**[СЛАЙД 1] – Приветствие**

Здравствуйте, уважаемые члены комиссии. Меня зовут Сальников Михаил Алексеевич, я – студент группы ИУ7-54Б, представляю Вашему вниманию выполненную мной курсовую работу на тему “Конструктор композиции многогранных примитивов”.

**[CЛАЙД 2] – Цели и задачи**

Целью курсовой работы является разработка программы композиции и визуализации трехмерных многогранных примитивов с учётом их геометрических и оптических параметров, задаваемых пользователем. Для выполнения работы были поставлены следующие задачи:

* описание объектов сцены;
* анализ известных алгоритмов для генерации трехмерной сцены;
* выбор наиболее подходящие алгоритмы для достижения цели;
* проектирование архитектуры и графического интерфейса приложения;
* выбор средств реализации программы;
* реализация выбранных алгоритмов и структур данных;
* проведение исследований быстродействия разработанного приложения.

**[СЛАЙД 3] – Этап анализа**

На этапе анализа мной были выполнены следующие действия:

1. формализован набор объектов сцены. В него входят *примитивы, источник света, камера*;
2. проанализированы известные методы формализации модели. Были рассмотрены *каркасная модель*, *поверхностная модель*, *твердотельная модель.* Выбор был сделан в пользу поверхностной модели, заданной полигональной сеткой   
     
   (***Почему?*** *Потому что каркасная модель не предоставляет достаточной информации о модели, а твердотельная нуждается в информации о материале модели, который в данной работе отсутствует*);
3. проанализированы алгоритмы удаления невидимых линий и поверхностей. Были рассмотрены алгоритмы *Робертса, Варнока, прямой и обратной трассировки лучей, художника и Z-буфера.* Выбор был сделан в пользу алгоритма Z-буфера   
   (***Почему?*** *Потому что данный алгоритм прост в своей реализации, производителен и позволяет добиться достаточной детализации синтезируемого изображения)*;
4. проанализированы модели освещения. Были рассмотрены модели *Ламберта*, *Фонга*, *Блинна-Фонга.* Был сделан выбор в пользу модели Фонга   
   (***Почему?*** *Потому что данная модель учитывает как диффузное, так и зеркальное отражение света, что делает синтезируемое изображение более реалистичным и способствует возникновению бликов света*);
5. проанализированы алгоритмы закраски. Были рассмотрены алгоритмы *простой закраски, закраски Гуро и закраски Фонга.* Выбор был сделан в пользу алгоритма закраски Фонга.  
   (***Почему?*** *Потому что данный алгоритм несмотря на его вычислительные затраты позволит обеспечить реалистичное освещение и четкие блики*);
6. выбран модифицированный алгоритм Z-буфера для построения теней.

**[СЛАЙД 4] – Визуализация сцены. Режимы визуализации**

Одним из требований к программе является многорежимность отображения сцены. Было спроектировано 4 режима отображения сцены: “каркасный”, “реалистичный”, “световой” и “теневой”. “Каркасный” режим отображает модели на сцене в каркасном формате, “реалистичный” режим позволяет увидеть линии пересечения моделей, “световой” режим добавляет освещение на сцену, “теневой” режим включает построение теней. На данном слайде Вы можете наблюдать схему общего алгоритма визуализации сцены, а также более подробную схему алгоритма визуализации сцены в “теневом” режиме. Подробные схемы алгоритмов для остальных режимов отражены в РПЗ.

**[СЛАЙД 5] – Получение изображения от лица камеры**

Одним из требований к программе является наличие камеры на сцене и изменение её положения. Для этого был подробно изучен и описан метод получения изображения от лица камеры, а также произведена её формализация. В описании этого метода были раскрыты такие понятия как пирамида видимости, пространства модели, мира, камеры, отсечения и экрана. Сам метод заключается в том, чтобы последовательно перевести вершину из пространства модели через все пространства на экран. После описания метода на его основе был спроектирован соответствующий алгоритм, схему которого Вы можете наблюдать на слайде. Также путь перевода вершины из пространства модели на экран также отображен на слайде. Более подробные математические выкладки изложены в РПЗ.

