-	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 ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент Царев Антон Андреевич
фамилия, имя, отчество

Группа ИУ7-53Б

Тип практики технологическая

Название предприятия МГТУ им. Н. Э. Баумана

Студент _____ Царев А. А.
подпись, дата *фамилия, и.о.*

Руководитель практики _____ Куров А. В
подпись, дата *фамилия, и.о.*

Оценка _____

2022 г.

«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой _____
_____ _____
« ___ » _____ 20 __ г.

З А Д А Н И Е
на прохождение производственной практики
_____технологическая_____
Тип практики

Студент
_____Царев Антон Андреевич_____ _____ курса группы _____
Фамилия Имя Отчество № курса индекс группы

в период с _____. _____.20_____ г. по _____. _____.20_____ г.

Предприятие: _____МГТУ им. Н. Э. Баумана_____
Подразделение: _____кафедра ИУ7_____
(отдел/сектор/цех)
Руководитель практики от предприятия (наставник):
_____
(Фамилия Имя Отчество полностью, должность)
Руководитель практики от кафедры:
_____
(Фамилия Имя Отчество полностью, должность)

Задание:
1. Начать разработку программы моделирования детской игрушки “Лизун”
2. Определить способы представления объектов, проанализировать и выбрать алгоритмы для их обработки
3. Закрепить знания и навыки, полученные в ходе изучения курса компьютерной графики

Дата выдачи задания « ____ » _____ 20___ г.

Руководитель практики от предприятия _____/_____/

Руководитель практики от кафедры _____/_____/

Студент _____/_____/

Оглавление

Введение

В настоящее время компьютерная графика имеет широкое применение в различных сферах. В частности, визуализация объектов с помощью электронно-вычислительных машин используется в киноиндустрии, разработке компьютерных игр и моделировании физических процессов.

Особое внимание уделяется моделированию деформируемых тел, которые могут менять свою форму, внутреннюю структуру, объем и площадь поверхности под действием внешних сил. Одним из таких тел является детская игрушка “Лизун” (она же “Слайм”).

Целями данной работы являются:

- выбор и обоснование модели тела и алгоритмов, которые можно применить для реализации программы моделирования детской игрушки “Лизун”;

- разработка оконного интерфейса программы;

- закрепление знаний и навыков, приобретенных в ходе изучения курса компьютерной графики;

1. Аналитическая часть

1.1 Описание объектов сцены

Сцена состоит из следующих объектов: камера, источник света, пол, имеющий структуру, и слайм.

1.1.1 Камера

Камера представляет собой невидимый объект, содержащий в себе информацию о координатах положения камеры в пространстве и векторе направления взгляда и использующийся для получения изображения на дисплее.

1.1.2 Источник света

Источник света представляет собой материальную точку, которая испускает лучи света во все стороны. Данный объект хранит в себе информацию о координатах положения в пространстве источника света, интенсивности и цвете в формате RGB.

1.1.3 Пол

Пол представляет собой бесконечную плоскость, заданную уравнением $y=0$. Предполагается, что все объекты находятся над полом или на нем.

1.1.4 Слайм

В нашей программе слайм представляет собой вязкоупругое тело. При отсутствии внешних сил тело имеет форму шара.

1.1.4.1 Выбор модели представления трехмерного объекта

Для задания трехмерных моделей выделяют три формы: каркасную, поверхностную и объемную.

Каркасная модель является простейшим видом. В ней задается информация о вершинах и ребрах объекта. Однако ввиду своей простоты данный вид обладает серьезным недостатком: не всегда корректно передается представление об объекте.

Поверхностная модель предполагает хранение информации не только о вершинах и ребрах объекта, но и о его поверхности. Поверхность может быть описана как аналитически, так и с помощью задания участков поверхности как поверхностей того или иного рода. Недостатком данной модели является невозможность определения, с какой стороны поверхности находится материал объекта.

Объемная модель отличается от поверхностной лишь тем, что в ней мы храним информацию о расположении материала объекта, указывая направление вектора внутренней нормали.

Для данного проекта была выбрана поверхностная модель, так как каркасная модель не всегда правильно передает представление об объекте, а объемная модель является избыточной в рамках решаемой задачи.

1.1.4.2 Выбор модели представления поверхности объекта

Поверхность объекта может быть описана как аналитически, так и с помощью полигональной сетки. В аналитическом методе поверхность объекта рассматривается как множество точек, координаты которых удовлетворяют заданному уравнению, в то время как полигональная сетка представляет собой совокупность вершин, ребер и полигонов, соединенных таким образом, что каждое ребро принадлежит не более, чем двум многоугольникам.

Существует несколько способов описания полигональных сеток:

1) Список вершин

В этом случае информация о сетке хранится в виде перечня координат вершин, в котором каждая вершина записывается один раз в виде тройки координат. При этом каждый многоугольник задается множеством порядковых номеров вершин в списке.

