



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»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
K KURSOVOЙ РАБОТЕ
НА ТЕМУ:

*«Программа для построения железнодорожной
инфраструктуры»*

Студент группы ИУ7-56Б

_____ (Подпись, дата)

Мамврийский И. С.

(И.О. Фамилия)

Руководитель курсовой работы

_____ (Подпись, дата)

Мальцева Д. Ю.

(И.О. Фамилия)

2023 г.

Содержание

ВВЕДЕНИЕ	3
1 Аналитическая часть	4
1.1 Описание объектов сцены	4
1.2 Выбор модели представления объектов	4
1.3 Выбор способа задания поверхностных моделей	5
1.4 Выбор способа хранения полигональных моделей	6
1.5 Алгоритмы удаления невидимых линий и поверхностей	6
1.5.1 Алгоритм Робертса	7
1.5.2 Алгоритм Варнок	8
1.5.3 Алгоритм обратной трассировки лучей	9
1.5.4 Алгоритм, использующий z-буфер	10
1.5.5 Выбор алгоритма удаления невидимых линий и поверхностей	11
1.6 Выбор алгоритма построения теней	11
2 Конструкторская часть	13
2.1 Требования к программному обеспечению	13
2.2 Алгоритм, использующий z-буфер	13
2.3 Алгоритм для построения теней, использующий z-буфер	15
2.4 Используемые типы и структуры данных для представления объектов	18
3 Технологическая часть	19
3.1 Выбор языка программирования	19
3.2 Выбор среды разработки	19
3.3 Интерфейс программного обеспечения	19
3.4 Пример работы программы	23
4 Исследовательская часть	24
4.1 Постановка цели исследования	24
4.2 Технические характеристики	24

4.3 Средство замера времени	24
4.4 Результаты замера времения	25
4.5 Вывод	27
ЗАКЛЮЧЕНИЕ	28
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	29
Приложение А	30

ВВЕДЕНИЕ

Компьютерная графика — совокупность методов и способов преобразования информации в графическое представление при помощи ЭВМ.

Ее применение охватывает широкий спектр областей, включая медицину, архитектуру, науку, где необходимо наглядное отображение различной информации.

Для создания реалистичных изображений необходимо учитывать оптические явления преломления, отражения и рассеивания света, а также текстуру и цвет. Чтобы создать еще более реалистичные изображения, требуется брать в расчет дифракцию, интерференцию, вторичные отражения света.

Существует множество алгоритмов компьютерной графики, которые помогают нам решать эти задачи. Однако, эти алгоритмы зачастую требуют значительных вычислительных ресурсов, памяти и времени. Это является основной проблемой при создании реалистичных изображений и динамических сцен.

Целью данной курсовой работы является разработка программного обеспечения для построения железнодорожной инфраструктуры.

Для достижения данной цели необходимо выполнить следующие задачи:

- описать список доступных к размещению на сцене моделей, формализовать эти модели;
- выбрать алгоритмы компьютерной графики для визуализации сцены и объектов на ней;
- выбрать язык программирования и среду разработки;
- реализовать выбранные алгоритмы визуализации;
- разработать программное обеспечение для визуализации и редактирования железнодорожной инфраструктуры

1 Аналитическая часть

В данной части будут рассмотрены объекты сцены, модели их представления, способы задания и способы хранения. Также будут описаны алгоритмы удаления невидимых линий и алгоритмы построения теней.

1.1 Описание объектов сцены

Площадка сцены — правильный параллелепипед с заданной сеткой, по которой расставляются модели. Объекты располагаются только на одной из сторон площадки. Границы задаются количеством ячеек (квадратов) по длине и ширине. Размер ячеек — константная величина, определяемая внутри программы. Пользователь может использовать камеру для движения по сцене.

Объекты сцены — модели, расположенные на площадке сцены, которые занимают определенное количество клеток сетки (ячеек). Каждая модель представляет собой набор граней, описываемых точками в пространстве, которые соединены ребрами. Все модели определены заранее, в программе не предусмотрена возможность добавления новых или изменения старых моделей. В программном обеспечении доступно изменение положения, добавление или удаление объектов на сцене.

1.2 Выбор модели представления объектов

Отображением формы и размеров объектов являются модели. Они могут быть следующих видов:

- 1) **Каркасная модель.** Одна из простейших форм задания модели, так как мы храним информацию только о вершинах и ребрах нашего объекта. Основная проблема отображения объекта с помощью данной модели заключается в том, что модель не всегда однозначно может передать представление о форме объекта;
- 2) **Поверхностная модель.** Такая информационная модель содержит данные только о внешних геометрических параметрах объекта. Дан-

ный тип модели часто используется в компьютерной графике. При этом могут использоваться различные типы поверхностей, ограничивающих объект, такие как полигональные модели, поверхности второго порядка и др. Недостатком данной модели является отсутствие информации о том, с какой стороны поверхности находится материал;

- 3) **Объемная модель.** Данная форма задания модели отличается от поверхностной наличием информации о том, с какой стороны расположен материал. Это реализуется с помощью указания направления внутренней нормали.

Для решения поставленной задачи будет использована поверхностная модель. Этот выбор обусловлен тем, что каркасные модели будут приводить к неправильному восприятию форм объекта, а объемные не подходят, так как нам не важно из какого материала сделаны объекты сцены.

1.3 Выбор способа задания поверхностных моделей

Существует несколько способов задания поверхностных моделей:

- **Аналитический способ.** Данный способ задания модели характеризуется описанием модели объекта, которое доступно в неявной форме. То есть для получения поверхности необходимо вычислять функцию, зависящую от параметра;
- **Полигональная сетка.** Данный способ представляет собой совокупность вершин, ребер и полигонов, соединенных таким образом, что каждое ребро принадлежит не более, чем двум многоугольникам. Ребра ограничиваются двумя вершинами, а полигоны замыкаются цепочкой из последовательно соединенных ребер [1].

Из двух способов задания поверхностных моделей наиболее оптимальным является использование полигональной сетки, так как данный вариант позволяет более быстро выполнять операции над объектами.

