|  | **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»\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_Клименко Алексей Константинович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа\_\_\_\_\_\_ИУ7-45Б\_\_\_\_\_\_

Тип практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_учебная\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Название предприятия\_\_\_\_\_\_\_\_ МГТУ им. Н. Э. Баумана\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Клименко А. К.\_\_\_\_\_**

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Майков К. А.\_\_\_\_\_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2021 г.*

**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)**

УТВЕРЖДАЮ

Заведующий кафедрой \_ИУ7\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_И. В. Рудаков\_\_

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Компьютерная графика\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_\_\_ИУ7-45Б\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Клименко Алексей Константинович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Тема курсовой работы \_\_\_Программа построения реалистичного изображения сцены,

состоящей из трёхмерных объектов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направленность КР (учебная, исследовательская, практическая, производственная, др.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Учебная\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_Кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения работы: 25% к \_3\_ нед., 50% к \_6\_ нед., 75% к \_9\_ нед., 100% к \_14\_ нед.

***Задание*** \_\_Разработать программу для построения реалистичного изображения сцены, состоящей из набора трёхмерных объектов и источников света. Предусмотреть возможность добавления на сцену объектов из перечня: сфера, цилиндр, конус, тор. Положение, размер и поворот объектов в пространстве, а также спектральные характеристики поверхности объектов задаются пользователем. Пользователем на сцену могут быть добавлены источники точечного и/или направленного освещения. Положение и спектральные характеристики каждого из источников света задаются пользователем. Предусмотреть возможность создания нескольких камер и переключения между ними. Добавить функцию сохранения полученного изображения в файл.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление курсовой работы:***

Расчетно-пояснительная записка на 25-35 листах формата А4.

Расчётно-пояснительная записка должна содержать постановку задачи, введение, аналитический раздел, конструкторский раздел, технологический раздел, экспериментально-исследовательский раздел, заключение, список литературы, приложения.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.): на защиту проекта должна быть представлена презентация, состоящая из 15-20 слайдов. На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, диаграмма классов, интерфейс, характеристики разработанного ПО, результаты проведенных исследований.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « 15 » \_июня\_ 2021 г.

**Руководитель курсовой работы**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_К. А. Майков\_\_\_\_\_\_

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_** А. К. Клименко **\_\_\_**

(Подпись, дата) (И.О.Фамилия)

# Содержание

[**Содержание**](#_gjdgxs) **2**

[**ВВЕДЕНИЕ**](#_1fob9te) **3**

[**1. Аналитический раздел**](#_3znysh7) **4**

[1.1. Описание моделей трехмерных объектов в сцене](#_2et92p0) 4

[1.2. Алгоритмы построения трехмерного изображения](#_pxmrfsl1639p) 7

[1.2.1. Алгоритм художника](#_ml1du2o0f19t) 7

[1.2.2. Алгоритм Варнока](#_kty8g444p5b1) 7

[1.2.3. Алгоритм Вейлера-Азертона](#_s0piekkvwshk) 7

[1.2.4. Алгоритм с использованием Z-буфера](#_imphnee5iuoz) 7

[1.2.5. Алгоритм трассировки лучей](#_q3otks2utljw) 8

[1.2.6. Вывод](#_u9qz6duguux3) 8

[1.3. Методы закраски](#_pk0d2ei1mxks) 8

[1.3.1. Простая закраска](#_gg050xatbh5i) 8

[1.3.2. Закраска по Гуро](#_tjqfexc36h2j) 8

[1.3.3. Закраска по Фонгу](#_5x3lhb4y43yn) 8

[**2. Конструкторский раздел**](#_2dxdvyialjkl) **9**

[2.1. Требования к программе](#_2s8eyo1) 9

[**3. Технологический раздел**](#_3rdcrjn) **10**

[3.1. Выбор языка программирования и среды разработки](#_26in1rg) 10

[**4. Экспериментальный раздел**](#_35nkun2) **11**

[4.1. Технические характеристики компьютера](#_44sinio) 11

[4.5. Выводы по исследовательской части](#_2jxsxqh) 11

[**ЗАКЛЮЧЕНИЕ**](#_z337ya) **12**

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**](#_3j2qqm3) **13**

[Приложение А. Пользовательский интерфейс](#_1y810tw) **17**

# 

# ВВЕДЕНИЕ

В современном мире всё более и более востребованным становится использование компьютерной графики в разных сферах деятельности. Она используется в компьютерном моделировании, компьютерных играх, а также в системах автоматизированного проектирования. В связи с нарастающей потребностью к реалистичному изображению трехмерных объектов, появляется необходимость создавать программное обеспечение, способное учитывать широкий спектр оптических явлений при отрисовке изображения.

