

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

## 2020 г.

### **Индивидуальное задание**

Разработать программу для моделирования и трёхмерной визуализации настольной игры в бильярд. Проанализировать методы построения реалистичных изображений и обосновать их выбор для решения поставленных задач.

## Оглавление

<b>Введение .....</b>	<b>4</b>
<b>1 Аналитическая часть.....</b>	<b>5</b>
1.1 Детализация задачи .....	5
1.2 Описание моделей визуализируемых объектов .....	5
1.3 Анализ алгоритмов удаления невидимых линий и поверхностей .....	6
1.4 Анализ алгоритмов закрашивания .....	9
1.5 Анализ алгоритмов освещённости .....	10
1.6 Физическая модель поведения объектов.....	11
1.7 Изменяемые параметры модели .....	11
<b>2 Конструкторская часть .....</b>	<b>13</b>
2.1 Алгоритм программы .....	13
2.2 Алгоритм, использующий z-буфер .....	14
2.3 Локальная модель освещения .....	15
2.4 Метод закрашки по Гуро.....	16
2.5 Алгоритм механического взаимодействия объектов.....	16
2.6 Используемые классы .....	19
<b>3 Технологическая часть .....</b>	<b>20</b>
3.1 Выбор средств программной реализации .....	20
3.2 Формат конфигурационных файлов.....	20
3.3 Описание интерфейса.....	21
3.4 Программная модель поведения шаров .....	21
<b>Заключение .....</b>	<b>23</b>
<b>Список использованных источников .....</b>	<b>24</b>

## **Введение**

Компьютерная графика занимает важное место в современных информационных технологиях. На эту область приходится решение таких задач как визуализация построек при конструировании, создание симуляций управления самолётами, поездами и т.п. Широко используется компьютерная графика и в видеоиграх. Спектр проблем в данных областях достаточно широк, поэтому существует множество алгоритмов для визуализации трёхмерных изображений. Их разнообразие объясняется тем, что не существует универсальных методов, одновременно создающих высоко реалистичное изображение и показывающих высокое быстродействие, поэтому различные алгоритмы позволяют сделать упор на наиболее важную характеристику.

Целью практической работы является создание ПО для моделирования игры в бильярд. Актуальность данной темы объясняется тем, что разработанная модель позволит лучше понять особенности данной игры, разобраться с теоретическими основами построения траектории движения шаров после удара и столкновений, научиться выбирать направление и силу удара, что даст возможность потенциальному игроку отработать на компьютерной симуляции практические навыки, которые можно будет применить и в реальной игре.

Программа будет предоставлять пользователю трёхмерную модель бильярдного стола и шаров с возможностью осуществления ударов и дальнейшей визуализацией их поведения. Пользователь сможет изменять положение камеры и источников света вокруг стола. В данной задаче основные требования к реалистичности изображения касаются в основном траектории движения шаров, которая должна строиться с учётом физических законов. Само движение при этом должно передаваться плавно. Поэтому при выборе алгоритмов предпочтение будет отдано тем, которые обеспечивают большую частоту вывода кадров на экран.

## **1 Аналитическая часть**

### **1.1 Детализация задачи**

Целью практической работы является создание программы для моделирования настольной игры в бильярд. В программе должна иметься возможность взаимодействия с шарами, задания их расстановки, рассмотрения стола с различных ракурсов и изменения его освещённости.

Для реализации всех перечисленных требований необходимо решить следующие задачи:

- Выделить объекты сцены и выбрать модель их представления
- Проанализировать, выбрать и реализовать алгоритмы визуализации объектов. Обязательным требованием является реалистичность создаваемого трёхмерного изображения
- Разработать физическую модель поведения объектов, обеспечивающую правдоподобное поведение объектов сцены
- Предоставить возможность задания начальных конфигурационных параметров игры
- Реализовать графический интерфейс для предоставления пользователю вышеописанных возможностей взаимодействия с игрой

### **1.2 Описание моделей визуализируемых объектов**

В качестве объектов визуализируемой сцены можно выделить следующие сущности:

- Камера. Представляет собой наблюдателя, поэтому задаётся при помощи трёхмерной точки положения и трёхмерного вектора направления обзора.
- Источник света. Задаётся при помощи трёхмерной точки положения и значения интенсивности света.

- Бильярдный стол. Составной объект, в свою очередь состоящий из следующих составляющих:
  - Плиты, покрытая сукном. Является параллелепипедом, задаётся при помощи размеров и положения.
  - Бортики с лузами. Являются параллелепипедами, имеющими кругообразные вырезы под лузы. Задаются при помощи размера, положения, а также радиусом и положением луз.
  - Ножки. Также являются параллелепипедами, задаются размерами и положением.
- Бильярдные шары. Являются шарами, в качестве задающих параметров являются положение и радиус.

Для задания всех объектов сцены наиболее удобным будет использование каркасной модели объектов.

