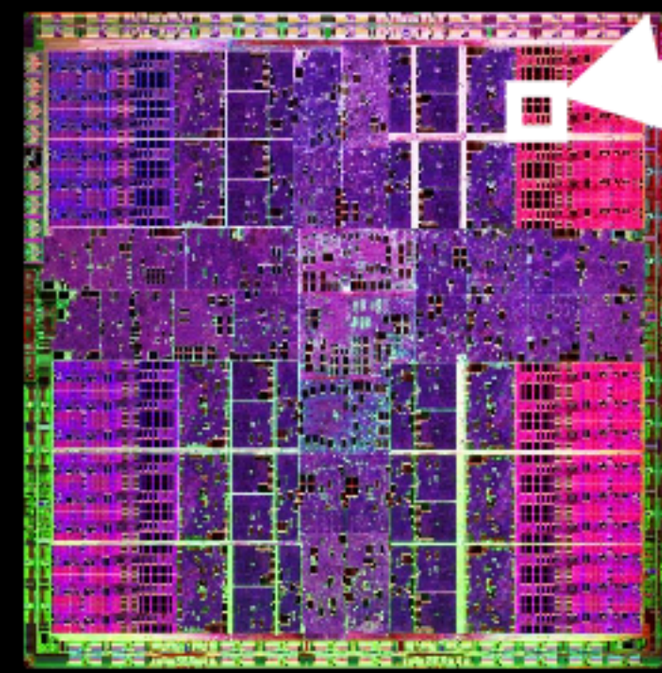
"Shader"