2) Список ребер

Базовую информацию дает список вершин, но многоугольники описываются ссылками на список ребер, в котором каждое из них упоминается только один раз. В свою очередь, ребра представляются в виде пары граничных вершин и перечня смежных многоугольников, число которых не превышает двух.

3) Список граней

В данном способе хранятся список вершин и список граней. В каждой грани хранится информация о трех вершинах.

Для данного проекта была выбрана модель полигональной сетки, ввиду простоты реализации и отсутствия серьезных проблем при описании сложных объектов. Кроме того, хранение информации о сетке будет осуществляться с помощью списка граней, так как данный способ предоставляет полную информацию о гранях, которая может быть крайне полезна при реализации алгоритмов удаления невидимых ребер и поверхностей.

1.1.4.3 Метод физического моделирования объекта

Для моделирования процессов, протекающих при деформации слайма, необходимо выбрать соответствующий метод описания физических свойств тела.

1) Метод конечных элементов

В данном методе объект разбивается на конечные элементы, соединяющиеся между собой в узлах. Каждый конечный элемент должен быть достаточно простым, чтобы имелась возможность легко определить перемещения и напряжения в любой его части по заданным перемещениям узлов. По точкам составляется система уравнений, которая решается относительно перемещений рассматриваемых точек. Данная модель имеет высокую точность, однако требует знания характеристик материала объекта, необходимых для рассчета соотношений для каждого элемента.

2) Метод граничных элементов

Метод граничных элементов является интересной альтернативой метода конечных элементов, так как все преобразования осуществляются над точками поверхности объекта, что дает прирост к скорости вычислений. Основная сложность реализации данного метода заключается в преобразовании интегральной формы уравнения движения тела в поверхностный интеграл.

3) Модель масс-с-пружинами

Система масс с пружинами подразумевает хранение объекта, как совокупности точечных масс, соединенных между собой сетью невесомых пружин. Пружины обычно моделируются как упругие, то есть сила, действующая на массу i со стороны пружины, вычисляется по закону Гука:

$$F_i = k_s\left(\left|x_{ij}\right| - l_{ij}\right)\frac{x_{ij}}{\left|x_{ij}\right|}, \qquad (1)$$

где k_s – жесткость пружины, соединяющий массы i и j; x_{ij}- разница радиус-векторов масс j и i соответственно; l_{ij}- длина покоя, то есть расстояние между массами i и j, при котором сила упругости равна нулю.

Физические тела не являются идеально упругими, так как они рассеивают энергию во время деформации. Чтобы учесть это, для затухания относительного движения используются вязкоупругие пружины. Обычно они моделируются следующим выражением:

$$F_i = k_d(v_j - v_i), \tag{2}$$

где k_d - коэффициент затухания; v_i, v_j- векторы скоростей масс i и j соответственно. Однако данная формула неудобна для использования, так как гасит вращение твердого тела. Поэтому предпочтительнее применять немного измененное выражение:

$$F_i = k_d\left(\frac{v_{ij}^T x_{ij}}{x_{ij}^T x_{ij}}\right)x_{ij}, \tag{3}$$

где $v_{ij} = v_j - v_i$. По сути, мы проецируем разность скоростей на вектор разности радиус-векторов масс и допускаем силу только в этом направлении.

Для описания динамики движения слайма была выбрана модель масс с пружинами, так как она позволяет выполнять простые вычисления и дает приемлемую точность моделирования.

1.2 Выбор алгоритма удаления невидимых ребер и поверхностей

Перед выбором алгоритма удаления невидимых ребер выделим несколько свойств, которыми должен обладать выбранный алгоритм, чтобы обеспечить оптимальную работу и реалистичное изображение:

1) Алгоритм может работать как в объектном пространстве, так и в пространстве изображений;

2) Алгоритм должен быть достаточно быстрым и использовать мало памяти;

3) Алгоритм должен иметь высокую реалистичность изображения.

Рассмотрим следующие алгоритмы:

1) Алгоритм Робертса

Алгоритм Робертса работает в объектном пространстве, осуществляя удаление невидимых линий выпуклых тел. Выполнение данного алгоритма можно разделить на 4 этапа: подготовка исходных данных; удаление ребер, экранируемых самим телом; удаление ребер, экранируемых другими телами; удаление линий пересечения тел, экранируемых самими телами и другими телами, связанными отношением протыкания.

На первом этапе для каждого тела формируется матрица, каждый столбец которой содержит коэффициенты уравнения плоскости $ax+by+cz+d=0$, проходящей через соответствующую грань:

$$V=\begin{pmatrix} a_1 & a_2 & & a_n \\ b_1 & b_2 & \cdots & b_n \\ c_1 & c_2 & & c_n \\ d_1 & d_2 & & d_n \end{pmatrix}. \qquad (5)$$

Матрица тела должна быть сформирована корректно. Это значит, что любая точка, лежащая внутри тела, должна лежать по положительную сторону от каждой грани тела. Если для какой-либо грани это условие не выполняется, соответствующий столбец матрицы умножается на -1.