1.4 Выбор способа хранения полигональных моделей

Существует несколько способов хранения полигональных моделей:

- **Вершинное представление.** Хранится информация о вершинах, указывающих на другие вершины, которые с ними соединены;
- **Список граней.** Объект представляется как множество граней и вершин, любая грань состоит из минимум трех вершин;
- **Таблица углов.** Обход данной таблицы неявно задает полигоны. Такое представление более компактно и более производительно для нахождения полигонов. Но из-за того, что вершины присутствуют в описании нескольких углов, операции по их изменению медленны;
- **Крылатое представление.** Каждая точка ребра указывает на две вершины, две грани и четыре ребра, которые ее касаются.

Способом хранения полигональной сетки выбран список граней, так как это даст явное описание граней. Этот способ позволит эффективно преобразовывать модели, так как структура будет включать в себя список вершин.

1.5 Алгоритмы удаления невидимых линий и поверхностей

Задача удаления невидимых линий и поверхностей является одной из наиболее сложных в машинной графике. Алгоритмы удаления невидимых линий и поверхностей служат для определения ребер, поверхностей или объемов, которые видимы или невидимы для наблюдателя, находящегося в заданной точке пространства.

Для оптимальной работы реализуемого программного обеспечения алгоритм должен быть достаточно быстрым при работе со множеством объектов сцены, чтобы не было долгой загрузки изображения, мог работать в любом пространстве, так как скорость важнее точности.

1.5.1 Алгоритм Робертса

Данный алгоритм работает в объектном пространстве, решая задачу только с выпуклыми телами.

Алгоритм выполняется в 3 этапа:

1) Этап подготовки исходных данных

Для каждого объекта сцены формируется матрица тела V . Размерность матрицы – $4 * n$, где n – количество граней тела.

Каждый столбец матрицы представляет собой четыре коэффициента уравнения плоскости $ax + by + cz + d = 0$, проходящей через очередную грань.

Таким образом, матрица тела будет представлена в следующем виде:

$$V = \begin{pmatrix} a_1 & a_2 & \dots & a_n \\ b_1 & b_2 & \dots & b_n \\ c_1 & c_2 & \dots & c_n \\ d_1 & d_2 & \dots & d_n \end{pmatrix}. \quad (1.1)$$

Для корректного формирования матрицы необходимо осуществить проверку, что любая точка, находящаяся внутри тела, расположается на положительной стороне каждой грани этого тела. При не выполнении условия соответствующий столбец матрицы надо умножить на -1 .

2) Этап удаление ребер и граней, которые перекрываются самим телом

На данном этапе рассматривается вектор взгляда $E = \{0, 0, -1, 0\}$.

Для определения невидимых ребер и граней достаточно умножить вектор E на матрицу тела V . Отрицательные компоненты полученного вектора будут соответствовать невидимым граням.

3) Этап удаления невидимых ребер, экранируемых другими телами сцены

На данном этапе происходит сравнение видимых ребер каждого тела с каждым из оставшихся тел для определения того, какая его часть или части, если таковые есть, перекрываются этими телами [2].

Особенности:

- математические методы, используемые в алгоритме Робертса просты и точны;
- вычислительная трудоемкость растет прямо пропорционально квадрату количества объектов сцены;
- могут возникнуть трудности на этапе подготовки информации об объектах сцены;
- возможность работы только с выпуклыми объектами.

1.5.2 Алгоритм Варнок

Алгоритм Варнока основан на разбиении картинной плоскости на части, для каждой из которых исходная задача может быть решена достаточно просто.

Поскольку данный алгоритм нацелен на обработку картинки, он работает в пространстве изображения. В пространстве изображения рассматривается окно и решается вопрос о том, пусто ли оно, или его содержимое достаточно просто для визуализации. Если это не так, то окно разбивается

на фрагменты до тех пор, пока содержимое фрагмента не станет достаточно простым для визуализации или его размер не достигнет требуемого предела разрешения [2].

Особенности:

- чем меньше часть изображения содержит информации, тем меньше затраты по времени и наоборот;
- алгоритм работает рекурсивно.

1.5.3 Алгоритм обратной трассировки лучей

Алгоритм обратной трассировки лучей работает в пространстве изображения. Наблюдатель видит объект посредством испускаемого источником света, который падает на этот объект и согласно законам оптики некоторым путем доходит до глаз наблюдателя.

Данный алгоритм состоит из следующих этапов:

- 1) для каждого пикселя на дисплее проводится прямой луч от наблюдателя до элемента сцены;
- 2) пересечение используется для определения цвета пикселя как функции пересекаемой поверхности элемента;
- 3) проводятся вторичные лучи от точек пересечения до разных источников света для определения освещённости пикселя;
- 4) если луч блокируется, то точка находится в тени, которую отбрасывает рассматриваемый источник света, иначе источник света влияет на освещение.

Для более реалистичного изображения необходимо проводить лучи отражения и лучи преломления.

Особенности:

- реалистичное изображение объекта, построенное по физическим законам;
- большая трудоемкость вычислений.

1.5.4 Алгоритм, использующий z-буфер

Идея z-буфера является простым обобщением идеи о буфере кадра. Буфер кадра используется для запоминания атрибутов (интенсивности) каждого пикселя в пространстве изображения, z-буфер — это отдельный буфер глубины, используемый для запоминания координаты z или глубины каждого видимого пикселя в пространстве изображения.

Процесс работы алгоритма, использующего z-буфер:

- 1) всем элементам буфера кадра присвоить фоновое значение;
- 2) инициализировать z-буфер минимальным значением глубины;
- 3) глубина z каждого нового пикселя, который нужно занести в буфер кадра, сравнивается с глубиной того пикселя, который уже занесен в z-буфер;
- 4) если сравнение показывает, что новый пиксель расположен впереди пикселя, находящегося в буфере кадра, то новый пиксель заносится в этот буфер;
- 5) если же сравнение дает противоположный результат, то никаких действий не производится.

По сути, алгоритм является поиском по x и y наибольшего значения функции $Z(x, y)$ [2].

Особенности:

- простая реализация;

- возможность модификации алгоритма для работы с тенями;
- работа с любой сложностью поверхностей;
- большой объем требуемой памяти для хранения буфера.

