7.2. Алгоритмы, основанные на использовании BSP-деревьев

В настоящее время алгоритмы этой группы являются наиболее распространенными. Множество существующих алгоритмов этой группы можно разделит на алгоритмы, использующие 2,5D-пространства и алгоритмы, использующие 3D-пространства.

7.2.1. 2,5D-алгоритмы

Алгоритмы данной группы позволяют строить изображения трехмерных сцен, используя описания плоских слоев. В них невозможно реализовать полную трехмерность, так как само их ядро не позволяет размещать один слой описания сцены над другим. В отличие от алгоритмов отсечения лучей, базовым описанием сцены является не регулярная схема, а схема, состоящая из объектов произвольной формы (точнее – имеющих плоскую проекцию произвольной формы).

Описание сцены состоит из вершин и соединяющих их отрезков. Отрезки образуют замкнутый многогранник, который разбивает сцену на секторы. Каждый сектор при этом описывается совокупностью следующих данных:

1) высота пола;
2) высота потолка;
3) набор базовых текстур для пола, потолка и боковых граней.

Каждая точка на плоской схеме объектного пространства соответствует вертикальной линии, которая пронизывает объекты сцены, при этом могут возникать три вида пересечений:

1) две точки (пол и потолок);
2) отрезок (при прохождении линии внутри стены);
3) два отрезка (при попадании на стык двух стен).

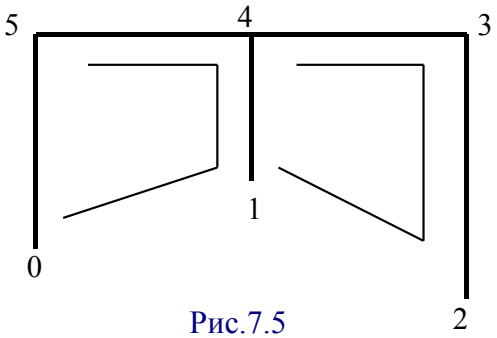


Рис.7.5

Многоугольники, которые определяют секторы, рассматриваются состоящими из направленных отрезков. Направление выбирается так, чтобы сектор располагался всегда по одну сторону от отрезков, составляющих многоугольник (рис.7.5).

При этом могут существовать двусторонние отрезки, для которых сектор может располагаться как слева, так и справа. На рисунке - это отрезок 1-4 или 4-1. С каждой стороной сектора связана текстура, обеспечивающая реалистичное отображение вертикальных граней. При этом различают три вида текстур, накладываемых на вертикальные поверхности (см. рис.7.6).

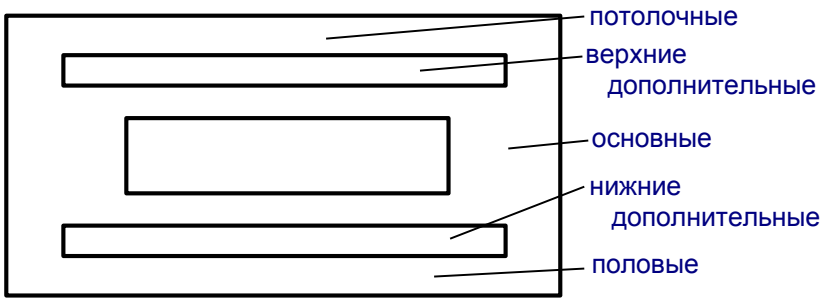


Рис.7.6

1) основные, для заполнения основных стен объектов сцены.
2) нижние дополнительные текстуры, для заполнения промежутков, соединяющих полы разного уровня.
3) верхние дополнительные текстуры, позволяющие заполнить расстояние между потолками разного уровня.

Перед выполнением процедуры рендеринга все секторы преобразуются в выпуклые области. Кроме того, осуществляется сортировка в порядке удаления от наблюдателя (*FTB* - процедура).

При построении разбиений и упорядочивания секторов используется один из вариантов *BSP*-дерева.

Алгоритмы двоичного разбиения пространства

Алгоритмы данной группы относятся к алгоритмам сортировки многоугольников. Принцип работы эти многоугольников заключается в следующем: если в объектном пространстве провести вертикальную плоскость, то она разделит все объектное пространство на два полупространства. Если продолжать разбиение полупространств далее, располагая в каждом новом полупространстве по одному объекту сцены, то в итоге получится разбиение пространства на кластеры, внутри которых находится по одному многоугольнику или грани (рис.7.7).

