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/ПРОИЗВОДСТВЕННОЙ) ПРАКТИКЕ

Студент Брянская Екатерина Вадимовна

фамилия, имя, отчество

Группа ИУ7-42Б

Тип практики учебная

Название предприятия _____

Студент _____ Брянская Е.В.

подпись, дата *фамилия, и.о.*

Руководитель практики _____

подпись, дата *фамилия, и.о.*

Оценка _____

2020 г.

Оглавление

Введение

В современном мире роль компьютерной графики огромна, она применима везде, где нужно создание и обработка изображений и каких-либо цифровых данных. Спецэффекты в фильмах, реклама, компьютерные игры – всё это проявления компьютерной графики, которая используется повсеместно. Она применяется в большинстве инженерных и научных дисциплинах для передачи информации, её наглядного восприятия.

Одной из основных задач компьютерной графики является создание наиболее приближенного к реальной жизни изображения. В процессе работы необходимо учитывать свойства не только самого объекта, но и окружающих тел, среды. Существует много алгоритмов, которые решают данную задачу, но зачастую они достаточно затратны как по времени, так и по используемой памяти. Поэтому при моделировании какого-либо объекта или процесса необходимо корректно подбирать алгоритм, исходя из конкретной цели.

Цель данной работы - разработать программу моделирования работы песочных часов.

Измерительное время пользователь выбирает из предлагаемых вариантов. Программа даёт возможность задания в соответствующем текстовом поле положения источника освещения, камеры, её перемещения вокруг объекта. Пользователь может начать повторный отсчет по истечении времени работы прибора. Программа должна предусматривать построение теней часов и песка в них. При разработке программы необходимо учитывать прозрачность стекла и блики от источников освещения.

Для достижения поставленной цели необходимо решить следующие задачи:

1) описать из каких объектов состоит сцена, как они представляются и как будут заданы в программе;
2) проанализировать алгоритмы для визуализации трёхмерной сцены, при необходимости рассмотреть модификации, и обосновать выбор конкретного алгоритма;

3) разработать физическую модель песочных часов, наиболее приближенную к реальности;
4) реализовать выбранные алгоритмы;
5) реализовать интерфейс, отвечающий заданным требованиям и предоставляющий пользователю заявленные возможности.

1. Аналитический раздел

1.1 Объекты сцены

Сцена состоит из следующих объектов:

- ограничивающая плоскость – расположена параллельно плоскости oxz;
  Песочные часы – располагаются на ограничивающей плоскости. В них можно выделить следующие составляющие:
  1. стеклянная часть – её свойства необходимо учитывать при изображении теней, бликов, моделировании песка;
  2. песок – совокупность мелких песчинок, которая приходит в движение при отсчете времени;
  3. непрозрачные подставки (сверху и снизу) – представляют собой параллелепипеды, основания которых параллельны ограничивающей плоскости, а боковые ребра перпендикулярны ей;
- источник освещения – предполагается, что источник находится в бесконечности. Начальное положение источника указывается по умолчанию, но пользователь может его поменять;
- камера – начальное положение указывается по умолчанию, но пользователь может его изменить, указав координаты в соответствующих текстовых полях.

Непрозрачные подставки и стеклянная часть часов будут описываться с помощью каркасной модели.

Песок «разбивается» на части:

- песок, который ещё не пересыпался, и песок, который уже пересыпался – будут представляться в виде полигональной сетки;
- движущиеся частички – представляются в виде каркаса треугольных пирамид;

Использование объёмной модели в данной задаче не рационально в силу наличия в ней излишней информации для данных объектов и затрат по памяти.

1.2 Анализ алгоритмов удаления невидимых линий и поверхностей

Задача удаления невидимых линий и поверхностей является достаточно сложной, поэтому существует много алгоритмов её решения. При этом универсального решения, которое можно считать наилучшим для любой задачи, нет, поэтом в каждом конкретном случае необходимо обосновывать выбор алгоритма, исходя из цели моделирования.

В рамках данной практической работы наиболее сложной проблемой является достижение реалистичной визуализации процесса пересыпания песка из одной части стеклянной колбы в другую. Для этого нужно добиться плавности и равномерности движения частиц, так как именно эти факторы будут влиять на качество анимации. Ещё одним важным критерием выбора алгоритма удаления невидимых линий является быстродействие.

Рассмотрим некоторые из основных алгоритмов удаления невидимых линий и поверхностей, проанализируем их и выберем наиболее подходящий.

1.2.1 Алгоритм Робертса

Плюсом данного алгоритма является то, что используемые в нём математические методы являются достаточно мощными и точными, а сам алгоритм прост для понимания.

На первом этапе удаляются те рёбра или грани, которые экранируются самим телом, далее те, что экранируются другими телами сцены. И если тела, связаны отношением взаимного «протыкания», то удаляются невидимые линии пересечения тел.

К минусам алгоритма следует отнести тот факт, что вычислительная ёмкость растёт как квадрат числа объектов. То есть при большом количестве объектов на сцене алгоритм будет работать достаточно медленно, однако существуют различные способы его оптимизации, например, с использованием предварительной приоритетной сортировки вдоль оси z[1].

1.2.2 Алгоритм, использующий z-буфер

К достоинствам можно отнести то, что это один из простейших алгоритмов удаления невидимых линий и поверхностей. Также он позволяет визуализировать пересечения этих поверхностей. Элементы сцены не нужно сортировать, следовательно, на это не тратится время.

