Построение корректного освещения с использованием теневых фрагментов.

23 марта 2008 г.

Введение

В трёхмерных играх обычно требуется создавать качественное освещение. На сегодня стандартным способом для создания такого освещения являются карты освещённости либо используется технология теневых объёмов (что наглядно продемонстрировал DOOM3). К сожалению, первый способ требует большого количества памяти для карт, поскольку для красивого, качественного освещения необходимо достаточно высокое разрешение карты освещённости и плохо подходит для "динамического" освещения (под "динамическим" здесь мы будем понимать мигающие источники света, изменения цвета освещения, но не перемещение источника света в пространстве). Второй же способ требует большой нагрузки на процессор и видеокарту, поскольку освещение вычисляется непрерывно. Однако, возможен ещё один способ построения качественного освещения, свободного от некоторых недостатков вышеперечисленных методов. Этот способ, однако, не подходит для перемещающихся источников света, но для статичных источников света качество полученного освещения будет очень хорошим, при этом окажется возможным использовать "динамические" источники света.

Суть этого способа в том, что у OpenGL есть поддержка как минимум восьми источников освещения. Если для каждой грани виртуального мира составить список освещающих её источников и выбрать из них 8 наиболее значительных, то при выводе грани можно будет включать эти источники и оперативно их настраивать, после чего уже выводить саму грань. Определить такие источники несложно на этапе подготовки карты и сделать это надо всего один раз, а потом можно уже просто пользоваться готовым списком. Для реализации такого освещения потребуется алгоритм, позволяющий построить тень от выпуклого многоугольника на другом выпуклом многоугольнике. Далее мы рассмотрим возможную реализацию такого алгоритма. Предполагается, что читатель знаком с основами векторной геометрии; если это не так, рекомендуется обратиться к соответствующей литературе по математике.

1 Используемые функции.

Для реализации алгоритма нам понадобятся некоторые функции

1.1 Функция, определяющая позицию точки относительно некоторой плоскости.

Входные данные:

- Точка плоскости $P(P_x, P_y, P_z)$;

- Вектор нормали к плоскости $n(n_x, n_y, n_z)$;

- Точка, для которой определяется положение относительно плоскости $V(V_x, V_y, V_z)$.

Возвращаемое значение: 1 - точка находится выше плоскости, -1 - точка находится ниже плоскости, 0- точка находится в плоскости.

Для реализации этой функции воспользуемся уравнением плоскости, тогда

$$t = n_x(V_x - P_x) + n_y(V_y - P_y) + n_z(V_z - P_z). \quad (1)$$

Полученное t даёт расстояние от точки V до плоскости в единицах длины вектора нормали. Следовательно, функция должна возвращать -1, если $t < 0$, возвращать 1, если $t > 0$ и возвращать 0, если $t = 0$. Однако, мы работаем на разрядной сетке ЭВМ, поэтому сравнивать с нулём нельзя. Вместо этого мы будем здесь и далее сравнивать с некоторым малым ε, значение которого проще всего подобрать опытным путём так, чтобы алгоритм не давал сбоев. В этом случае функция должна возвращать -1, если $t < -\varepsilon$, возвращать 1, если $t > \varepsilon$ и возвращать 0 во всех остальных случаях.

1.2 Функция, определяющая точку пересечения прямой с плоскостью.

Входные данные:

- Точка плоскости $P(P_x, P_y, P_z)$;
- Вектор нормали к плоскости $n(n_x, n_y, n_z)$;
- Первая точка прямой $A(A_x, A_y, A_z)$;
- Вторая точка прямой $B(B_x, B_y, B_z)$;

Возвращаемое значение: координаты точки пересечения либо флаг, показывающий что пересечения нет.

Вектор $L(L_x, L_y, L_z)$, задающий направление прямой вычисляется как

$$\begin{aligned} L_x &= B_x - A_x, \\ L_y &= B_y - A_y, \\ L_z &= B_z - A_z. \end{aligned}$$

Тогда скалярное произведение между вектором нормали и вектором прямой будет $E = L_x \cdot n_x + L_y \cdot n_y + L_z \cdot n_z$. Если $E = 0$ (в нашем случае это условие примет вид $E \geq -\varepsilon$и $E \leq \varepsilon$), то прямая и плоскость пересекаться не могут, поскольку вектора перпендикулярны. Если же пересечение возможно, то координаты точки пересечения могут быть получены по формулам

$$\begin{aligned} V_x &= A_x - L_x \frac{t}{E} \\ V_y &= A_y - L_y \frac{t}{E}, \\ V_z &= A_z - L_z \frac{t}{E}. \end{aligned}$$

где t- расстояние от точки до плоскости, полученное по формуле 1.

1.3 Функция, определяющая точку пересечения луча с плоскостью.

