Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра алгоритмических языков

Отчёт о выполнении задания практикума

Модель оптических экспериментов в зеркальной комнате

Выполнил:
Студент 425 группы
Ивачев Федор Владиславович

Москва 2019 г.

Оглавление

Краткая постановка задачи

Зеркальная комната представляет в плане произвольный замкнутый M-угольник (4 <= M), каждая стена - плоское зеркало. Для проведения экспериментов необходимо определить для каждой стены комнаты вид зеркала (плоское или сферическое), а для каждого сферического зеркала - его тип (вогнутое или выпуклое) и радиус кривизны. Основная функция программной системы - проведение оптического эксперимента, при котором из некоторой точки на одной из стен комнаты, под определенным углом к этой стене (угол может варьироваться от 0 до 180 градусов) выпускается луч света, и затем показывается его путь внутри комнаты с учетом отражений от зеркал. Траектория луча определяется физическими законами отражения от зеркальных поверхностей. Цель моделирования - подбор пользователем системы параметров зеркал и исходного угла выпущенного луча, при которых луч, отражаясь от зеркальных стен, попадает в нужную точку (зону) комнаты. При визуализации оптического эксперимента должен быть показан план комнаты и изображен путь луча в комнате.
Пользователь системы должен иметь возможность:
• определять число M стен комнаты и рисовать ее план (например, указывая мышью на экране компьютера угловые точки комнаты);
• задавать и изменять параметры зеркал (вид, тип, радиус кривизны), точку выпускания луча и его исходный угол;
• запоминать в файле копию оптического эксперимента, сохраняя все его параметры, и считывать сохраненную копию из файла в рабочее окно.
Требуется, чтобы указанные действия пользователь мог производить в произвольном, удобном для него порядке, и изменение одного параметра эксперимента не должно затрагивать другие установленные параметры.