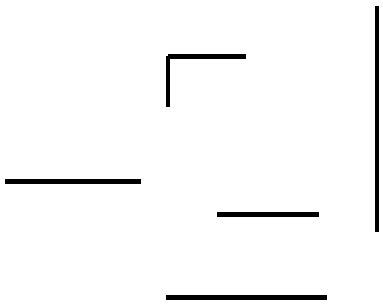


Рис.7.7

Отличительной особенностью этого разбиения является то, что элемент, расположенный в полупространстве, где нет наблюдателя, не могут перекрывать элементы полупространства, в котором находится наблюдатель. Это позволяет значительно сократить объем перебора взаимных перекрытий объектов.

На практике разбиение пространства продолжится до тех пор, пока в каждом полупространстве (кластере) не останется одна грань. Последующее отображение граней, начиная с самого дальнего кластера, позволяет получить корректные проекции трехмерных объектов.

Алгоритмы просмотра полупространств (граней) можно представить в виде двоичного дерева, описывающего разбиение пространств. Это дерево называется *Binary Space Partitioning (BSP) Tree*. Узлами *BSP*-дерева являются плоскости, вдоль которых производится разбиение. Каждая плоскость описывается уравнением Ax+By+Cz=D или в нормальном вида $\bar{p} \cdot \bar{n} = D$, где $\bar{n}$ - вектор нормали к поверхности.

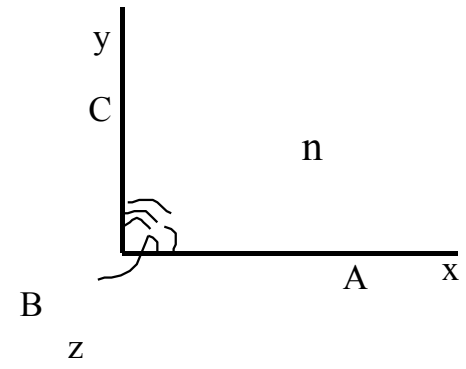


Рис.8.8.

Каждый узел BSP-дерева взвешивается информацией, оформленной в виде следующей структуры:

1) идентификатор грани, вдоль которой проходит плоскость разбиения;
2) координаты грани;
3) нормаль к поверхности грани;
4) указатель на вершину BSP-дерева, связанную с соседним положительным полупространством, то есть для которого $\bar{p} \cdot \bar{n} > D$;
5) указатель на вершину из отрицательного полупространства $\bar{p} \cdot \bar{n} < D$.

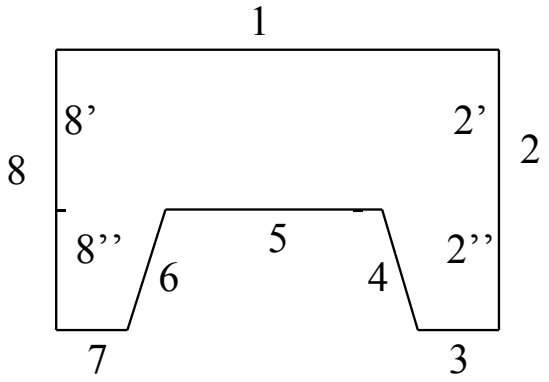
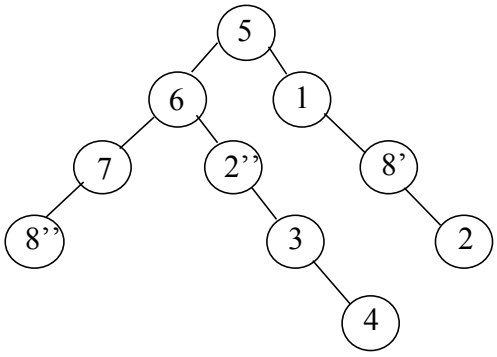


Рис.7.9.

Обычно в качестве разбивающей плоскости выбирается плоскость, проходящая через одну из граней объектов сцены. Все грани, пересекаемые этой плоскостью, разбиваются на две части, которые помещаются в соответствующие ветви дерева (поддеревья).

Если разбивающая грань группы 5, то она делит грани 2 и 8 на две части. После этого деления образовалось два кластера (верхний и нижний). В верхнем выберем за плоскость разбиения грань 1, а в нижнем – грань 6.



В нижнем кластере после деления по направлению 6-ой грани кластер делится на две части: левую и правую. Деление продолжается далее, пока в кластер не будет входить по одной грани. *BSP*-дерево для этого случая выглядит таким образом, как представлено на рис.7.10.

Особенность этого метода в том, что получающиеся при двоичном делении деревья сильно зависят от выбора на каждом шаге разбивающей грани.

Для приведенного примера можно выбрать другую последовательность разбиений, которая описывается другим *BSP*-деревом (рис.7.11). При этом не требуется разбиение граней.

