Реализация алгоритма иерархического z-буфера (с использованием октодерева) удаления невидимых поверхностей

Студент: Жабин Д.В.,
ИУ7-54Б
Руководитель: Погорелов Д.А.

Москва, 2021 г.

Цели и задачи

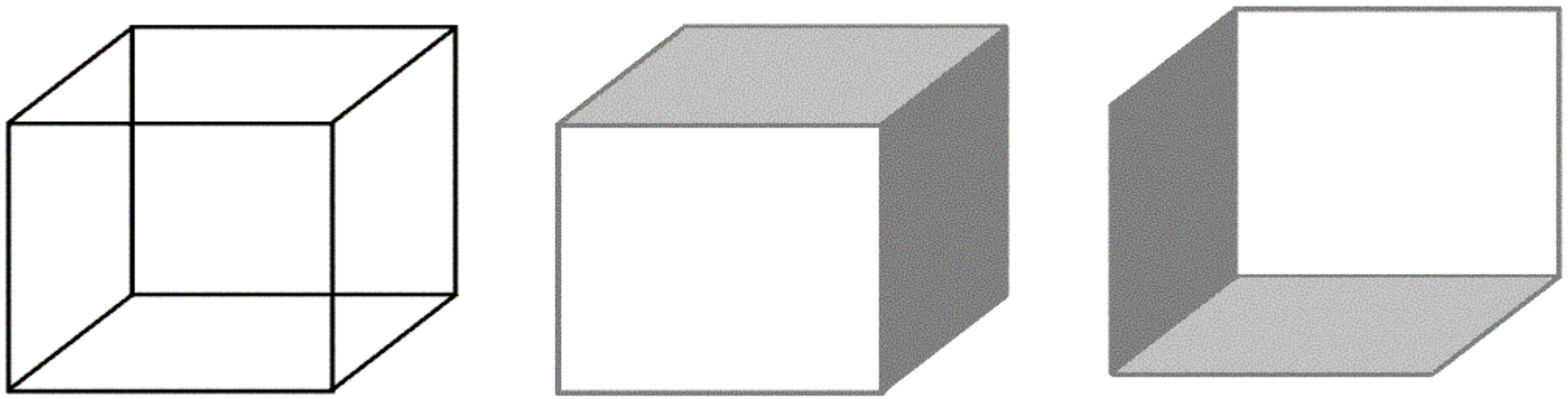
Цель работы – реализовать алгоритм иерархического z-буфера (с использованием октодерева) удаления невидимых поверхностей и произвести анализ его эффективности по сравнению с более простыми модификациями алгоритма z-буфера.

Для достижения поставленной цели требуется решить следующие задачи:

- формализовать объекты сцены;
- изучить модификации алгоритма z-буфера удаления невидимых поверхностей;
- изучить существующие модели освещения сцены и выбрать наиболее подходящую для данной задачи;
- реализовать алгоритмы удаления невидимых поверхностей с использованием z-буфера;
- провести сравнение временных характеристик реализаций алгоритмов.

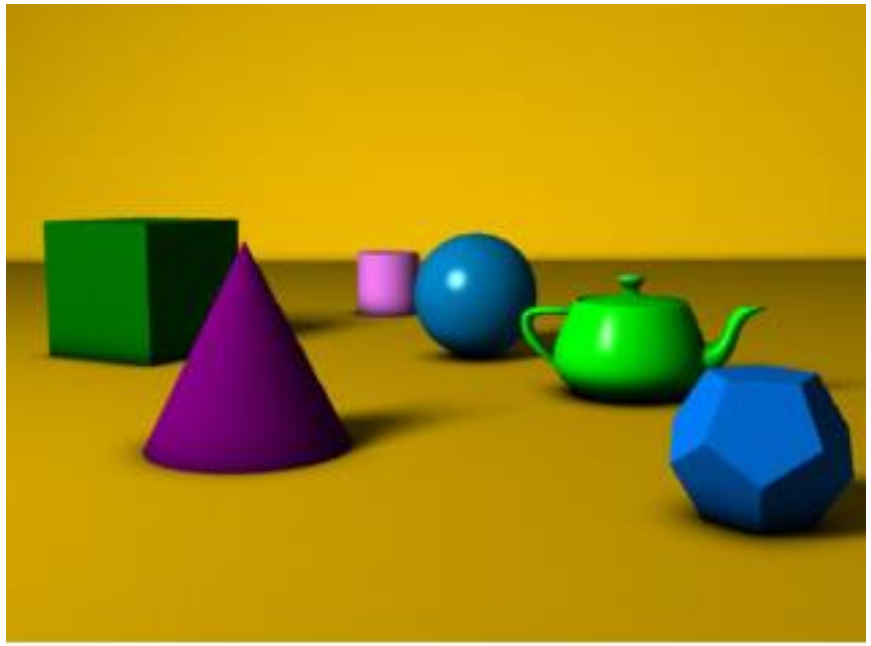
Алгоритмы удаления невидимых линий или поверхностей:

