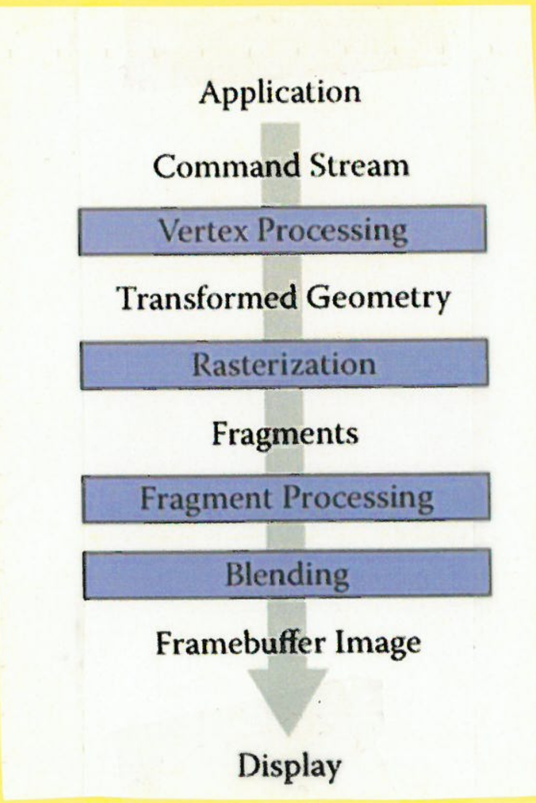
Hardware & OpenGL

OpenGL:

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

Hardware



GPUs are Fast

SIMD (Single instruction multiple data)

function!

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

Sol1 (iterative solution)

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

```
iter 0 [ i=0
         call addOne (10)
         store x[0]
iter 1 [ i=1
         call addOne (20)
              ⋮
iter 2 [      ⋮
```

Sol.2 (parallel)