Итерационный процесс построения *BSP*-деревьев реализуется за три шага:

1) выбор разбивающей плоскости или грани;
2) разбиение множества всех граней на две части: положительная или отрицательная. При этом может возникать дробление грани;
3) рекурсивное обращение для каждой из получившихся после деления частей.

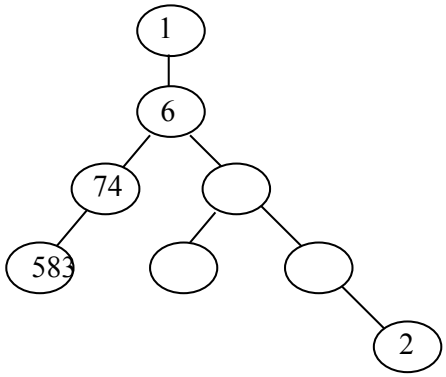


Рис.7.11.

Так как построение *BSP*-дерева для одной и той же сцены не однозначно, то возникает вопрос выбора построения оптимального *BSP*-дерева.

В качестве критерия оптимизации данного процесса может быть использован один из двух следующих показателей.

1. **Высота *BSP*-дерева** . Минимальная высота h_{min} гарантирует получение максимальной сбалансированности относительно левой и правой ветвей дерева. Это обеспечивает минимальное число проверок.

2. **Количество разбиения граней *S*** . Появление дополнительных граней, после их деления, увеличивает затраты по памяти и по времени на отображение сцены. Поэтому стремятся уменьшить этот показатель.

Критерии h_{min} и S_{min} являются взаимно противоречивыми. Поэтому на практике выбирают компромиссный вариант – аддитивный критерий, включающий в себя оба взвешенных критерия:

$$F = k_1 \cdot h_{min} + k_2 \cdot S_{min} .$$

Получить точное оптимальное решение почти невозможно из-за большого числа граней в реальных трехмерных сценах. Поэтому при программной реализации алгоритмов, основанных на *BSP*-деревьях, используют эвристические приемы, позволяющие сократить объем вычислений. Например, на каждом шаге итерационного алгоритма рассматриваются не все кандидаты на разбиение, а какая то их часть. Количество рассматриваемых граней-кандидатов и принцип отбора влияет на достоверность используемой эвристики.

После построения BSP-дерева можно приступать к выполнению процедуры рендеринга или визуализации. В алгоритмах данной группы используются два варианта алгоритма рендеринга:

1) *BTF*–алгоритм;
2) *FTB*–алгоритм.

BTF–алгоритмы позволяют получать окончательный результат наиболее естественным путем. В процессе рендеринга появляющиеся позже ближние объекты перекрывают удаленные. *Недостатком* этого варианта является большая трудоемкость, связанная с просчетом невидимых в итоге граней удаленных объектов.

Алгоритм *FTB* позволяет устранить этот недостаток, так как первыми на экран отображаются ближние к наблюдателю грани. Чтобы не выполнять лишних расчетов для невидимых дальних граней используется механизм отслеживания заполненных пикселей экрана. Для этого могут использоваться маски или счетчики. Как только все пиксели заполнены, обход *BSP*-дерева прекращается.

Недостатком *FTB*-алгоритма являются дополнительные вычислительные затраты на поддержку механизма отслеживания заполненных пикселей.

Достоинства алгоритмов, базирующихся на построении BSP-деревьев:

1) возможность визуализации нерегулярных пространств;
2) независимость BSP-дерева от расчетов проецирования, что удобно для построения видеоряда (в пределах одного сегмента).

Недостатки :

1) необходимость разбиения грани, что ведет к увеличению вычислительной сложности, особенно при обработке больших сцен;
2) нелокальность BSP-деревьев: незначительное изменение сцены может привести к изменению всего BSP-дерева;
3) невозможность добиться реалистичного освещения;
4) использование для движущихся объектов технологии спрайтов.

7.2.2. 3D-алгоритмы

От 2,5D-алгоритмов данная группа отличается тем, что позволяет обеспечивать следующие возможности:

1) полная трехмерность, то есть поддержка шести степеней свободы, реалистичность освещения;
2) поддержка трехмерных объектов и методов их анимации;
3) поддержка списка потенциально видимых граней (*PVS*), что позволяет повысить эффективность рендеринга в реальном масштабе времени.

При использовании алгоритмов данной группы все объекты делятся на две группы:

- статические объекты;
- динамические объекты.

