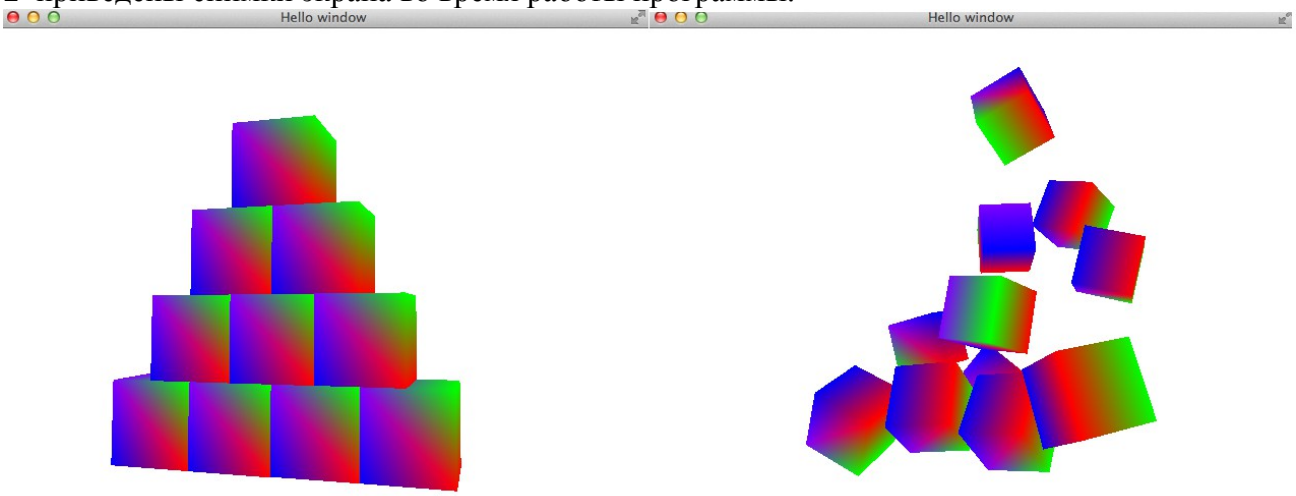

Рисунок 2: Снимок экрана во время работы программы

Вывод

Была разработана программа для симуляции взаимодействия твердых тел, изучены основы и базовые принципы работы физических движков, в частности Bullet, основы работы с OpenGL.

Список литературы

1: The OpenGL Graphics System: A Specification (Version 4.1 (Core Profile)) -2010 - c.518
2: Real-Time Physics Simulation http://bulletphysics.org/wordpress/
3: DirectX 11 - Microsoft Windows http://windows.microsoft.com/ru-ru/windows7/products/features/directx-11
4: Newton Game Dynamics http://newtondynamics.com/forum/newton.php
5: Tokamak Open Source Physics Engine http://www.tokamakphysics.com/
6: Davor Jovanoski, The Gilbert-Johnson-Keerthi(GJK) Algorigthm, 2008
7: John Kessenich, Dave Baldwin, Randi Rost The OpenGL Shading Language -2010 - c.161
8: GLFW - An OpenGL library http://www.glfw.org/