МИНОБРНАУКИ РОССИИ

–––––––—————––––––––––——––––––––––––

Санкт-Петербургский государственный электротехнический

университет « ЛЭТИ» им. В. И.Ульянова(Ленина)

——————————————————––––——

Задание для лабораторной работы № 6

## " 3D графика"

Преподаватель: Герасимова Т.В.

Санкт-Петербург

2018 г.

# Трёхмерная графика

## Координатный метод описания объектов. Обобщённые координаты. Матрицы преобразования координат

Основные параметры сцены:

* Расположение объектов
* Расположение источников освещения
* Расположение камеры

Простейшие манипуляции:

* Перемещение (x:=x+∆x; y:=y+∆y; z:=z+∆z)
* Поворот (вокруг z: x’ := x cos α + y sin α; y’ := -x sin α + y cos α; z’ := z)
* Масштабирование (x := kxx)
* Отражение по оси: \*(-1)

Некоторые преобразования приходится делать в несколько этапов. Например, увеличение сцены в определённом месте будет выполняться в три шага:

* Смещение начала координат в нужную позицию
* Изменение масштаба
* Возврат начала координат в исходное положение

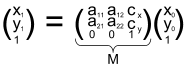
Виды координат:

* декартовы (x, y, z)
* обобщённые (x, y, z, w) — позволяют задать бесконечно удалённые точки (при w=0)

Связь:

* (x, y, z, w)об = (xдек, yдек, zдек, 1)
* (x, y, z)дек = (xоб, yоб, zоб) / w — не всегда возможно

**Аффинное преобразование** — преобразование, которое можно задать в виде:

A2X2 ; или 

Комбинацию аффинных преобразований можно представить как одно аффинное преобразование, матрица которого есть произведение матриц комбинируемых преобразований:

* x1 = M1x0
* x2 = M2x1
* …
* xn = Mnxn-1 = Mn(Mn-1xn-2) = Mn(Mn-1(Mn-2xn-3)) =...= (MnMn-1...M1)x0 = Mx0

Примеры аффинных преобразований координат:

* Мировые → видовые;
* Проекционные → экранные.

## Проецирование. Виды координат (модельные, видовые, проекционные, экранные)

Координаты бывают:

1. *Мировые* (точка в пространстве).
2. *Видовые* (получаются путём передвижения мировых).
3. *Проекционные* (относительно плоскости проецирования, Z — расстояние до объекта).
4. *Экранные* (координаты пикселей).

**Поверхность проецирования** — поверхность, на которой строится изображение (обычно плоскость).

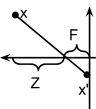
**Проецирование** — процесс (задача) сопоставления каждой точке трёхмерной сцены некоторой точки поверхности проецирования.

**Луч проецирования** — луч, который идет из объекта к поверхности проецирования.

***Виды проецирования:***

* **Параллельное**: все лучи проецирования параллельны друг другу. Подвиды:
  + **Косоугольное**: лучи не перпендикулярны плоскости проецирования.
  + **Аксонометрическое**: лучи перпендикулярны плоскости проецирования. Если использовать систему координат (x,y,z), в которой z — это расстояние от поверхности проецирования до объекта, то все лучи будут параллельны оси Z (hi, co?). В зависимости от соотношения коэффициентов масштабирования kx, ky, kz (по осям X,Y,Z соответственно) выделяют следующие модификации:
    - **Изометрическое**: kx = ky = kz.
    - **Диметрическое**: kx = ky ≠ kz.
    - **Триметрическое**: kx ≠ ky ≠ kz.
* **Перспективное**: все лучи проецирования проходят через одну точку, они не параллельны. Чем меньше фокусное расстояние, тем меньше размер проекции на плоскости.

С фокусным расстоянием связано увеличение и угол обзора. Чем меньше фокусное расстояние, тем больше угол обзора, и наоборот.

Фокусное расстояние человеческого глаза приблизительно равно 22 мм; каждый глаз по отдельности имеет угол зрения порядка 120-200°, соответственно, зона перекрытия двух глаз составляет порядка 130° — она практически настолько же широка, как у объектива типа "рыбий глаз".

Однако по эволюционным причинам наше периферийное зрение пригодно только для обнаружения движения и крупных объектов. Наш центральный угол зрения — порядка 40-60° — максимально влияет на наше восприятие. Субъективно это соотносится с углом, в пределах которого вы сможете вспомнить объекты, не двигая глазами. Это близко к углу зрения портретного объектива с фокусным расстоянием 52 мм.

**Кроп-фактор** — это отношение физического размера пленки к размеру матрицы, бывает от 1 до 2. **Full-frame** — это фотоаппарат, у которого размеры матрицы и пленки совпадают (кроп-фактор = 1).

Если *x* — координата объекта, а *x’* — координата на поверхности проецирования, то . Таким образом, перспективное преобразование можно задать как аффинное (с помощью матрицы 4х4 с обобщёнными координатами).

## Классификация способов визуализации (рендеринга) трёхмерных сцен

**Рендеринг** — это построение двумерного изображения трехмерной сцены согласно заданному положению камеры, освещению, моделей объектов и пр.

