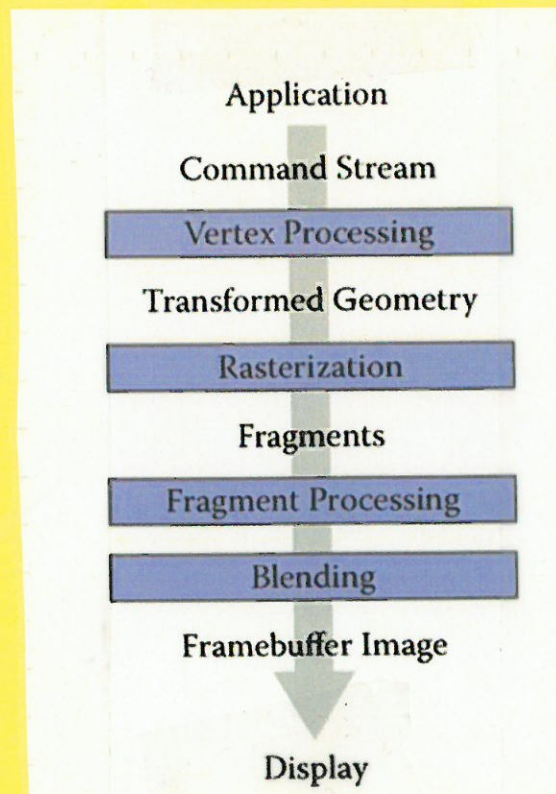


OpenGL:

- started as API for a seq of actions
- Old OpenGL `glBegin` & `glEnd`
- New version of OpenGL
 - write small "programs"
 - transfered over to GPU
 - C-like lang GLSL

Hardware



158

Hardware 2 Open GL

Open GL

- started as API for a set of actions

- Old Open GL 2.1/2.0

- New version of Open GL

- "merged" into 3.0

- transition over to GLN

- C-like lang GLSL

Hardware

GPUs are Fast

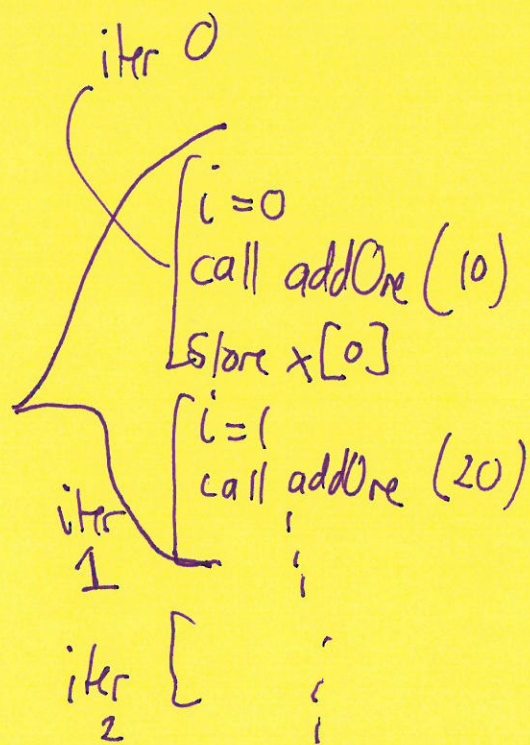
SIMD (Single instruction multiple data)

function!

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

Sol 1 (iterative solution)

```
int X[] = {10, 20, ..., 1000}  
for (int i=0; i < 100; i++) {  
    X[i] = addOne(X[i])  
}
```



Sol. 2 (parallel)

iter	thread 0	thread 1	---	thread 15
iter 0	i=0 x[0] = addOne(x[0])	i=1 x[1] = addOne(x[1])	---	i=15 x[15] = ---

2-11-2 are fast
 SIMD (single instruction multiple data)

function:

```

  {
    int addOne(X)
    return X+1
  }

```

int i;
 for (i=0; i<100; i++)
 {
 call addOne(i)
 // do something
 }
 int j;
 for (j=0; j<100; j++)
 {
 call addOne(j)
 // do something
 }

2-11-2 (iterative solution)
 int X[3] = {10, 20, 30};
 for (int i=0; i<100; i++)
 {
 X[i] = addOne(X[i])
 }

2-11-2 (parallel)

int i	for (i=0; i<100; i++)	int j	for (j=0; j<100; j++)	int k	for (k=0; k<100; k++)
0	X[0] = addOne(X[0])	101	X[101] = addOne(X[101])	201	X[201] = addOne(X[201])

Vertex Shader

```
#version 330 core

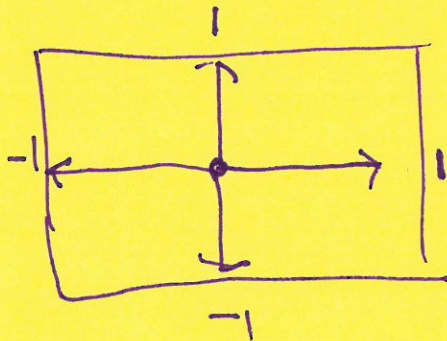
layout(location=0) in vec3 in_Position;
void main(void)
{
    gl_Position = vec4(in_Position, 1.0);
}
```