Сцена может быть любой сложности, а так как размеры изображения ограничены размером экрана, то трудоёмкость линейно зависит от числа поверхностей.

Но у этого алгоритма есть и существенные недостатки: большой объём требуемой памяти и трудоёмкость реализации таких эффектов, как прозрачность и освещение, правда существуют и специальные модификации, направленные на разрешение проблемы с их визуализацией.

1.2.3 Алгоритм обратной трассировки лучей

В данном алгоритме считается, что точка зрения находится в бесконечности на положительной полуоси z и поэтому все световые лучи параллельны этой оси. Траектория каждого луча отслеживается, чтобы определить, какие именно объекты сцены, если таковые существуют, пересекаются с данным лучом.

Этот алгоритм позволяет получать такие эффекты как отражение, преломление и т.д. Другими словами, изображение становится более реалистичным. Расчёт отдельной точки изображения производится независимо от других, поэтому в этом алгоритме поддерживается высокая степень параллельности вычислений.

К недостаткам алгоритма обратной трассировки лучей следует отнести высокую вычислительную стоимость расчетов. Каждая точка изображения требует множества вычислительных операций. Очень много времени тратится на поиск пересечений, что зачастую тормозит работу алгоритма[1].

1.2.4 Алгоритм Варнока

Единой версии этого алгоритма не существует. Но все модификации основываются на рекурсивном разбиении окна изображения.

Для каждого окна определяются многоугольники, которые связаны с ним, и те, у которых легко определить видимость, изображаются. Если нет, то разбиение на подокна продолжается до тех пор, пока либо нельзя будет принять однозначное решение, либо размер окна не станет равным одному пикселю.

На каждом этапе разбиения осуществляется определение расположения многоугольников относительно текущего окна.

Насколько быстро будет работать данный алгоритм, зависит от сложности сцены. Так как было решено использовать полигональную сетку, то это может затормозить выполнение алгоритма.

1.2.5 Вывод

Для наибольшей наглядности была составлена Таблица 1:

Таблица 1 - Сравнение алгоритмов

	Алгоритм Робертса	Алгоритм z-буфера	Алгоритм обратной трассировки лучей	Алгоритм Варнока
Пространство, в котором работает алгоритм	Объектное пространство	Пространство изображения	Пространство изображения	Пространство изображения
Сложность, N – количество граней; C –	$O(N^2)$	O(CN)	O(CN)	O(CN)

количество пикселей				
Эффективность для сложных сцен	Низкая	Высокая	Низкая	Средняя
Сложность реализации	Высокая	Низкая	Средняя	Средняя

Проанализировав алгоритмы удаления невидимых линий и поверхностей, можно сделать вывод, что наиболее предпочтительным для данной задачи является алгоритм с использованием z-буфера, так как он позволяет работать со сценой любой сложности и обеспечивает её обработку за меньшее время. Кроме того, при работе с освещением и закраской z-буфер легко подстраивается под модификации.

1.3 Анализ алгоритмов закраски

1.3.1 Простая закраска

Один из самых быстрых алгоритмов. В его основе лежит закон Ламберта, который говорит о том, что интенсивность отражённого света пропорциональна косинусу угла между направлением света и нормалью к поверхности.

Поверхность предметов, которая была изображена с помощью этого алгоритма, выглядит визуально матовой и блеклой. Это происходит потому, что вся грань закрашивается с постоянной интенсивностью [1].

1.3.2 Закраска методом Гуро

Благодаря этому алгоритму можно получить сглаженное изображение. Сначала определяется интенсивность вершин, а затем с помощью билинейной интерполяции вычисляется интенсивность соответствующего пикселя.

Недостаток алгоритма – появление полос Маха, которое можно исправить, если увеличить число обрабатываемых граней, но это приводит к замедлению процесса визуализации [3].

Закраску Гуро лучше всего использовать с простой моделью с диффузным отражением, так получается более реалистичное изображение [1].

1.3.3 Закраска Фонга

В закраске по Фонгу интерполирование происходит по вектору нормали, в отличие от закраски по Гуро, где используется значение интенсивности. Тем самым изображение становится более реалистичным, а зеркальные блики выглядят более правдоподобно.

Но у этого метода есть существенный недостаток: большие вычислительные затраты, так как сначала производится работа с нормалями, а затем по ним находятся соответствующие интенсивности [1].

1.3.4 Вывод

Так как в текущей задаче будут использоваться полигональные сетки и, желательно, чтобы были сглажены границы многоугольников, то лучше всего подойдёт алгоритм закраски по Гуро.

1.4 Анализ алгоритмов построения теней

Для придания естественности необходимо учитывать тени от объектов. Из всех алгоритмов, выполняющих данную задачу, нужно выбрать наиболее подходящий для реализации текущей трёхмерной сцены.

В общем случае построение теней производится в два этапа удаления невидимых поверхностей (относительно положения наблюдателя и каждого источника освещения). Поэтому построение теней лучше всего внедрить в выбранный алгоритм удаления невидимых поверхностей [1].

При построении теней может использоваться матрица теней, в которой помечаются затеняемые многоугольники и многоугольники, отбрасывающие тень.

Определять видимость объектов сцены можно с помощью алгоритмов z-буфера, Робертса и других. Удобнее работать с алгоритмом z-буфера, так как он позволит сделать это быстрее.

Важно помнить тот факт, что положение теней зависит только от расположения источника освещения, а не от наблюдателя.