*Источники освещения:*

1. *Точечные*: свет выходит из одной точки (характеризуется координатами). Свет падает под разными углами даже в близкие точки объекта.
2. *Бесконечно удаленные источники* (характеризуется направлением). Все лучи параллельны друг другу, угол падения одинаковый у любых точек (например, Солнце).
3. *Фоновое освещение*: не знаем, откуда берутся лучи; их невозможно проследить, в каждой точке луч падает под всеми углами. Результат всех возможных отражений света.
4. *Излучение*: часть поверхности некоторого объекта, которая сама по себе светится, но никакие другие предметы не освещает.

Луч может отражаться, преломляться, поглощаться, а также комбинировать эти действия. Мы видим то, что попало в камеру. *Уравнение рендеринга:* для каждой точки объекта

интенсивность исходящего луча = интенсивность отраженного + преломленного - поглощенного луча = сумма всех входящих лучей – сумма всех поглощенных.

Чем больше препятствий на пути от источника до предмета и от предмета до камеры мы учитываем, тем качественнее изображение. Методы рендеринга различаются тем, что учитывается.

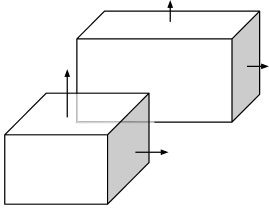
|  |  |
| --- | --- |
| **Метод** | **Что учитываем** |
| 1) Проволочная модель | Модель объекта и положение камеры |
| 2) Удаление невидимых линий (ребер)  а) с закраской  б) без закраски | Учитываем путь луча от объекта до камеры, наличие преград. Если с закраской, то закрашиваем без теней и оттенков. |
| 3) Учет освещения | Все грани рисуем закрашенными, цвет определяем исходя из параметров источника. |
| 4) Учет гладких поверхностей и текстур | Является ли грань частью гладкой поверхности, накладывается ли на эту грань изображение (текстура) |
| 5) Моделирование теней | Учитываются препятствия на пути луча от источника освещения до точек объекта. |
| 6) Учет отражений и преломлений | Ну тут и так всё понятно. |

## Алгоритм Робертса для многогранников

**Вход**: векторная модель объектов и положение камеры.

**Выход**: набор отрезков, которые надо нарисовать - удаляются невидимые грани (в экранных координатах).

Алгоритм:

1. Изначально множество отрезков пустое.
2. Для каждой грани вычисляется нормаль (векторное произведение двух её сторон) и с её помощью определяем ориентацию грани (направлена к нам “лицом” или нет). Обходим точки в направлении по часовой стрелке.
3. Из рассмотрения удаляются грани, нормаль которых “отвернута” от камеры: если скалярное произведение нормали на направление камеры (n, (0,0,1)) > 0, то угол острый, и грань не видна.
4. Все видимые грани сортируются в порядке удаления от камеры.
5. В цикле рассматриваются грани с самой дальней до самой ближней: для каждой грани просматривается множество уже накопленных отрезков (алгоритмом двумерного отсечения):
   1. Если отрезок полностью внутри грани, то он удаляется из результирующего множества
   2. Если отрезок полностью вне грани, то он остается в результате.
   3. Если отрезок пересекается с гранью, то он удаляется из множества, но вместо него добавляется видимая часть

После этого все ребра грани добавляются в множество, и берется следующая грань.

Проблема в двойном цикле: из-за этого низкое быстродействие.

## Алгоритм плавающего горизонта для отрисовки поверхностей

|  |  |
| --- | --- |
|  | Алгоритм применяется для поверхностей, заданных уравнением y = f(x,z). Поверхность можно задать набором функций y(x) = f(x, zi), i=0..n. Создаются два массива: максимальных и минимальных значений *y* для каждого *x*. |

Алгоритм:

1. Инициализируются массивы ymax[x] и ymin[x].
2. Рисуется y = f(x, zn) — это самая ближняя линия.
3. Цикл по *z* в порядке удаления от камеры
   * Цикл по *x:*
     + Вычисляется *y* = *f(x, z)* и проверяется:
       - Если *y* > *y*max[*x*], то изменяем ymax[x] и рисуем точку (*x, y, z)*.
       - Если *y* < *y*min[*x*], то изменяем ymin[x] и рисуем точку (*x, y, z)*.
       - Иначе точка не рисуется

Главный недостаток алгоритма — его узкая специализированность.

## Алгоритм с сортировкой по глубине

**Вход**: векторная модель (параметры сцены, расположение камеры).

**Выход**: растр.

Для каждой грани вычисляется удалённость (Z). Все грани сортируются по убыванию, и начиная с самой дальней, они рисуются с закраской цветом.

Недостатки:

* Низкая скорость, т. к. один и тот же пиксель перерисовывается несколько раз.
* Иногда невозможно определить, какая грань ближе, а какая дальше (например, пересекающиеся грани будут отрисованы неверно).

## Алгоритм Z-буфера и построчного сканирования

**Вход**: векторная модель (параметры сцены, расположение камеры)

**Выход**: растр.

Каждая грань рисуется модифицированным алгоритмом закраски многоугольника. Для каждого пикселя растрового изображения хранится ещё и z — расстояние от камеры до пикселя.