iter	therad 0	tread 1	- - -	thead 15

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

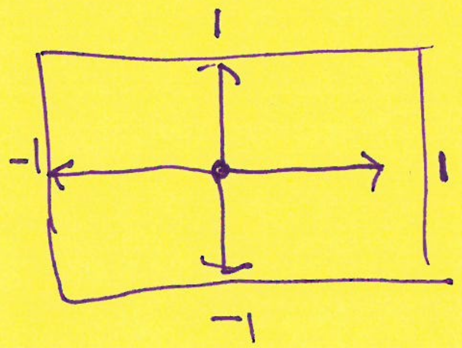
Vertex Shader

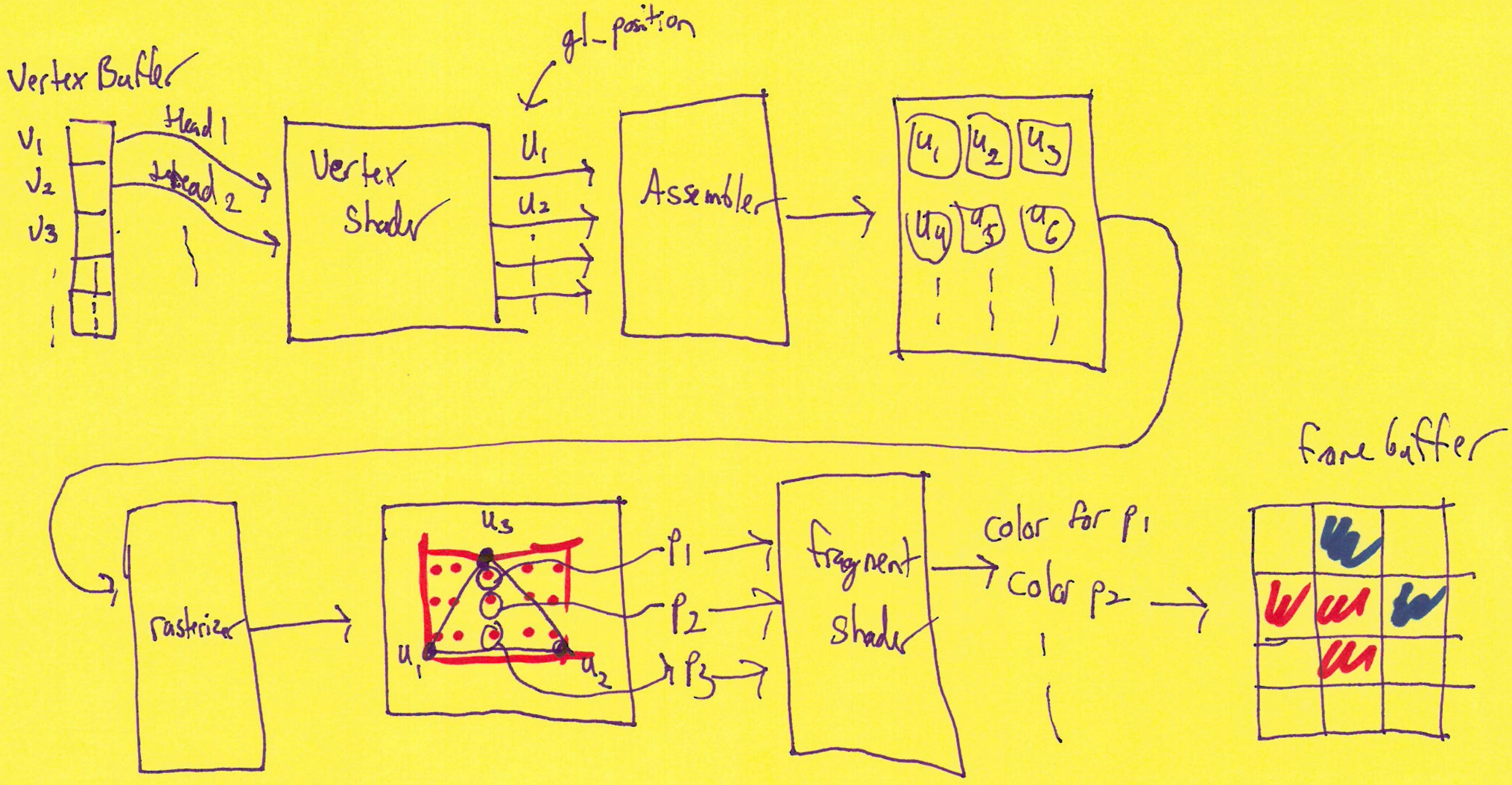