1) алгоритмы, работающие в объектном пространстве;
2) алгоритмы, работающие в пространстве изображения (экрана);
3) алгоритмы, формирующие список приоритетов.



Алгоритм z-буфера

Относится к алгоритмам, работающим в пространстве изображения, что позволяет воспользоваться преимуществом когерентности при растровой реализации.

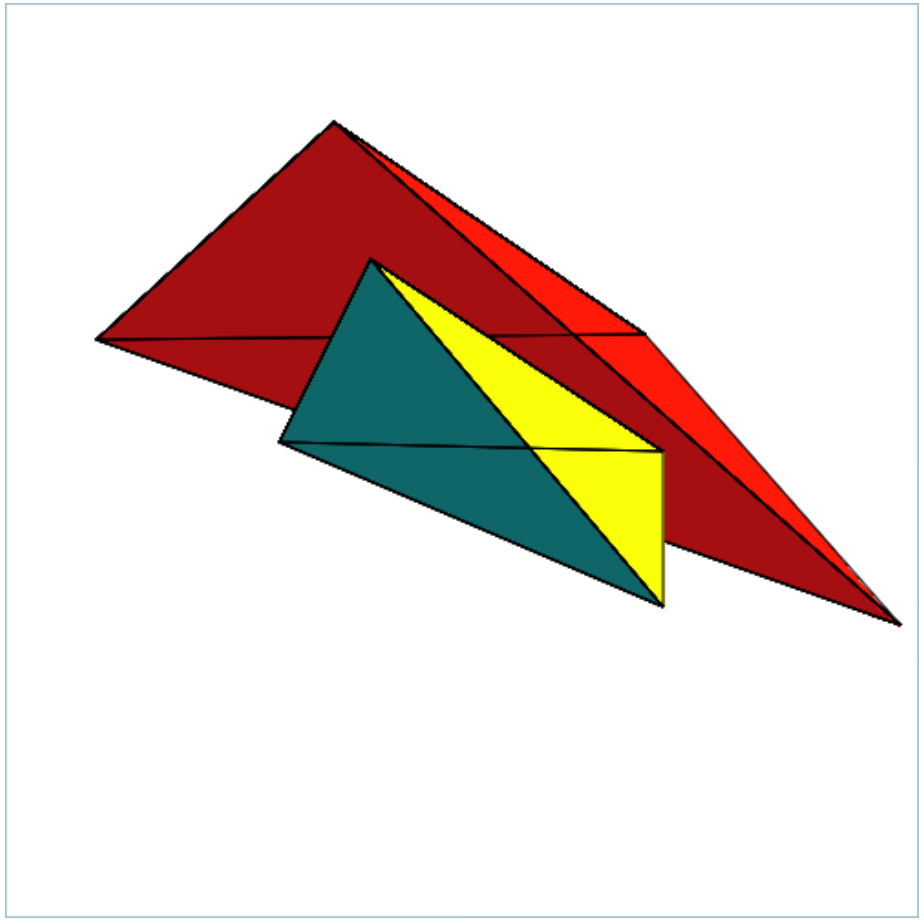


Алгоритм Z-буфера

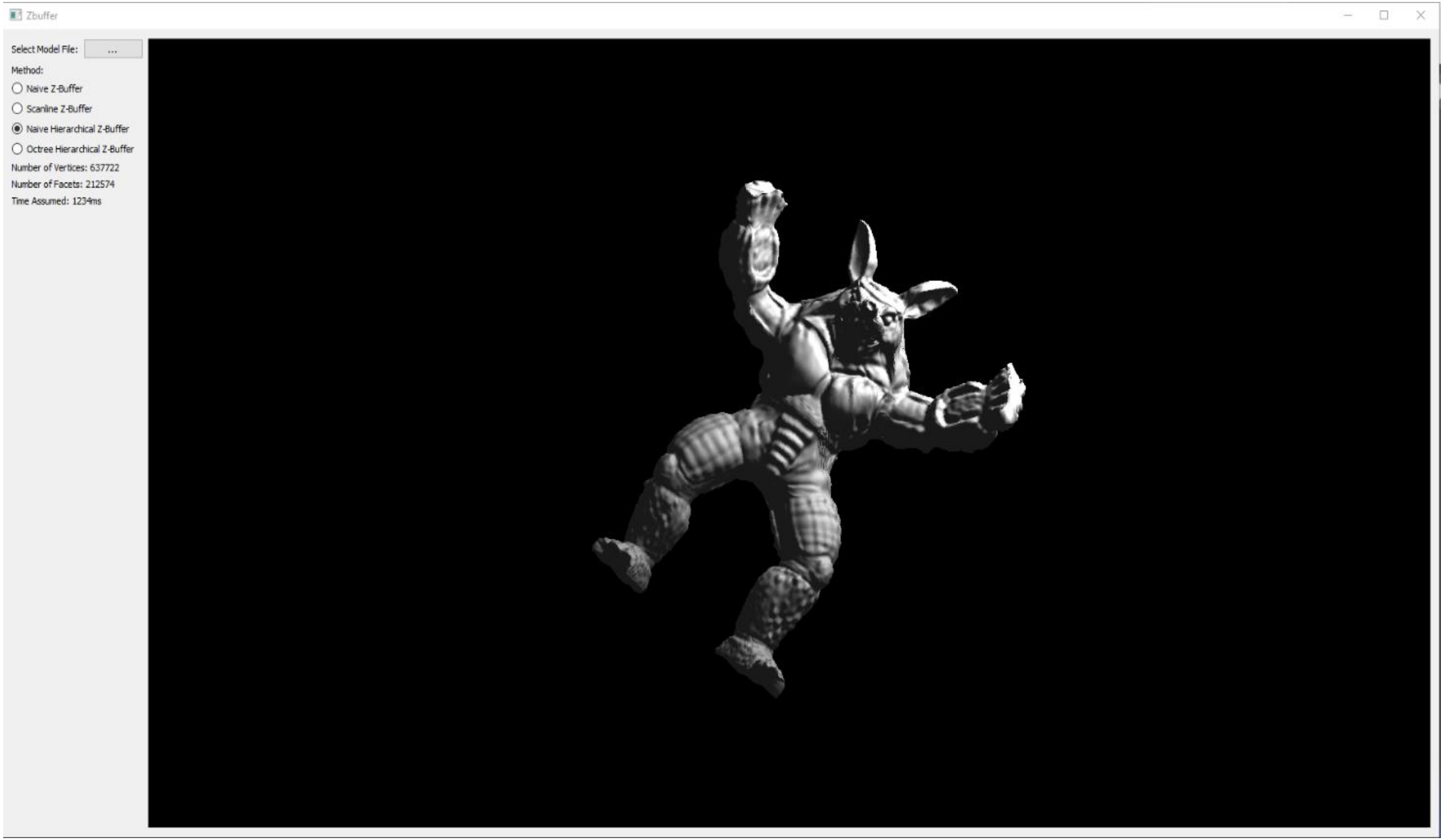
1. Заполнить буфер кадра фоновым значением интенсивности или цвета.
2. Заполнить z-буфер минимальным значением z.
3. Преобразовать каждый многоугольник в растровую форму в произвольном порядке.
4. Для каждого пиксела с координатами (x, y) в многоугольнике вычислить его глубину $z(x, y)$.
5. Сравнить глубину $z(x, y)$ со значением a, хранящимся в z-буфере в этой же позиции.
6. Если $z(x,y) > a$, то записать атрибут этого многоугольника (интенсивность, цвет и т. п.) в буфер кадра и заменить значение a в z-буфере на $z(x,y)$. В противном случае никаких действий не производить.