1.5.5 Выбор алгоритма удаления невидимых линий и поверхностей

Самым подходящим алгоритмом для удаления невидимых линий и поверхностей будет z-буфер. Выбор обусловлен тем, что данный алгоритм прост в реализации, работает с любой сложностью сцен, оценка вычислительной трудоемкости алгоритма не более чем линейна.

1.6 Выбор алгоритма построения теней

При выборе алгоритма создания теней важно учесть простоту реализации и время, необходимое для написания алгоритма. Если использовать алгоритм обратной трассировки лучей для удаления невидимых ребер и поверхностей, то тени будут автоматически созданы в ходе выполнения алгоритма. Это происходит потому, что пиксель затемняется, когда луч, испускаемый из данной точки, сталкивается с объектом, но не достигает источника света.

Однако, учитывая сложность алгоритма обратной трассировки, возможно решение модифицировать алгоритм z-буфера, добавив в него функцию вычисления теневого буфера.

Такая модификация позволит избежать необходимости написания большого объема дополнительного кода. Вместо этого, потребуется лишь внести небольшие изменения в уже использующийся алгоритм.

Вывод

Были рассмотрены способы задания и хранения трехмерных моделей, а также выбраны наиболее подходящие из них. Также были рассмотрены алгоритмы удаления невидимых ребер:

- алгоритм Робертса;
- алгоритм Варнока;
- алгоритм обратной трассировки лучей;
- алгоритм, использующий Z-буфер.

В качестве реализуемого был выбран алгоритм z-буфера, отвечающего главному требованию — быстрая работа со множеством объектов сцены. Были рассмотрены алгоритмы построения теней. В качестве реализуемого был выбран алгоритм, использующий теневые карты, в работе которого будет использоваться z-буфер.

2 Конструкторская часть

2.1 Требования к программному обеспечению

Программа должна предоставить следующие функции:

- создание сцены заданного размера;
- добавление, поворот, удаления объектов сцены;
- перемещение, поворот камеры сцены.

2.2 Алгоритм, использующий z-буфер

- Всем элементам буфера кадра присвоить фоновое значение.
- Инициализировать z-буфер минимальным значением глубины.
- Для каждого полигона сцены в произвольном порядке:
 - для каждого пикселя, который принадлежит полигону вычислить глубину $z(x, y)$;
 - сравнить вычисленную глубину пикселя со значением, которое хранится в z-буфере. Если $Z(x, y) > Zbuf(x, y)$, то $Zbuf(x, y) = Z(x, y)$ и $color(x, y) = pixelColor$.
- Вывести итоговое изображение.

На рисунке 2.1 представлена схема алгоритма для удаления невидимых линий и поверхностей с помощью z-буфера.

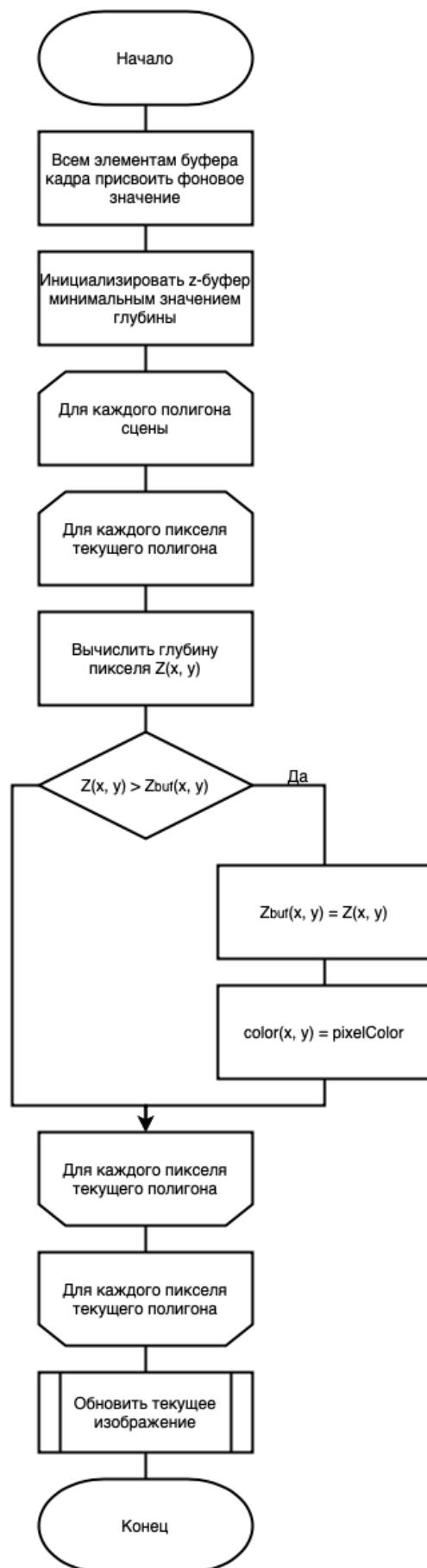


Рисунок 2.1 – Схема алгоритма, использующего z-буфер

2.3 Алгоритм для построения теней, использующий z-буфер

- Для каждого направленного источника света:
 - инициализировать теневой z-буфер минимальным значением глубины;
 - определить теневой z-буфер для источника.
- Выполнить алгоритм z-буфера для точки наблюдения. При этом, если некоторая поверхность оказалась видимой относительно текущей точки наблюдения, то проверить, видима ли данная точка со стороны источников света.
- Для каждого источника света:
 - координаты рассматриваемой точки (x, y, z) преобразовать из вида наблюдателя в координаты (x_0, y_0, z_0) в вид из рассматриваемого источника света;
 - сравнить значение $Z_{\text{shadowBuf}}(x_0, y_0)$ со значением $Z_0(x_0, y_0)$. Если $Z_0(x_0, y_0) < Z_{\text{shadowBuf}}(x_0, y_0)$, то пиксел высвечивается с учетом его затемнения, иначе точка высвечивается без изменений.
- Вывести итоговое изображение.

На рисунках 2.2 – 2.3 представлена схема алгоритма для построения теней с помощью z-буфера.