```
#version 330 core

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

fragment shader

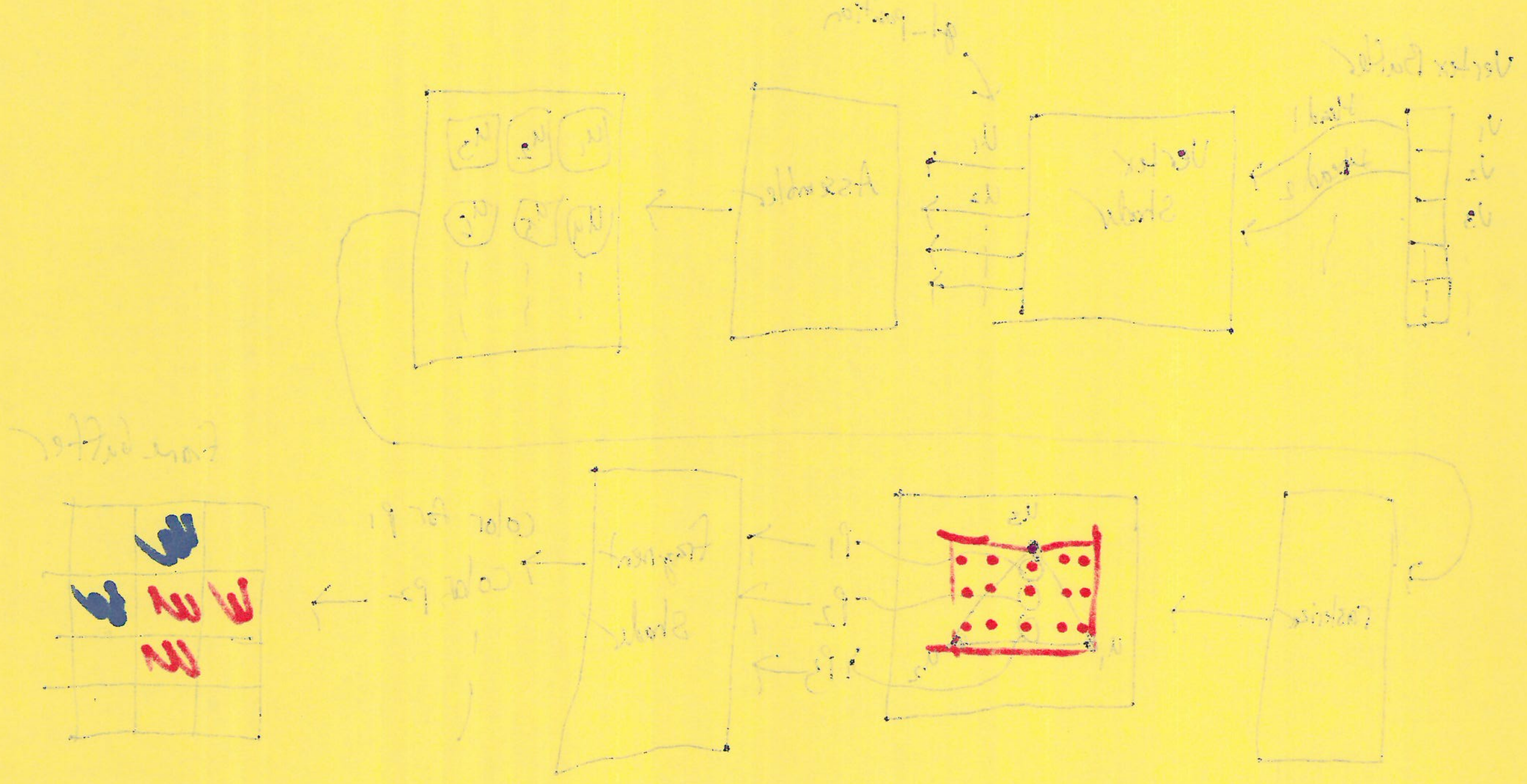
```
#version 330 core