Диаграмма классов

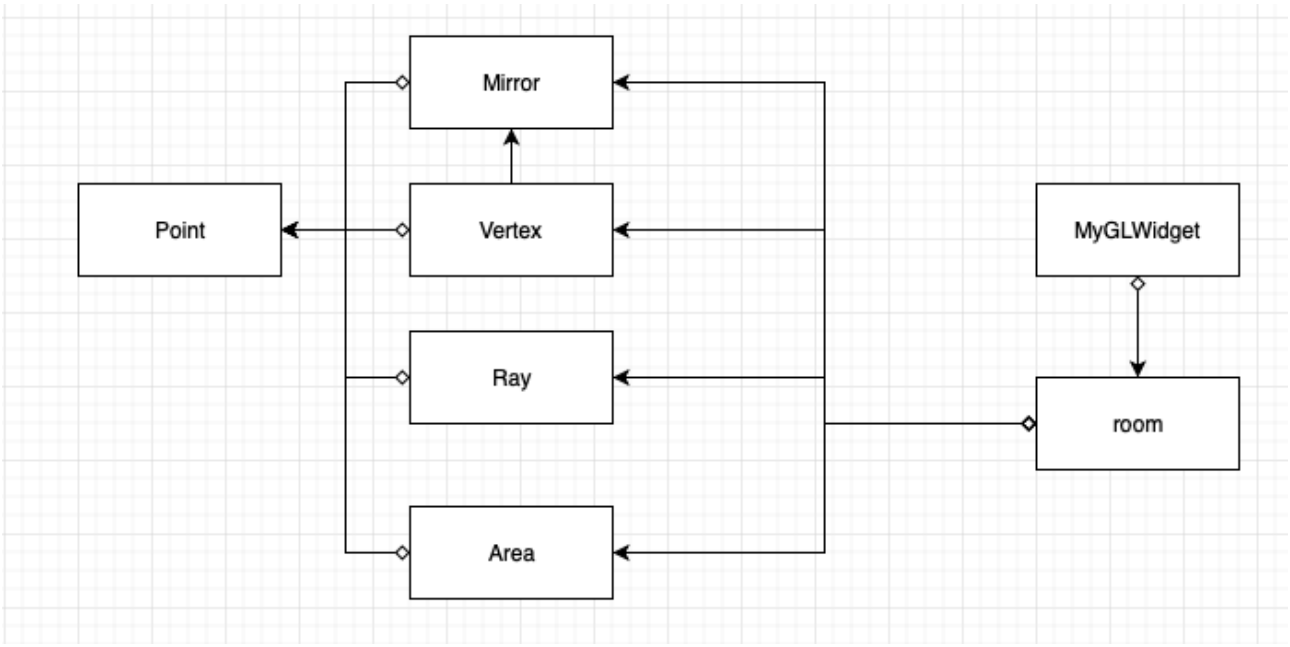
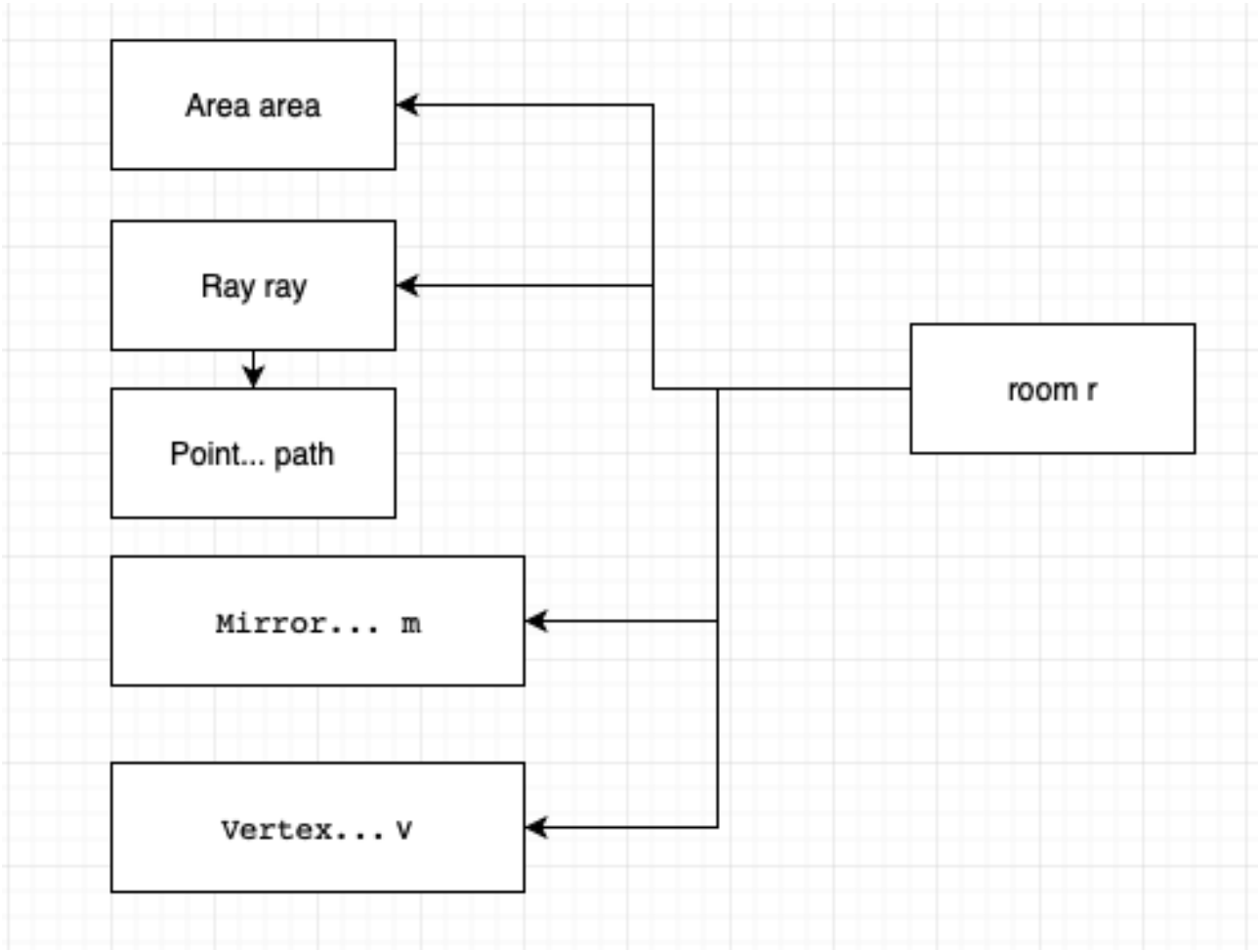


Диаграмма объектов



Текстовые спецификации интерфейса

```
class Point
{
public:
    Point(float x1, float y1);
    Point();
    float x;
    float y;
};
```

```
class Vertex
{
public:
    Vertex(Point cen, int col, bool is_m);
    Vertex();
    Point center; // координаты центра
    int color = 0; // цвет вершины
    bool is_moved = false; // в процессе перемещения ли
};

class Ray
{
public:
    Point Start; // координаты начала луча
    Point Second; // координаты второй точки луча (задает направление)
    QVector <Point> path = {}; // координаты пути луча
    int color = 0; // цвет
};
class Mirror
{
public:
    Vertex left; // первый край зеркала
    Vertex right; // второй край зеркала
    Mirror();
    Mirror(Vertex l, Vertex r);
    double r = 0; // радиус кривизны зеркала
};
```

```
class Area
{
public:
    Point Center; // координаты центра
    int color = 0; // цвет
    int ach = 0; // достигнута ли область лучом
    float r = 0.2; // радиус области
};

class room
{
public:
    room();
    void loadCoords(); // загрузка исходного сохраненного состояния
    void saveCoords(); // сохранение состояния
    void updateMirrorsCoord(); // обновление координат зеркал
    void update(); // обновление состояния комнаты
    void get_path(int strength); // построение пути луча
    QVector <Vertex> v = {}; // массив вершин зеркал
    QVector <Mirror> m = {}; // массив зеркал
};
```