Входные данные:

- Точка плоскости $P(P_x, P_y, P_z)$;
- Вектор нормали к плоскости $n(n_x, n_y, n_z)$;
- Первая точка луча (начало луча) $A(A_x, A_y, A_z)$;
- Вторая точка луча $B(B_x, B_y, B_z)$.

Возвращаемое значение: координаты точки пересечения либо флаг, показывающий что пересечения нет.

Задача сводится к предыдущей, но появляются дополнительные условия возможности пересечения, поскольку луч, в отличие от прямой, полубесконечен. Так, если получится, что$-t < E$ для $E > 0$ или $t > E$ для $E < 0$ (в нашем случае $-t < E + \varepsilon$ для $E > 0$ или $t > E - \varepsilon$ для $E < 0$), то луч плоскость не пересекает.

1.4 Функция, определяющая, нахождение точки внутри выпуклого многоугольника, лежащего в плоскости.

Входные данные:

- Точки многоугольника, лежащие в одной плоскости $P0(P0_x, P0_y, P0_z)...Pn(Pn_x, Pn_y, Pn_z)$;

- Точка, находящаяся в плоскости многоугольника, для которой проверяется её нахождение внутри многоугольника $V(V_x, V_y, V_z)$.

Возвращаемое значение: флаг, показывающий находится точка внутри многоугольника или нет.

Алгоритм работы функции выглядит так:

Введём переменную a=индекс последней точки многоугольника;

Цикл по i от 0 до a;

Введём переменные индексов трёх последовательных точек $i0 = i$,$i1 = i + 1$,$i2 = i + 2$;

Если $i1 > a$, то $i1 = i1 - a - 1$;

Если $i2 > a$, то $i2 = i2 - a - 1$;

Возьмём три последовательные точки многоугольника $A = P(i0)$,$B = P(i1)$,$C = P(i2)$;

Создадим три вектора: $F(F_x, F_y, F_z), G(G_x, G_y, G_z), H(H_x, H_y, H_z)$ по формулам

$$\begin{aligned} F_x &= B_x - A_x, \\ F_y &= B_y - A_y, \\ F_z &= B_z - A_z. \end{aligned}$$

$$\begin{aligned} G_x &= C_x - A_x, \\ G_y &= C_y - A_y, \\ G_z &= C_z - A_z. \end{aligned}$$

$$\begin{aligned} H_x &= V_x - A_x, \\ H_y &= V_y - A_y, \\ H_z &= V_z - A_z. \end{aligned}$$

Вычислим векторные произведения $N1(N1_x, N1_y, N1_z) = F \times G$ и $N2(N2_x, N2_y, N2_z) = F \times H$;

Вычислим скалярное произведение $R = N1_x * N2_x + N1_y * N2_y + N1_z * N2_z$;

Если $R < 0$(в нашем случае $R < -\varepsilon$), то точка внутрь многоугольника не попадает.

Конец цикла;

Точка попадает внутрь многоугольника.

Идея работы функции заключается в том, что три подряд идущие точки многоугольника задают два ребра многоугольника, преобразовав которые в вектора, можно вычислить векторное произведение векторов первого ребра со вторым и первого ребра с вектором, составленным из первой точки многоугольника и исследуемой точкой. При этом, направления получившихся векторов зависят от того, по какую сторону от первого ребра находится точка. Для вычисления согласованности получившихся векторов используется их скалярное произведение, которое будет отрицательным, если вектора направлены в противоположные стороны.

2 Алгоритм построения теневых фрагментов

Входные данные:

- Точки $PS(PS0_x, PS0_y, PS0_z)...PSn(PSn_x, PSn_y, PSn_z)$ выпуклого многоугольника, являющегося источником тени (далее по тексту МИТ - "многоугольник, источник тени"), лежащие в одной плоскости;

- Точки $PD(PD0_x, PD0_y, PD0_z)...PDn(PDn_x, PDn_y, PDn_z)$ выпуклого многоугольника, на который отбрасывается тень (далее по тексту МПТ - "многоугольник, приёмник тени"), лежащие в одной плоскости;

-Точка положения источника света $L(L_x, L_y, L_z)$ (далее по тексту ИС - "источник света");

Возвращаемое данные: флаг, показывающий существует тень или нет, и массив многоугольников, на которые разбивается МПТ тенью.

2.1 Получение точек тени в плоскости МПТ

Рассмотрим алгоритм получения точек тени в системе координат МПТ.

1. Проверим, что МИТ и МПТ имеют не менее трёх точек. Иначе это, строго говоря, вообще не многоугольники, и тени нет.

2. Проверим, может ли ИС освещать МПТ. Условимся, что если ИС лежит выше плоскости МПТ (нормаль МПТ направлена в сторону ИС), то он может освещать МПТ, иначе, это невозможно. Если ИС освещать МПТ не может, то тени нет.