1.5 Анализ моделей освещения

Освещение тоже играет не последнюю роль в данной задаче, а в силу того, что будет моделироваться стекло, нужно подобрать соответствующую модель.

1.5.1 Модель Ламберта

Является одной из самых простых моделей освещения (Рисунок 1). Она моделирует идеальное диффузное освещение, сама поверхность выглядит одинаково яркой со всех направлений.

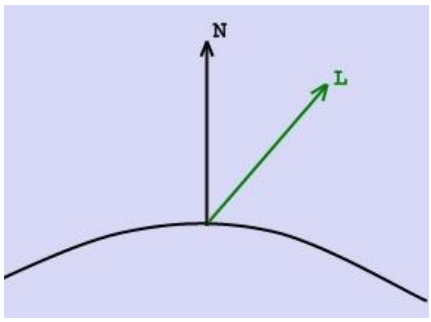

Рисунок 1 – Модель Ламберта

$I = k_d(\vec{N}, \vec{L})$, где

k_d – коэффициент диффузного освещения

$\vec{N}$ – вектор нормали к поверхности в данной точке

$\vec{L}$ – направление из точки на источник

Считается, что при попадании на поверхность свет равномерно рассеивается по всем направлениям. В основе расчетов лежит закон Ламберта.

1.5.2 Модель Фонга

Представляет классическую модель освещения, комбинируя диффузную составляющую (модель Ламберта) и зеркальную составляющую. Таким образом, на объекте может появиться ещё и блик, что придаёт больше реалистичности изображению. Нахождение блика на объекте определяется из закона равенства углов падения и отражения. Если наблюдатель находится вблизи углов отражения, то яркость соответствующей точки повышается [6].

Падающий и отраженный лучи лежат в одной плоскости с нормалью к отражающей поверхности в точке падения, и она делит угол между лучами на две равные части (Рисунок 2), где $\overrightarrow{V}$ – направление на наблюдателя, $\overrightarrow{L}$ – направление на источник, $\overrightarrow{R}$ – отраженный луч.

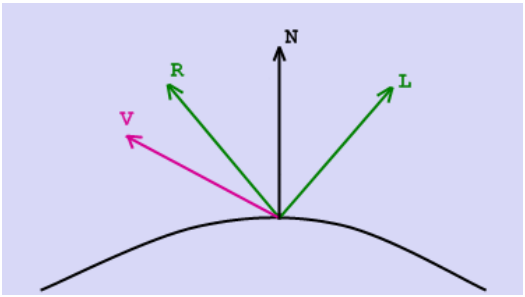

Рисунок 2 - Модель Фонга

Эта модель учитывает фоновую, рассеянную компоненты освещения и глянцевые блики.

Для удобства все векторы считаются единичными, чтобы сказать, что косинус угла между ними совпадает со скалярным произведением.

Интенсивность по модели Фонга рассчитывается следующим образом:

$I = k_a I_a + k_d(\vec{N}, \vec{L}) + k_S(\vec{N}, \vec{V})^p$, где

k_a – коэффициент фонового освещения;

I_a – мощность фонового освещения (заранее задаётся для всей сцены);

k_d – коэффициент диффузного освещения;

$\vec{N}$ – вектор нормали к поверхности в данной точке;

$\vec{L}$ – направление из точки на источник;

k_S – коэффициент зеркального освещения;

$\vec{V}$ – направление на наблюдателя;

p – коэффициент блеска.

При фиксированном положении поверхности относительно источника освещения фоновую и рассеянную составляющие можно вычислить только один раз, так как они не зависят от положения наблюдателя. А зеркальная компонента зависит, поэтому её надо вычислять каждый раз, когда меняется положение наблюдателя.

Модель Фонга учитывает только свойства текущей точки и источника освещения, а такие эффекты как рассеивание, отражение от других тел игнорируются.

1.5.3 Модель Блинна-Фонга

Есть большое сходство с моделью Фонга, только она исключает расчёт отражённого луча, что сильно упрощает вычисления, тем самым, экономится время работы. Но существенной разницы нет.

В этой модели используется медианный вектор, который является единичным и находится посередине между вектором, указывающим направление обзора, и вектором направления освещения.

Чем ближе такой вектор к нормали поверхности, тем больше будет вклад зеркальной компоненты.

Благодаря тому, что измеряется угол между нормалью и медианным вектором (а не между вектором направления наблюдения и вектором отражения, как в модели Фонга), не будет проблемы с резкой границей области зеркального отражения [7].

1.5.4 Вывод

В качестве подходящей модели освещения была выбрана модель Фонга, она позволяет изобразить более реалистичное изображение, чем модель Ламберта, так как учитывает зеркальную составляющую. Модель Блинна-Фонга тоже позволяет создать правдоподобное изображение, но требует дополнительных расчётов, что может сказаться на скорости работы.

1.6 Физическая модель поведения объектов сцены

Как было уже упомянуто, основной целью является моделирование реалистичного процесса работы песочных часов. Вместе с этим большое внимание уделяется эффективности отобранных алгоритмов. Задача непростая, так как песок будет состоять из большого числа частиц, каждая из которых обладает скоростью, размером и т.д. и которые должны равномерно пересыпаться.