**[СЛАЙД 6] – Алгоритм Z-буфера**

На этапе анализа для удаления невидимых линий и поверхностей был выбран алгоритм Z-буфера. Этот алгоритм был спроектирован как надстройка над алгоритмом получения изображения от лица камеры, чтобы после реализации самой камеры была возможность внедрить в нее данный алгоритм. Такое решение было принято для обеспечения многорежимности отображения сцены камерой, потому что алгоритм Z-буфера также позволяет внедрить в него расчет освещенности обрабатываемых объектов, а еще при таком решении реализация модифицированного версии алгоритма с учетом теней становится тривиальной. Схему алгоритма Z-буфера Вы можете наблюдать на слайде.

Одним из важных этапов алгоритма Z-буфера при обработке отдельного полигона модели является определение множества его точек. Схему алгоритма определения этого множества Вы можете также видеть на слайде.

**[СЛАЙД 7] – Диаграмма классов**

Была создана UML-диаграмма предложенных классов программы. В ней отражены основные классы и связи между ними. В нижней части диаграммы отражены низкоуровневые классы такие как камера, источник света и модели. В середине можно наблюдать классы сцена, система управления источниками света и система управления камерами. В верхней части диаграммы изображен высокоуровневый класс отображения изображения на экран и классы, отвечающие за организацию единого интерфейса управления.

(***Зачем нужен единый интерфейс управления?*** *Могу привести аналогию с управлением механическим транспортным средством. В нем у нас есть руль, педаль сцепления, тормоза и газа. Они выступают в роли единого интерфейса управления. Если бы их не было, то для того, чтобы выполнять простые действия такие как старт двигателя машины приходилось бы производить много действий*)

**[СЛАЙД 8] – Структура программы**

Для реализации программы был выбран язык программирования C#, потому что его стандартная библиотека поддерживает необходимые классы, определенные на этапе проектирования, а также потому что этот язык поддерживает объектно-ориентированное программирование и механизм многопоточности.

На слайде вы можете наблюдать реализованные классы представления камер. Класс Camera является базовым классом представления камеры, классы ViewingFrustum и последующие классы с аналогичными названиями являются продвинутыми реализациями камеры с внедренными в нее модификациями для улучшения получаемого изображения и ускорения его получения. Класс ViewingFrustumProcessor представляет из себя мультирежимную камеру, позволяющую получать изображение в выбранном режиме. Более подробное описание классов приведено в РПЗ.

**[СЛАЙД 9] – Интерфейс программы**

Интерфейс реализованной программы и пример её работы Вы можете наблюдать на слайде.

При запуске программы пользователю демонстрируется пустая сцена и панель, содержащая вкладки «Главная» и «Вид». Вкладка «Главная» позволяет пользователю добавлять примитивы на сцену, редактировать добавленные примитивы, очищать сцену и изменять режим отображения сцены. Вкладка «Вид» позволяет пользователю перемещать камеру, изменять спектральные характеристики и положение источника света.

При нажатии на кнопку «Редактор примитивов», расположенной в группе «Действия» вкладки «Главная» будет открыто окно редактирования примитивов, в котором можно изменять их геометрические, спектральные, цветовые и информационные параметры. Его Вы также можете видеть в правой части слайда.

**[СЛАЙД 10] – Исследования**

Были проведены исследования быстродействия разработанной программы. В результате исследований были получены следующие зависимости:

* времени визуализации сцены от количества полигонов;
* времени визуализации сцены от количества генерируемых примитивов;
* времени визуализации сцены от геометрических параметров примитива.

Результаты первых двух исследований отображены на слайде. По построенным графикам видно, что данные зависимости имеют линейный характер. Результаты третьего исследования и методы проведения всех исследований приведены в РПЗ.

**[СЛАЙД 11] – Заключение**

В ходе выполнения курсовой работы было разработана программа визуализации композиции трехмерных многогранных примитивов в соответствии с задачами, поставленными техническим заданием.  
В процессе разработки были решены все обозначенные задачи:

1. описаны объекты сцены;
2. проанализированы известные алгоритмы для генерации для трехмерной сцены;
3. выбраны наиболее подходящие алгоритмы для достижения цели;
4. спроектированы архитектура и графический интерфейс приложения;
5. выбраны средства реализации программы;
6. реализованы выбранные алгоритмы и структуры данных;
7. проведено исследование быстродействия разработанного приложения.

**[СЛАЙД 12] – Спасибо за внимание**

**СПАСИБООО**