Статическая часть сцены строится, как правило, при помощи очень большого числа граней и произвольного числа источников света. На предварительном этапе построения сцены для статических объектов производится построение *BSP*-дерева. При этом все пространство делится на набор выпуклых многоугольников, которые, в конечном счете, являются листьями *BSP*-дерева. Построенное один раз *BSP*-дерево может неоднократно использоваться в ходе моделирования виртуальной реальности, обеспечивая проход по вершинам в любом направлении. В отличие от 2,5D-алгоритмов, в данном случае для каждого элемента сцены задается параллелепипед. Внутри параллелепипеда присутствует множество граней с указанием их текстур и отражающих характеристик. Особенностью сцены виртуальной реальности является значительное общее количество граней. При этом число видимых граней составляет, как правило, достаточно малый процент от их общего числа. Данная особенность используется для того, чтобы сократить количество проверок, ускорив тем самым изображение моделируемой сцены.

Таким образом, алгоритмы данной группы базируются на отбрасывании из активного списка заведомо невидимых граней. При чем, чем эффективнее осуществляется анализ списка граней, тем выше быстродействие конкретного алгоритма.

Вся сцена разбивается на множество выпуклых областей. Для каждой из выделенных областей определяется список граней, видимых из этой области. Данный список носит название **списка потенциально видимых граней**(*Potentially Visible Set - PVS*). Использование PVS требует определенных затрат на предварительных этапах работы алгоритмов. Однако, затем присутствие этого списка обеспечивает эффективную работу в реальном времени. Существует множество алгоритмов, которые поддерживают PVS. Для удаления невидимых граней может использоваться один из традиционных алгоритмов (например, алгоритм Z-буфера). К наиболее известным алгоритмам данной группы относятся алгоритмы, основанные на методе порталов и алгоритмы, базирующиеся на иерархических сценах.

7.2.2.1. Метод порталов

Метод порталов позволяет осуществлять построение виртуальной реальности «на лету». Исходная сцена по методу порталов разбивается на множество выпуклых областей. Места соединения между собой, через которые можно видеть область друг из друга, называются порталами. При построении PVS в первую очередь в него заносятся все грани текущей области. На следующем шаге рассматриваются порталы (окна, двери), соединяющие текущую область с соседними.

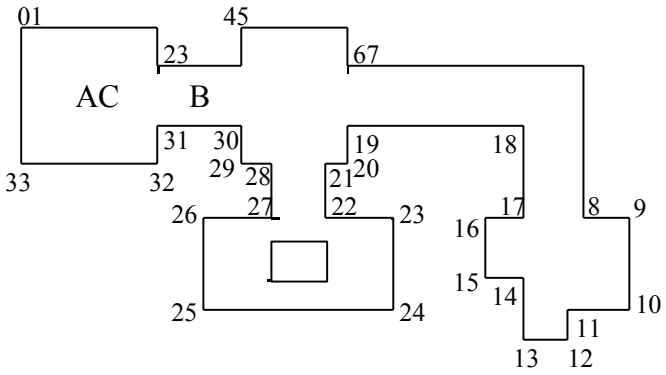


Рис.7.12.

Например, для области A таким порталом является портал 2-31 (рис.7.12). Далее рассматриваются соседние области и имеющиеся у них порталы. В этих областях выделяются те грани, которые видны из первой области (область A). Эти грани помещаются в *PVS*. Далее процедура продолжается рекурсивно. Рекурсивный алгоритм выполняется до тех пор, пока области перекрытия порталов не будут пустыми.

Так как в данном случае все области являются выпуклыми, то можно пользоваться простейшими алгоритмами отсечения.

7.2.2.2. Метод иерархических подсцен

Метод порталов можно усовершенствовать, сняв ограничения на выпуклость областей. В этом случае вся сцена разбивается на ряд подсцен при помощи алгоритма порталов. Каждая подсцена рассматривается как отдельный объект, наделенный внутренней структурой, с которым может быть связан алгоритм визуализации. Каждый такой объект – подсцена описывается структурой данных, включающей:

1) описание самой подсцены;
2) описание и характеристики метода визуализации.

Внутренняя организация и метод визуализации для каждой подсцены может быть уникальным, наиболее полно учитывающим ее особенности, Для подсцен, моделирующих внутренние помещения, целесообразно использовать алгоритмы, базирующиеся на *BSP*-деревьях, или в простейших случаях, алгоритмы с *Z*-буфером.

Для подсцен, моделирующих открытые участки (ландшафты), можно использовать алгоритмы, основанные на явной сортировке граней. При указанном подходе, каждая подсцена может состоять из более мелких подсцен. Таким образом, в результате разбиения моделируемой сцены мы приходим к иерархической модели подсцен, соединяющихся между собой посредством порталов.

При помощи данного метода можно эффективно реализовать алгоритмы визуализации универсального характера, ориентированные на применение в различных областях и использующие достаточно разнородные объекты.