В начальном положении часов весь песок находится в верхней части колбы. Далее с течением времени он пересыпается в нижнюю. То есть, объём песка в верхней части должен равномерно уменьшаться, причём должны соблюдаться соответствующие физические законы. Аналогичная картина будет наблюдаться в нижней части, только количество песка должно увеличиваться. При моделировании теней необходимо учитывать тот факт, что и движущийся песок отбрасывает тень, доля анимации должна присутствовать и здесь. Время пересыпания должно соответствовать реальному.

Песок будет разделён на три части: верхнюю, пересыпающуюся и нижнюю. На Рисунке 3 показаны песочные часы в действии, можно различить все выделенные для работы части.

Рисунок 3 - Песочные часы

Верхняя часть:

Разбивается на две составляющие: повторяющая форму часов боковая поверхность и образующая выемку в верхней части всего объёма. Обе компоненты задаются с помощью каркасной модели.

Боковая поверхность представляет собой перевёрнутую усечённую пирамиду с маленьким основанием в месте соприкосновения верхней и нижней части часов.

Поверхность, образующая выемку в песке, задаётся однородной полигональной сеткой, определяемой уравнением:

$$y(x,z) = -\frac{e^{-\frac{(mx^2+mz^2)^2}{10}}}{k},$$

где коэффициенты m, k зависят от времени.

С течением времени, когда глубина выемки достигнет своего максимума, вся система верхней части песка начнёт “проседать”, то есть уменьшаться в объёме. Другими словами, высота усечённой пирамиды будет равномерно уменьшаться, и поверхность с выемкой будет вместе с ней опускаться вниз. Так будет

продолжаться до тех пор, пока всё содержимое часов не окажется в нижней части.

Нижняя часть:

Процессы и представление песка аналогичны тем, что были описаны в верхней части, только вместо выемки будет наблюдаться растущая со временем горка из песка, поверхность которой описывается следующим образом:

$$y(x,z) = \frac{e^{-\frac{(mx^2+mz^2)^2}{10}}}{k},$$

где коэффициенты m, k зависят от времени.

С течением времени, горка расти перестанет, но будет продолжать увеличиваться объём пересыпанного песка, и вся система нижней части будет равномерно подниматься (увеличивается высота усеченной пирамиды, которая описывает боковую поверхность уже нижней части песка, и поднимается поверхность с горкой). Процесс продолжается до тех пор, пока не пересыплется весь песок.

Время, которое затрачивается на каждый из процессов, совпадает с временными затратами, описанными в предыдущем пункте.

Пересыпающаяся часть:

Как было упомянуто выше, эта часть представляет собой совокупность частиц, которые имеют форму треугольных пирамид, описанных с помощью каркасной модели.

Все частицы будут падать с ускорением, и их скорость будет изменяться по закону:

$$v = at,$$

где v – скорость, a – ускорение, t – время.

И соответственно ордината будет меняться в соответствии с формулой:

$$y = y_0 - \frac{at^2}{2},$$

где y_0, y – начальная и текущая ординаты.

1.7 Вывод из аналитической части

В данном разделе были рассмотрены алгоритмы удаления невидимых линий и поверхностей, закраски, алгоритмы построения теней, проанализирована физическая модель песочных часов.

В результате анализа для моделирования был выбран алгоритм z-буфера (удаление невидимых линий и поверхностей), метод Гуро (закраска), построение теней будет осуществляться с помощью z-буфера, а песок будет представляться в виде системы частиц, которая подчиняется соответствующим физическим законам.

Входным данным является измерительное время, которое пользователь выбирает из предлагаемого списка. Пользователь может поменять расположение камеры, источника освещения. Все изменения, которые могут быть внесены пользователем, должны быть учтены при разработке графического интерфейса программного продукта.

2. Конструкторский раздел

В данном разделе описаны используемые методы и алгоритмы.

2.1 Возможности программы

Программа предоставляет возможность:

- выбора из предлагаемого списка измерительного времени;
- задания в соответствующем текстовом поле положения источника освещения и камеры;
- перемещения камеры вокруг объекта;
- повторного отсчета по истечении времени работы прибора.

2.2 Общий алгоритм работы программы

Общий алгоритм выглядит следующим образом (Рисунок 4):