### **1.3 Анализ алгоритмов удаления невидимых линий и поверхностей**

Основным фактором, на который делается акцент в данной задаче является частота вывода изображения на экран. Пользователь должен получить достаточно плавную для восприятия анимацию, в то время как реалистичностью и детализацией изображения можно в разумных пределах пренебречь. Это обусловлено тем, что в данной программе, как было сказано ранее, более значимую роль играет физическая модель поведения объектов, в то время как отдельные детали визуализации играют уже второстепенную роль. Поэтому, при выборе алгоритма следует ориентироваться в первую очередь на его быстродействие.

#### **Алгоритм Варнока**

Принцип алгоритма Варнока, работающий в пространстве изображения, можно описать как “Разделяй и властвуй”. Область рисунка разбивается на несколько частей, для каждой из которых производится анализ попавших в неё

многоугольников. В случае, если цвет области не определяется одним многоугольником, вышеописанная операция повторяется уже для данной области. Иначе, область полностью закрашивается цветом многоугольника.

Преимуществом алгоритма является то, что с помощью него можно эффективно работать с многоугольниками, занимающими большие области в пространстве изображений.

Недостатком является низкая эффективность, в случае, когда на сцене присутствует множество небольших по размеру многоугольников. В таком случае, для заливки объектов придётся делать достаточно много разбиений, а сами конечные заливаемые области по размеру будут стремиться к пикселю. Данный недостаток является существенным для решаемой, так как шары будут задаваться с помощью множества аппроксимирующих многоугольников.

### **Алгоритм, использующий Z-буфер**

Данный алгоритм работает в пространстве изображения. Ключевым понятием в нём является буфер кадра, являющийся матрицей, содержащей некую информацию о каждом пикселе изображения. Основным буфером является Z-буфер, содержащий координату  $z$  точки на наиболее приближённом объекте в данном пикселе. Алгоритм сравнивает значения Z-буфера для каждого объекта с уже имеющимся. В случае, если для какого-либо пикселя значение  $z$  объекта больше, информация о данном пикселе обновляется в соответствии с данным объектом для каждого буфера, например, буфера интенсивности света.

Преимуществом алгоритма является линейная зависимость от числа визуализируемых объектов. Также не важен и порядок анализа объектов, что позволяет не осуществлять сортировку. Также, временные затраты практически не зависят от количества рассматриваемых многоугольников, что важно для шаров, имеющих большое количество граней.

Недостатком является объём занимаемой буферами памяти. Однако, данный недостаток не является столь существенным, ввиду достаточного количества памяти в современных условиях.

## Алгоритм обратной трассировки лучей

Алгоритм работает в пространстве изображения. Для каждого пикселя от точки зрения наблюдателя исходит луч, направление которого определяется положением пикселя. Для луча ищутся пересечения со всеми объектами, после чего в соответствии с ближайшим из них происходит закраска пикселя.

Преимуществом алгоритма является линейная зависимость от количества объектов на сцене. Также данный алгоритм удобно использовать вместе с глобальной моделью освещения, что позволяет создавать реалистичные изображения.

Недостатком является большое количество вычислений для каждого из лучей, а также для его отражений, что является недопустимым в данной задаче, так как приведёт к низкой частоте кадров обновления изображения.

Результаты анализа можно свести таблицу 2.1. Проанализировав данные алгоритмы, можно прийти к выводу, что наиболее подходящим решением для данной программы будет использование алгоритма z-буфера. Данный алгоритм позволит выполнять синтез изображения достаточно быстро, а также упростит задачу закраски и освещения объектов за счёт соответствующих буферов.

Таблица 2.1. Сравнение алгоритмов удаления невидимых линий

Критерий Алгоритм	Сложность (N – грани, P – пиксели)	Пространство алгоритма	Сложность реализации	Эффективность для большого числа объектов
Варнока	$O(P*N)$	Изображения	Средняя	Средняя
Z-буфер	$O(P*N)$	Изображения	Низкая	Высокая
Обратной трассировки лучей	$O(P*N)$	Изображения	Средняя	Низкая



## **1.4 Анализ алгоритмов закрашивания**

Существует несколько методов закрашки, позволяющих по-разному передать цвет и освещённость граней объектов. Рассмотрим их, для того чтобы определиться с наиболее подходящим для поставленной задачи.

### **Простая модель закрашки**

В данном методе вся грань закрашивается с одинаковой интенсивностью, определяемой по нормали к данной поверхности. Алгоритм является самым быстродействующим, однако полученное изображение не будет отличаться реалистичностью отображения освещённости.

### **Закраска по Гуро**

В данном методе используется интерполяция интенсивности. За счёт усреднения нормалей граней, сходящихся в одной точке, вычисляется нормаль к вершинам многогранника, по которой вычисляется интенсивность в вершине. Далее, полученные значения интерполируются сначала по граням, а потом и по плоскостям многогранника. В итоговом изображении происходит сглаживание ребристости объектов, что хорошо в случае аппроксимации гладких объектов с помощью каркасной модели.