В компьютерной графике алгоритмы создания реалистичного изображения требуют особых затрат ресурсов, тратится много времени и памяти для получения результата. Затраты времени на синтез изображения не всегда позволяют удобно и эффективно подобрать параметры отрисовки.

Цель данной работы – создать программное обеспечение, которое позволило бы синтезировать изображение трехмерной сцены с настраиваемыми параметрами ее объектов в реальном времени и сменой режимов быстродействия.

Для достижения поставленной цели, необходимо решить следующие задачи:

1. описать трехмерную сцену и объекты, из которых она состоит;
2. анализ и видоизменение существующих алгоритмов отрисовки, визуализирующих трехмерную сцену;
3. реализовать выбранные алгоритмы;
4. разработать программное обеспечение, позволяющее в реальном времени менять параметры сцены, а также параметры отрисовки.

# 1. Аналитический раздел

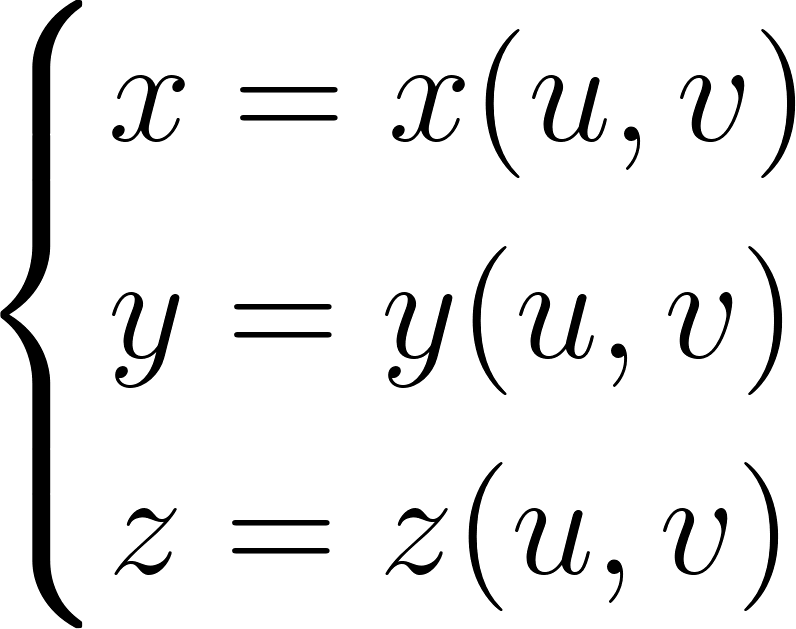
В данном разделе проводится обзор существующих способов описания трехмерных моделей, алгоритмов построения трехмерного изображения. Осуществляется анализ существующих моделей освещения и алгоритмов закраски полигонов.

## 1.1. Описание моделей трехмерных объектов в сцене

Существует множество способов представления трехмерных объектов в сцене. Различным алгоритмам отрисовки требуется разная информация о самом объекте.

Одним из простейших способов задания трехмерных объектов можно назвать параметрическое задание. Возможны несколько вариантов

1. Поверхность такого объекта задается системой уравнений:

[](https://www.codecogs.com/eqnedit.php?latex=%20%5Cbegin%20%7Bcases%7D%20x%20%3D%20x(u%2C%20v)%20%5C%5C%20y%20%3D%20y(u%2C%20v)%20%5C%5C%20z%20%3D%20z(u%2C%20v)%20%5Cend%20%7Bcases%7D%20#0)

каждая точка на поверхности объекта преобразуется в мировое пространство.

2. Внутренний объем такого объекта выражается неравенством (или системой неравенств):

f(x, y, z) > 0

для всех точек, расположенных внутри объекта данная функция (называемая пробной) должна удовлетворять неравенству.

Особенности данного представления:

* Параметрический (или его еще можно назвать/называют аналитическим) способ представления позволяет получить максимальный уровень детализации при отрисовке, наилучшую сглаженность формы объекта.
* Достаточно сложная форма может иметь довольно простое параметрическое представление, из чего следует меньший расход памяти при хранении представления объекта.
* Трансформации над объектом производятся очень просто. Достаточно применить преобразования к описывающим объект уравнениям или неравенствам.
* Наложение текстур в случае с внутренне-определёнными объектами является затруднительным. (А в случае с гранично-определёнными - нет)
* Процесс отыскания параметрического представления невыпуклых многогранников может быть затруднительным.

