Оглавление

Введение

Программное моделирование загородной местности с каждым годом набирает все большее значение в области ландшафтного дизайна, архитектуры и планирования. Применение современных технологий дает возможность разработки детализированных трехмерных моделей, обеспечивая тем самым более глубокое понимание и визуализацию географических и ландшафтных особенностей, а также предстоящих изменений. Использование метода 3D-визуализации при планировании местности преобразует сложные концепции в ясные визуальные образы, что значительно упрощает процесс коммуникации и минимизирует риск возникновения недопонимания между заказчиком и исполнителем [1].

Цель работы - разработка программного обеспечения для визуализации и макетировании загородной местности. ?

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

- Формально описать структуру моделей.
- Выбрать алгоритмы трехмерной графики для визуализации сцены и объектов.
- Выбрать язык программирования и среду разработки.
- Реализовать выбранные алгоритмы.
- Реализовать программное обеспечения для визуализации и редактирования загородной местности. ?

1 Аналитическая часть

1.1 Способ определения моделей

Определение пространственных объектов возникает в двух случаях: при описании существующего объекта и при задачах автоматизированного проектирования. В первом случае объект уже существует, и его описание может быть комбинацией математических поверхностей или требовать точного соответствия в определенных точках. Во втором случае конструктор создает пространственную форму в интерактивном режиме и использует удобное представление объекта для работы. В системах проектирования используются различные типы моделей, такие как каркасные модели (проволочные), поверхностные модели и монолитные модели.

1.1.1 Каркасные модели

Каркасные модели представляют объекты, созданные из соединенных ребер, похожие на объекты, сделанные из проволоки. В таких моделях грани объекта не определены, но их границы представлены ребрами. Каркасная модель не имеет поверхности, которые бы скрывали ребра, поэтому она выглядит прозрачной. Однако, без визуальной поддержки, например, в виде перспективы, трудно понять, как выглядит объект на самом деле.

Каркасная модель определяется линейными размерами, но так как она не содержит граней, понятие объема в ней не применимо. Каркасные модели являются наиболее простыми и широко используются в системах проектирования с низкой производительностью. Они состоят из списка координат вершин и связей между ними.

Каркасные модели применяются для визуализации объектов, особенно в случаях, когда объекты имеют более сложные поверхности. При этом

поверхность объекта может быть приближенно интерполирована плоскими гранями, но полученное изображение будет условным, а не реальным.

Каркасные модели широко применяются в задачах визуализации, а также в создании и тестировании программ управления движениями роботов. В последнем случае каркасные модели анализируются в реальном времени на компьютере, что позволяет эффективно отслеживать динамику их действий.

1.1.2 Поверхностные модели

Поверхностные модели включают как ребра, так и поверхности, что позволяет более точно представить объект, чем каркасные модели. В поверхностных моделях грани, расположенные спереди, перекрывают грани, находящиеся на заднем плане. Когда изображение выводится на монитор, можно получить более реалистичное представление трехмерного объекта. Поверхностные модели имеют объем, но не имеют массы.

Поверхностные модели широко применяются в разработке динамических поверхностей, которые взаимодействуют с внешней средой. Примеры таких поверхностей включают фюзеляжи самолетов, обводы судов, кузова автомобилей

Недостаток - отсутствует информация, о том, с какой стороны поверхности находится материал, а с какой - пустота.

1.1.3 Монолитные модели

Монолитные модели являются наиболее полным и сложным представлением объекта. Они описывают внутренние особенности и устройство объекта, имеют объем, массу и содержат характеристики материала. Большинство систем проектирования, использующих монолитные модели, обладают

средствами автоматического расчета этих параметров и широко применяются в машиностроении.

1.1.4 Вывод

Для решения указанной задачи было выбрано использовать поверхностные модели объектов.

Поверхностные модели предоставляют возможность описывать геометрическую форму объекта, не уделяя внимание его внутренним особенностям и материалам. Этот подход особенно ценен, когда ориентация материала объекта не имеет существенного значения для проведения анализа. Таким образом, необходимость моделирования внутренних особенностей и материалов отпадает.

1.2 Способ задания поверхностных моделей

Для представления трехмерных поверхностей существуют два широко используемых метода:

1) Полигональные сетки: Этот метод представляет объект в виде связанной между собой сетки плоских многоугольников. Полигональная сетка позволяет описать поверхность объектов, таких как здания, но обладает низкой точностью.

2) Параметрические бикубические куски: В этом методе поверхность представляется с помощью параметрических бикубических кусков, где уравнения, описывающие координаты точек поверхности, имеют два параметра со степенями не выше третьей. Параметрические бикубические куски позволяют более точно описать поверхность объекта и требуют меньшего количества элементов по сравнению с полигональными сетками

Существует несколько способов описания полигональных сеток, каждый из которых имеет свои преимущества и недостатки в зависимости от конкретных требований и ограничений приложения. Для оценки эффективности и удобства использования этих способов описания полигональных сеток, можно использовать следующие показатели:

Объем памяти, необходимый для хранения полигональной сетки. Вычислительная эффективность процедур, связанных с поиском и обработкой вершин, ребер и многоугольников в сетке. Эффективность процедур подготовки и визуализации полигональной сетки на устройстве вывода. Эффективность процедур проверки корректности представления сетки, включая обнаружение ошибок и дублирование ребер и вершин.