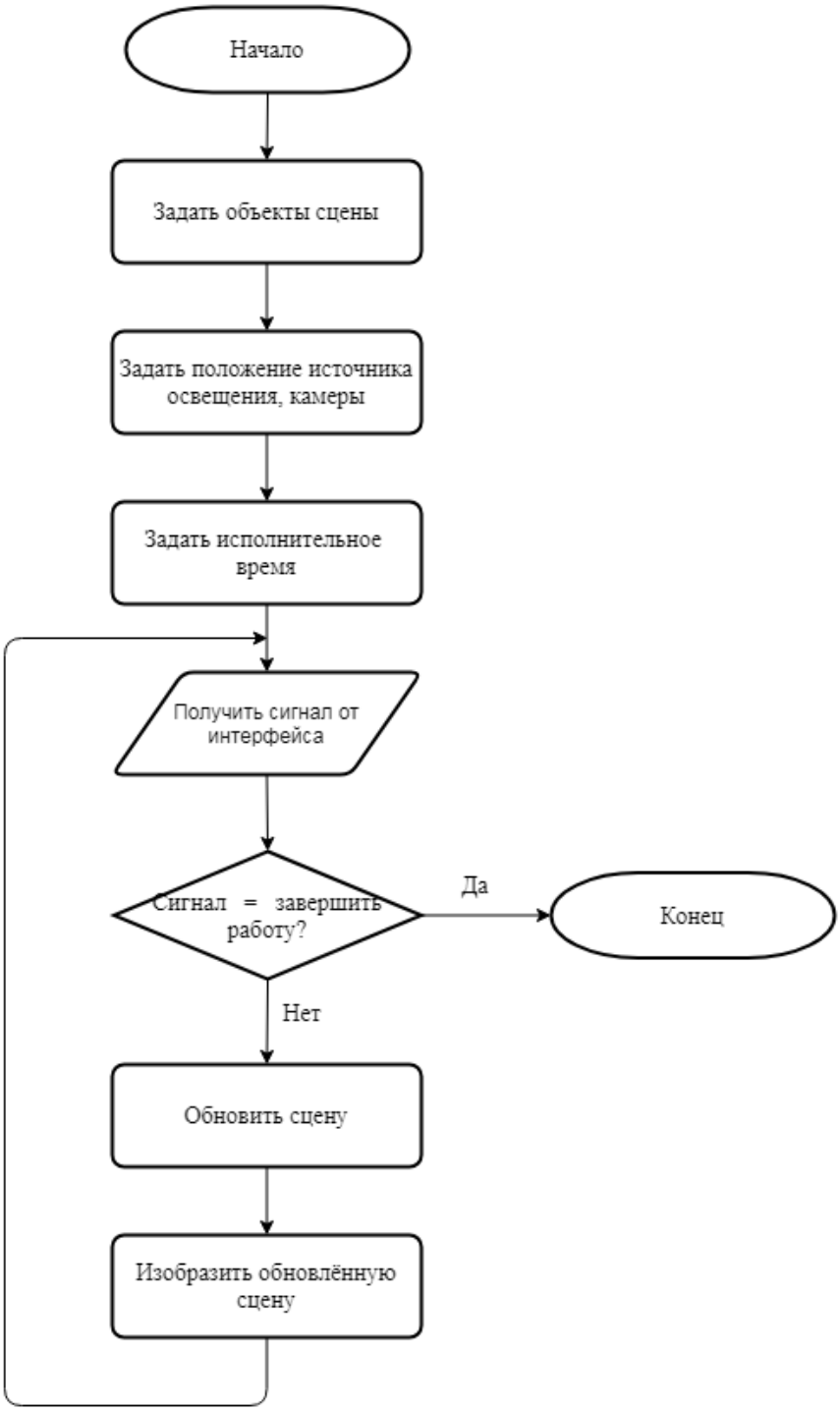

Рисунок 4 - Общий алгоритм работы

В качестве сигнала может выступать команда перемещения камеры вокруг объекта, смены положения источника освещения, начала работы, повторного отсчёта времени (только при окончании уже запущенного процесса), завершения работы.

2.3 Алгоритм z-буфера

Блок-схема этого алгоритма представлена ниже на Рисунке 5:

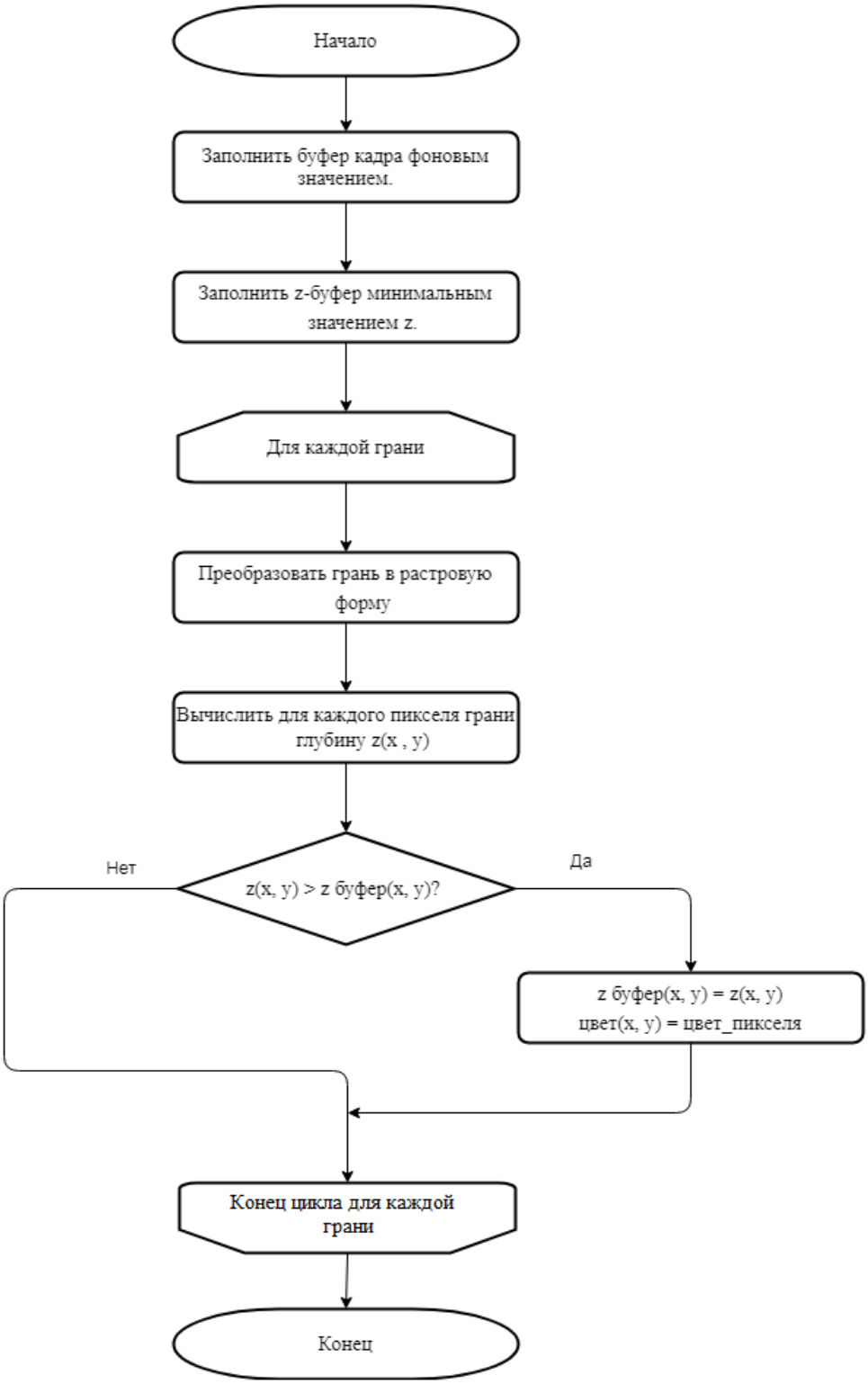

Рисунок 5 - Алгоритм z-буфера

Трудоёмкость этого алгоритма линейно зависит от числа поверхностей на сцене.

2.4 Алгоритм закраски по Гуро

Сначала определяется интенсивность вершин, а затем вычисляется интенсивность соответствующего пикселя. Блок-схема представлена на Рисунке 6.

Этот метод хорошо сочетается с построчным сканированием. Для каждой сканирующей строки определяются её точки пересечения с рёбрами. В этих точках интенсивность вычисляется с помощью линейной интерполяции интенсивностей в вершинах рёбра. Затем для всех пикселей, находящихся внутри многоугольника и лежащих на сканирующей строке, аналогично вычисляется интенсивность.

Можно заранее высчитать нормали к каждой вершине, чтобы не делать эти вычисления в процессе работы алгоритма. Интенсивность в вершинах полигона определяется как скалярное произведение вектора светового луча и нормали в вершине.

Рисунок 6 - Алгоритм закраски по Гуро

2.5 Физика песка

На Рисунке 7 изображена блок-схема процесса пересыпания песка в модели:

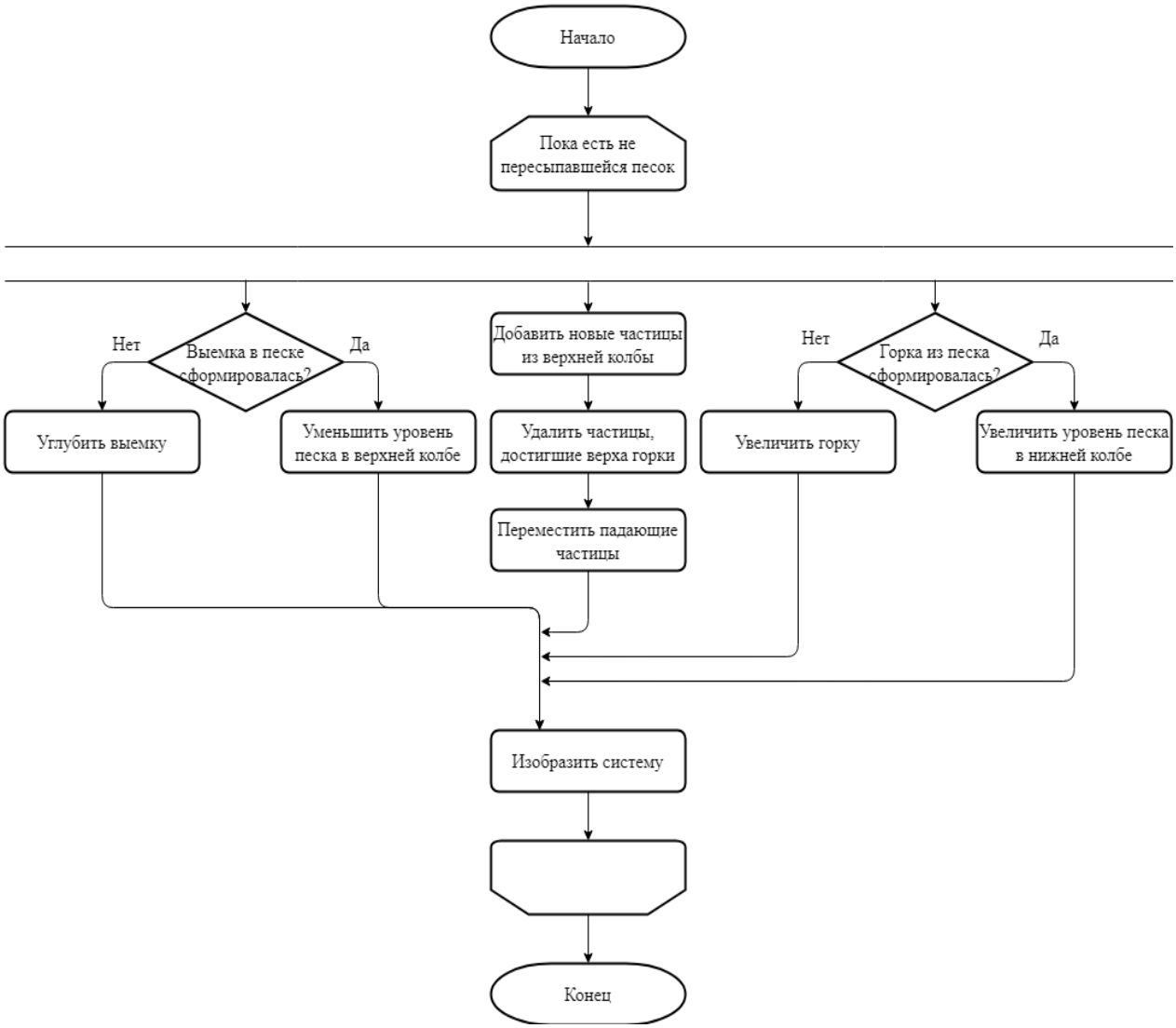

Рисунок 7 – Процесс пересыпания песка

Изначально весь песок находится в верхней части часов. Постепенно он пересыпается в нижнюю. Весь процесс происходит до тех пор, пока песок остаётся в верхней колбе. Количество песка в верхней части должно равномерно уменьшаться, в нижней, наоборот, увеличиваться. Отдельное внимание уделено формированию выемке в песке в верхней колбе и горке в нижней.

2.6 Создание полигональной сетки

В ходе работы программы будет активно использоваться полигональная сетка. Это совокупность вершин, рёбер и граней, которые определяют форму многогранного объекта в трёхмерной компьютерной графике и объёмном моделировании.

Гранями обычно являются треугольники, четырёхугольники или многоугольники, при этом у каждой грани высчитывается нормаль.

Наиболее удобно использовать треугольник, так как:

1) любой многоугольник можно разделить на несколько треугольников
2) треугольник также можно разбить на другие треугольники меньшей площади
3) треугольник не может быть невыпуклым, и это свойство упрощает работу
4) позволяют эффективно использовать память, так как можно организовать хранение так, чтобы для каждого следующего треугольника нужно будет добавлять только одну вершину

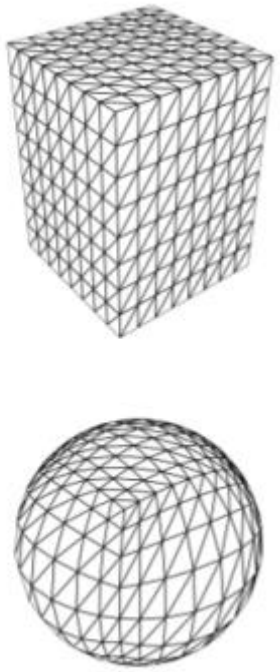

Рисунок 8 - Использование полигональной сетки

Примеры использования полигональной сетки приведены на Рисунке 8.

Построение полигональной сетки осуществляется следующим образом: зная число разбиений по осям, можно по рассматриваемой формуле плоскости найти неизвестное, тем самым заполнив «ячейку» сетки. Далее все полученные точки соединяются в четырёхугольники, и в каждом четырёхугольнике строится диагональ, которая разбивает его на два треугольника. Все параметры разбиения заданы таким образом, чтобы обеспечить не только качественную визуализацию, но и минимальные, насколько это возможно затраты по памяти.

2.7 Строение объектов сцены и отношения между ними

На Рисунке 9 представлена схема взаимодействия объектов сцены и показаны их составляющие.