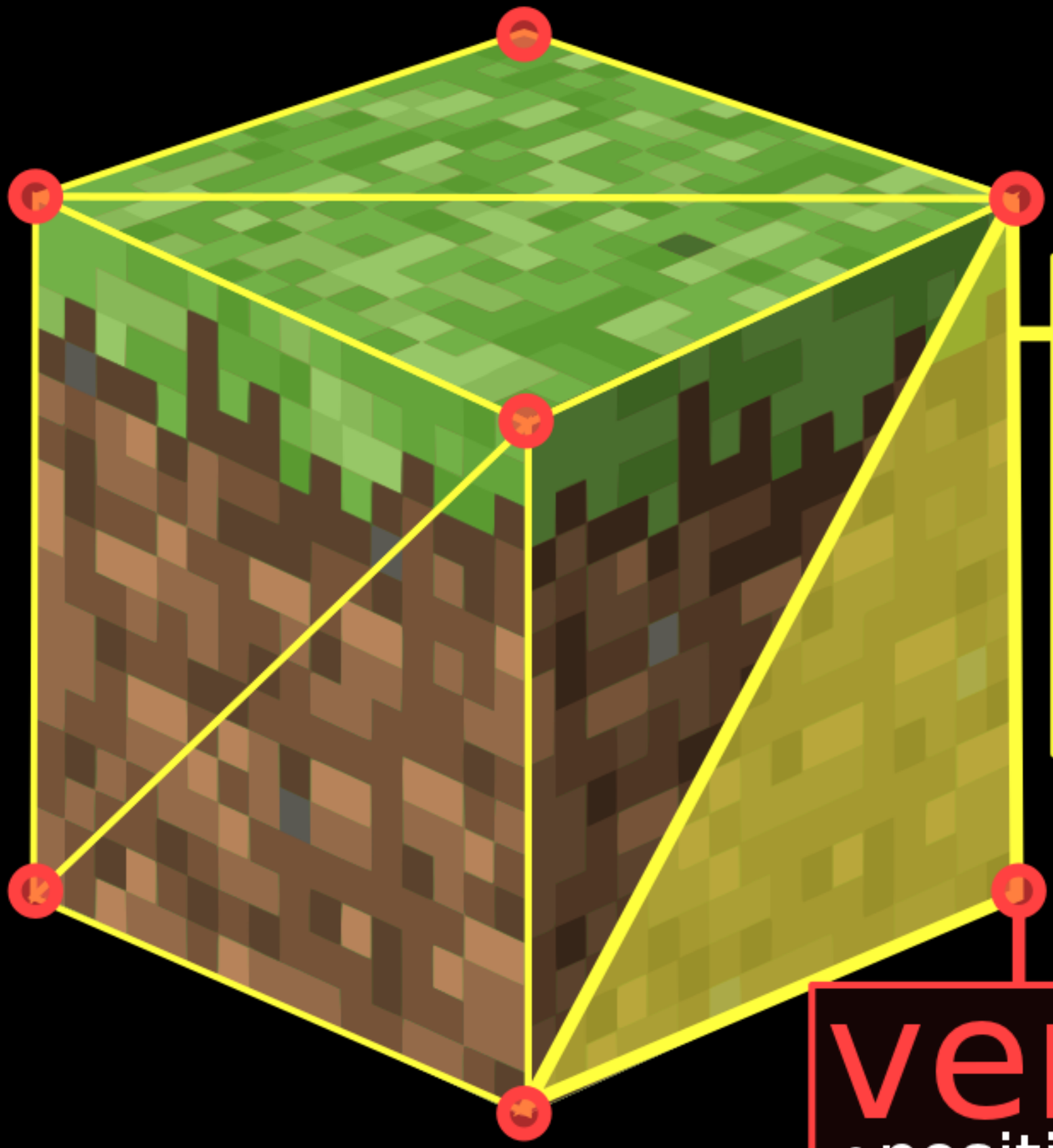
1. shader core



2. shader program

```
void main()
{
  float bla = blah();
  bla += foo(bla)*bla;

  gl_FragColor = bla;
}
```