На втором этапе указывается расположение наблюдателя и направление его взгляда. Как правило, наблюдатель располагается в бесконечности на положительной полуоси z и смотрит в сторону начала координат. В однородных координатах вектор такого направления равен $E=[0,0,-1,0]$.

Для определения невидимых граней нужно умножить вектор E на матрицу тела V. Отрицательные компоненты результирующего вектора соответствуют невидимым ребрам тела.

На третьем этапе работы алгоритма для определения невидимых линий строится луч, соединяющий точку наблюдения с точкой на ребре. Точка невидима, если луч проходит через тело.

На четвертом этапе ведется поиск ребер, образовавшихся в результате взаимного протыкания тел. Вновь полученные ребра проверяются на экранирование телами.

Данный алгоритм дает высокую точность вычислений, однако сложен в реализации.

2) Алгоритм Варнока

Данный алгоритм работает в пространстве изображений. 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 подокна не станет

достаточно простым для визуализации или его размер не достигнет требуемого предела разрешения.

Конкретная реализация алгоритма Варнока зависит от метода разбиения окна и от критериев, используемых для того, чтобы решить, является ли содержимое окна достаточно простым. В оригинальной версии алгоритма Варнока каждое окно разбивается на четыре одинаковых подокна. Окно является пустым, когда все многоугольники сцены являются внешними по отношению к этому окну. Пределом разбиения является размер окна в 1 пиксель. Тогда определяем глубину каждого из рассматриваемых многоугольников в этой точке и изображаем точку многоугольника, наиболее близкого к пользователю.

Данный алгоритм прост в реализации, однако возможны большие затраты по времени, если рассматривается область с большим количеством информации.

3) Алгоритм, использующий z-буфер

Данный алгоритм работает в пространстве изображений. В нем используются буфер кадра и z-буфер. Буфер кадра используется для запоминания интенсивности каждого пиксела в пространстве изображения, в то время как z-буфер — это отдельный буфер глубины, используемый для запоминания глубины (координаты z) каждого видимого пиксела в пространстве изображения.

В процессе работы значение z каждого нового пикселя, который нужно занести в буфер кадра, сравнивается с глубиной того пиксела, который уже занесен в z-буфер. Если это сравнение показывает, что новый пиксел расположен впереди пиксела, находящегося в буфере кадра, то новый пиксел заносится в этот буфер, и производится корректировка z-буфера новым значением z. Если же сравнение дает противоположный результат, то никаких действий не производится.

Главное преимущество алгоритма — простота. Кроме того, этот алгоритм решает задачу об удалении невидимых поверхностей и делает тривиальной визуализацию пересечений сложных поверхностей. Однако, несмотря на свою простоту, алгоритм требует использование большого объема памяти. Кроме того, недостаток алгоритма заключается еще в трудоемкости реализации эффектов прозрачности.

4) Алгоритм обратной трассировки лучей

Основная идея, лежащая в основе трассировки лучей, заключается в том, что наблюдатель видит любой объект посредством испускаемого неким источником света, который падает на объект и затем каким-то путем доходит до наблюдателя согласно законам оптики. Однако не все лучи света доходят до наблюдателя, что делает процесс отслеживания этих лучей вычислительно неэффективным. Поэтому лучше отслеживать лучи в обратном направлении, то есть от наблюдателя к объекту, причем испуская лучи от точки наблюдателя через каждый пиксел экрана.

Для данной работы был выбран алгоритм обратной трассировки лучей, так как он, несмотря на малую производительность из-за большого количества вычислений, позволяет получить реалистичное изображение с учетом оптических явлений, таких как отражение и преломление.

1.3 Выбор модели освещения

Построение реалистических изображений включает в себя как физические, так и психологические процессы. В частности, при визуализации сцены учитываются законы оптики: отражение, преломление, поглощение света и т. д. Поэтому необходимо выбрать модель освещения для построения изображения.

1) Модель Ламберта

Модель Ламберта описывает диффузное освещение, то есть свет точечного источника диффузно отражается от идеальной поверхности по закону косинусов Ламберта: интенсивность отраженного света пропорциональна косинусу угла между направлением света L и нормалью к поверхности N (см. рисунок 1), то есть

$$I = I_l k_d \cos(\theta), \tag{7}$$

где I - интенсивность отраженного света; I_l - интенсивность точечного источника в направлении L; k_d - коэффициент диффузного отражения; θ - угол между направлением света L и нормалью к поверхности n, $0 < \theta < \frac{\pi}{2}$ (рисунок 1).

