Hardware Rendering

25 Jan

OpenGL:

glBegin & glEnd - immediate mode - depricated

New verson - small programs called shaders

SIMP:

```
int addOne(int x) {
    return x+1
}
```

Data

```
int x[] = {10, 20, 30, 40, ---, 1000}
for (int i=0 ; i<100; i++) {
    x[i] = addOne(x[i])
}
```

Iteratively

iter 1: i = 0
call addOne(0)
Store x[0] value from before

iter 2: i = 1
call addOne(1)
Store x[1] value from

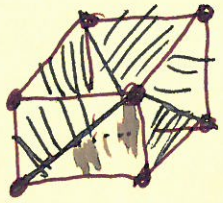
⋮

code
[α
callAddOne(α)
Store α]

Thread 1: call code α = 0
Thread 2: call code α = 1
Thread 3: call code α = 2

⋮

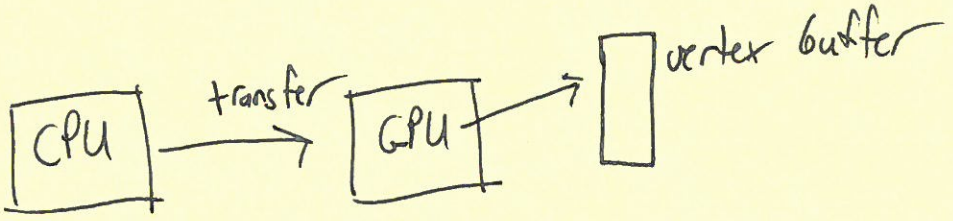
Representing w/ Δs



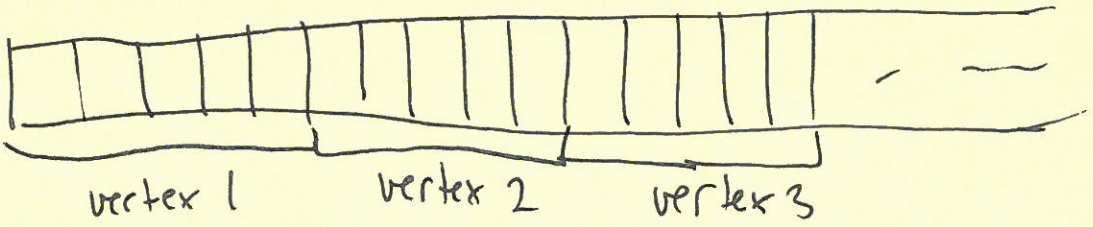
0 representations

- geometric coords
- texture coords
- color values
- normal vector

transfer vertex data from CPU → GPU

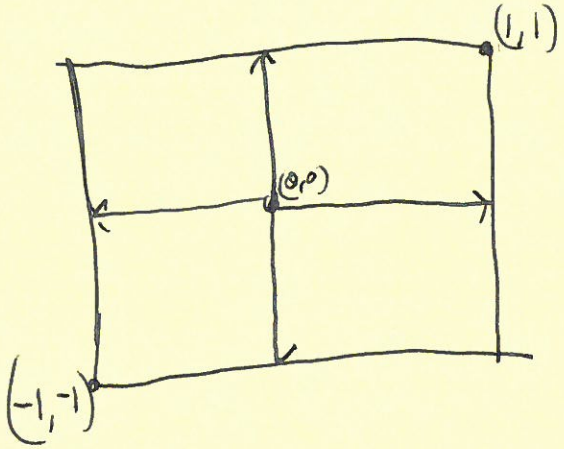


vertex buffer

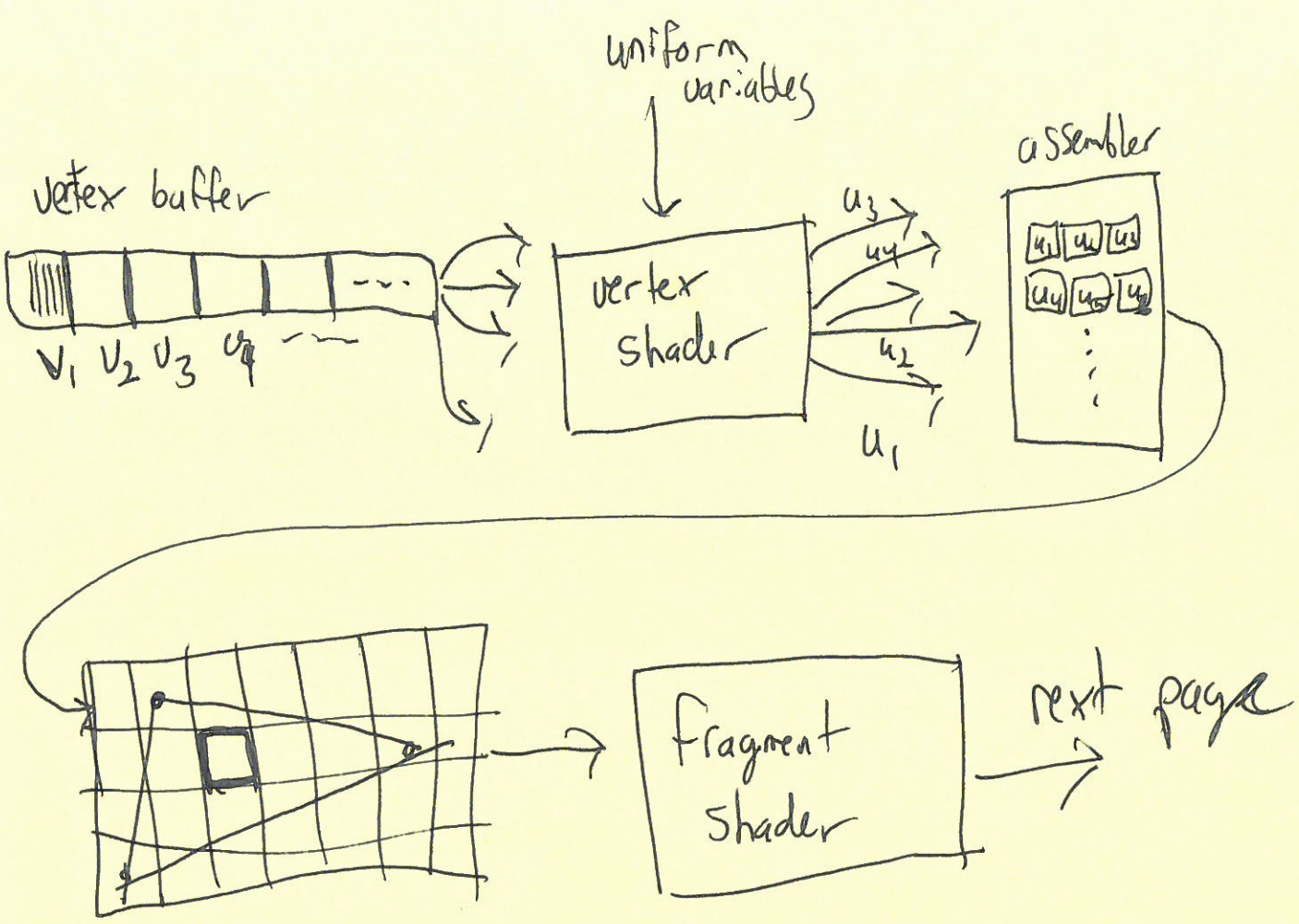


Uniform variables
- global
- updated between draw calls

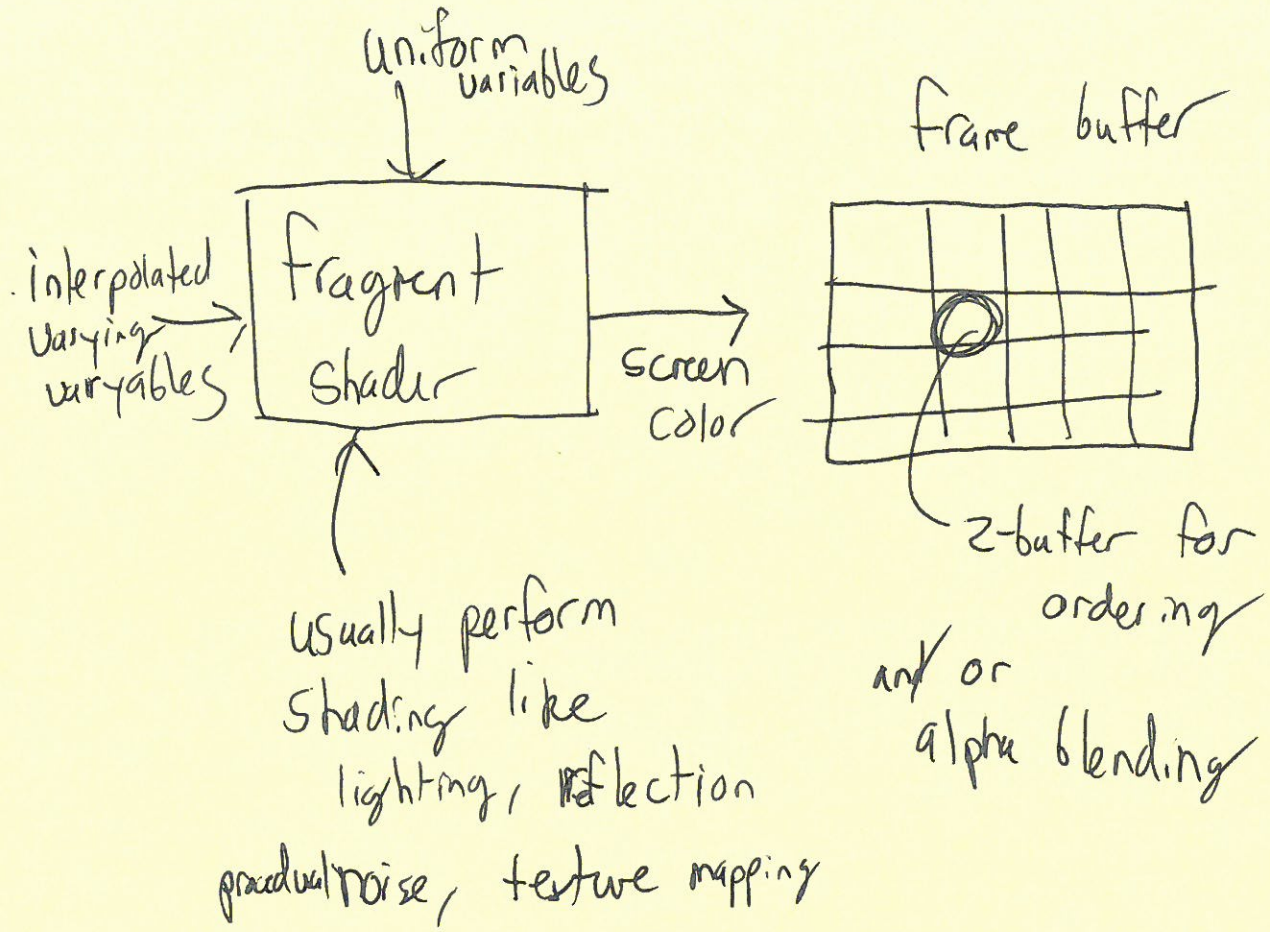
Vertex Shader
- implended by developer
- mapping to screen
- gl_Position
- in window coordinate



Assembler
- after vetex is processed
  grouped as tripples


uniform variables
vertex buffer
v_1 v_2 v_3 v_4
vertex Shader
u_3
u_4
u_2
u_1
assembler

fragment Shader
next page


uniform variables
frame buffer
interpolated varying variables
fragment shader
screen color
z-buffer for ordering
and or alpha blending
usually perform shading like lighting, reflection
procedural noise, texture mapping