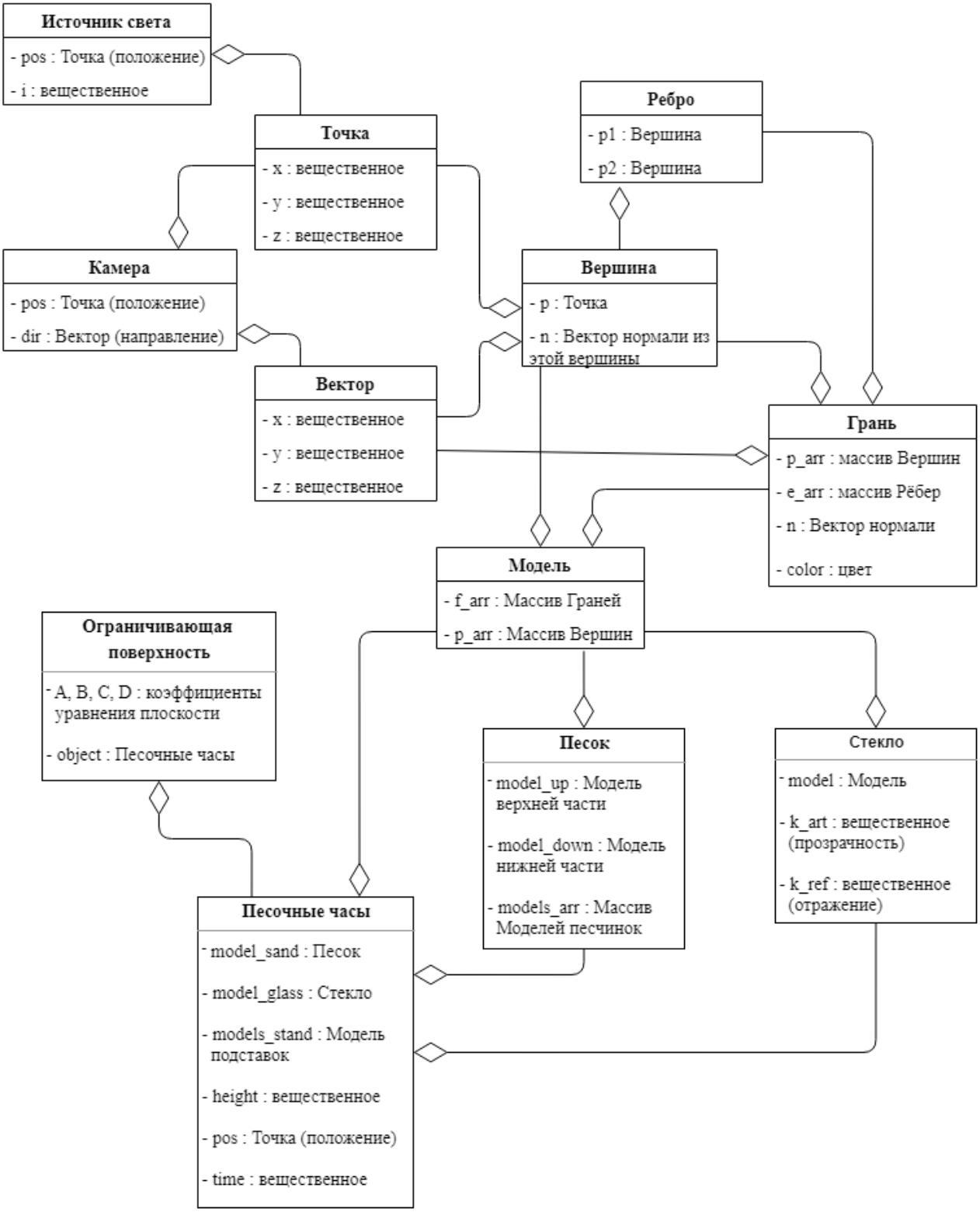

Рисунок 9 - Объекты сцены

3.Технологический раздел

3.1 Выбор и обоснование языка программирования и среды разработки

При выборе языка программирования важно учитывать много факторов в зависимости от задачи. Из-за специфики модели в приложении на сцене будет большое количество объектов. К тому же происходящее на сцене будет постоянно изменяться, что даёт дополнительную нагрузку.

В качестве языка программирования (ЯП) был выбран Си++. На это есть несколько причин:

1. при разработке этого программного продукта предусматривается использование объектно-ориентированного подхода, а его как раз поддерживает этот ЯП;
2. этот язык обладает большой производительностью, что важно в данной задаче;
3. предоставляет большое число шаблонов и библиотек, которые ускоряют работу и улучшают эффективность алгоритмов;
4. накопившийся опыт работы с этим языком во время обучения, что упростит задачу, также в свободном доступе в большом количестве есть необходимая литература и документация.

В качестве среды разработки был выбран QtCreator, который:

1. хорошо знаком, так как активно использовался во время обучения;
2. предоставляет удобную графическую библиотеку;
3. позволяет работать с графическим интерфейсом;
4. является бесплатным приложением.

3.2 Формат входных данных

Входным данным выступает измерительное время, которое пользователь выбирает из предлагаемого списка.

Пользователь также может задать через интерфейс расположение камеры, источника освещения.

3.3 Описание интерфейса

Пользователь может изменять положение источника освещения, камеры через графический интерфейс. Как было указано выше, через него также задаётся измерительное время. Кроме того, можно, не дублируя данные, повторить отсчёт времени (после того, как текущий процесс завершился), нажав на соответствующую кнопку.

Заключение

В процессе выполнения поставленной задачи была визуализирована трёхмерная сцена с множеством объектов. Были изучены, проанализированы и реализованы алгоритмы удаления невидимых линий и поверхностей, закраски, построения теней. Все алгоритмы выбирались, исходя из поставленной задачи. Была разработана физическая модель поведения объектов на сцене с учётом особенностей всей системы.

Для того чтобы проверить корректность работы физической модели, была написана программа в двумерном представлении, где была смоделирована работа песочных часов, с учётом всех особенностей и возможностей, которые были заявлены ранее.

Список использованной литературы

1. Роджерс Д. Алгоритмические основы машинной графики: Пер. с агл. – М.: Мир, 1989. – 512 с.
2. Методы представления дискретных трехмерных данных [Электронный ресурс]. – Режим доступа: https://www.graphicon.ru/oldgr/ru/library/multires_rep/index.html (дата обращения 02.07.20)
3. Алгоритмы закраски [Электронный ресурс]. – Режим доступа: https://studbooks.net/2248060/informatika/odnotonnaya_zakraska_metod_graneniya (дата обращения 03.07.20)
4. Обзор алгоритмов построения теней в реальном времени [Электронный ресурс]. – Режим доступа: https://www.ixbt.com/video/realtimeshadows.shtml (дата обращения 04.07.20)
5. Ошаровская Е.В., Солодка В.И. Синтез трехмерных объектов с помощью полигональных сеток, Цифровые технологии, 2012, № 12, 2-9
6. Простые модели освещения [Электронный ресурс]. – Режим доступа: http://grafika.me/node/344 (дата обращения 05.07.20)
7. Продвинутое освещение. Модель Блинна-Фонга [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/353054/ (дата обращения 06.07.20)
8. Куров А.В., Курс лекций по дисциплине “Компьютерная графика” [Текст]
9. Польский С.В., Компьютерная графика: учебн.-методич. Пособие. – М. : ГОУ ВПО МГУЛ, 2008. – 38 с.

Приложение