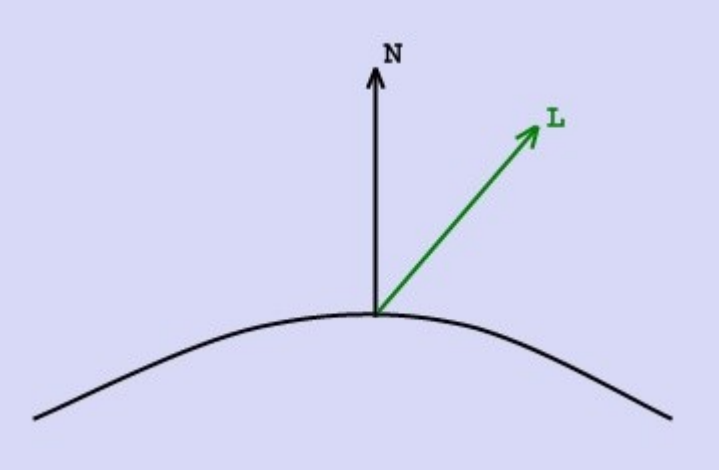

Рисунок 1. Модель Ламберта

2) Модель Фонга

Модель Фонга — это классическая модель освещения, представляющая собой совокупность диффузной и зеркальной составляющих. Работает модель таким образом, что кроме равномерного освещения на материале могут появляться блики. Местонахождение блика на объекте определяется из закона равенства углов падения и отражения. Чем ближе наблюдатель к углам отражения, тем выше яркость соответствующей точки.

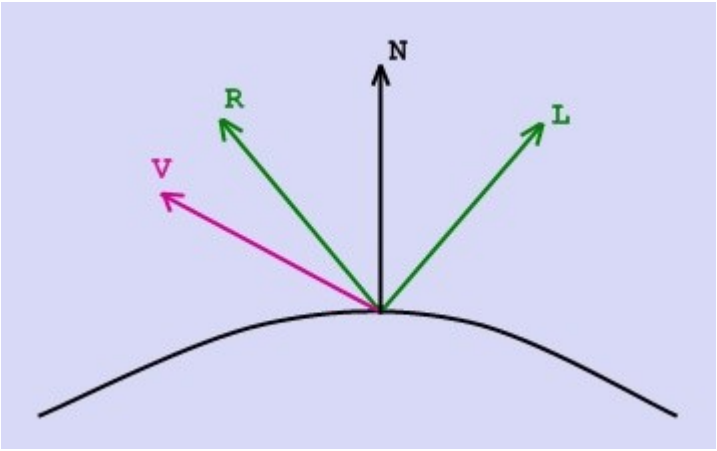

Рисунок 2. Модель Фонга

Зеркальная составляющая описывается выражением

$$I_s = I_l w(i,\lambda)\cos^n \alpha, \tag{8}$$

где $w(i,\lambda)$ - кривая отражения, представляющая отношение зеркально отраженного света к падающему как функцию угла падения i и длины волны λ; n - степень, аппроксимирующая пространственное распределение зеркально отраженного света; α - угол между вектором направления отраженного луча R и вектором направления на наблюдателя V (рисунок 2). Функция $w(i,\lambda)$ довольно

сложна, поэтому обычно ее заменяют константой, которая либо выбирается из эстетических соображений, либо определяется экспериментально.

3) Глобальная модель освещения Уиттеда с трассировкой лучей

Рисунок 3. Глобальная модель освещения Уиттеда

В данной модели луч V, падающий на поверхность в точке Q, отражается в направлении r и, если поверхность прозрачна, преломляется в направлении p (рисунок 3). Тогда наблюдаемая интенсивность *I* выражается формулой

$$I = k_a I_a + k_d \sum_j I_{l_j}(n; L_j) + k_s \sum_j I_{l_j}(S; R_j)^n + k_s I_s + k_t I_t, \qquad (9)$$

где k_a, k_d, k_s - коэффициенты рассеянного, диффузного и зеркального отражения; k_t - коэффициенты пропускания; I_t - интенсивность света, падающего падающего в точку Q по направлению p; I_s - интенсивность зеркально отраженного света, падающего в направлении r и отраженного к наблюдателю в точке Q; n -

нормаль к поверхности в точке Q; L_j - направление на j-й источник света, S и R - локальные векторы наблюдения отражения, n - степень пространственного распределения Фонга для зеркального отражения из формулы (8).

Для вычисления интенсивности преломленного света можно применить закон Бугера-Ламберта-Бера для учета поглощения энергии при прохождении луча света через вещество:

$$I_t = I_0 e^{-k_\lambda l}, \tag{10}$$

где l - толщина слоя вещества, через который прошел свет; I_0 - интенсивность света на входе в вещество; k_λ - показатель поглощения.

