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 (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

Студент Иванов Всеволод Алексеевич
фамилия, имя, отчество

Группа ИУ7-42Б

Тип практики Программа моделирования игры бильярд

Название предприятия МГТУ им. Н.Э. Баумана, кафедра ИУ7

Студент _____ Иванов В.А.
подпись, дата *фамилия, и.о.*

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

Оценка _____

2020 г.

Индивидуальное задание

Разработка программы для моделирования и трёхмерной визуализации настольной игры бильярд. Проанализировать методы построения реалистичных изображений и обосновать их выбор для поставленных задач.

Оглавление

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 самолётов, поездов и т.п., и в компьютерных играх. Спектр проблем в данных областях достаточно широк, и поэтому существует множество алгоритмов для визуализации трёхмерных изображений. Их разнообразие объясняется тем, что не существует методов, одновременно создающих высоко реалистичное изображение и показывающих высокое быстродействие, поэтому различные алгоритмы позволяют сделать упор на наиболее важную характеристику.

Целью практической работы является создания ПО для моделирования игры бильярд. Актуальность данной темы объясняется тем, что для получения многих теоретических знаний о бильярде (понимание физики игры, знание стандартных приёмов и игровых ситуаций), компьютерная симуляция может оказаться даже более эффективной, чем реальная игра.

Программа будет предоставлять пользователю трёхмерную модель бильярдного стола и шаров с возможностью осуществления ударов и дальнейшей визуализацией поведения шаров. Для удобства пользователя также будет предоставлена возможность изменения положения камеры и источников света вокруг стола. В данной задаче реалистичность изображения не имеет столь значительную роль, как плавность изображения и правдоподобность движения шаров. Поэтому при дальнейшем анализе предпочтение будет отдано алгоритмам, обеспечивающим большую частоту вывода кадров на экран.

Основная часть

1 Аналитическая часть

1.1 Детализация задачи

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

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 (наблюдатель)
- Источник света
- Ограничивающая плоскость
- Бильярдный стол. В свою очередь состоит из:
    - Плита, покрытая сукном

    - Бортики с лузами
    - Ножки
- Бильярдные шары

1.3 Анализ алгоритмов удаления невидимых линий и поверхностей

Основным фактором, на который стоит сделать 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 более значительная роль отводится под саму природу поведения объектов, в то время как реалистичность отображаемого играет уже второстепенную роль. Поэтому, при выборе алгоритма следует ориентироваться в первую очередь на быстродействие.

Опираясь на вышеизложенное, требуется рассмотреть существующие алгоритмы, оценить их качества, и на основании этого выбрать наиболее подходящий вариант.

Алгоритм Робертса

Данный алгоритм работает в пространстве объектов. В первую очередь, для каждого из объектов формируются матрица тела, описывающих с помощью 4х коэффициентов уравнение плоскости каждой из граней тела. После чего, производится ориентация плоскостей так, чтобы нормаль каждой из них была внутренней (для этого используется заведомо внутренняя точка тела). Следующим этапом происходит удаление граней тела, экранируемых им самим. Далее, каждое ребро тела проверяется на факт экранирования частями других объектов, удаляются невидимые части.

Преимуществом данного алгоритма является простота вычислений, и в то же время точность результата, которая главным образом достигается за счёт работы в объектном пространстве.

Недостатком является временная сложность, выражаемая как число объектов на сцене в квадрате. Существуют также более оптимизированные версии алгоритма, например использующие сортировку объектов по z, однако они заметно более сложные в реализации.

Учитывая тот факт, что из-за наличия множества шаров, на сцене будет большое количество объектов, алгоритм Робертса будет не самым оптимальным вариантом.

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 в данном случае, 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-любо 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 советующий луч, для которого 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.

Вывод

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.

1.4 Анализ алгоритмов закрашивания

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

Простая модель закраски

В данном методе вся грань закрашивается с одинаковой интенсивностью, определяемой по нормали к данной поверхности. Алгоритм является самым быстродейственным, 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 закраской 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 требования вычислять интенсивность для каждой из нормалей.

Вывод

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. Также рассматривается 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.

Вывод

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

1.6 Физическая модель поведения объектов

Физическая составляющая является ключевой в данном проекте, так как сама актуальность задачи основывается на том, что пользователь в первую очередь сможет получить реалистичную модель поведения объектов, что, вероятно, сможет применить и на практике.

