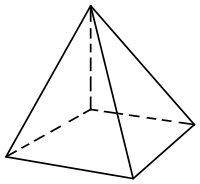
Компьютерная графика в 3D-пр-ве

Задание сцен: 1 сцена – треуг. призма
сцена будет описана мн-м треугольников, которые её представляют

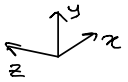


4Δ – грани + 2Δ в основании

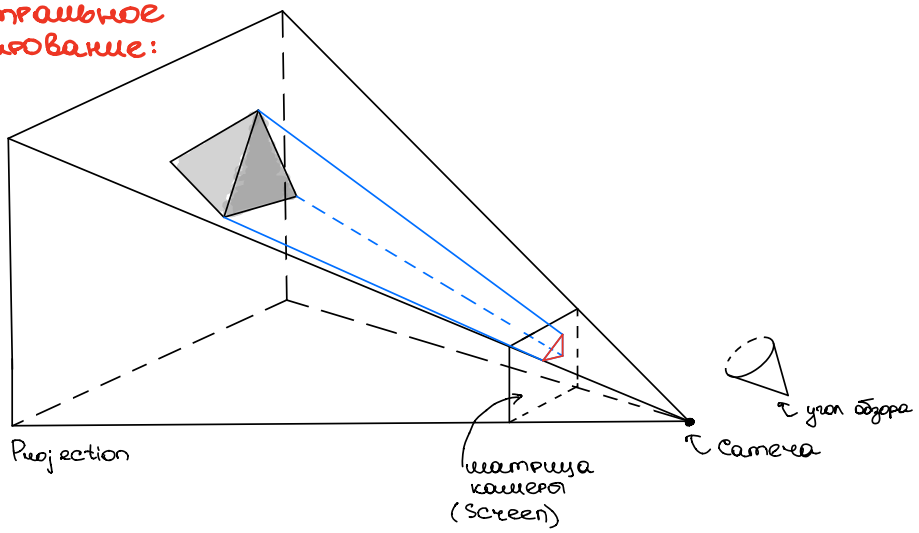
Чтобы задать мн-во треугольников:

- задаём координаты вершин в 3-х мерном пр-ве
- описание топологии задания треугольников на осн. вершин

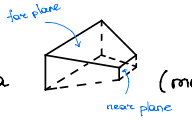
Вводится система координат для „камеры" (Сатеча)



Центральное проецирование:

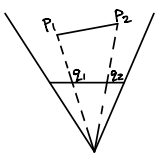


Все стороны спроецируются, получим мн-во проекций (треугольников)
Потом происходит нек. масштабирование – отображ. окошек в метрах в пиксели.



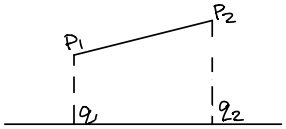
View frustum – усечённая пирамида (тот объем, который мы видим)
View port – проекция на screen
View window = screen

Центральное проецирование



(привычнее для человека)
позволяет оценить перспективу

Параллельное проецирование



объекты имеющие одинак. разм. в проекции тоже одинак.

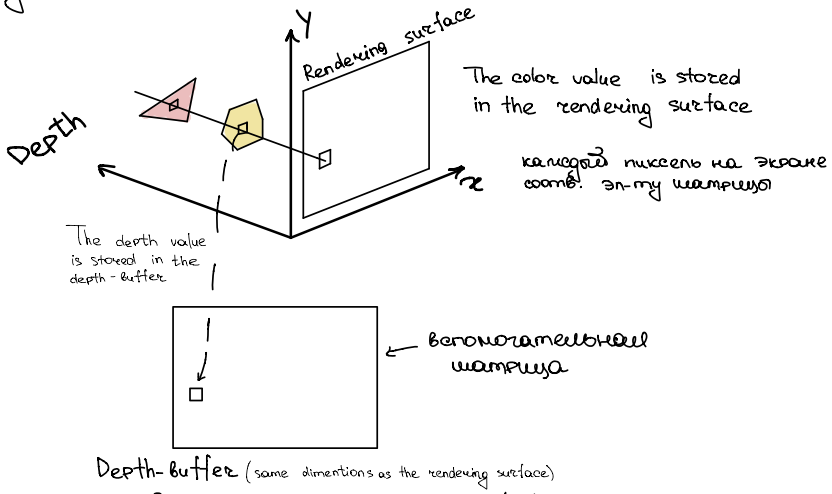
Порядок вывода треугольников вапсен

Сортировать треугольники нельзя: 1. треугольников очень много

2. сортировка не может быть вып. за $O(n \log n)$ в любом случае кв. порядок $O(n^2)$, т.к. топологическая сортировка.

Также GPU работает с примитивами, и GPU не может работать в realtime, если мы хотим визуализировать больше чем ~1000 примитивов. (> 1000 треуг. не вывести)

Нужно предлагает аппаратно реализ. алгоритм Z-test / Z-buffer



Крч. тот пиксель, который имеет меньшее знач. depth, тот и занимает место. При очистке buff'ра ставим наиб. знач.

Если объект не попал в видимый объём, то отбрасывается.
Отбрасывание — culling.
Если объект не вошёл целиком, а лишь частично ⇒ разрезаем.
Разрезание — clipping

Это выполняется аппаратно

И много, много

Graphics pipeline:

Application
Geometry
Transform and lighting
Back culling, clipping
Projection
Rasterization (interpolation) (закрашивание пробелов)
Pixel processing (пример: цвет пикс.)
Z-test and others
Target buffer
Screen