Для данной работы была выбрана модель освещения Уиттеда, так как она позволяет учесть такие оптические явления, как отражение и преломление света, что позволит нам получать более реалистичное изображения, чем модели Ламберта и Фонга.

2. Конструкторская часть

2.1 Общие сведения

Положение объектов сцены в пространстве описывается с помощью мировой системы координат, при этом ось Oy должна направлена вверх. Частота смены кадров во время работы программы — 60 кадров в секунду (то есть каждый кадр длится приблизительно 17 миллисекунд).

Общий алгоритм работы программы:

1) Задать объекты сцены (камера, пол, источник света, слайм);

2) Для каждого кадра:

а) Если пользователь изменил характеристики объектов сцены, установить новые характеристики объектов;

б) Если пользователь пытается захватить точку слайма, найти и запомнить ее;

в) Определить силы, действующие на точки слайма, не захваченные пользователем;

г) Изменить скорости точек слайма, не захваченных пользователем, в соответствии с действующими на них силами;

д) Переместить точки слайма, не захваченные пользователем, в соответствии с их скоростями;

е) Вывести изображение на дисплей

2.2 Генерация слайма в начале работы программы

Для хранения информации о слайме используется линейный односвязный список граней тела и линейный список точек масс.

Каждая точка массы хранит в себе информацию о ее положении в пространстве, массе и векторе скорости. Помимо этого точка массы содержит в себе массив точек, с которыми связана данная точка с помощью пружин, и значение коэффициента жесткости этих пружин.

Каждая грань тела хранит в себе массив из трех точек, образующих плоскость данной грани, и четыре коэффициента уравнения плоскости $ax+by+cz+d=0$ грани.

Чтобы получить внешние точки слайма, программа производит рекурсивное деление исходных треугольных граней тела на новые треугольники посредством бисекции, то есть деления сторон треугольников пополам. До разбиения граней шар представляет собой икосаэдр (правильный многогранник с 20 гранями). Таким образом, поверхность слайма будет представлять собой трехмерную фигуру с треугольными гранями (см. рисунок 4).

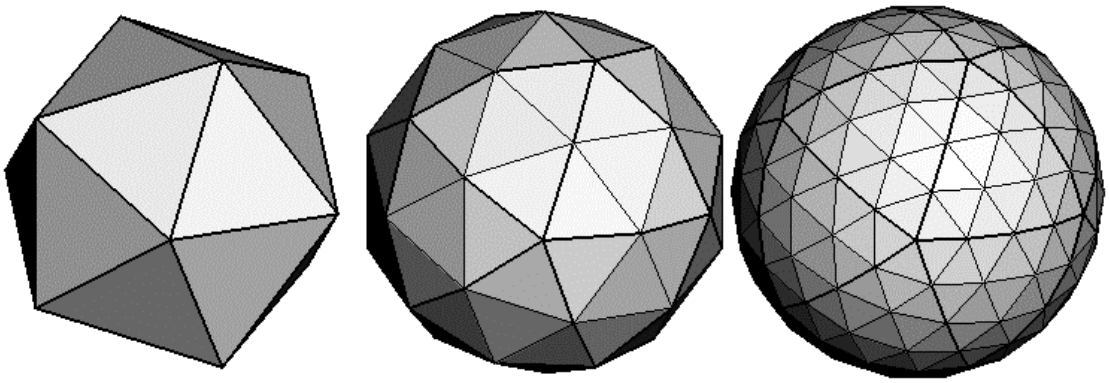

Рисунок 4. Процесс разбиения поверхности шара

После получения внешних точек программа производит поиск внутренних точек тела. Для этого из центра шара проводятся радиусы ко всем внешним точкам шара. Каждый построенный отрезок разбивается на одинаковое количество равных интервалов. Полученные точки, включая центр шара, отмечаются как внутренние точки масс шара.

Масса каждой точки шара равна значению массы тела, деленной на количество этих точек. Начальная скорость точек масс равна нулю. После разбиения точки масс соединяются пружинами с соседними точками.

2.3 Расчет сил, скоростей и координат точек слайма

Векторы сил, действующих на точки слайма со стороны пружин, определяются по формуле (3). Помимо этого учитывается векторы сил тяжести, действующих на точки, которые определяются по формуле

$$F_T = mg, \tag{12}$$

где m - масса точки; g - вектор ускорения свободного падения, направленный вниз. В нашей программе $|g| = 9.81 \frac{м}{с^2}$.

Если точка слайма имеет координату $y = 0$, и на точку действует сила реакции опоры, то вычисляется сила трения, действующая на эту точку, по формуле

$$F_{тр} = \mu N, \tag{13}$$

где μ - коэффициент трения; N – сила реакции опоры. Направление вектора силы трения должно быть противоположно направлению вектора скорости точки, который, в свою очередь, параллелен плоскости $y = 0$.