Fragment Shader

```
#version 330 core

layout(location=0) out vec4 out_FragmentColor;
void main(void)
{
    out_FragmentColor = vec4(0.49, 0.87, 0.59, 1.0);
}
```

Coord frame for GL Position



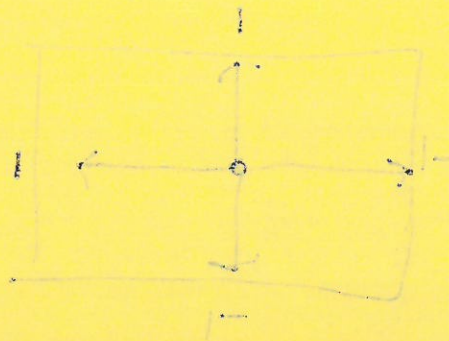
Vector sketch

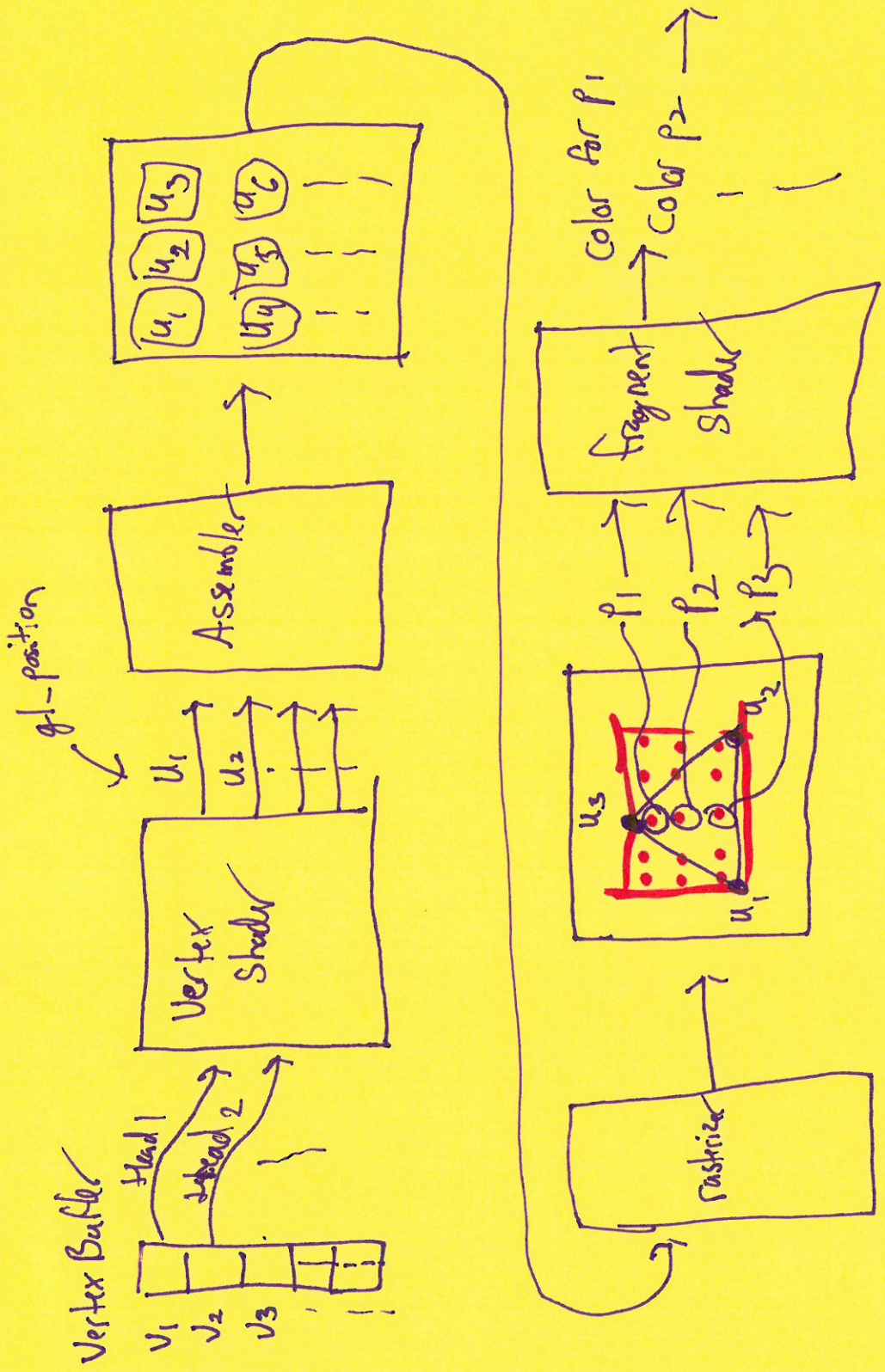


Force diagram

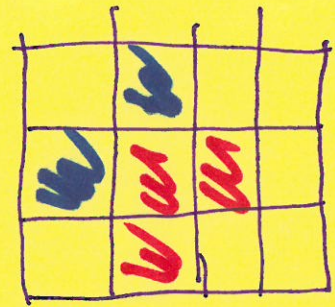


Load from floor slab

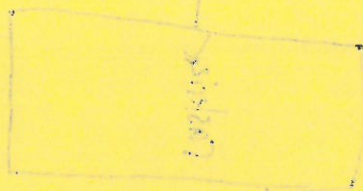
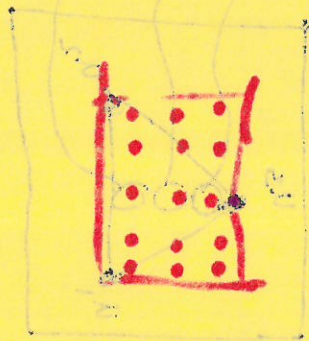