### **Закраска по Фонгу**

Данный метод схож с закрашкой по Гуро. Различием является то, что вместо интенсивности происходит интерполяция по значению самой нормали. Это позволяет получить улучшенную аппроксимацию кривизны поверхности по сравнению с Гуро и лучше передаёт блики, однако требует в 3 раза больше вычислений из-за интерполяции трёх значений вместо одного и необходимости вычислять интенсивность для каждой из нормалей.

В поставленной задаче лучше всего подойдёт метод Гуро, так как он сможет обеспечить достаточно реалистичное изображение множества используемых в сцене закруглённых объектов, и при этом будет иметь

приемлемую ресурсозатратность. Также данный метод будет достаточно удобно сочетать с выбранным алгоритмом z-буфера.

## **1.5 Анализ алгоритмов освещённости**

Неотъемлемой частью реалистичного изображения является освещённость объектов. Рассмотрим модели существующие модели освещения.

### **Локальная модель освещённости**

Данная модель рассматривает только однократное отражение лучей от объектов, поэтому не передаёт взаимного освещения между разными объектами сцены. Модель позволяет определить только факт освещённости или затемнённости частей объектов. В данной модели рассматриваются только от точных источников света, которые, в частности, могут располагаться на бесконечном удалении от сцены. Для отражённых лучей требуется рассчитать только интенсивность и цвет.

### **Глобальная модель освещённости**

Глобальная модель позволяет создать более реалистичное изображение за счёт учёта отражённых и преломлённых лучей. Данная модель рассматривает сцену как единую систему и учитывает лучи, попавшие на объект не непосредственно от источника света, а отразившись от другого объекта. Такой подход позволяет создать более реалистичное изображение, однако значительно возрастают вычислительные затраты.

Для поставленной задачи лучше подходит локальная модель освещённости, так как она является более быстросействующей. Также в сцене отсутствуют объекты, обладающие зеркальными или преломляющими свойствами, поэтому использование более качественной глобальной модели не требуется.

## **1.6 Физическая модель поведения объектов**

Программная реализация поставленной задачи будет обладать практической ценностью только в том случае, если пользователь сможет получить реалистичную модель поведения объектов.

Главными элементами системы, участвующими в кинетических процессах, являются бильярдные шары. Также во взаимодействии учувствуют бортики стола, но они считаются неподвижными и служат лишь для отражения от них шаров. Помимо бортиков, шары обязательно должны иметь возможность взаимодействовать друг с другом. При этом следует учесть потери энергии, возникающие при перемещении и столкновении, что позволит системе приходить в состояние покоя за правдоподобное время.

Для выполнения этих требований, в основу физического поведения модели должны лечь законы трения и модель абсолютно упругого удара. Система работает в дискретном временном пространстве, поэтому её основной задачей будет изменение параметров за время  $\Delta t$ , прошедшее после текущего момента. Физический модуль будет выполнять перемещение объектов и поворот в соответствии с текущей скоростью, применять потери от диссипативных сил, разрешать возникшие коллизии и моделировать удары.

Также обязательно должна быть обеспечена возможность вмешательства в покоящуюся систему приданием одному из шаров импульса произвольной силы и направления. Это необходимо для симуляции удара кия.

## **1.7 Изменяемые параметры модели**

В различных версиях бильярда могут варьироваться некоторые параметры, которые можно учесть для создания более универсальной программы:

- Высота ножек бильярдного стола.
- Ширина и длина стола.
- Размер и цвет шаров.
- Количество и расстановка шаров.

Возможность изменения данных параметров нужно предусмотреть при разработке пользовательского интерфейса.

## 2 Конструкторская часть

Выполнив анализ имеющейся задачи, были выбраны наиболее подходящие алгоритмы. В данном разделе представлены конструкторские решения, принятые для реализации выбранных алгоритмов.

### 2.1 Алгоритм программы

На рисунке 2.1 представлен общий алгоритм работы программы.



Рисунок 2.1 Блок-схема общего алгоритма программы

В качестве сигналов интерфейса могут служить такие действия как:

- Обновления положения и направления камеры.
- Обновление положения источников света.
- Совершение удара по одному из шаров (задаются направление и сила удара).
- Завершение работы программы.

В соответствии с ними сцена может обновлять свою конфигурацию. Также параметром для обновления системы может послужить изменение текущего времени в случае, если система находится в движении.

## 2.2 Алгоритм, использующий z-буфер

Основой метода вывода изображения на экран был выбран алгоритм, использующий z-буфер. На рисунке 2.2 представлена блок-схема данного алгоритма.

Для уменьшения количества вычислений, нахождению  $X_{min}$ ,  $X_{max}$  и  $Z(x, y)$  можно придать итеративный характер. Для этого, следует использовать список активных рёбер (САР) для каждой грани. Каждое ребро в такой структуре будет хранить в себе:

- $x$  – текущее пересечение со сканирующей строкой;
- $dx$  – приращение по  $X$  при переходе к следующей строке;
- $z, dz$  – аналогично  $x, dx$ ;
- $y$  – количество строк, пересекающих это ребро;
- $y_{max}$  – наивысшая строка, пересекающая ребро;

Значения для всех рёбер вычисляются в начале работы с очередной гранью. Все рёбра изначально хранятся в списке неиспользованных рёбер, откуда переносятся в САР по мере достижения строки значения  $y_{max}$ . При переходе к следующей сканирующей строке,  $x$  и  $z$  увеличиваются на  $dx, dz$ , а  $y$  уменьшается на 1. В случае, если значение  $y$  ребра становится  $< 0$ , оно удаляется из САР.