Сила реакции опоры вычисляется как проекция на ось Oy равнодействующей сил тяжести и упругости, действующих на эту точку. Если результат неотрицательный, то на точку не действует сила реакции опоры. В ином случае результат умножается на -1.

Как только для каждой точки тела были определены равнодействующие всех сил, программа вычисляет новые векторы скоростей этих точек и перемещает их. Для этого для каждой точки вычисляется вектор ускорения по второму закону Ньютона:

$$a(a_x, a_y, a_z) = \frac{F_p(F_x, F_y, F_z)}{m}, \tag{14}$$

где $F_p(F_x, F_y, F_z)$ - вектор равнодействующей всех сил, действующих на точку. Новый вектор скорости точки вычисляется по формуле

$$v(v_x, v_y, v_z) = v_0(v_{0x}, v_{0y}, v_{0z}) + a(a_x, a_y, a_z)\, t, \tag{15}$$

где $v_0(v_{0x}, v_{0y}, v_{0z})$ - вектор начальной скорости точки; t - промежуток времени.

Если скорость направлена вниз ($v_y \le 0$), и координата y точки равна 0, то компонента вектора скорости по оси Оу v_y обнуляется.

Наконец, новое положение точки тела в пространстве вычисляется по формуле

$$P(x, y, z) = P_0(x_0, y_0, z_0) + v(v_x, v_y, v_z)\, t, \tag{16}$$

где $P_0(x_0, y_0, z_0)$ - радиус-вектор начального положения точки в пространстве.

После того, как были изменены положения всех точек тела, программа должна произвести перерасчет уравнений плоскостей, описывающих поверхность слайма.

2.4 Захват пользователем вершины слайма

Если пользователь хочет схватить вершину слайма, программа считывает координаты курсора мыши на дисплее, а затем испускает луч от точки наблюдателя через данный пиксель. Далее, находится ближайшая точка пересечения этого луча с поверхностью слайма. Если точка пересечения нашлась, то в пересеченной грани находится ближайшая к данной точке пересечения вершина. Перемещение захваченной вершины осуществляется в плоскости, параллельной плоскости экрана и проходящей через эту вершину. При этом скорость захваченной вершины обнуляется.

2.5 Алгоритм обратной трассировки лучей

При моделировании слайма нужно учитывать такие оптические явления, как отражение и преломление света. С этой задачей помогает справляться алгоритм обратной трассировки лучей.

Пусть есть камера и экран, находящийся на некотором расстоянии от нее. Камера должна смотреть в центр экрана под прямым углом. Через каждый пиксел по очереди из камеры строится первичный луч и находится ближайшая к экрану точка пересечения этого луча с объектами сцены. Из точки пересечения строятся теневые лучи ко всем источникам света для определения их видимости из этой точки. Далее, по выбранной модели освещения выбирается цвет. Если луч не пересекает объекты сцены, то пиксел закрашивается в цвет фона.

Однако нам нужно учесть такие явления, как отражение и преломление света. Поэтому из точки пересечения строятся вторичные лучи по оптическим законам, если пересеченный объект зеркальный или преломляющий. Новые лучи рекурсивно трассируются дальше, а точки их соударений формируют дерево пересечений. По завершении рекурсии интенсивность пиксела рассчитывается путем суммирования значений интенсивности в узлах дерева с использованием модели освещения. В нашей программе интенсивность пиксела рассчитывается по формулам (9) и (10). Рекурсия завершается, если достигнуто максимальное число вторичных лучей, значение текущей интенсивности незначительно или нет пересечений с объектами сцены.

Для ускорения работы алгоритма обратной трассировки лучей слайм содержит информацию о сферической объемлющей оболочке, радиус и центр которой будут обновляться при изменении положения тела в пространстве. Центр оболочки определяется как центр масс всех внешних точек слайма, а радиус – как длина отрезка, соединяющего центр оболочки с наиболее отдаленной от нее внешней точкой слайма.

2.6 Управление камерой

Программа предоставляет пользователю возможность управления камерой. Камеру можно как перемещать взад и вперед вдоль вектора взгляда, так и поворачивать ее вокруг оси, проходящей через точку наблюдателя параллельно оси Oy.

3. Технологическая часть

3.1 Выбор языка программирования и среды разработки

Для выполнения данной работы был выбран высокоуровневый язык программирования C++. Данный выбор обусловлен следующими факторами:

1) C++ поддерживает технологию объектно-ориентированного программирования, которая позволяет легко модифицировать программу и эффективно организовывать взаимодействия между объектами;

2) C++ обладает очень высокими показателями вычислительной производительности, что также обеспечивает достойную скорость исполнения кода;

3) Для C++ было создано множество библиотек, которые могут быть использованы, в частности, для математических расчетов.

