УДК 51

ПРИМЕНЕНИЕ МАТЕМАТИКИ В ОБРАБОТКЕ, ХРАНЕНИИ И РЕНДЕРЕ 3D ОБЪЕКТОВ

Сафонова А.В., студентка гр. ПИб-191, II курс
Романин К.П., студент гр. ПИб-191, II курс
Гутова Е.В., ст. преподаватель кафедры математики
Кузбасский государственный технический университет
имени Т.Ф. Горбачева,
г. Кемерово

Абсолютно все процессы, проходящие на программном уровне в компьютере связаны с математикой. Начиная простейшими операциями с двоичными величинами и заканчивая плавным движением персонажа в вашей любимой игре. О втором пункте и будет идти речь. В этом докладе разобраны основы работы с трехмерными моделями в компьютерах со стороны использующихся в них математических формул и законов.

Для того, чтобы создать и сохранить трехмерную модель нужно сначала определить область, в которой она будет находится. Обратившись к геометрии можно найти несколько возможных систем координат, которые способны хранить трехмерные объекты: Декартова с тремя осями, цилиндрическая и сферическая. Последние две реализовать вполне возможно, но для определения координат точек в пространстве, потребуются расчеты с использованием тригонометрических функций, что существенно замедлит обработку даже относительно простых сцен. Декартова же система предоставляет возможность хранить и обрабатывать большое количество точек, без надобности в дополнительных вычислениях.

В созданном пространстве сохраняется два типа объектов: точки и вектора. Под точкой в пространстве принято понимать какое-то местоположение, заданное в виде трех координат, соответственно трем осям: абсцисс (X), ординат (Y) и аппликат (Z). Вектор же имеет тот же набор параметров, но обозначает уже не конкретное место в пространстве, а направление.

Определившись с системой, которая будет определять положение и размер объектов, можно построить простейшую геометрическую фигура, например, куб. Для этого достаточно сохранить в системе 8 точек с разными координатами, которые имеют одинаковое расстояние между двумя соседними точками. Таким образом хранятся абсолютно все модели начиная от геометрических фигур и заканчивая оцифрованным лицом актера. Для удобного хранения и последующей обработки координаты всех его вершин объединяются в одну матрицу размером 3×8. С помощью таких матриц становится легче определять принадлежность точек к отдельным фигурам, но их главная задача – это упрощение процессов перемещения и вращения.

Теперь попробуем получить немного больше контроля над моделью, перемещение в пространстве самый базовый из возможных инструментов присущих программам для обработки 3D графики. На этом этапе понадобятся векторы. Как уже описывалось ранее, они не имеют координаты начала, а используются только для того, чтобы указать направление и длину шага, на который будет сдвигаться модель. Рассмотрим пример: Допустим необходимо сдвинуть модель на 5 условных единиц расстояния (допустим сантиметров) по положительному направлению оси абсцисс (X), для этого используется вектор $\{5,\ 0,\ 0\}$. К координатам каждой из вершин куба прибавляется значение вектора. Получается простая формула: (старые координаты) + (значение вектора) = (новые координаты).

После того успешного перемещения модели в пространстве, было бы неплохо научится еще и вращать ее. Это завершит базовый набор инструментов, на основе которых можно создавать полноценные анимации, а также управлять камерой изменением тангажа, рысканья и крена. Для того-чтобы повернуть модель нужно определить ось вращения и угол на который мы будем смещать вершины модели по окружности.

Работа с углами и движением по окружности сразу отсылает к тригонометрии, а именно функций sin и cos которые используются в следующих формулах:

$$x_1 = x_0 \cdot \cos\alpha - y_0 \cdot \sin\alpha$$
$$y_1 = y_0 \cdot \cos\alpha - x_0 \cdot \sin\alpha$$

Как видно из самих формул, они позволяют найти новые координаты точки x1, после поворота на угол alpha, но только для двумерной плоскости. Для третьего измерения меняется немного: Для того, чтобы повернуть трехмерную модель нужно зафиксировать положение ее вершин по одной из осей, при этом изменяя координаты по другим. Таким образом, получается, что вращение в плоскости XOY будет осуществляться по следующей формуле:

$$x_1 = x_0 \cdot \cos\alpha - y_0 \cdot \sin\alpha$$
$$y_1 = y_0 \cdot \cos\alpha - x_0 \cdot \sin\alpha$$
$$z_1 = z_0$$

И по аналогии для вращения в плоскостях XOZ и YOZ. Фиксированной будет координаты по y и по x соответственно.