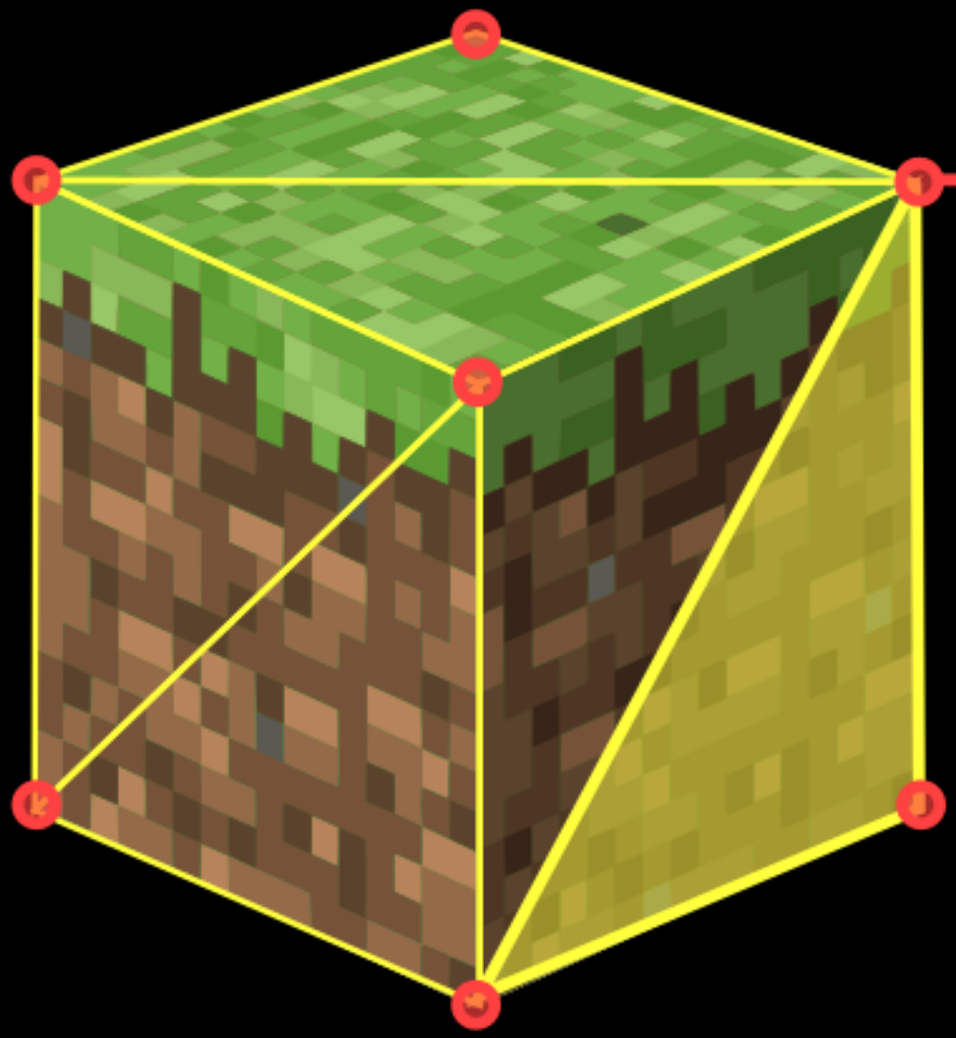


triangle

- vertex1
- vertex2
- vertex3
- texture

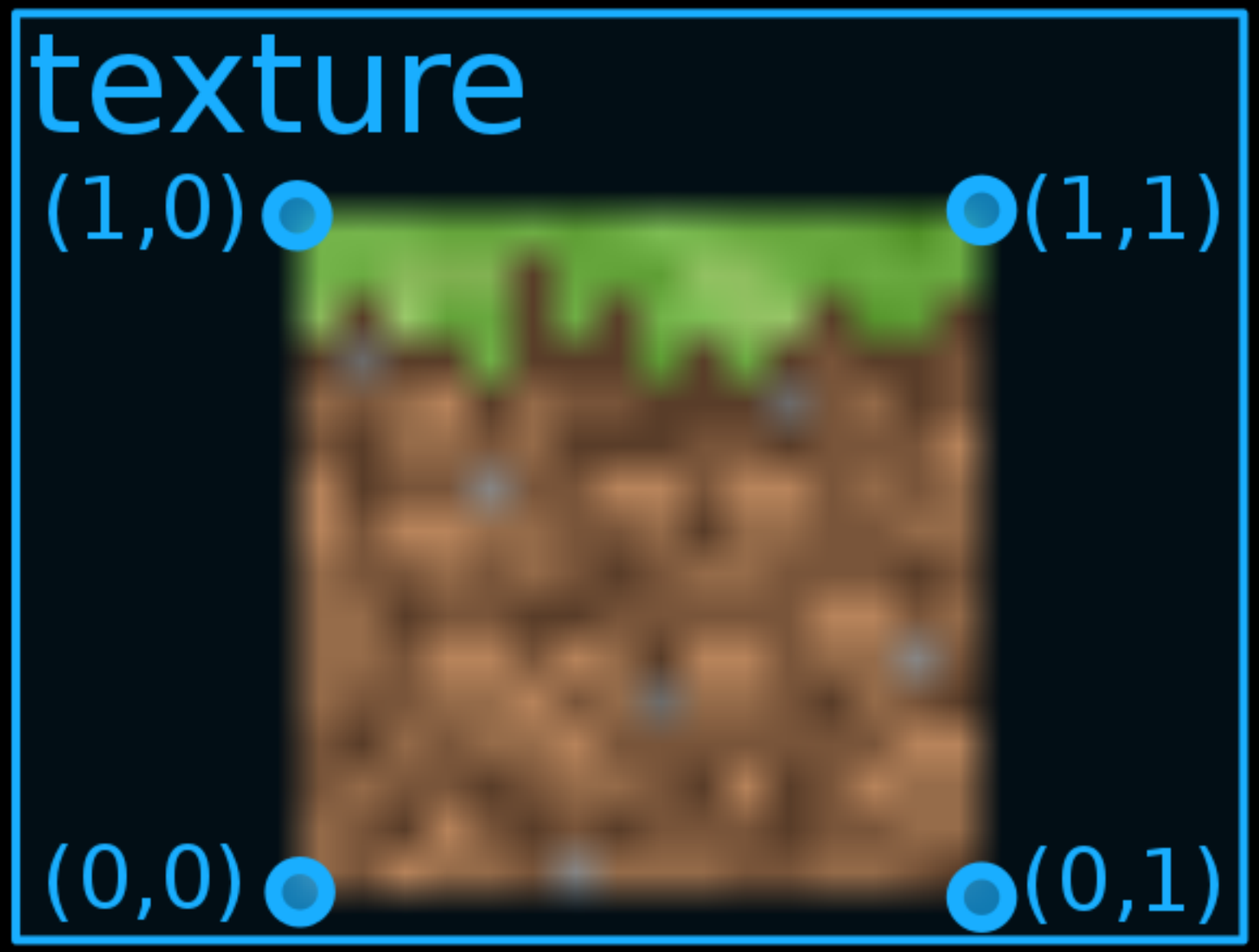
vertex