Главными элементами системы, участвующих в кинетических процессах являются бильярдные шары. Также в взаимодействии учувствуют бортики стола, но они считаются неподвижными и служат лишь для отражения от них шаров. Помимо бортиков, шары обязательно должны иметь взаимодействие между собой. Также следует учесть потери энергии при перемещении и столкновении, что позволит системе приходить в покой 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 изменить параметры системы, приняв Δt времени, прошедшего после ныне хранящейся модели системы. 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 обеспечить возможность вмешательства в покойную 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
- Ширина и длинна стола
- Размер, масса, и цвет шаров
- Количество и расстановка шаров

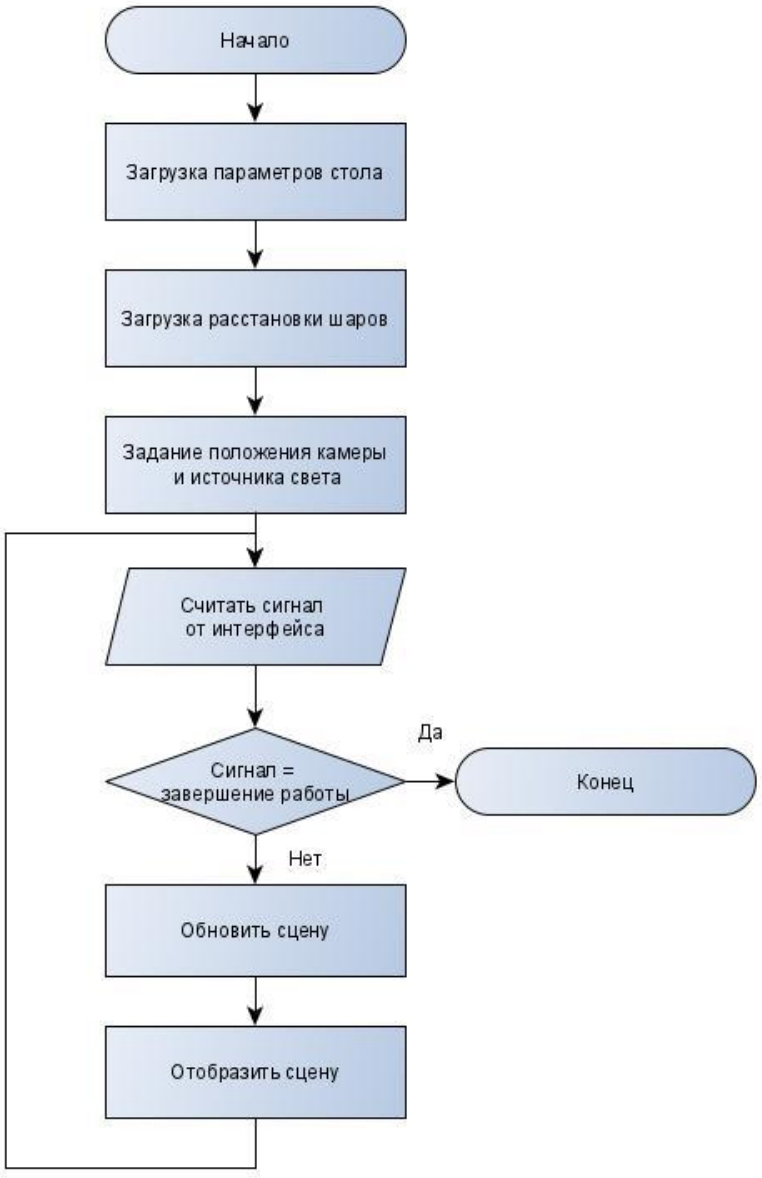
Возможность изменения данных параметров нужно предусмотреть при разработке пользовательского интерфейса.

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

Ниже представлен общий алгоритм работы программы.



В качестве сигналов интерфейса могут служить такие действия как:

- Обновления положения и направления камеры
- Обновление положения источников света
- Совершения 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. Справа представлена блок-схема данного алгоритма.

Для уменьшения количества вычислений, нахождению Xmin, Xmax 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 с сканирующей строкой
- dx – приращение по X при переходе к следующей строке
- z, dz – аналогично x, dx
- y – количество строк, пересечкающих это ребро
- ymax – 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 ymax. 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 1. Блок-схема алгоритма, использующего z-буфер

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

При вычсилении значения Z(X, Y) для каждого пикселя также используется итеративное приращения 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 $cos\alpha = \frac{(N,L)}{|N| * |V|}$.

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

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

Для трёх точек P1(x1, y1, z1), P2(x2, y2, z2), P3(x3, y3, z3), 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 является значениями вектора нормали N(A, B, C).

Полученное значение интенсивности может быть использовано для определения цвета точки грани. Так как при нулевой интенсивности поверзность 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 закраски по Гуро

Для определения цвета пикселей был выбран метод закраски Гуро. Его можно описать при помощи представленной блок-схемы.