Рисунок 2.2 – Схема алгоритма для построения теней, использующего z-буфер

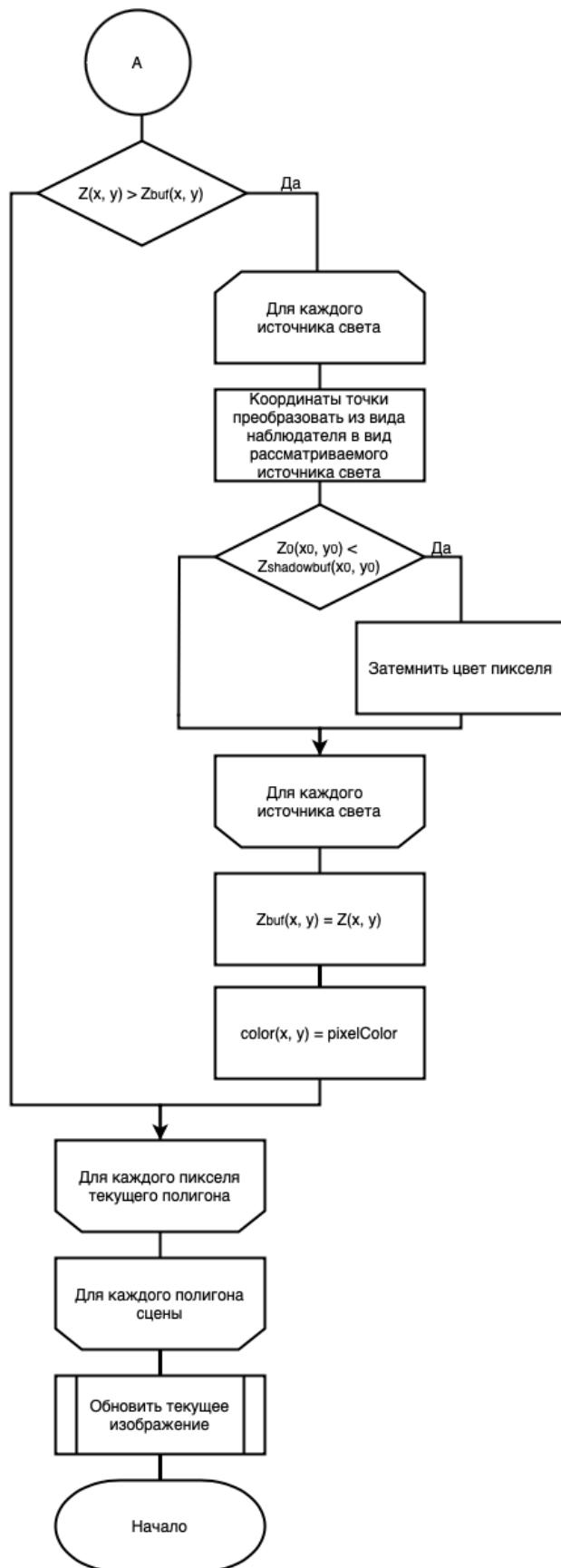


Рисунок 2.3 – Схема алгоритма для построения теней, использующего z-буфер

2.4 Используемые типы и структуры данных для представления объектов

Для решения поставленных задач курсового проекта необходимо определить представление объектов разрабатываемой программы. В таблице 2.1 представлены объекты и выбранные для них типы и структуры данных.

Таблица 2.1 – Выбранный типы и структуры данных для представления объектов

Объект	Типы и структуры данных
Точка в трехмерном пространстве	Вектор, состоящий из четырех координат типа float64: x, y, z, w
Вершина	Точка в трехмерном пространстве
Полигон	Структура, содержащая три вершины
Камера	Структура, содержащая направление и матрицу преобразования вершины в пространство камеры вершины в пространство камеры типа float64
Двигатель программы	Структура, содержащая: камеру, источник света, z-буфер, теневой буфер, матрицу типа float64 для преобразования из пространства камеры в пространство источника света

Вывод

В этом разделе были определены требования к программному обеспечению, выбраны типы и структуры данных. Также были разработаны алгоритм для удаления невидимых линий и поверхностей, использующий z-буфер и алгоритм, для построения теней с помощью z-буфера.

3 Технологическая часть

В этом разделе будут выбраны язык программирования для разработки программного обеспечения. Также будет представлен интерфейс программы и продемонстрирован ее функционал.

3.1 Выбор языка программирования

Для реализации данной лабораторной работы выбран компилируемый язык программирования Go [3]. Данный выбор обусловлен следующими причинами:

- статическая типизация, помогающая выявлять ошибки на этапе компиляции;
- наличие сборщика мусора, который помогает облегчить работу с памятью;
- личное желание расширить свои знания в области применения данного языка.

3.2 Выбор среды разработки

В качестве среды разработки программного обеспечения был выбран Visual Studio Code. Данный выбор обусловлен следующими причинами:

- предоставляет встроенный отладчик;
- наличие опыта работы с данной средой разработки.

3.3 Интерфейс программного обеспечения

На рисунке 3.1 представлен интерфейс программы, который поделен на две части, меню для взаимодействия со сценой и пространство, где происходит отрисовка сцены и объектов на ней.



Рисунок 3.1 – Интерфейс программного обеспечения

Меню управления сценой состоит из следующих разделов:

- «Работа со сценой»— раздел, отвечающий за создание сцены определенного размера, очистка изображения от сцены и объектов, расположенных на ней;
- «Управление камерой»— раздел, предназначенный для управления движением камеры;
- «Работа с объектами сцены»— раздел, обеспечивающий создание, удаление и поворот выбранного объекта;
- «Выбор темы интерфейса»— раздел, дающий возможность поменять тему программного обеспечения.

Раздел «Работа со сценой» представлен на рисунке 3.2. Данный раздел позволяет ввести количество квадратов, из которых будет состоять сцена. Максимальный размер сцены 15x15. Также предоставляется очистка изображения, будет удалена сцена вместе с объектами, расположенными на ней.