Ниже рассмотрены несколько широко распространенных способов описания полигональных сеток:

Сетка с использованием списка граней: В этом подходе объект представляется в виде множества граней и множества вершин. Каждая грань содержит информацию о своих вершинах и ребрах, которые ее ограничивают. Этот подход обеспечивает простое и интуитивно понятное представление сетки, но может потребовать больше памяти для хранения данных.

"Крылатое"представление: В этом подходе каждая точка ребра указывает на две вершины, две грани и четыре ребра, которые касаются этой точки. "Крылатое"представление обеспечивает быстрый доступ к информации о гранях и ребрах, но требует больше памяти для хранения данных.

Полуреберные сетки: Этот подход похож на "крылатое"представление, но использует информацию только о половине грани. Полуреберные сетки обеспечивают более компактное представление сетки и меньшие требования к памяти по сравнению с "крылатым"представлением.

Четырехреберные сетки: В этом подходе хранятся ребра, полуребра и вершины без указания полигонов. Полигоны могут быть найдены путем обхода структуры данных. Этот подход обеспечивает компактное представление сетки, а требования по памяти аналогичны полуреберным сеткам.

Таблица углов: В этом подходе вершины хранятся в предопределенной таблице, а обход таблицы неявно задает полигоны. Таблица углов представляет собой "веер треугольников который используется в аппаратном рендеринге. Это компактное и производительное представление для нахождения полигонов, но операции по изменению медленнее. Однако, таблицы углов не полностью представляют сетку, и для представления большинства сеток может потребоваться несколько таблиц углов (вееров треугольников).

Вершинное представление: В этом подходе представлены только вершины, которые указывают на другие вершины. Информация о гранях и ребрах выражена неявно. Простота представления позволяет проводить множество эффективных операций над сеткой.

1.2.1 Вывод

Сетка, представленная в виде списка граней, обеспечивает легкость изменения модели путем добавления, удаления и модификации граней, вершин и ребер. Это метод хранения данных, который особенно полезен для динамического изменения моделей в реальном времени, например, в интерактивных приложениях и симуляциях. Список граней обеспечивает удобный доступ к элементам сетки и позволяет выполнять различные операции на них, такие как вычисление нормалей и текстурирование. Однако следует учитывать, что использование списка граней может требовать больше памяти по сравнению с некоторыми другими методами хранения сеток, и это следует учитывать при работе с большими моделями или при ограниченных ресурсах памяти. В целом, список граней обеспечивает эффективность изменения модели и удобство работы с ней при наличии достаточных ресурсов памяти.

1.3 Алгоритмы удаления невидимых линий и поверхностей

После того, как вершины прошли все этапы геометрических преобразований и процедуру отсечения, «на конвейере» остались только геометрические объекты, которые потенциально могут попасть в формируемое изображение. Но перед тем, как приступать к их преобразованию в растр, нужно решить еще одну задачу – удалить объекты, перекрываемые с точки зрения наблюдателя другими объектами

Алгоритм Робертса

Алгоритм Робертса (в пространстве объектов) Первый из алгоритмов удаления невидимых линий. Требует, чтобы каждая грань была выпуклым многоугольником:

шаг 1: Отбросить ребра, обе инцидентных грани которых являются нелицевыми;

шаг 2: Проверка закрывания каждого оставшегося ребра лицевыми гранями

2.1. грань ребра не закрывает;

2.2. грань полностью закрывает ребро;->раннее отсечение

2.3. частично закрывает – ребро разбивается на части и оставляются только видимые части, <= 2

Алгоритм, использующий Z буфер.

• Это один из простейших алгоритмов удаления невидимых поверхностей.

• Работает в пространстве изображений.

• В этом алгоритме используется идея о буффере кадра.

