Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Компьютерная графика»

Студент: Т. А. Бучкин
Преподаватель: Г. С. Филиппов
Группа: М8О-301Б

Дата: 5 ноября 2024 г.
Оценка:
Подпись:

Москва, 2024

Курсовая работа

Тема: Основы 2D-графики и трансформаций

Задание: В данной лабораторной работе вам предстоит научиться работать с графическим API для отрисовки 2D-примитивов, освоить основные 2D-трансформации (перемещение, масштабирование, поворот) и изучить алгоритмы построения 2D-кривых.

Требования: Вы должны использовать C++ (OpenGL + SFML) или C# (OpenTK).

Программа должна работать в реальном времени, обновляя изображение в цикле.

Визуальный результат необходимо продемонстрировать на экране с возможностью управления через интерфейс.

Вариант №11. Анимация по траектории кривой Безье: Реализуйте анимацию движения объекта (например, круга) по траектории кривой Безье.

Анимация должна быть плавной, объект должен перемещаться по кривой с равномерной скоростью.

Добавьте управление скоростью анимации через интерфейс.

Дополнительно: Реализуйте изменение размера объекта в зависимости от его положения на кривой.

1 Решение

Кривую Безье зададим вектором опорных точек. (Все координаты будем задавать в нормализованном виде, т.е. на отрезке [-1, 1], это значительно упростит вершинный шейдер). Далее в классе кривой Безье опишем важный метод получения точки на данной кривой во время $t \in [0, 1]$. С её помощью мы будем отрисовывать кривую и расчитывать положение объекта на кривой.

Построение кривой Безье: В конструктор кривой Безье прокидывется в качестве параметра степень детализации, т.е. количество рассчитываемых точек для отрисовки. Для хранения данных о координатах заведем буфер вершин(vbo), чтобы хранить координаты прямо в памяти видеокарты. Далее указываем, что у нас там хранятся именно координаты (переменная вершиного шейдера с индексом 0) через функцию glVertexAttribPointer. Для хранения состояния vbo создадим объект вершинного массива(vao). Собственно, конструкция кривой Безье происходит следующим образом: расчитываем n равномерно расположенных на кривой точек, где n – степень детализации. Далее создается и привязывается vao, vbo. На vbo кладутся рассчитанные координаты, вызывается glVertexAttribPointer. После этого у нас готовы данные для отрисовки кривой Безье.

Построение объекта на кривой Безье: В качестве объекта я выбрал прямоугольник, т.к. у него простейшая геометрия(всего 2 треугольника). Ситуация аналогичная

построению кривой Безье: рассчет координат точек, vao, vbo, glVertexAttribPointer. Единственное отличие в том, что объект будет перемещаться, т.е. на каждом кадре у него надо будет пересчитывать координаты и соответственно обновлять оные на GPU. В общем для этого после каждого расчета нужно вызывать glBufferData, для vbo прямоугольника.

Отрисовка: Для отрисовки достаточно прикрепить соответствующий vao и шейдерную программу и вызвать glDrawArrays, которая принимает в качестве аргументов тип отрисовываемого примитива, с какого начинать и количество отрисовываемых вершин. Для кривой Безье в качестве типа указываем GL_LINE_STRIP(Ломанная линия), для квадрата – GL_TRIANGLE_FAN(Треугольники с общей вершиной, позволяет задать квадрат всего 4 точками, вместо 6 для обычного GL_TRIANGLES).

GUI: Для управления скоростью перемещения я буду использовать систему событий(она без особых наворотов описана в файле events.hpp). Для кнопок будем использовать цветные прямоугольники подписанные на событие нажатия на экран. Каждая кнопка будет иметь собственный обработчик, один для ускорения, другой для замедления.

Шейдерные программы: На момент описания первой лабы я еще плохо разбирался в шейдерах, поэтому использовал самый примитивный вариант. Для вершинного шейдера – проброс входной координаты в фрагментный шейдер без преобразования, во фрагментном шейдере просто задается фрагменту один цвет. Сообственно для красного, зеленого и синего цветов я использовал отдельные шейдеры.

Изменение размера объекта: Мне показалась прикольной идей задавать ширину и высоту прямоугольника через тригонометрические функции, что я и сделал. Получился интересный эффект скручивания прямоугольника.

Демонстрация: В общем можно все скомпилировать и запустить, все построено на базе glfw и glew. Но также прилагаю видео демонстрацию, чтобы вам особо с этим не заморачиваться(В видео почему-то кривая Безье отображается как азбука Морзе, возможно проблемы с записью. Отрисовка в режиме GL_LINE_STRIP дает спошную линию).

2 Вывод

В своей первой лабе я начал понемногу вникать в pipeline работы с opengl, из-за недостаточного опыта, у меня получилась нерасширяемая архитектура приложения, поэтому для следующих лаб пришлось все переписывать заного с нуля, но для того чтобы сохранилась возможность скомпилировать и запустить первую лабу я переместил её исходники в include_old для заголовочных файлов, src_old для исходников.