Для вычисления начальных значений ребра, лежащего между вершинами  $P1$  и  $P2$  (где  $y_1 > y_2$ ), используются формулы:



Рисунок 2.2 Блок-схема алгоритма, использующего z-буфер

$$dx = \frac{x_2 - x_1}{y_2 - y_1}, \quad dz = \frac{z_2 - z_1}{y_2 - y_1}, \quad y = y_1 - y_2, \quad x = x_1, \quad z = z_1,$$

$$y_{max} = y_1$$

При вычислении значения  $Z(X, Y)$  для каждого пикселя также используется итеративное приращение значения  $z$  от левого ребра к правому с изменением формулы  $dz = \frac{z_{\text{прав}} - z_{\text{лев}}}{x_{\text{прав}} - x_{\text{лев}}}$ .

### 2.3 Локальная модель освещения

Для нахождения интенсивности в точке, данная модель использует формулу  $I = I_{\text{ист}} k \cos \alpha$ , где  $I_{\text{ист}}$  – интенсивность источника света,  $k$  – коэффициент диффузного отражения света,  $\cos \alpha$  – угол между нормалью  $N$  и вектором, проведённым от источника света до рассматриваемой точки  $L$ .  $\cos \alpha$  находится как  $\cos \alpha = \frac{(N, V)}{|N| * |V|}$ .

Вычисление нормали к плоскости возможно осуществить путём вычисления уравнения плоскости по трём произвольным вершинам грани.

$$Ax + By + Cz + D = 0$$

Для трёх точек  $P1(x_1, y_1, z_1)$ ,  $P2(x_2, y_2, z_2)$ ,  $P3(x_3, y_3, z_3)$ , уравнение плоскости будет находиться как определитель матрицы:

$$\begin{vmatrix} x - x_1 & y - y_1 & z - z_1 \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix}$$

То есть, коэффициенты  $A$ ,  $B$ ,  $C$  можно выразить как:

$$\begin{cases} A = (y_2 - y_1)(z_3 - z_1) - (y_3 - y_1)(z_2 - z_1) \\ B = (z_2 - z_1)(x_3 - x_1) - (z_3 - z_1)(x_2 - x_1) \\ C = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) \end{cases}$$

Что и будет являться значениями вектора нормали  $N(A, B, C)$ .

Полученное значение интенсивности может быть использовано для определения цвета точки грани. Так как при нулевой интенсивности поверхность выглядит чёрной, а при 100% имеет собственный цвет, интенсивность выступает в роли соотношения, в котором нужно смешать чёрный и собственный цвет грани.

## 2.4 Метод закрашки по Гуро

Для определения цвета пикселей был выбран метод закрашки Гуро. Его можно описать при помощи представленной на рисунке 2.3 блок-схемы.

Имея уравнения всех смежных вершине граней можно найти усреднённую нормаль  $N_{\text{вер}} = \sum N_{\text{грани}}$ . Полученный вектор нормали используется для нахождения интенсивности освещения в вершине в соответствии с вышеприведённой формулой.

После получения значений интенсивности в вершинах, они интерполируются сначала на связывающие их рёбра, а после на грани. Несложно заметить, что процесс нахождения интенсивности  $I(X, Y)$  схож с применённым в алгоритме z-буфера способе вычисления  $Z(X, Y)$ . Поэтому, для совмещения данных алгоритмов, можно добавить в информацию о каждом ребре значение интенсивности в данной точке, а также её приращение при переходе к следующей сканирующей строке.

## 2.5 Алгоритм механического взаимодействия объектов

Задачей алгоритма служит обеспечение реалистичного поведения подвижных элементов системы,

т.е. бильярдных шаров, с использованием законов трения и абсолютно упругого удара, которых для решения данной задачи вполне достаточно. Блок-схема алгоритма моделирования приведена на рисунке 2.4. Для описания движения шара требуется задать скорости шара по осям  $x$  и  $z$ :  $V_x, V_z$  (скорость по перпендикулярной плоскости стола оси  $OY$  считается равной 0).