Рисунок 2. Алгоритм закраски по Гуро

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 снчала на связывающие их рёбра, а после на грани. Несложно заметить, что процесс нахождения интенсивности I(X, Y) схож с применённым в алгоритме z-буффера 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 ниже. 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).

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 Δt времени будет уменьшать модуль вектора скорости на $a_{\text{тр}}\Delta t$ м/с.

$$V_0 = \sqrt{{\text{V}_x}^2 + {\text{V}_z}^2}, \qquad V_1 = V_0 - a_{\text{тр}}\Delta t$$

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

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

$$\text{V}_x = \text{V}_x * \alpha, \qquad \text{V}_z = \text{V}_z * \alpha$$

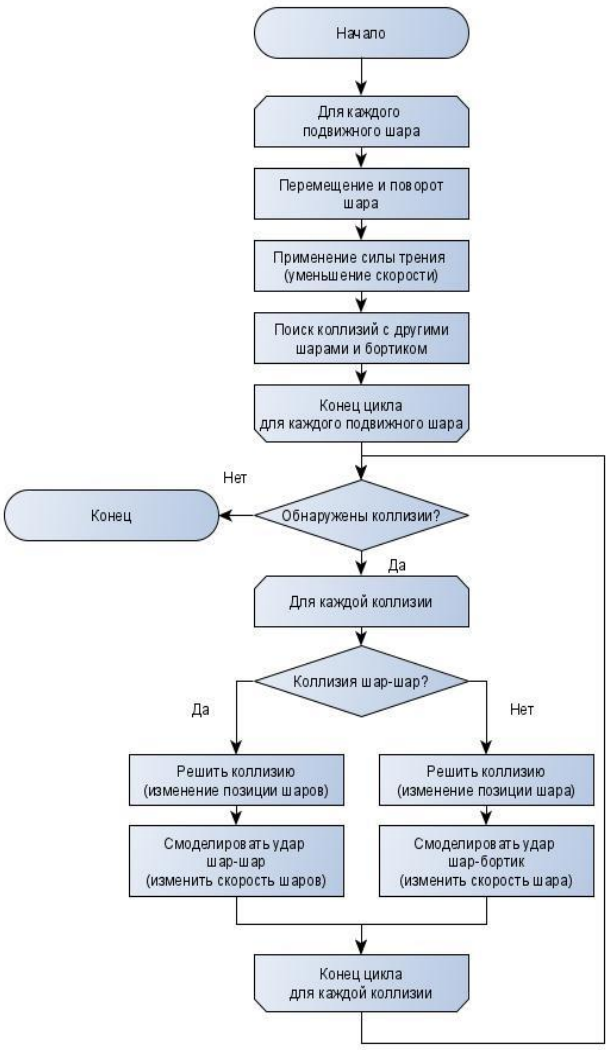


Рисунок 3. Блок-схема алгоритма механического взаимодействия

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, 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:

$\triangle V1 = sin\alpha\left(V1_y - V2_y\right) + cos\alpha(V1_x - V2_x)$,что в проекции даёт

$\triangle V1_x = sin\alpha(sin\alpha\left(V1_y - V2_y\right) + cos\alpha(V1_x - V2_x))$

$\triangle V1_y = cos\alpha(sin\alpha\left(V1_y - V2_y\right) + cos\alpha(V1_x - V2_x))$

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 $\beta = 360^{o} * \frac{l}{2\pi R}$, где l – перемещение. Так как скорость, как и поворот выражается через составляющие по осям OX и OZ, формулы принимают вид

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

2.6 **Используемые структуры данных**

Для реализации всех выделенных объектов системы требуется использовать следующие структуры данных:

- Точка - координаты x, y, z
- Вектор - значения проекций на оси x, y, z.
- Ребро - две точки.
- Грань - множество вершин, рёбер, вектор нормали и собственный цвет.
- Модель – множество вершин, граней.

- Камера – точка (положение) и вектор (направление взгляда)
- Источник света – точка (положение), интенсивность света.
- Бильярдный стол – множество объектов (ножки, бортики, плита), которые, в свою очередь, задаются при помощи модели.
- Бильярдные шары – модель (является аппроксимацией поверхности шара), точка (задаёт центр), радиус, масса, вектор скорости.

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 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, что важно по вышеописанным причинам.
- Высокая эффективность языка и графических библиотек, что позволит обеспечить требуемую частоту обновления изображения.

3.2 Формат конфигурационных файлов

Задачей алгоритма служит обеспечение реалистичного поведения

3.3 Описание интерфейса

Задачей алгоритма служит обеспечение реалистичного поведения

Заключение

Ф

Список использованных источников

https://ru.wikipedia.org/wiki/Бильярд

Приложения

Ф