3. Поскольку МИТ может пересекать МПТ, то нужно скорректировать МИТ так, чтобы он всегда был выше МПТ. Для этого нужно пройтись циклом по рёбрам МИТ, оставляя точки, которые лежат выше плоскости МПТ и отбрасывая точки, лежащие ниже плоскости МПТ, причём, если точки какого-либо ребра находятся по разные стороны от плоскости, задаваемой МПТ, то нужно добавить к новому МИТ эти точки пересечения. Таким образом, у нас будет новый МИТ, лежащий точно выше МПТ. Если после коррекции точек МИТ меньше трёх, то тени нет.

4. Возможна ситуация, когда МПТ и ИС находятся по одну сторону от МИТ. В этом случае тоже тени нет. Это можно проверить определив положение ИС относительно плоскости МИТ и сравнивая с этим положением положения точек МПТ относительно плоскости МИТ. Если хоть одна точка МПТ лежит по другую сторону плоскости МИТ, нежели ИС, то тень существует, иначе тени нет.

5. Теперь нужно сформировать теневой объём. Всё, что попадает в теневой объём, ИС не освещается. Фигура, задающая теневой объём, ограничивается сверху МИТ, снизу её ничто не ограничивает (так как объём простирается в бесконечность), а по бокам её ограничивают четырёхугольные грани, у которых две вершины так же находятся в бесконечности, а две другие совпадают с вершинами ребра МИТ. Поскольку МИТ уже задан, то требуется найти четырёхугольные грани. Так как нам от этих граней нужно сейчас только то, что они задают плоскости, то заменим бесконечные вершины на конечные. Для этого возьмём рёбра МИТ, и пустим луч от ИС через вершины рёбер. Для примера возьмём первые две точки МИТ $PS0$ и $PS1$, тогда координаты точек

грани теневого объёма $V0, V1, V2, V3$ определяются как

$$
\begin{aligned}
V0_x &= PS0_x + N0_x;\\
V0_y &= PS0_y + N0_y;\\
V0_z &= PS0_z + N0_z;\\
V1_x &= PS0_x;\\
V1_y &= PS0_y;\\
V1_z &= PS0_z;\\
V2_x &= PS1_x;\\
V2_y &= PS1_y;\\
V2_z &= PS1_z;\\
V3_x &= PS1_x + N1_x;\\
V3_y &= PS1_y + N1_y;\\
V3_z &= PS1_z + N1_z,
\end{aligned}
$$

где

$$
\begin{aligned}
N0_x &= PS0_x - L_x;\\
N0_y &= PS0_y - L_y;\\
N0_z &= PS0_z - L_z;\\
N1_x &= PS1_x - L_x;\\
N1_y &= PS1_y - L_y;\\
N1_z &= PS1_z - L_z.
\end{aligned}
$$

6. Поскольку МИТ мог быть ориентирован как угодно, то нормали граней теневого объёма могут быть направлены как внутрь объёма, так и наружу. Условимся, что нормали теневого объёма должны быть направлены наружу. Для этого сравним положение точки, точно попадающей в теневой объём относительно плоскостей, задаваемых гранями теневого объёма. Такая точка может быть получена, если найти геометрический центр МИТ

$$
\begin{aligned}
x_c &= \frac{\sum_{i=0}^{max} PSi_x}{max};\\
y_c &= \frac{\sum_{i=0}^{max} PSi_y}{max};\\
z_c &= \frac{\sum_{i=0}^{max} PSi_z}{max}.
\end{aligned}
$$

и пустить луч от ИС через этот центр, так же, как это делалось для нахождения вершин граней теневого объёма. Если при определении положения этой точки относительно плоскостей граней будет получено для какой-либо грани, что точка находится выше выбранной грани, то грань следует "перевернуть", т.е. поменять знаки координат вектора нормали грани на противоположные. Направление обхода самих точек, задающих грань можно не менять, так как это в дальнейшем не понадобится.

7. Теперь мы должны найти точки тени. Сначала мы найдём точки МПТ, лежащие внутри теневого объёма. Делается это так: каждую точку МПТ нужно сравнить со всеми плоскостями, задаваемыми гранями теневого объёма, и если окажется, что для всех граней она находится ниже плоскости грани (или в грани), то такая точка точно лежит внутри теневого объёма. После этого надо найти точки пересечения теневого объёма с МПТ. Для этого мы для каждой грани теневого объёма проходим по всем рёбрам МПТ и находим пересечения ребёр с плоскостью данной конкретной грани, проверяя, попадает ли точка пересечения внутрь теневого объёма. Однако, это ещё не все точки из которых может собираться тень. Нужны ещё точки пересечения граней теневого объёма с МПТ. Для того, чтобы их найти, нам нужно вычислить точки пересечения плоскости МПТ и лучей, выходящих из ИС и проходящих через вершины МИТ, проверив, что все полученные точки лежат внутри МПТ. В конечном итоге, у нас должен получиться массив точек, из которых собирается тень на плоскости МПТ. Однако эти точки ещё не упорядочены.