Если проецирование ведётся на плоскость XY, а проецируемая грань (треугольник) лежит в плоскости {Ax + By + Cz + D = 0}, то в точку (x,y) экрана будет спроецирована точка (x, y, (– Ax – By – D) / C), причём последняя координата будет обозначать расстояние до камеры.

|  |
| --- |
|  |

Изначально матрица значений Z (Z-буфер) инициализируется некоторым очень большим числом.

* Если Z рисуемого пикселя меньше соответствующего ему элемента в Z-буфере, то пиксель рисуется, и в Z-буфер заносится его значение Z.
* Иначе пиксель пропускается.

Для ускорения вычислений Z можно вычислять его только для одной вершины, а затем добавлять частные производные:

Достоинства:

* Простота реализации.
* Приемлемость для сцен любой сложности.
* Возможность распараллеливания (делим изображение на несколько частей, и каждую обрабатываем в отдельном потоке).

Недостатки:

* Большие объемы памяти
* Z — это вещественное число, вычисление производится с высокой точностью
* При реализации полупрозрачности требуется несколько z: z1, z2... ⇒ еще больше ресурсов. В этом случае можно рисовать сначала все непрозрачные объекты, а потом — все полупрозрачные

**Алгоритм построчного сканирования** (модификация предыдущего алгоритма).

Делаем полигон объектом, который имеет методы:

1. Init
2. GetCurrentY — на какой строке сейчас стоит алгоритм
3. DrawCurrentLine — рисует текущую часть объекта
4. bool next — существует ли следующая строка

В цикле по всем строкам изображения проверяем, пересекает ли текущая строка какой-нибудь полигон, и если да, то рисуется строка полигона (DrawCurrentLine и next) с использованием Z-буфера.

Достоинства:

* храним Z-буфер только для текущей сканируемой строки ⇒ экономия памяти.

## Алгоритм Варнока

**Вход**: векторная модель (параметры сцены, расположение камеры).

**Выход**: растр.

Для каждой грани вычисляются экранные координаты её вершин. Рассматривается множество многоугольников и прямоугольное окно.

|  |  |
| --- | --- |
| Если все многоугольники внешние, то закрашиваем окно цветом фона. |  |
| Если только один многоугольник пересекается с окном, то выполняется его отсечение (окном) и рисуется только видимая часть. |  |
| Если с окном пересекается несколько многоугольников, но при этом самый ближний из них является охватывающим (окно целиком лежит внутри него), то окно закрашивается цветом этого многоугольника. |  |
| Если размер окна = 1 пиксель, то он закрашивается цветом ближайшего многоугольника и на этом работа алгоритма прекращается. | *янезнаюкакэтонарисовать.jpg* |
| Если с окном пересекается несколько многоугольников и ни один из них не является охватывающим, то окно делится на четыре части и для каждой части опять запускается алгоритм с начала. |  |

Замечания

* Глубина рекурсии оказывается небольшой (при размерах 1024х1024 глубина = 10)
* Алгоритм может быть распараллелен.

## Алгоритм Вейлера-Айзертона

На входе и на выходе векторная модель.

1. Все полигоны сортируются по глубине и находится самый близкий к камере многоугольник.
2. Все многоугольники отсекаются этим ближайшим многоугольником и формируются два списка: внутренний (то, что внутри отсекающего многоугольника) и внешний (то, что вне отсекающего многоугольника)
3. Если внутренний список пуст, то отсекающий многоугольник ни с чем не пересекается. Добавляем его в результат, и весь алгоритм выполняется для внешнего списка.
4. Иначе, сравниваются Z отсекающего многоугольника и каждого из внутреннего списка:
   * Если там есть хотя бы один многоугольник, который ближе отсекающего, то к нему рекурсивно применяется весь алгоритм.
   * Иначе отсекающий многоугольник заносится в результат и весь алгоритм выполняется для внешнего списка

## Задание на лабораторную работу 6

**Задание 6\_72**

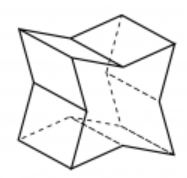
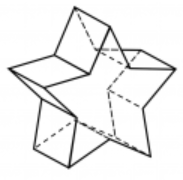
Написать программу, рисующую проекцию трехмерного каркасного объекта.

**Требования**

1. Грани объекта рисуются с помощью доступных функций рисования отрезка в координатах окна. При этом использовать шейдеры GLSL и OpenGL
2. Вывод многогранника с удалением или прорисовкой невидимых граней;
3. Ортогональное и перспективное проецирование;
4. перемещения, повороты и масштабирование многогранника по каждой из осей независимо от остальных.
5. Генерация многогранника с заданной мелкостью разбиения.
6. Д.б. установлено изменение свойств источника света (интенсивность).
7. При запуске программы объект сразу должно быть хорошо виден.
8. Пользователь имеет возможность вращать фигуру (2 степени свободы) и изменять параметры фигуры.
9. Возможно изменять положение наблюдателя.

.

**Звезды**

****

**63. 11-угольная**.