Рисунок 2.3 Алгоритм закрашки по Гуро



Закон трения рассматривается только между шаром и сукном, и поэтому описывается формулой:

$$F_{\text{тр}} = \mu N = \mu m_{\text{шара}} g.$$

По второму закону Ньютона, ускорение трения  $a_{\text{тр}} = \frac{F_{\text{тр}}}{m_{\text{шара}}} = \mu g$ . Данное ускорение за  $\Delta t$  времени будет уменьшать модуль вектора скорости на  $a_{\text{тр}} \Delta t$  м/с.

$$V_0 = \sqrt{V_x^2 + V_z^2}, \quad V_1 = V_0 - a_{\text{тр}} \Delta t$$

$$\alpha = \frac{V_1}{V_0} = 1 - \frac{a_{\text{тр}} \Delta t}{\sqrt{V_x^2 + V_z^2}}$$

Уменьшение компонент  $V_x, V_z$  происходит пропорционально  $V$ , то есть в  $\alpha$  раз. В случае, если  $\alpha < 0$ , скорости обнуляются.

$$V_x = V_x * \alpha, \quad V_z = V_z * \alpha$$

Наиболее важной задачей является моделирование ударов. В моделируемой системе возможно два типа соударений: шар-бортик и шар-шар. Помимо изменений векторов скоростей шаров, требуется также устранить их коллизии, которые при визуализации будут выглядеть как “проваливание” одного объекта в другой. Рассмотрим возможные случаи:

### Соударение шар-бортик

В данной модели движением бортика стола при соударении с шаром можно пренебречь ввиду незначительности и незаметности этого процесса в реальной игре. Таким образом, при ударе шар будет абсолютно упруго отскакивать от неподвижной поверхности, которая, как ранее было оговорено,

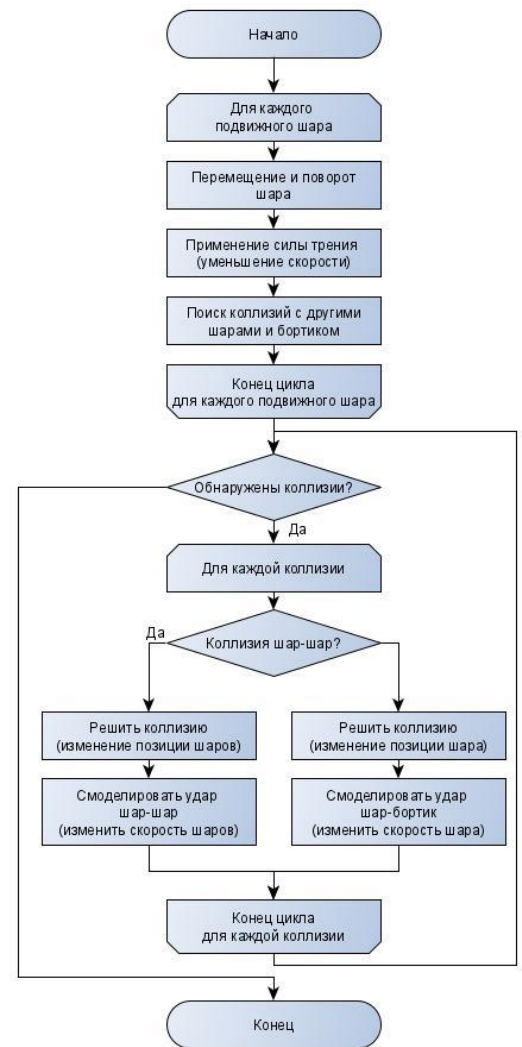


Рисунок 2.4 Блок-схема алгоритма механического взаимодействия

расположена вдоль оси OX или OZ. В случае удара об бортик, расположенный вдоль OX, по закону отражения, будет меняться знак у  $V_z$  (аналогично для бортика вдоль OZ).

Проблема коллизии в таком случае решается перемещением шара от бортика на расстояние, которое он “провалился” внутрь бортика.

### **Соударение шар-шар**

Моделирование такого столкновения является более сложным, так как оба объекта являются подвижными. Как и в настоящем бильярде, в данной модели все шары имеют одинаковый размер и массу, что позволяет рассмотреть частный и более простой случай столкновения двух шаров.

При столкновении двух шаров, сила действует от точки соприкосновения к центру масс тела, следовательно изменение скорости шаров при соударении будет происходить вдоль прямой, соединяющей центры шаров, в то время как составляющие скоростей, направленных перпендикулярно этой линии для обоих шаров останется неизменным. Из этого, используя закон сохранения импульса в вышеописанной системе координат, получаем формулы для вычисления векторов скоростей после удара. Изменение скорости вдоль оси удара:

$$\Delta V_1 = \sin\alpha(V_{1y} - V_{2y}) + \cos\alpha(V_{1x} - V_{2x}), \text{ что в проекции даёт}$$

$$\Delta V_{1x} = \sin\alpha(\sin\alpha(V_{1y} - V_{2y}) + \cos\alpha(V_{1x} - V_{2x}))$$

$$\Delta V_{1y} = \cos\alpha(\sin\alpha(V_{1y} - V_{2y}) + \cos\alpha(V_{1x} - V_{2x}))$$

Для шара №2 по ЗСИ расчёты будут те же, но со знаком минус.

Коллизии для этой ситуации устраняются путём раздвигания шаров вдоль оси, связывающей их центры.