Frame buffer



Left side



Right side

Left side

Eg 1

vertex 3 {

float x, y, z

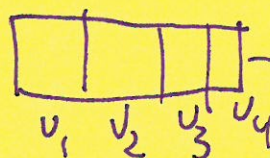
float r, g, b

}

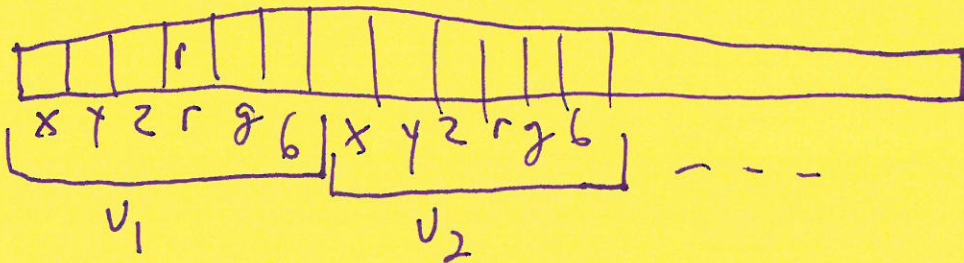
concept



on CPU



on GPU



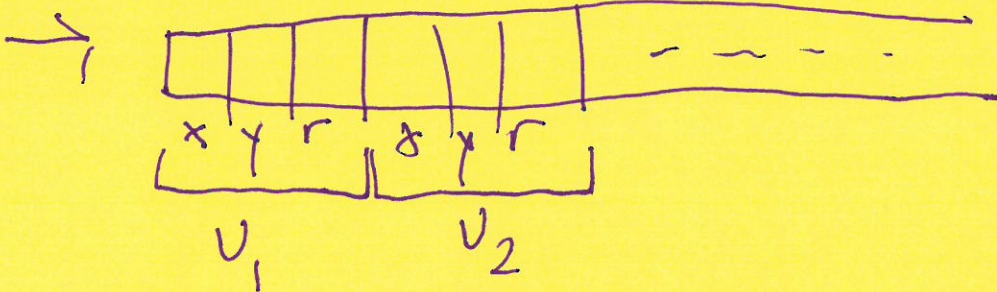
vertex 2 {

float x, y

float ref, t, i, v, y

}

on GPU

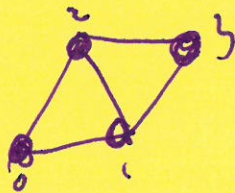
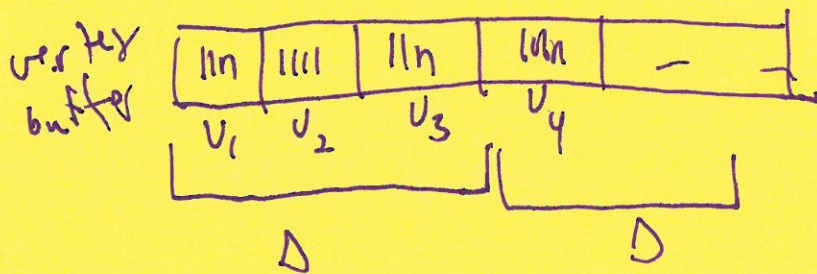


OpenGL

- vertex buffer

w/ a description of data layout

- Δ rep



EBO rep

$\{0, 1, 2\}$
 $\{1, 2, 3\}$

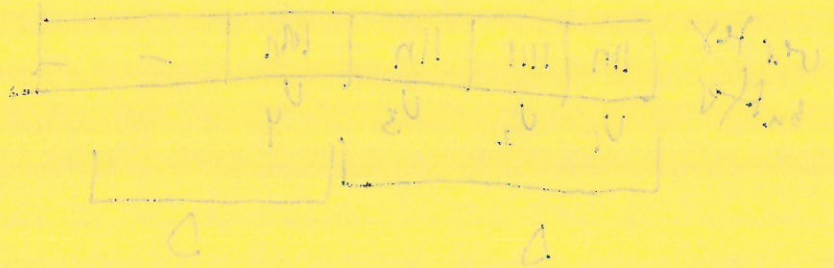
* LoGL

Open GL

vertex buffer

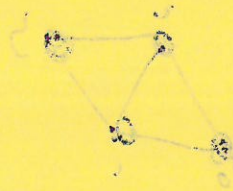
+ need a description of data layout

for 4



for 4

{
v1, v2, v3
}



* for GL