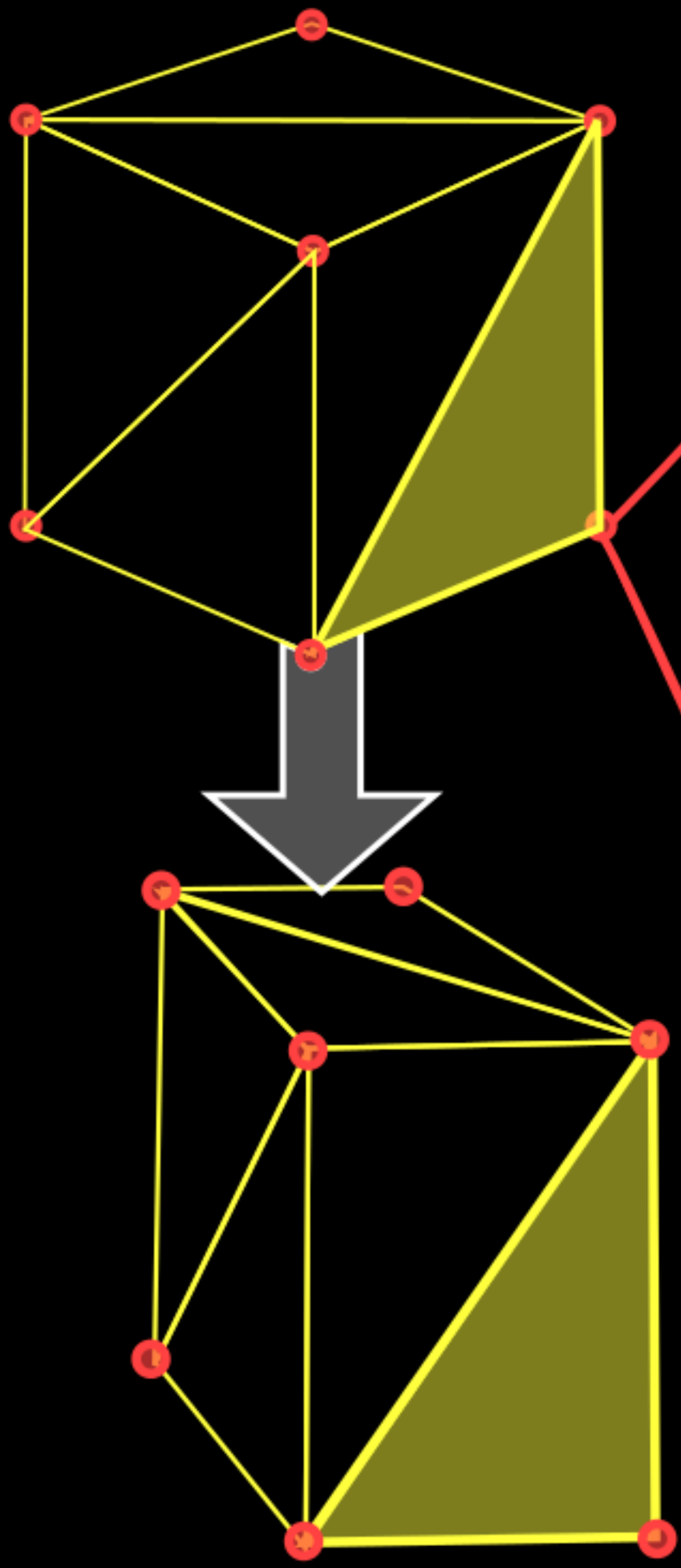
- position (x,y,z)
- texture coordinates (u,v)



vertex

- position (x,y,z)
- texture coordinates (u,v)