Теперь, когда созданную модель можно двигать и вращать пора заняться второй по важности вещью в 3D моделировании - самообманом. Дело в том, что мозг человека воспринимает тела объемными, благодаря разности освещенности их поверхности в зависимости от типа материала и расположения источника освещения. Если попытаться сейчас создать инструмент по выводу на экран полноценного изображения, то, так как на созданной сцене нет никаких источников света, в лучшем случае получится однотонный квадрат, а если куб будет повернут к камере не под прямым углом, к какой-либо из сторон, то получится однотонный многоугольник. Чтобы избежать этой проблемы, нужно создать источник освещения, который будет делать светлее поверхности, направленные на него, и затемнять направленные

по вектору освещения. Для этого нужно сначала узнать в какие стороны направлены поверхности, их нормали.

Нормаль к грани – это вектор, который идет из центра модели к ее грани под прямым углом. Для того чтобы вычислить направления векторов нормали для куба (нельзя вручную задать значения для этих векторов, так как куб может вращаться) нужно транспонировать обратную матрицу куба. Полученная матрица нормалей сравнивается с вектором источника освещения с помощью разницы векторов: чем меньше полученный в результате вектор, тем более светлой должна быть поверхность. Также выставляется ограничитель, который не позволяет изменять яркость поверхностей, которые повернуты от источника света более чем на 90°.

Попробуем уйти от наблюдения за созданным нами кубом в виде массива координат его вершин. Займемся камерой. Камера это, пожалуй, самый важный объект в любом 3D движке, она позволяет создать плоскую проекцию объемного мира, смоделированного нами в памяти компьютера. Из множества различных методов реализации камеры выберем raycast. Идея этого метода довольно проста: допустим, необходимо получить картинку размером 1920×1080 пикселей, для этого понадобится всего лишь 2073600 раз отправить из точки, в которой будет располагаться камера, по направлению ее взгляда, единичный вектор, постепенно смещая направление.

Для каждого отправленного вектора требуется проверить условие пересечения этого вектора с кубом. Если регистрируется пересечение, то соответствующий пиксель красится в какой-либо цвет, например, синий. Тут же проводится проверка освещенности поверхности, на ее основе увеличиваем яркость пикселя в зависимости угла поворота к источнику света.

Самый наглядный способ нахождения пересечений между лучом и кубом – это нахождение пересечений с какой-либо из шести граней куба. При поиске пересечения с плоскостью нужно убедится, что точка пересечения находится внутри соответствующей грани. Для этого сначала находится пересечение с плоскостью, а после проверяется входит ли точка в грань. Проверить это можно несколькими способами. Один из них заключается в вычислении площадей четырехугольника и четырех треугольников. Если сумма площадей четырех треугольников, $(S_1 + S_2 + S_3 + S_4)$, равна площади четырехугольника S, то точка принадлежит грани.

Создавать кубы в пространстве довольно просто, но что на счет модели машины? Такая модель почти не имеет ровных и прямых плоскостей и если попытаться взять куб за единицу строения модели (воксель), то получится воксельная графика - угловатая, и сильно ограниченная. Возникает вопрос: как же современные программы позволяют создавать такие красивые и детализированное модели? Для этого используется другой метод их строения – полигональная сетка. Его смысл заключается в создании множества небольших плоскостей (полигонов) соединенных друг с другом в вершинах. Таким образом можно создавать имитацию круглых, рельефных, плоских и многих других моделей, а лучшая детализация легко достигается с ростом

числа полигонов, количество которых, теоретически, можно наращивать вплоть до бесконечности.