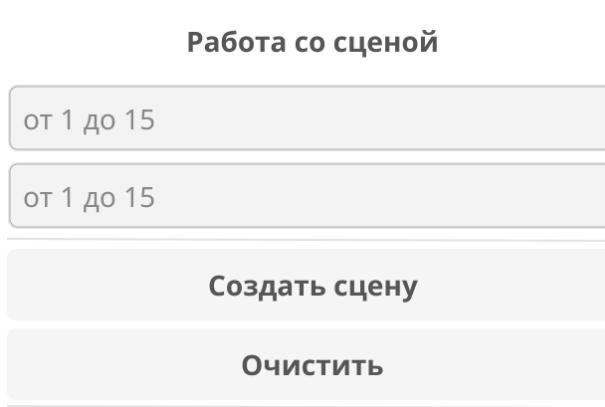


Рисунок 3.2 – Раздел «Работа со сценой»

Раздел «Управление камерой», представлена на рисунке 3.3. Данный раздел помогает управлять камерой без использования клавиатуры. С помощью стрелок \leftarrow , \rightarrow , \uparrow , \downarrow осуществляется движение камеры влево, вправо, вверх и вниз соответственно. Кнопки w, s, a, d помогают приближать, отдалять, поворачивать камеру налево и направо.

Управление камерой



Рисунок 3.3 – «Управление камерой»

Раздел «Работа с объектами сцены», показанный на рисунке 3.4, предназначен для создания, поворота и удаления объектов сцены. Для построения объекта необходимо выбрать объект в подразделе «Выберите объект» и «создать» в подразделе «Действие», далее нужно выбрать ячейку, в которой будет располагаться объект. Для удаления или поворота объекта необходимо выбрать нужное действие, затем выбрать ячейки, в которых располагается нужный объект.

Работа с объектами сцены

Выберите объект	Действие
<input type="radio"/> станция	<input type="radio"/> создать
<input type="radio"/> головной вагон	<input type="radio"/> удалить
<input type="radio"/> вагон	<input type="radio"/> повернуть
<input type="radio"/> закругленные рельсы	номер по X
<input type="radio"/> прямые рельсы	номер по Y
<input type="radio"/> дерево	

Выполнить

Рисунок 3.4 – Раздел «Работа с объектами сцены»

Раздел «Выбор темы интерфейса», представленный на рисунке 3.5, позволяет выбрать тему интерфейса.

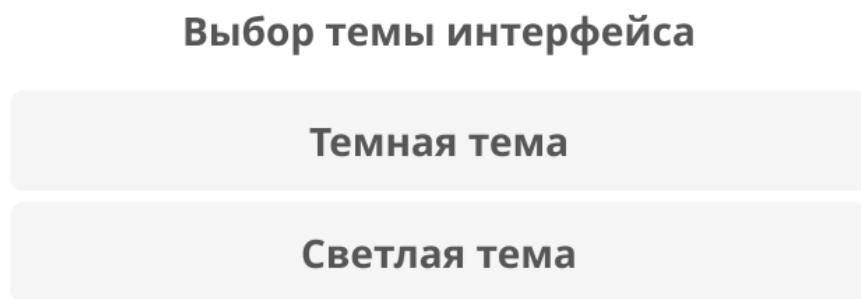


Рисунок 3.5 – Интерфейс программного обеспечения

3.4 Пример работы программы

На рисунке 3.6 продемонстрирована работа программы при расположении на сцене нескольких объектов.