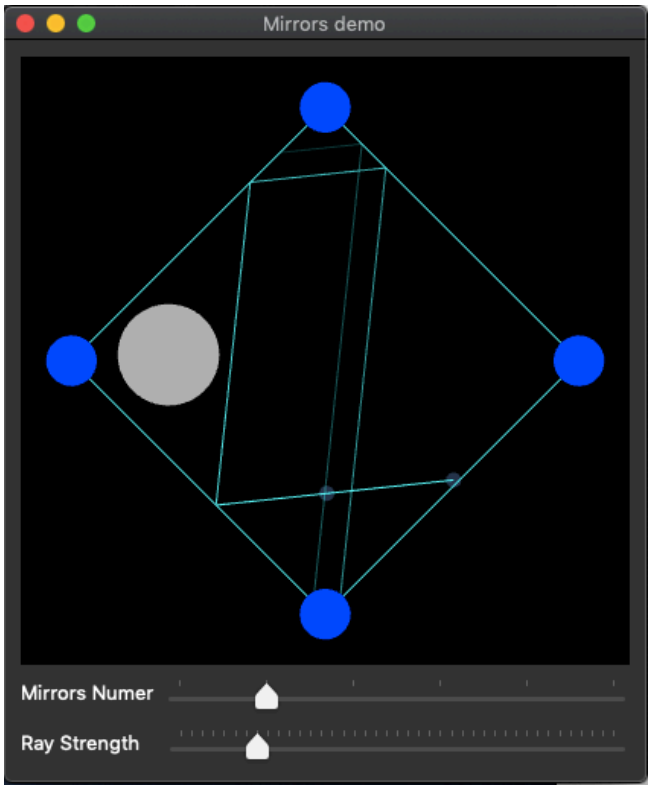
Инструментальные средства:

- Язык разработки: C++11
- Среда разработки: Qt Creator
- Библиотеки: OpenGL, Qt 5.14 ****

Файловая структура:

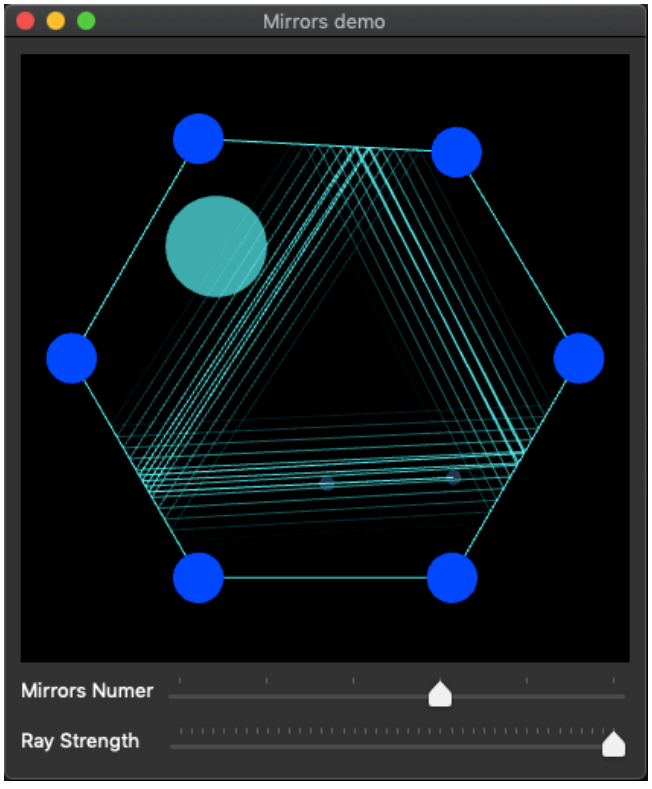
- main.cpp - основной файл
- window.ui - ui-файл окна программы
- window.cpp - вспомогательный файл отрисовки ui
- MyGLWidget.h / .cpp - отрисовка OpenGL
- room.h / .cpp - построение модели комнаты
- point.h / cpp - описание и классов внутри комнаты
- settings.txt - файл-автосохранение состояния комнаты

Пользовательский интерфейс:



Интерфейс интуитивно понятен: Слайдерами внизу окна можно выбрать количество вершин в многоугольнике, силу луча (то же, что и количество отражений).

Вершины многоугольника перемещаются мышкой, так же, как и начало луча (две точки, задающие его направление), и финальная область. При попадании луча в область, она подсвечивается.



Также в программе реализовано автосохранение. Если пользователь завершит программу, то при новом ее запуске комната будет аналогична той, что и при ее завершении. Экспортировать координаты вершин, луча и финальной области можно, просто скопировав файл settings.txt.