Другим представлением трехмерных объектов в сцене является представление в виде полигональной сетки.

Полигональная сетка является аппроксимацией поверхности представляемого объекта. От её плотности зависит качество изображения объекта.

Так как полигоны в сети могут представляться вне зависимости друг от друга, данный способ позволяет с меньшей сложностью описать сложные объекты в сцене.

Особенности данного представления:

* Просто создать топологию поверхности.
* Преобразования над объектом сводятся к преобразованию полигональной сетки, что в свою очередь раскладывается на независимые преобразования над отдельными полигонами.
* Наложение текстур на поверхность является несложной процедурой.
* Повышение плотности сетки приводит к увеличению объема затрачиваемой памяти для ее хранения.

Наконец, есть еще один способ представления трёхмерных объектов - представление воксельной сеткой.

Воксельная сетка позволяет непосредственно представить объем объекта, его внутреннее содержание.

Особенности данного представления:

* Просто представляются объекты сцены.
* Легко выполняются отсечения частей объекта.
* Текстурирование тесно связано с воксельной сеткой. Нельзя изменить текстуру объекта без изменения воксельной сетки.
* Поворот и масштабирование объекта не могут быть осуществлены без перерасчета сетки.
* Чем больше объем объекта, тем больше затрачивается памяти на хранение его представления.

## 1.2. Алгоритмы построения трехмерного изображения

Проведем сравнительный анализ алгоритмов построения трехмерного изображения для выбора наиболее подходящего. Целевыми показателями, на которые следует обратить внимание являются: зависимость скорости работы от числа изображаемых объектов на сцене, требуемый объем памяти, а также возможность распараллеливания процесса работы алгоритма.

### 1.2.1. Алгоритм художника

Принцип данного алгоритма заключается в том, что объекты на экране рисуются в порядке от дальних к ближним, при этом происходит наложение - ближние объекты рисуются поверх дальних. Этот алгоритм удобен в случаях, когда процесс распознавания видимых частей объекта является более затратным, чем собственно его полная отрисовка. Но при этом этот алгоритм производит лишнюю работу по отрисовке невидимых частей объектов.

### 1.2.2. **Алгоритм Варнока**

Алгоритм Варнока работает по принципу “Разделяй и властвуй” и разбивает окно на подокна в случаях, когда его отрисовка нетривиальна. Такие разбиения могут продолжаться до тех пор, пока размер окна не станет равным единице растеризации или пока в окне не останется всего одной простой для отрисовки фигуры.

### 1.2.3. Алгоритм Вейлера-Азертона

Алгоритм Вейлера-Азертона позволяет выполнить отсечение невидимой части объекта перед его отрисовкой по контурам загораживающих его объектов. Этот прием позволяет не делать лишнюю работу при отрисовке частично видимых объектов.

### 1.2.4. Алгоритм с использованием Z-буфера

Алгоритм с использованием Z-буфера базируется на том, что вместо того, чтобы определять очередность в которой должны отрисовываться объекты, для каждого пикселя при отрисовке сохраняется дополнительный атрибут - глубина, использование которого гарантирует корректное отображение пикселя, соответствующего наиболее близкому объекту.

### 1.2.5. Алгоритм трассировки лучей

Алгоритм трассировки лучей базируется на физическом представлении света и законах геометрической оптики. Он позволяет с высокой степенью точности моделировать различные оптические эффекты.

### 1.2.6. Вывод

Для программной реализации отображения сцены на экран было решено выбрать алгоритм трассировки лучей как основной для создания реалистического изображения, а при временной отрисовке использовать алгоритм Z-буфера. Сравнение алгоритмов растеризации приведено в таблице 1.1.

Таблица 1.1. Сравнение алгоритмов растеризации

| **Алгоритм** | **Художника** | **Варнока** | **Вейлера-Азертона** | **Z-буфера** | **Трассировки лучей** |
| --- | --- | --- | --- | --- | --- |
| Необходимость предварительной сортировки объектов | есть | есть | есть | нет | нет |
| Объем необх. доп. памяти[[1]](#footnote-0) | O(1) | O(1) | O(N) | O(p) | O(1) |
| Возможность параллельного выполнения | нет | есть | нет | нет | есть |

## 1.3. Методы закраски

При использовании алгоритмов, основанных на закрашивании полигонов, следует подобрать модель освещения в сцене, т.к. это напрямую влияет на изменение цвета в пределах одного полигона.