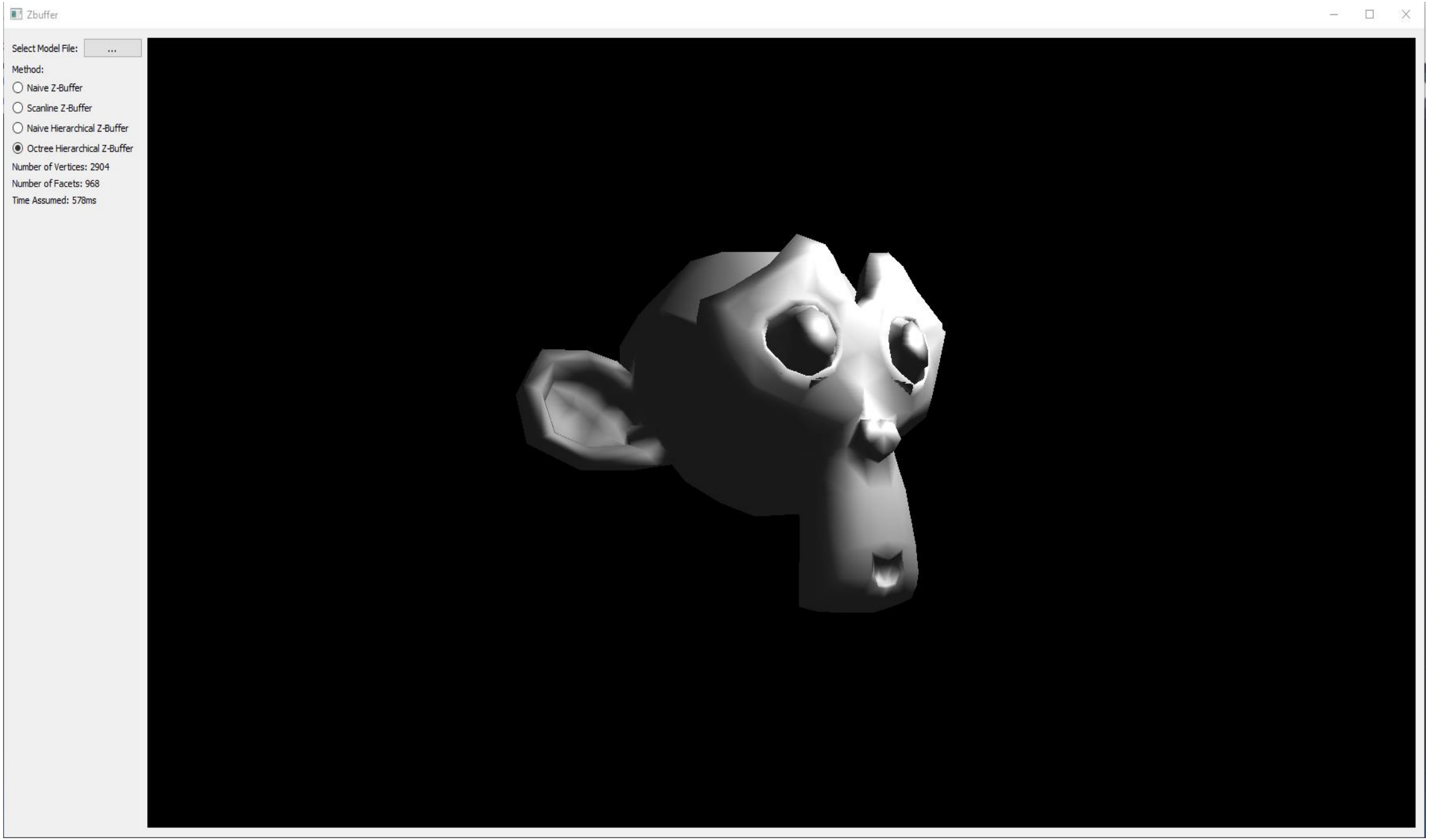
Иерархический алгоритм z-буфера

Основная идея: рекурсивное разбиение изображаемого трехмерного пространства на 8 октантов до тех пор, пока в каждом октанте (узле полученного дерева) изображение объектов сцены не будет однозначно – каждому узлу должно соответствовать не более 1 многоугольника.