Также, при движении шара будет происходить вращение в направлении движения. Для передачи этого требуется применить преобразование поворота

шара относительно его центра. Величина угла, на которую требуется совершить поворот рассчитывается как  $\beta = 360^\circ * \frac{l}{2\pi R}$ , где  $l$  – перемещение. Так как скорость, как и поворот выражается через составляющие по осям OX и OZ, формулы принимают вид

$$\beta_z = 360^\circ * \frac{\Delta t * V_x}{2\pi R}, \quad \beta_x = 360^\circ * \frac{\Delta t * V_z}{2\pi R}, \text{ т.к. ось вращения } \perp \text{ движению}$$

## 2.6 Используемые классы

На рисунке 2.5 представлена схема классов, представляющих ранее выделенные сущности системы.

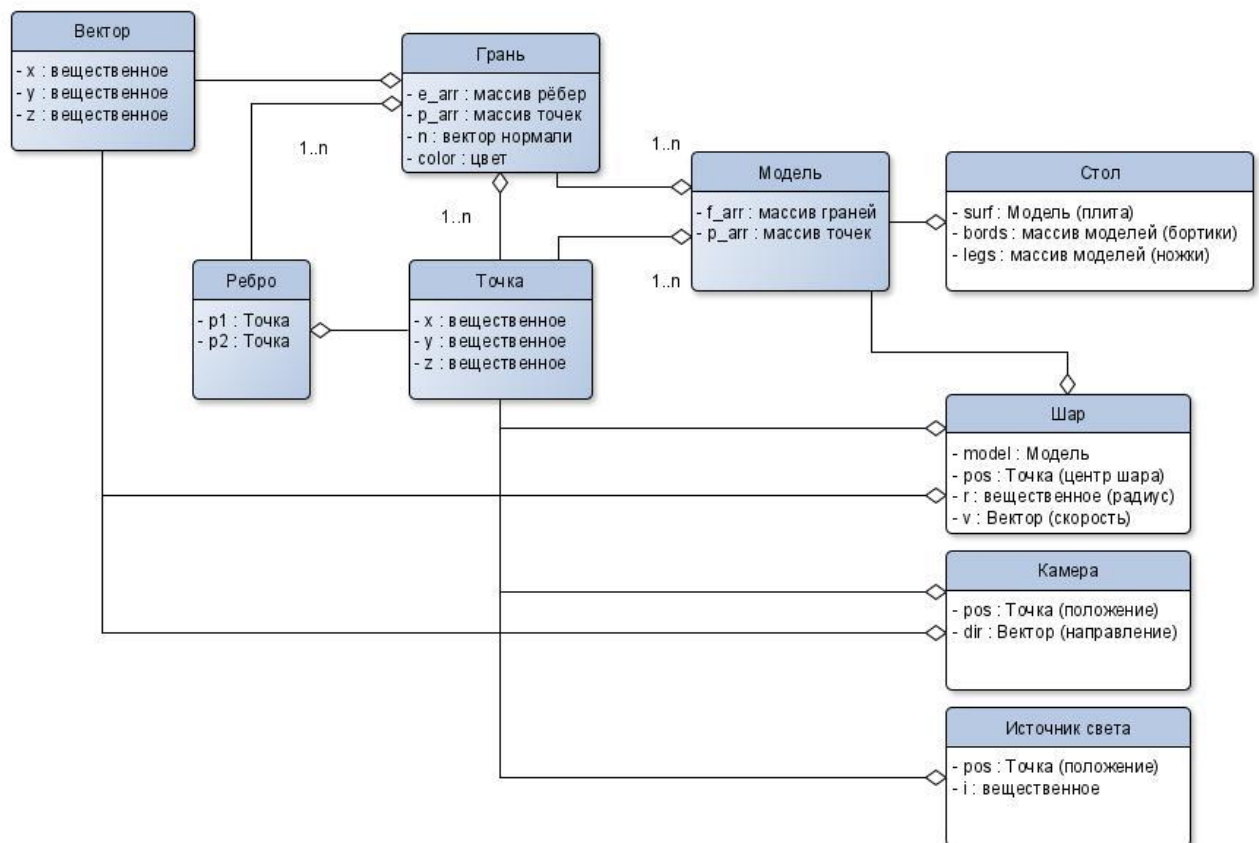


Рисунок 2.5. Структура и иерархия классов сущностей

### **3 Технологическая часть**

#### **3.1 Выбор средств программной реализации**

В качестве парадигмы программирования был выбран Объектно-ориентированный подход. Причиной этого послужила необходимость структурировать достаточно большое количество различных объектов, имеющих много общих свойств.

В качестве языка программирования был выбран C++, по следующим причинам.

- Знание языка, наличие опыта работы. Ввиду ограниченного времени проведения практической работы, этот фактор является важным, так как позволит не тратить время для изучения нового ЯП.
- Поддержка ООП.
- Высокая эффективность языка и графических библиотек, что позволит обеспечить требуемую частоту обновления изображения.

В качестве среды разработки был выбран “QtCreator”, так как:

- В нём также накоплен существенный опыт разработки.
- В рамках фреймворка Qt представлена многофункциональная и производительная графическая библиотека.
- Представлен удобный способ создания графических интерфейсов.