В качестве среды разработки была выбрана Visual Studio Code по следующим причинам:

1) Данная среда разработки бесплатна;

2) В данной среде присутствуют встроенный отладчик, инструменты для работы с Git и средства навигации по коду, автодополнения кода и контекстной подсказки;

3) Имеется возможность расширить любой функционал, включая компиляцию и отладку.

3.2 Структура классов. Диаграмма классов

Для решения данной задачи используются следующие классы:

1) MainWindow — оконный интерфейс, через который пользователь взаимодействует с программой;

2) Plot — содержит в себе информацию о дисплее и осуществляет вывод изображения на экран;

3) Scene — совокупность объектов сцены;

4) Timer — используется для отсчета времени каждого кадра;

5) Object — абстрактный класс объекта сцены;

6) Camera — камера, заданный точкой положения в пространстве и направляющим вектором;

7) LightSource — точечный источник света;

8) Slime — моделируемый объект (слайм);

9) Floor — пол, имеющий структуру;

10) Texture — абстрактный класс обработчика текстур, осуществляющего загрузку текстур их наложение на объекты;

11) FloorTexture — обработчик текстур для пола;

12) RGBColor — работа с цветом в формате RGB;

13) PlaneFace — плоская поверхность объекта;

14) SphereCover — сферическая оболочка слайма, нужна для оптимизации алгоритма трассировки лучей по времени;

15) MassPoint — точка масс слайма;

16) Point — геометрическая точка;

17) Vector3d — трехмерный вектор;

18) Ray — трехмерный луч, заданный точкой начала и направляющим вектором.

Диаграмма классов представлена на рисунке 5.

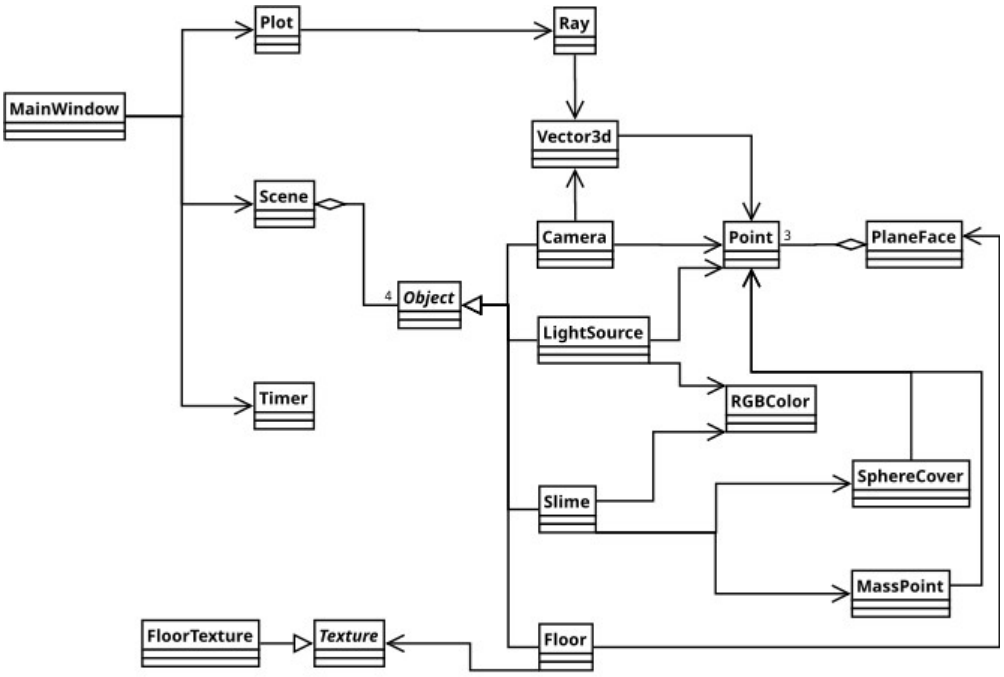

Рисунок 5. Диаграмма классов

3.3 Интерфейс программы

Интерфейс программы представлен на рисунке 6.

Цвет слайма задается с помощью ползунков синего, зеленого и красного цветов.

Степень прозрачности объекта задается с помощью двух параметров: коэффициента пропускания k_t из формулы (9) и показателя поглощения k_λ из формулы (10).

Кнопки со стрелками предназначены для управления камерой: стрелки вверх и вниз — перемещение камеры вперед и назад соответственно, стрелки вправо и влево — поворот камеры вправо и влево вокруг оси, параллельной оси Oy и проходящей через точку наблюдателя. Также камерой можно управлять с помощью клавиатурных клавиш со стрелками.

В правой стороне окна расположен экран, на котором будет выводится изображение. Через него пользователь с помощью мыши может взаимодействовать со слаймом.