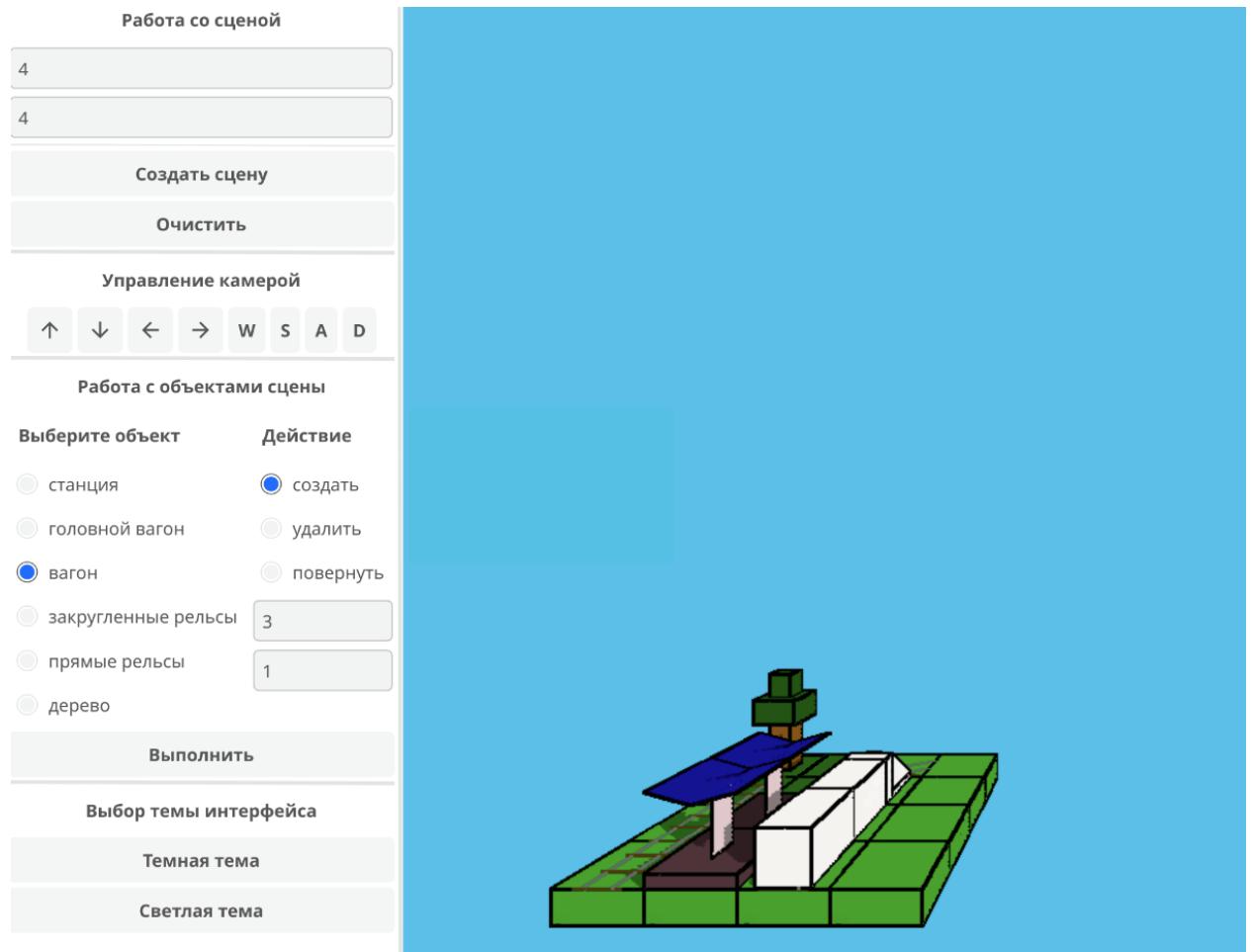


Рисунок 3.6 – Пример работы программного обеспечения

4 Исследовательская часть

В этом разделе представлены параметры технического оборудования, на котором осуществлялись замеры времени выполнения программного обеспечения, а также полученные результаты измерений времени работы.

4.1 Постановка цели исследования

Целью исследования является проведение анализа скорости реализации работы алгоритма генерации одного кадра с помощью алгоритма Z-буфера в зависимости от количества полигонов и шага полигональной сетки.

4.2 Технические характеристики

Технические характеристики устройства, на котором выполнялись замеры времени, представлены далее:

- процессор – 2 ГЦ 4-ядерный процессор Intel Core i5;
- оперативная память – 16 ГБайт;
- операционная система – macOS Sonoma 14.1.1.

4.3 Средство замера времени

Измерение времени выполнения программы будет проводиться с использованием пакета cgo [4], который внедряет код на языке C [5]. Эта вставка кода будет предназначена для оценки процессорного времени работы программы.

4.4 Результаты замера времени

Для исследования влияния количества полигонов на время генерации изображения проводились замеры, в ходе которых размерность сцены изменялась от 10 до 18. Также в процессе исследования происходило изменение шага полигональной сетки. Замеры происходили в миллисекундах. Результаты замеров времени приведены в таблицы 4.1. На рисунке 4.1 представлена зависимость времени выполнения программы от шага полигональной сетки и от размерности сцены.

Таблица 4.1 – Результаты замера времени (мс)

Шаг полигон. сетки	Размерность сцены								
	10	11	12	13	14	15	16	17	18
0.5	119	172	213	274	322	391	555	916	1932
0.75	125	179	218	283	355	460	569	945	1994
1	129	182	224	286	363	465	623	1016	2098
1.25	146	185	229	295	387	485	663	1039	2280
1.5	148	197	232	321	398	495	743	1452	2520

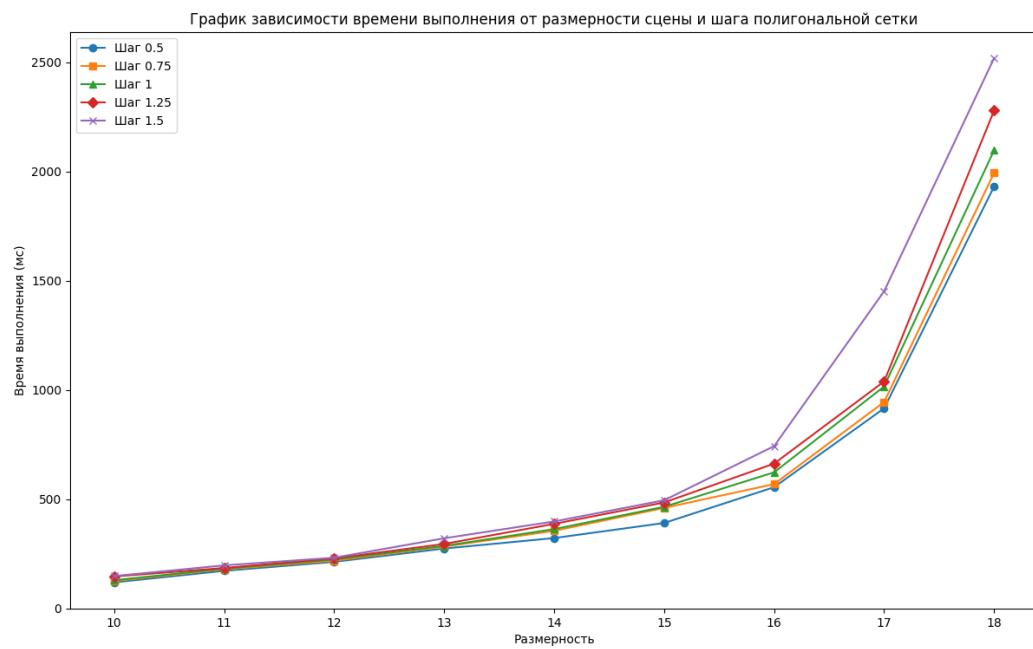


Рисунок 4.1 – График зависимости времени генерации одного кадра в зависимости от размерности сцены и шага полигональной сетки

На рисунке 4.2 представлен приближенный график, иллюстрирующий изменение времени генерации одного кадра при размерностях сцены от 10 до 13. Это приближенное представление помогает лучше понять различия в выполнении программы.