#### **3.2 Формат конфигурационных файлов**

В соответствии с заданными требованиями изменяемости системы, конфигурационные файлы должны иметь следующую структуру:

##### **Файл конфигурации бильярдного стола**

В данном файле должны быть указаны: высота ножек стола, ширина и длина стола (в сантиметрах).

## **Файл конфигурации бильярдных шаров**

Данный файл должен содержать:

- радиус шаров (в сантиметрах);
- количество шаров (не более 20);
- для каждого шара указывается позиция –  $x$ ,  $z$  (в сантиметрах, центром координат считается центр стола), цвет (в формате RGB).

### **3.3 Описание интерфейса**

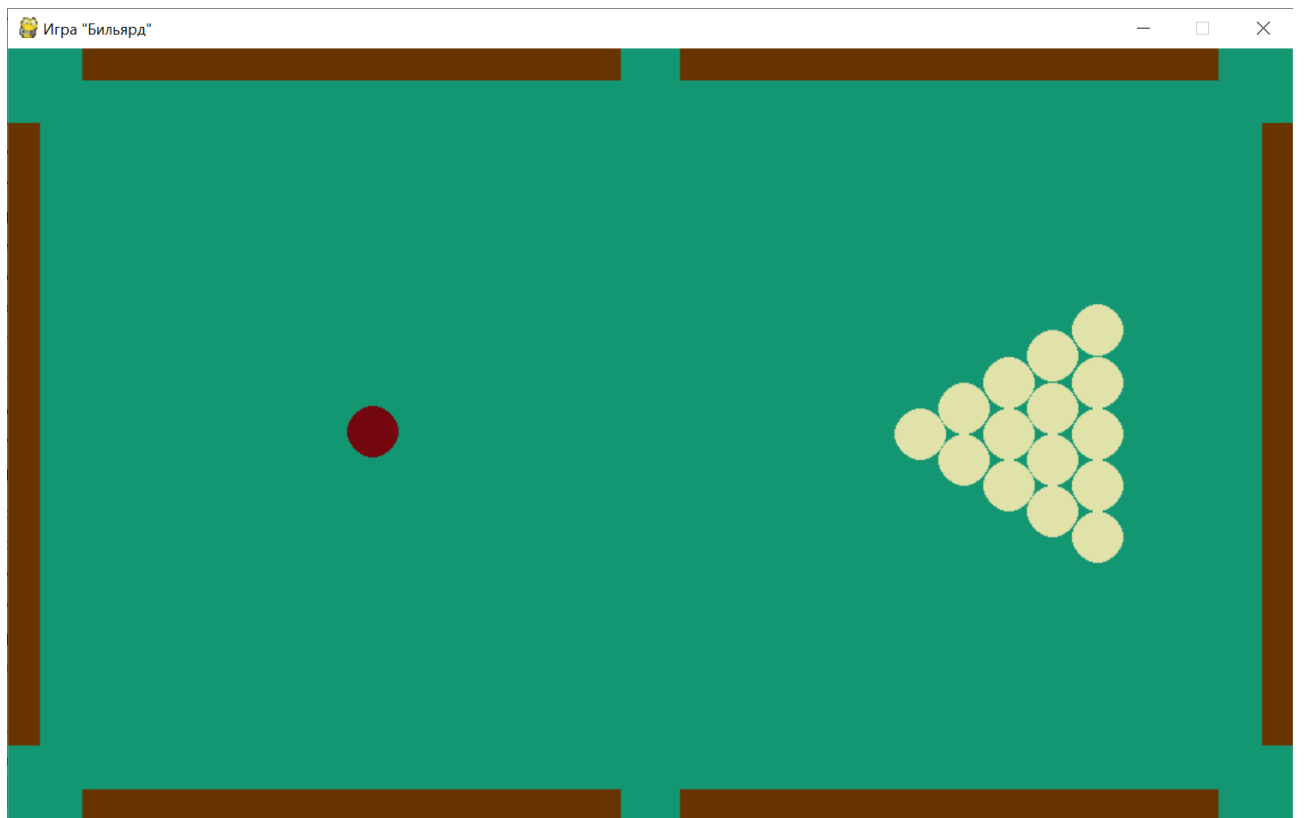
Помимо указанных в предыдущем пункте данных, требуется изменять ещё ряд параметров системы. Для этого должен быть разработан графический интерфейс, позволяющий задавать следующие параметры.

- Положение и направление камеры.
- Интенсивность и положение источника света.
- Параметры удара по шару: выбор шара, направления и силы удара.
- Файл для загрузки конфигурации игры.