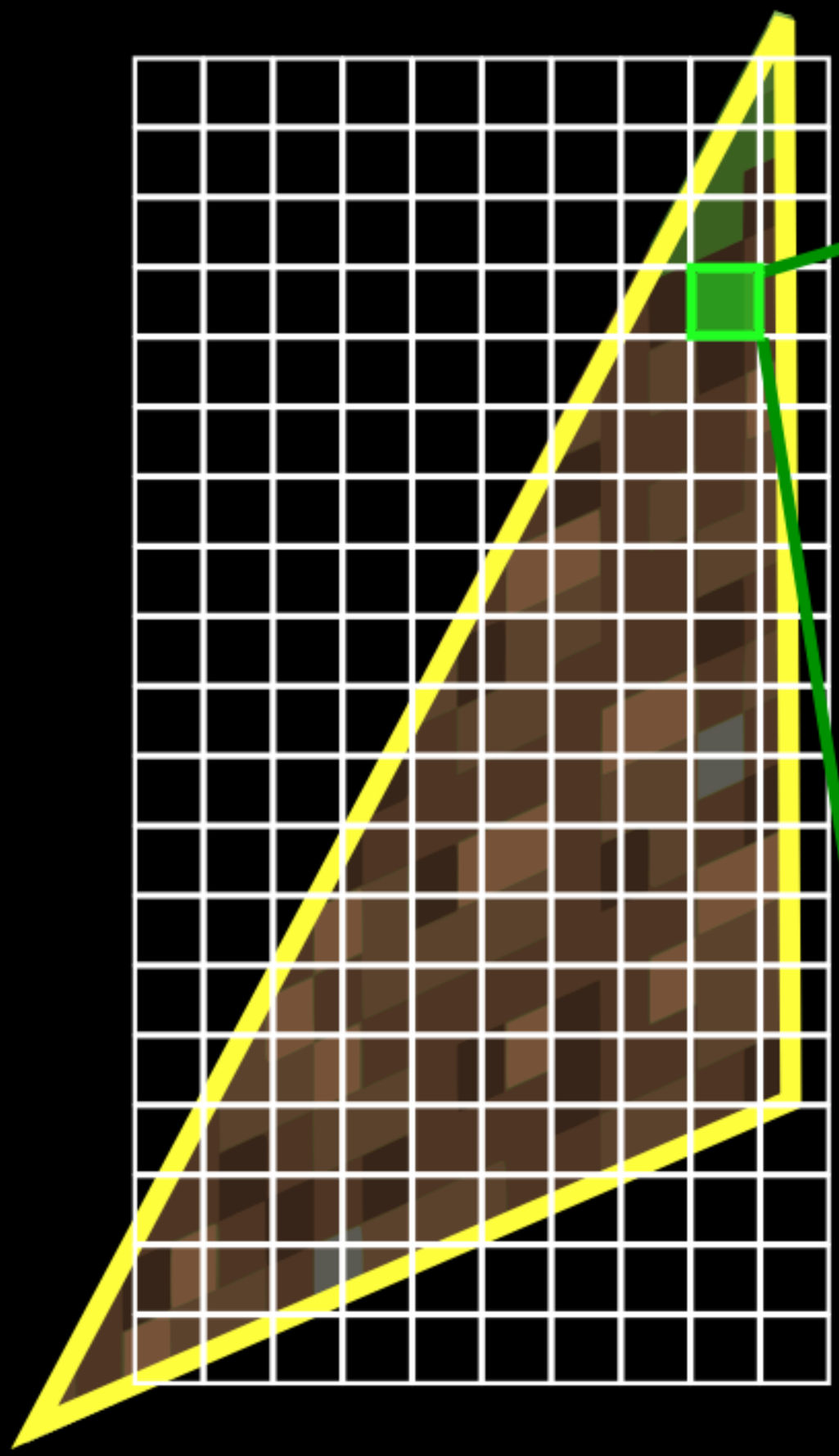




vertex shader

```
void main()
{
  blah bla = bla();
  bla += blah(bla)*bla;
  bla = blah(bla);
  blah();

  gl_Position = bla;
}
```

