Instancing

Instancing is the ability to draw the same object multiple times. It uses all the same vertices and the same graphics pipeline data structure each time. It avoids the overhead of the program asking to have the object drawn again, letting the GPU/driver handle all of that.

```
vkCmdDraw(CommandBuffers[nextImageIndex], vertexCount, instanceCount, firstVertex, firstInstance);
```

But, this will only get us multiple instances of identical objects drawn on top of each other. How can we make each instance look differently?

BTW, when not using instancing, be sure the `instanceCount` is 1, not 0!

---

Making each Instance look differently -- Approach #1

Use the built-in vertex shader variable `gl_InstanceIndex` to define a unique display property, such as position or color.

- `gl_InstanceIndex` starts at 0

In the vertex shader:

```glsl
layout( std140, set = 0, binding = 0 ) uniform sporadicBuf
{
    int uMode;
    int uUseLighting;
    int uNumInstances;
} Sporadic;

void main( )
{
    float DELTA = 3.0;
    float s = sqrt(float(Sporadic.uNumInstances));
    float c = ceil(float(s));
    int cols = int(c);
    int fullRows = gl_InstanceIndex / cols;
    int remainder = gl_InstanceIndex % cols;
    float xdelta = DELTA * float(remainder);
    float ydelta = DELTA * float(fullRows);
    vColor = vec3(1., float((1.+gl_InstanceIndex))/float(Sporadic.uNumInstances), 0.);
    vec4 vertex = vec4(aVertex.xyz + vec3(xdelta, ydelta, 0.), 1.);
    gl_Position = PVM * vertex;
}
```

In the vertex shader:

- `uNumInstances = 16`
Making each Instance look differently -- Approach #2

Put the unique characteristics in a uniform buffer array and reference them

Still uses gl_InstanceIndex

In the vertex shader:

```glsl
layout(std140, set = 4, binding = 0) uniform colorBuf

vec3 uColors[1024];

out vec3 vColor;

... int index = gl_InstanceIndex % 1024; // gives 0 - 1023 vColor = Colors.uColors[index];
... vec4 vertex = vec4(aVertex.xyz + vec3(xdelta, ydelta, 0.), 1.);
gl_Position = PVM * vertex;
```