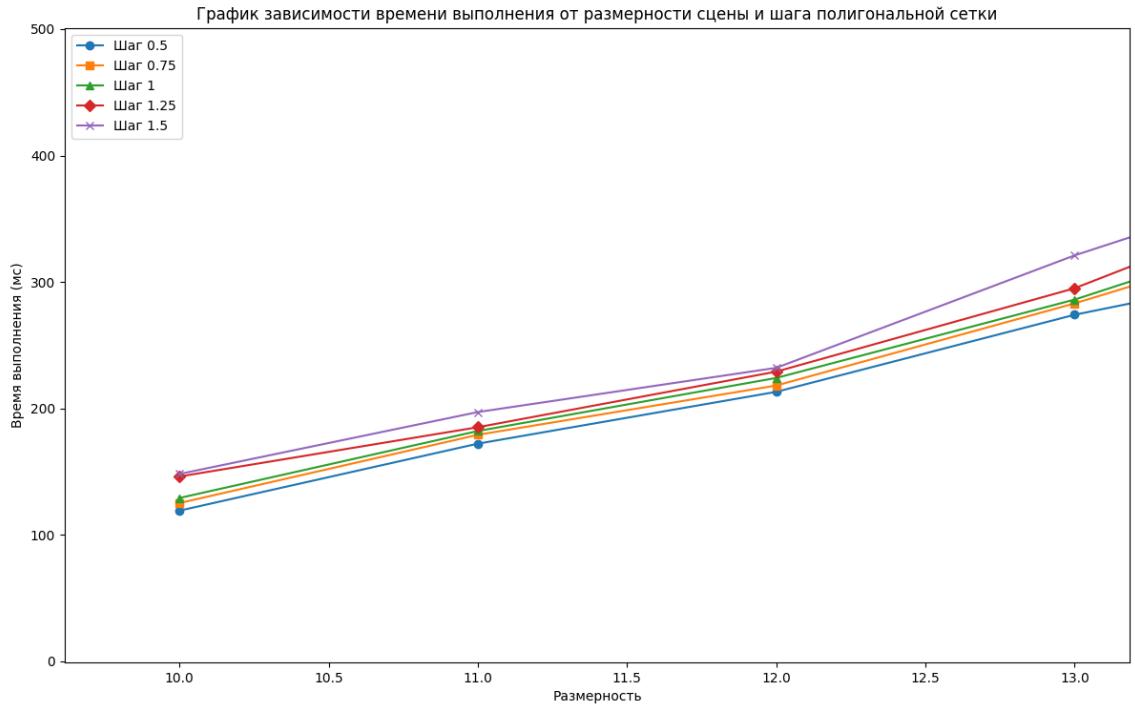


Рисунок 4.2 – График зависимости времени генерации одного кадра в зависимости от размерности сцены и шага полигональной сетки

По результатам данных замеров времени можно сделать следующие выводы:

- время генерации одного кадра линейно зависит от размерности сцены, что ожидаемо, так как с увеличением числа полигонов требуется больше вычислительных ресурсов;
- увеличение размера шага полигональной сетки приводит к увеличению времени выполнения программы, так как с увеличением шага происходит увеличение размера полигонов, что требует дополнительных вычислительных операций для генерации изображения.

4.5 Вывод

В данном разделе было проведено исследование, направленное на выявление зависимостей времени генерации одного кадра при использовании алгоритма Z-буфера от характеристик сцены, таких как размерность (количество полигонов) и размер шага полигональной сетки.

ЗАКЛЮЧЕНИЕ

Цель работы, которая заключалась в разработке программного обеспечения для построения железнодорожной инфраструктуры выполнена. Также в ходе работы были решены следующие задачи:

- описан список доступных к размещению на сцене моделей, формализовать эти модели;
- выбраны алгоритмы компьютерной графики для визуализации сцены и объектов на ней;
- выбран язык программирования и среду разработки;
- реализованы выбранные алгоритмы визуализации;
- разработано программное обеспечение для визуализации и редактирования железнодорожной инфраструктуры.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Н. Божко А. Компьютерная графика. Изд-во МГТУ им.Н. Э. Баумана, 2007. С. 174–177.
2. Ю. Демин А. Основы компьютерной графики. Изд-во Томского политехнического университета, 2011. С. 130–151.
3. The Go Programming Language [Электронный ресурс]. Режим доступа: <https://golang.org/> (дата обращения: 25.09.2023).
4. Документация CGo [Электронный ресурс]. Режим доступа: <https://pkg.go.dev/cmd/cgo> (дата обращения: 20.11.2023).
5. Документация С [Электронный ресурс]. Режим доступа: <https://learn.microsoft.com/ru-ru/cpp/c-language/?view=msvc-170> (дата обращения: 20.11.2023).

Приложение А

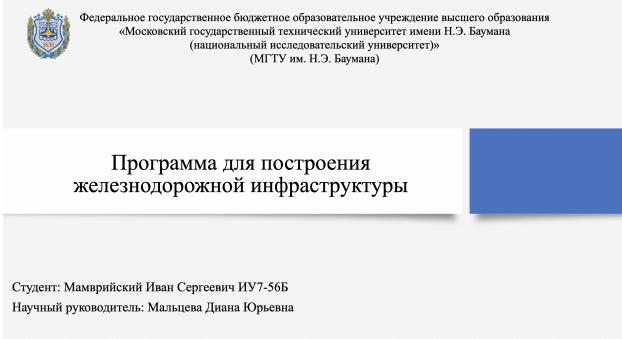


Рисунок 4.3 – Титульный лист (слайд 1)

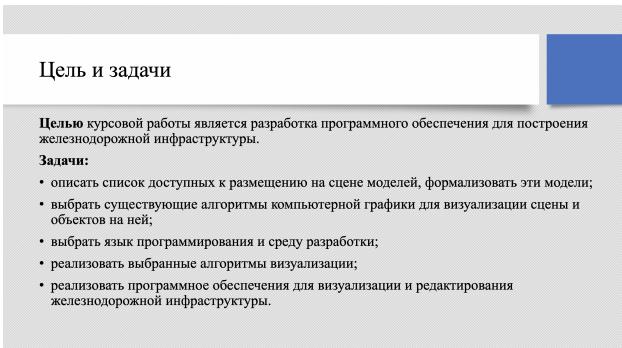


Рисунок 4.4 – Цели и задачи (слайд 2)

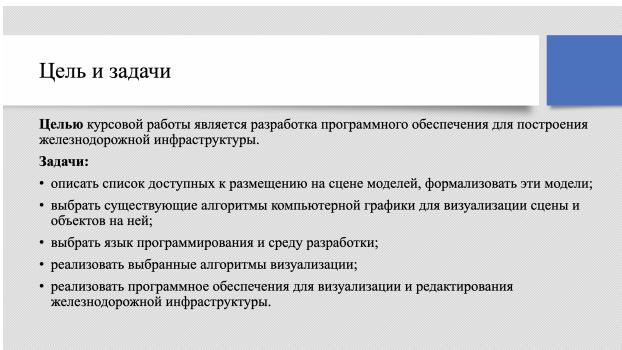


Рисунок 4.5 – Цели и задачи (слайд 2)

Описание и формализация объектов сцены

Объекты сцены:

- площадка сцены;
- объекты сцены:
 - станция;
 - головной вагон;
 - вагон;
 - закругленные рельсы;
 - прямые рельсы;
 - дерево.
- источник света.