Примеры работы


Zbuffer
Select Model File:
...
Method:
Naive Z-Buffer
Scanline Z-Buffer
Naive Hierarchical Z-Buffer
Octree Hierarchical Z-Buffer
Number of Vertices: 637722
Number of Facets: 212574
Time Assumed: 1234ms

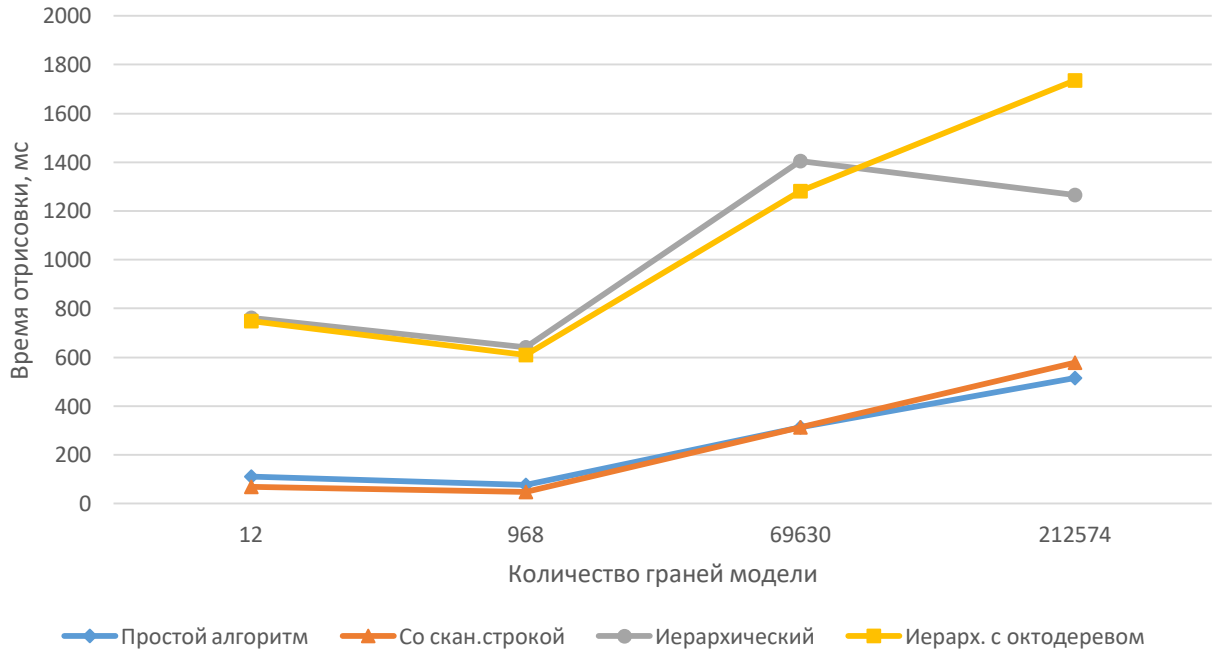

Zbuffer
Select Model File:
...
Method:
Naive Z-Buffer
Scanline Z-Buffer
Naive Hierarchical Z-Buffer
Octree Hierarchical Z-Buffer
Number of Vertices: 2904
Number of Facets: 968
Time Assumed: 578ms

Сравнение реализаций алгоритмов

Файл	Количество вершин	Количество граней
cube.obj	36	12
monkey.obj	2904	968
bunny.obj	208890	69630
armadillo.obj	637722	212574

Время (мс)	Cube.obj	Monkey.obj	Bunny.obj	Armadillo.obj
Простой алгоритм	110	78	312	515
Алгоритм со сканирующей строкой	70	47	313	578
Иерархический алгоритм	761	641	1406	1266
Иерархический алгоритм с октодеревом	750	610	1281	1735

Выводы



Реализация алгоритма иерархического z-буфера проигрывает реализации простого алгоритма z-буфера по временным характеристикам на использованных моделях, но при усложнении модели временные затраты на выполнение иерархического алгоритма растут значительно медленнее, чем затраты на выполнение простого алгоритма. Из этого можно сделать вывод о том, что при достижении определенного уровня сложности модели иерархический алгоритм окажется эффективнее простого алгоритма.

Спасибо за внимание

Москва, 2021 г.