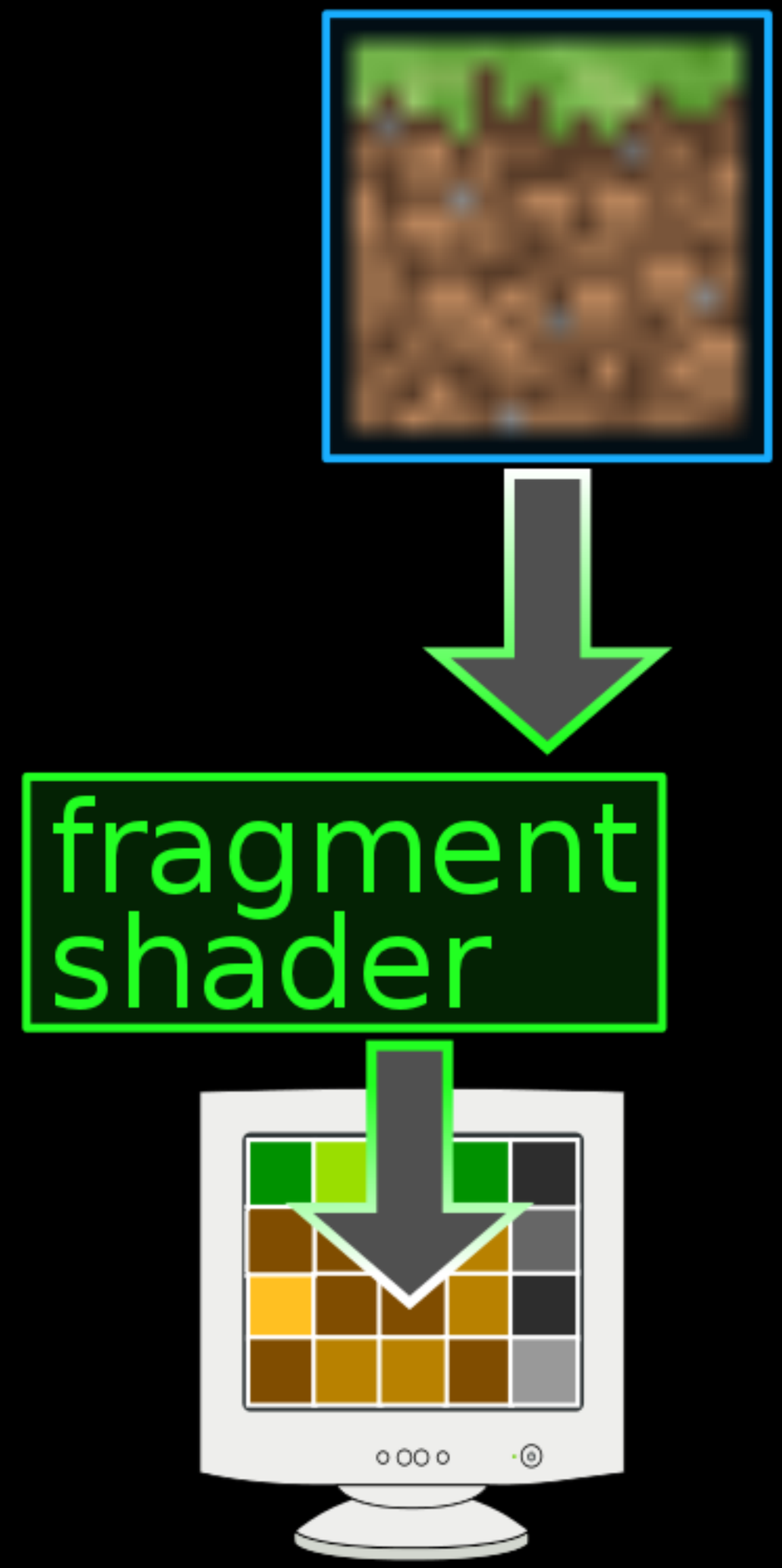