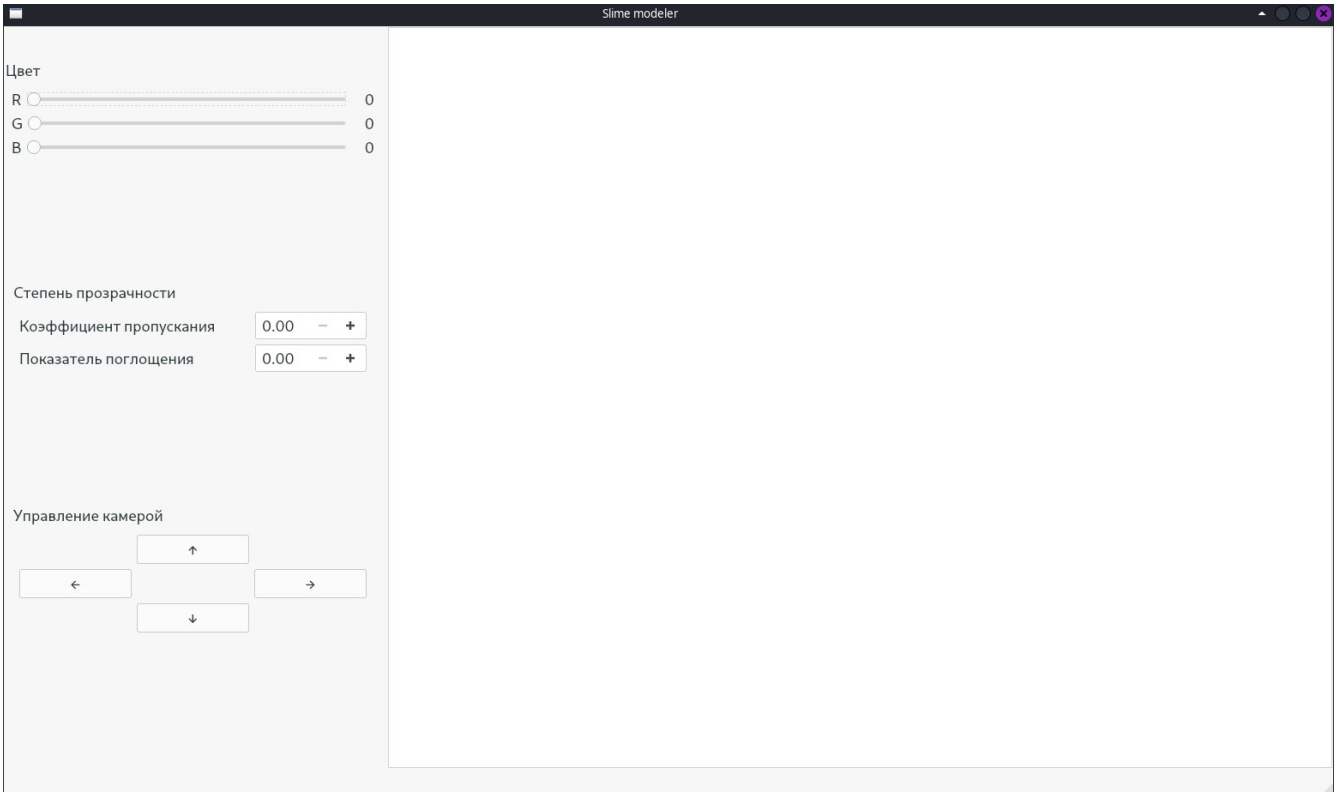

Рисунок 6. Интерфейс программы

Заключение

Во время выполнения практической работы были проанализированы основные способы представления и задания трехмерных моделей, а также рассмотрены и основные алгоритмы удаления невидимых линий и методы

освещения. Также были проанализированы достоинства и недостатки представленных алгоритмов и выбраны наиболее оптимальные для решения поставленной задачи. Был решен вопрос о методе физического моделирования рассматриваемого объекта. Кроме того, были закреплены знания и навыки, полученные в ходе изучения курса компьютерной графики. Был разработан интерфейс для взаимодействия пользователя с программой.

Список использованных источников

1. Роджерс Д., Алгоритмические основы машинной графики: пер. с англ.— М.: Мир, 1989.— 512 с.: ил.

2. Божко А.Н., Жук Д.М., Маничев В.Б., Компьютерная графика: Учеб. пособие для вузов. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2007. - 392 с.: ил. - (Информатика в техническом университете.)

3. Nealen A., Müller M., Keiser R., Boxerman E., Carlson M., Physically Based Deformable Models in Computer Graphics [Электронный ресурс] - Режим доступа: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.124.4664 (дата обращения 06.07.22)

4. Задорожный А.Г., Компьютерная графика: введение в трассировку лучей: учебное пособие — Новосибирск: Изд-во НГТУ, 2021. — 64 с.

5. Сивухин Д. В. Общий курс физики. — Издание 3-е, стереотипное. — М.: Физматлит, МФТИ, 2002. — Т. IV. Оптика. — 792 с.