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



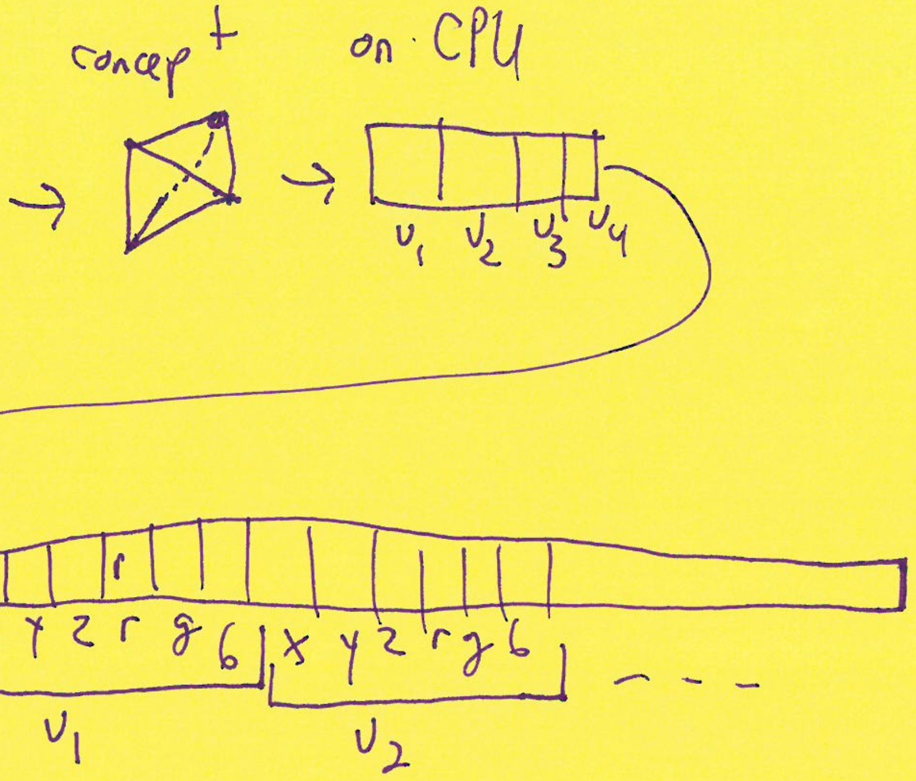


Vertex Buffer
v_1
v_2
v_3
Head 1
thread 2
Vertex Shader
gl_position
u_1
u_2
Assembler
u_1 u_2 u_3
u_4 u_5 u_6
rasterizer
u_3
u_1
u_2
p_1
p_2
p_3
fragment Shader
color for p_1
color p_2
frame buffer

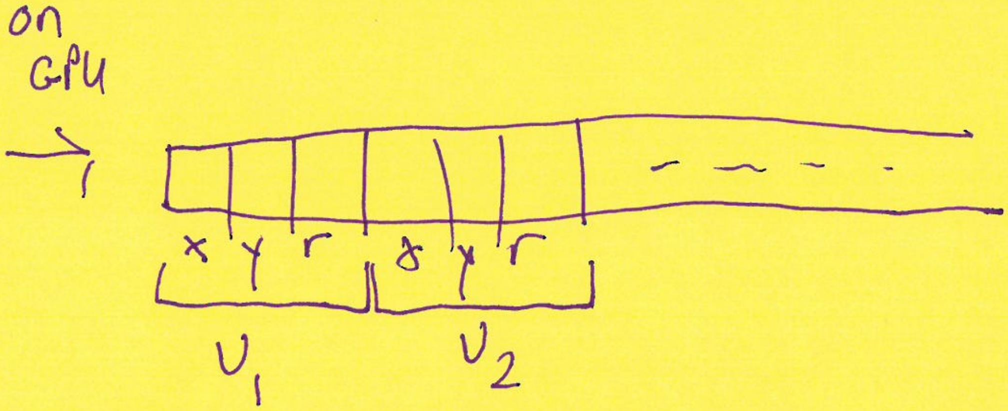


Eg1

```
vertex3 {
    float x,y,z
    float r,g,b
}
```



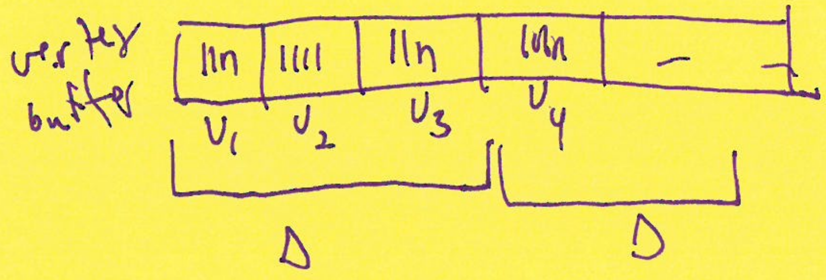
```
vertex2 {
    float x,y
    flatreflectivity
}
```



Open GL

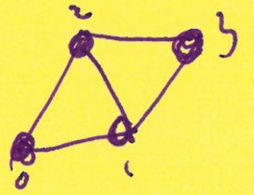
- vertex buffer
  w/ a description of data layout

- Δ rep



EBO rep

{0,1,2
1,2,3}



* LoGL