### 1.3.1. Простая закраска

При реализации простой закраски для каждой отдельно взятой грани объекта используется информация об источниках света для расчета цвета, в который впоследствии закрашивается эта грань. Простая закраска хорошо подходит для граненых объектов, которым не требуется сглаживание.

### 1.3.2. Закраска по Гуро

Закраска по Гуро учитывает кривизну поверхности и использует бикубическую интерполяцию для расчета цвета очередного пикселя. Интерполируемым значением является цвет граничных пикселей. Данный вид закраски позволяет с достаточной степенью точности изображать гладкие кривые поверхности.

### 1.3.3. Закраска по Фонгу

Закраска по Фонгу также учитывает искривленные поверхности, но за счёт интерполяции не цвета пикселя, а вектора нормали, для которого и происходит перерасчет значения интенсивности цвета пикселя. Такой метод позволяет гораздо точнее учитывать блики на поверхности объектов, а также позволяет корректно обрабатывать случаи, когда источник света расположен вблизи освещаемого объекта.

# 

# 2. Конструкторский раздел

Данный раздел посвящен конструированию приложения, а также описанию алгоритмов и методов, которые будут применяться в разрабатываемой программе.

## 2.1. Требования к программе

Создаваемая программа должна предоставлять следующий набор функций:

* Загрузка заготовленных полигональных моделей из файлов
* Добавление в сцену объектов из заготовленного набора примитивов
* Независимое трансформирование положения объектов в сцене
* Навигация по сцене с помощью перемещения и поворота главной камеры
* Просмотр и изменение специфических свойств объектов в сцене
* Сохранение текущего изображения сцены в файл на диске
* Сохранение созданной сцены в файл на диске
* Загрузка готовой сцены из файла

## 2.2. Описание основных алгоритмов

### 2.2.1. Общий алгоритм визуализации трехмерной сцены

1. Инициализация z-буфера

2. Цикл по всем объектам в сцене

2.1. Определение видимости объекта: если невидим, то переход к следующему объекту в сцене

2.2. Цикл по всем полигонам объекта

2.2.1. Определение видимости полигона: если невидим, то переход к следующему полигону

2.2.2. Вычисление параметров освещенности полигона

2.2.3. Отображение полигона

### 2.2.2. Алгоритм трассировки лучей

Цикл по всем пикселям на экране

1. Вычислить параметры луча для текущего пикселя

2. Цикл по всем видимым объектам в сцене

2.1. Если луч пересекает объект, то переход к 2.2, иначе переход к 2.6

2.2. Вычислить точку пересечения луча с объектом

2.3. Если эта точка ближе ближайшей, то переход к 2.4, иначе переход к 2.6

2.4. Присвоить ближайшему объекту текущий объект

2.5. Присвоить ближайшей точке текущую точку пересечения

2.6. Конец цикла

3. Если ближайший объект найден, то переход к 4, иначе переход к 5

4. Вычислить цвет пикселя согласно глобальной модели освещения

5. Конец

### 2.2.3. Методы освещения

## 2.3. Выбор используемых типов и структур данных

Математические абстракции:

* Четырехкомпонентный вектор
* Матрица 4х4
* Луч. Состоит из двух векторов - начала и направления
* Прямоугольная двухмерная область

Примитивы растеризации:

* Пиксель (в формате ARGB32)
* Цель отрисовки. Содержит буфер из пикселей, в который происходит отрисовка, и размеры окна - ширина и высота.

Сцена и объекты сцены:

* Вершина полигона. Состоит из позиции и вектора нормали.
* Полигон. Состоит из трех вершин полигона.
* Полигональная модель. Состоит из массива полигонов.
* Источник света. Содержит тип (точечный/направленный), вектор положения/направления, интенсивность и цвет.
* Камера. Содержит матрицу проецирования на экранную плоскость.
* Сцена из объектов. Состоит из списка полигональных моделей и списка источников света.

# 3. Технологический раздел

Данный раздел посвящен выбору средств для программной реализации разработанного метода, описанию основных моментов разработки программного обеспечения.

## 3.1. Выбор языка программирования и среды разработки

Для написания программы в качестве языка программирования был выбран язык С++ по следующим причинам:

1. Имеет статическую типизацию, является компилируемым языком программирования;
2. Позволяет напрямую работать с памятью, а также контролировать все обращения к менеджеру памяти;
3. Имеет мощный механизм шаблонов;
4. Позволяет писать программы как в структурном стиле, так и в объектно-ориентированном;

В качестве среды разработки была выбрана “CLion”.