Буффер кадра используется для заполнения атрибутов (интенсивности) каждого пикселя в пространстве изображения. В данном алгоритме используется два буффера: буффер регенерации и собственно сам z-буффер, куда можно помещать информацию о координате z для каждого пикселя. В начале z-буффер кладут минимально возможные значения z, а в буффер регенерации кладут пиксели, описывающие фон. Затем каждый многоугольник приводят к растровому виду, и записывают в буффер регенерации (без упорядочивания) В процессе подсчета глубины нового пикселя (который надо занести в буффер кадра), сравнивается с тем значением, что лежит в z-буффере. Если новый пиксель расположен ближе, то он заносится в буффер кадра, при этом происходит корректировка z-буффера: в него заносится глубина нового пикселя. В сущности, алгоритм для каждой точки ищет максимальное значение z для каждой точки (x, y). Вычисление глубины z Многоугольник описывается уравнением: Ax + By + Cz + D = D, отсюда получаем: z = -(Ax + By + D) / C При c = 0 - многоугольник для наблюдателя вырождается в линию. Ясно, что для сканирующей строки y = const. Поэтому, можем рекуретно считать z' для каждого x' = x + dx. z' - z = -(ax' + d) / c + (ax + d) / c = a(x - x') / c, откуда z' = z - (a / c), потому что (x - x') = dx = 1 (единичный шаг растра) Глубина пикселя, являющегося пересечением сканирующей строки с ребром многоугольника Сначала определяют ребра грани, вершины которых лежат по разные стороны от сканирующей строки, так как только в этом случае сканирующая строка пересекает ребро. Затем из найденых точек пересечения выбирают ближайшую к наблюдателю. Глубину определяют по соотношению:

z3 = z2 + (y3 - y2) / (y2 - y1) * (z2 - z1), где

• (y1, z1), (y2, z2) - координаты вершин проекции ребра на плоскость YOZ.

• (x3, z3) - координаты проекции точки пересечения на ту же плоскость.

Оценка эффективности

Плюсы:

• Сцены могут быть произвольной сложности.

• Не нужна сортировка, как в других алгоритмах.

• Трудоемкость линейно зависит от числа рассматриваемых поверхностей

Недостатки:

• Большой объем памяти (под буфферы).

• Трудоемкость устранения лестничного эффекта.

• Трудность реализации эффектов прозрачности.

Псевдокод (алгоритм)

1. Инициализация буффера кадра фоновым значением.

2. Инициализация z-буффера минимальным значеним Z.

3. Растровая развертка каждого многоугольника (в произвольном порядке).

4. Вычисление глубины z = (x, y) для каждого пикселя, принадлежащего многоугольнику.

5. Сравнение полученой глубины z со значеним z, лежащей в буфере (для пикселя (x, y)).

если полученая глубина больше значения в буфере, то записать атрибут многоугольника в буфер кадра и заменить значение в Z-буфере на полученное значение

Для невыпуклых многогранников, предварительно надо удалить нелицевые грани. Алгоритм так же можно применять для построения сечений поверхностей, в таком случае изменится только операция сравнения:

1.3.1 Вывод

На основании рассмотренных алгоритмов удаления невидимых поверхностей, я пришел к выводу, что использование алгоритма Z-буфера является предпочтительным для данной задачи. Размер изображения не является очень большим, поэтому использование Z-буфера не вызовет проблем с памятью. Кроме того, данный алгоритм обеспечивает более эффективную обработку множества объектов в сцене.

Одним из главных преимуществ выбора алгоритма Z-буфера является его легкость в понимании и отладке. Алгоритм основан на простом принципе сравнения глубины пикселей с текущим значением в Z-буфере, что облегчает понимание его работы и возможность быстрой и эффективной отладки.

Таким образом, на основании этих факторов, я буду использовать алгоритм Z-буфера для удаления невидимых объектов в данном контексте.

1.4 Алгоритм построения теней

Учет теней в алгоритмах удаления невидимых поверхностей может быть осуществлен путем модификации Z-буфера. Z-буфер является буфером глубины, который хранит информацию о глубине каждого пикселя в сцене. При обработке геометрических объектов и определении их видимости, Z-буфер используется для сравнения текущей глубины пикселя с сохраненным значением в буфере. Однако, для учета теней необходимо внести изменения в этот процесс.

Одним из подходов к учету теней является модификация Z-буфера путем введения дополнительной информации о тени. Для каждого пикселя в Z-буфере помимо значения глубины сохраняется также значение, указывающее, находится ли данный пиксель в тени или нет. Это значение может

быть булевым флагом или числовым значением, указывающим интенсивность тени.

При рендеринге сцены с учетом теней, перед сравнением глубины текущего пикселя с сохраненным значением в Z-буфере, также проверяется, находится ли данный пиксель в тени. Если пиксель находится в тени, то его глубина не обновляется, и он может быть отображен с учетом соответствующей интенсивности тени.

Важным аспектом при использовании модифицированного Z-буфера для учета теней является правильное определение, какие пиксели находятся в тени. Для этого необходимо провести предварительные вычисления и определить, какие объекты или грани находятся между источником света и точкой наблюдения, и соответственно, создают тень.

Таким образом, модификация Z-буфера позволяет эффективно учитывать тени при рендеринге сцены и одновременно выполнять удаление невидимых поверхностей. Этот подход требует дополнительных вычислений и хранения информации о тени в буфере, но позволяет достичь реалистичного отображения теней на изображении.

Литература

[1] A. Lovett K. Appleton B. Warren-Kretzschmar. Using 3D visualization methods in landscape planning: An evaluation of options and practical issues // Landscape and Urban Planning. 2015. С. 85–94.