Деревья, станции, рельсы разрешено расставлять только на свободных клетках сетки сцены, вагоны – только на железнодорожных путях.

Рисунок 4.6 – Описание и формализация объектов сцены (слайд 3)

Анализ и выбор алгоритмов

- Алгоритмы удаления невидимых ребер и поверхностей:
 - алгоритм Роберта;
 - алгоритм Варнока;
 - алгоритм обратной трассировки лучей;
 - алгоритм, использующий z-буфер.
- Алгоритмы построения теней:
 - алгоритм обратной трассировки лучей;
 - алгоритм, использующий z-буфер.

Рисунок 4.7 – Анализ и выбор алгоритмов (слайд 4)

Алгоритм z-буфера

- Всем элементам буфера кадра присвоить фоновое значение.
- Инициализировать z-буфер минимальным значением глубины.
- Для каждого многоугольника сцены в произвольном порядке:
 - для каждого пикселя, который принадлежит многоугольнику вычислить его глубину $z(x, y)$;
 - сравнить вычисленную глубину пикселя со значением, которое находится в z-буфере. Если $z(x, y) > zbuf(x, y)$, то $zbuf(x, y) = z(x, y)$ и $colour(x, y) = specColour$.
- Вывести итоговое изображение.

Рисунок 4.8 – Анализ z-буфера (слайд 5)

Модифицированный алгоритм z-буфера для построения теней

- Для каждого направленного источника света:
 - инициализировать теневой z-буфер минимальным значением глубины;
 - определить теневой z-буфер для источника.
- Выполнить алгоритм z-буфера для точки наблюдения. При этом, если некоторая поверхность оказалась видимой относительно текущей точки наблюдения, то проверить, видима ли данная точка со стороны источников света.
- Для каждого источника света:
 - координаты рассматриваемой точки (x, y, z) линейно преобразовать из вида наблюдателя в координаты $(x0, y0, z0)$ на виде из рассматриваемого источника света;
 - сравнить значение $ZshadowBuf(x0, y0)$ со значением $Z0(x0, y0)$. Если $Z0(x0, y0) < ZshadowBuf(x0, y0)$, то пиксель высвечивается с учетом его затмения, иначе пиксель высвечивается без затмения.

Рисунок 4.9 – Модифицированный алгоритм z-буфера для построения теней (слайд 6)

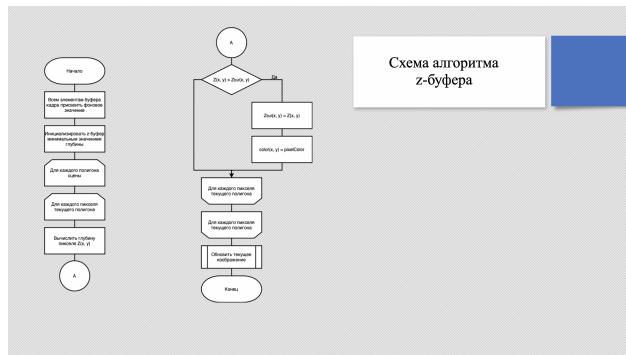


Рисунок 4.10 – Схема алгоритма z-буфера (слайд 7)

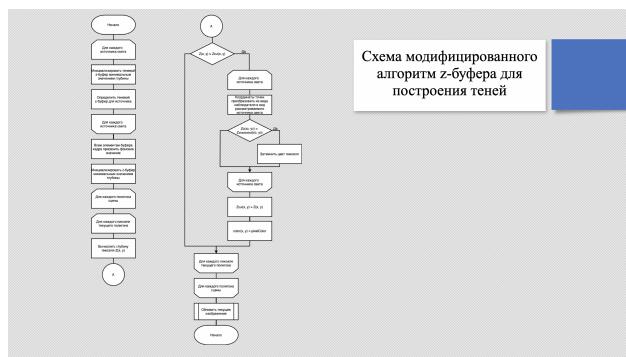


Рисунок 4.11 – Схема модифицированного алгоритма z-буфера для построения теней (слайд 8)

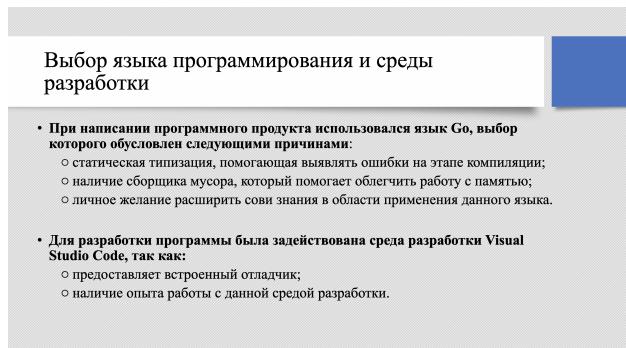


Рисунок 4.12 – Выбор языка программирования и среды разработки (слайд 9)

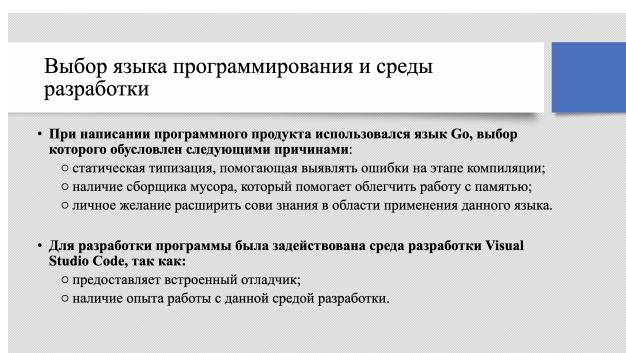


Рисунок 4.13 – Выбор языка программирования и среды разработки (слайд 9)