## 3.2. Схема доменов

Программа состоит из трех доменов: прикладного, архитектурного и интерфейсного. Прикладной домен отвечает за процесс визуализации сцены, интерфейсный - за взаимодействие с пользователем, а архитектурный служит связующим звеном между двумя вышеупомянутыми доменами.

Диаграмма 3.1. Схема взаимодействия доменов



## 3.3. Прикладной домен

Прикладной домен реализуется в структурном стиле, так как такой подход обеспечивает наибольшую скорость работы и простоту реализации, а также в дальнейшем не нуждается в частых модификациях.

Диаграмма 3.2. Иерархия функций прикладного домена

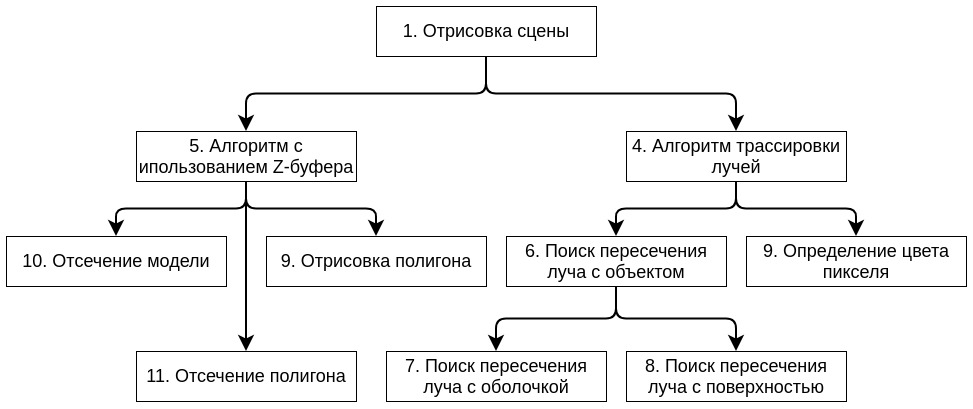
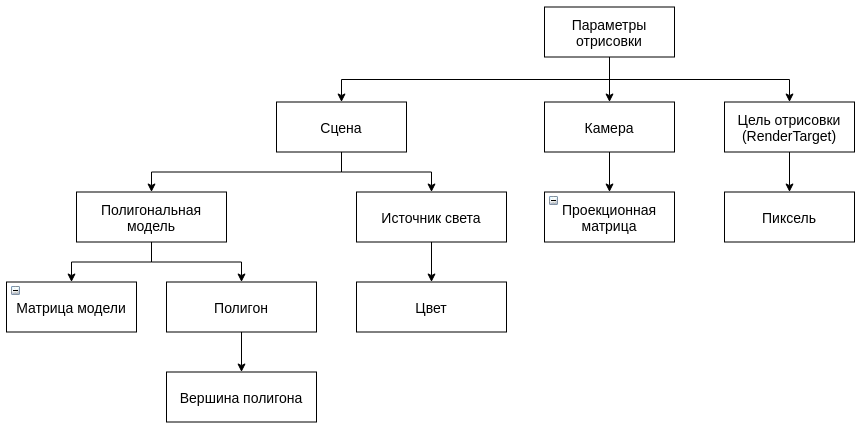