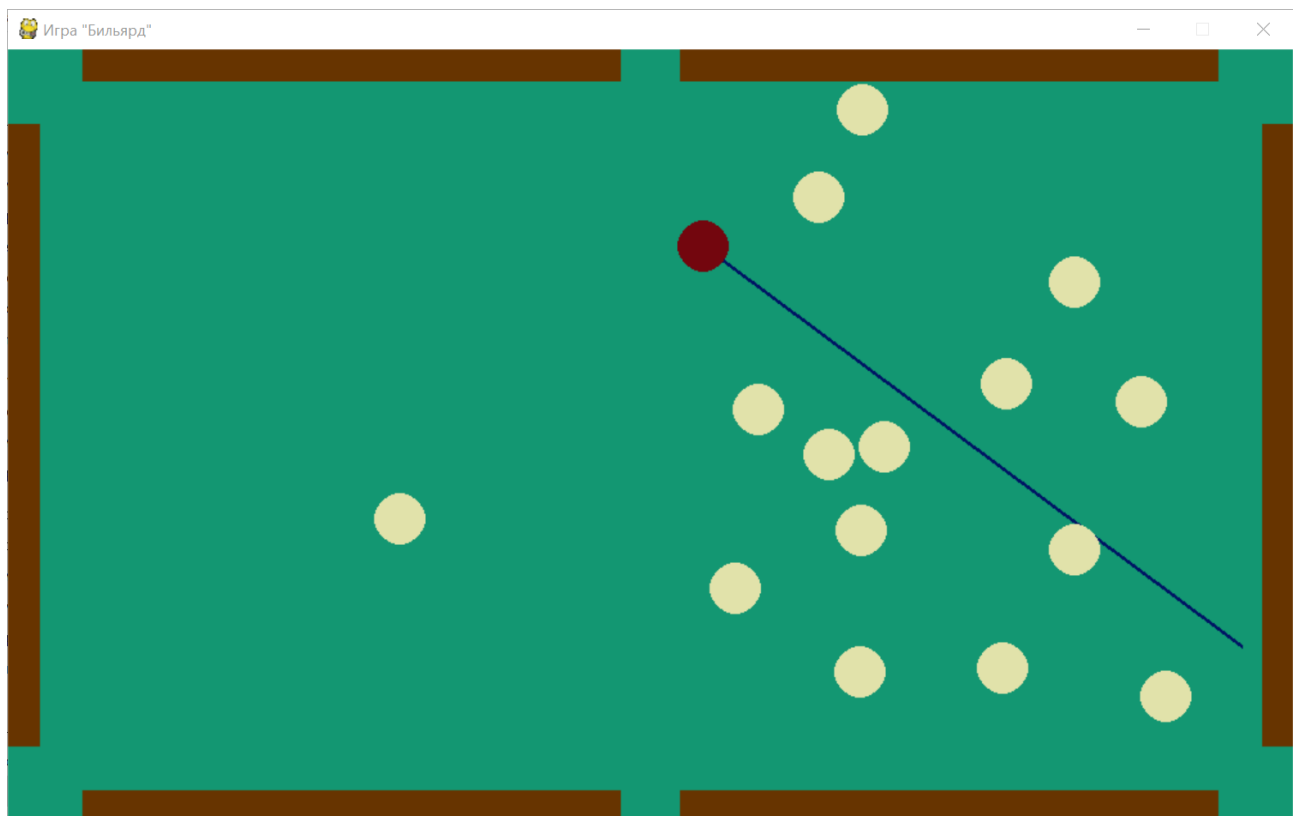
### **3.4 Моделирование поведения шаров**

С целью проверки корректности полученных физических формул и алгоритма действия физической модели была написана программа на языке Python 3 для моделирования поведения бильярдных шаров в двухмерном формате (рисунки 3.1, 3.2).

Созданная модель может быть использована в качестве основы для реализации трёхмерной модели игры.



*Рисунок 3.1. Программа моделирования бильярда в двумерном формате*



*Рисунок 3.2. Программа моделирования бильярда в двумерном формате*

## **Заключение**

В процессе выполнения работы были рассмотрены, проанализированы и выбраны для достижения поставленных задач алгоритмы трёхмерной компьютерной графики: удаление невидимых линий, методы закрашивания, методы построения теней. Также были изучены физические аспекты взаимодействия объектов в рамках решаемой задачи, результатом чего стала спроектированная и реализованная в виде программы физическая модель взаимодействия объектов.

### Список использованных источников

1. Бильярд [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Бильярд> (дата обращения 02.07.20)
2. Роджерс Д. Алгоритмические основы машинной графики [Текст]. Пер. с англ. С.А.Вичеса, Г.В. Олохтоновой, П.А. Монахова. – М.: Мир, 1989. – 512 с.
3. В.Д. Иванов. Бильярд как вид спорта. Разновидности бильярда // Физическая культура. Спорт. Туризм. Двигательная рекреация. 2019. №4.
4. И.В. Савельев. Курс общей физики: Учебник. В 3-х тт. 1. Механика, колебания и волны, молекулярная физика. 11-е изд., стер. – СПб.: Издательство «Лань», 2011. – 432 с.: ил. – (Учебники для вузов. Специальная литература)
5. Куров А.В., Курс лекций по дисциплине “Компьютерная графика” [Текст].
6. Алгоритмы закрашки [Электронный ресурс]. – Режим доступа: [https://studbooks.net/2248060/informatika/odnotonnaya\\_zakraska\\_metod\\_graneniya](https://studbooks.net/2248060/informatika/odnotonnaya_zakraska_metod_graneniya) (дата обращения 05.07.20)