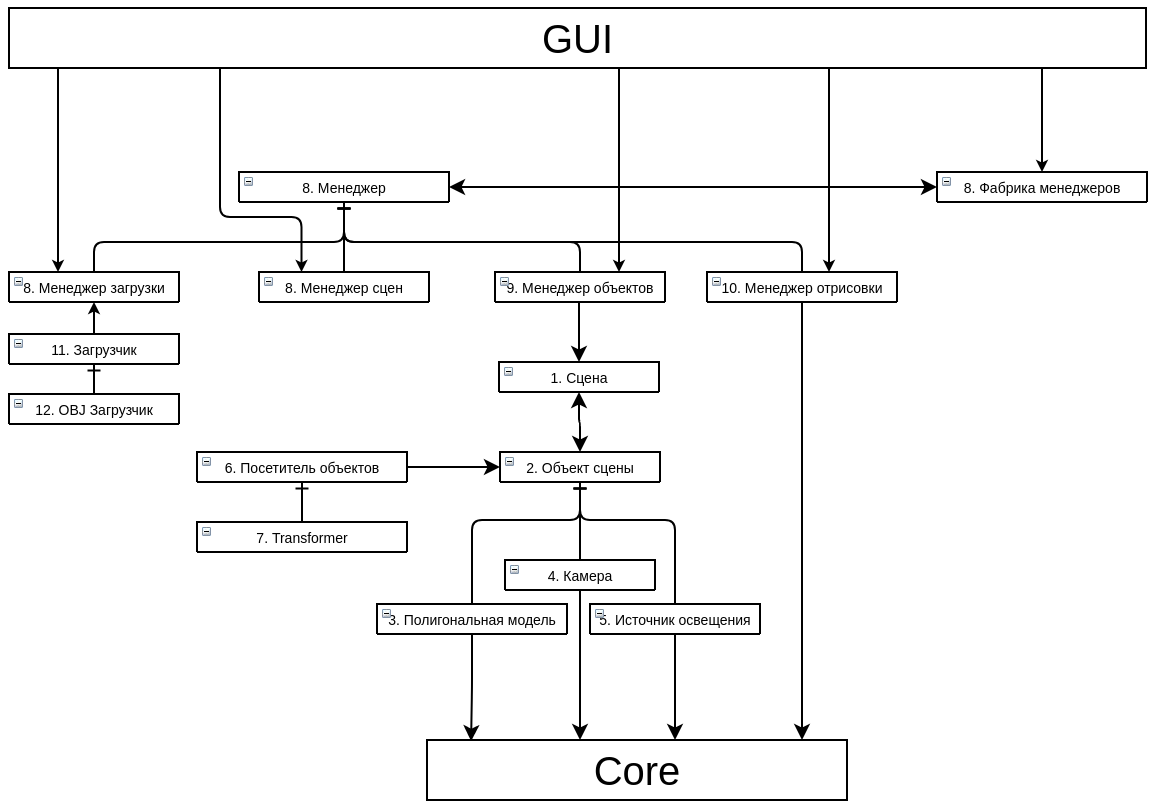
Диаграмма 3.3. Иерархия по данным



## 3.4. Архитектурный домен

Архитектурный домен реализуется согласно объектно-ориентированному подходу с использованием паттернов проектирования.

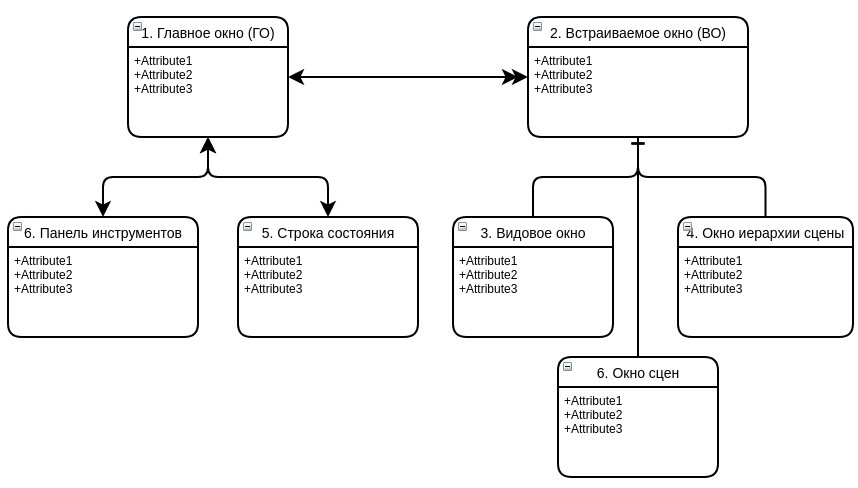
Диаграмма 3.4. Информационная модель архитектурного домена



## 3.5. Интерфейсный домен

Интерфейсный домен реализуется с использованием открытой библиотеки Qt.

Диаграмма 3.5. Информационная модель интерфейсного домена

****

# ЗАКЛЮЧЕНИЕ

В ходе работы был выполнен анализ существующих алгоритмов построения трехмерного изображения и алгоритмов, используемых в разрабатываемом программном обеспечении. Были выдвинуты более подробные требования к программному обеспечению, обоснован выбор инструментов разработки. В основной части были выбраны типы и структуры данных, выделены математические абстракции и описано представление объекта сцены. Была продумана архитектура приложение.

1. N - число объектов в сцене, p - число пикселей растра [↑](#footnote-ref-0)