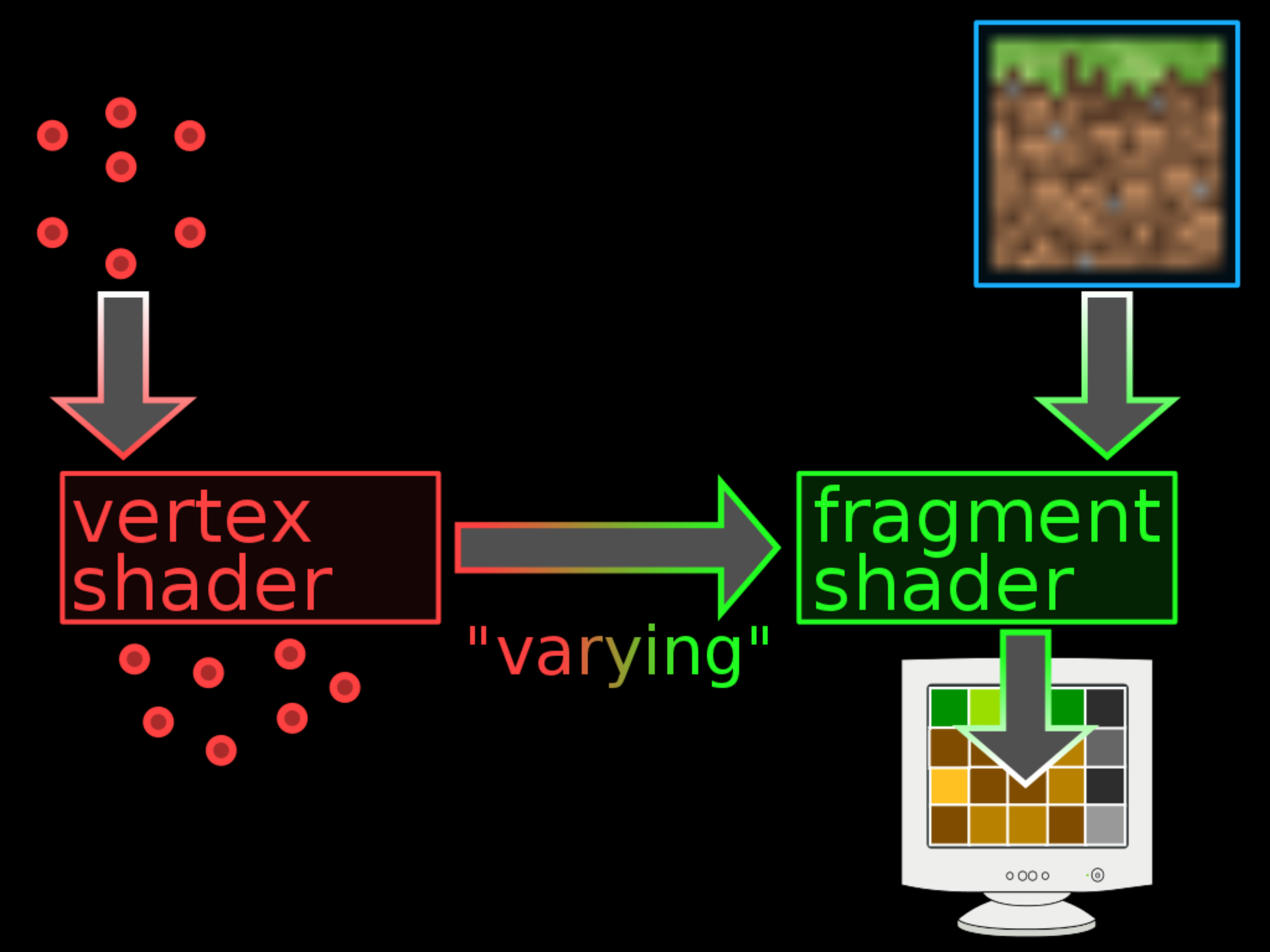
fragment shader

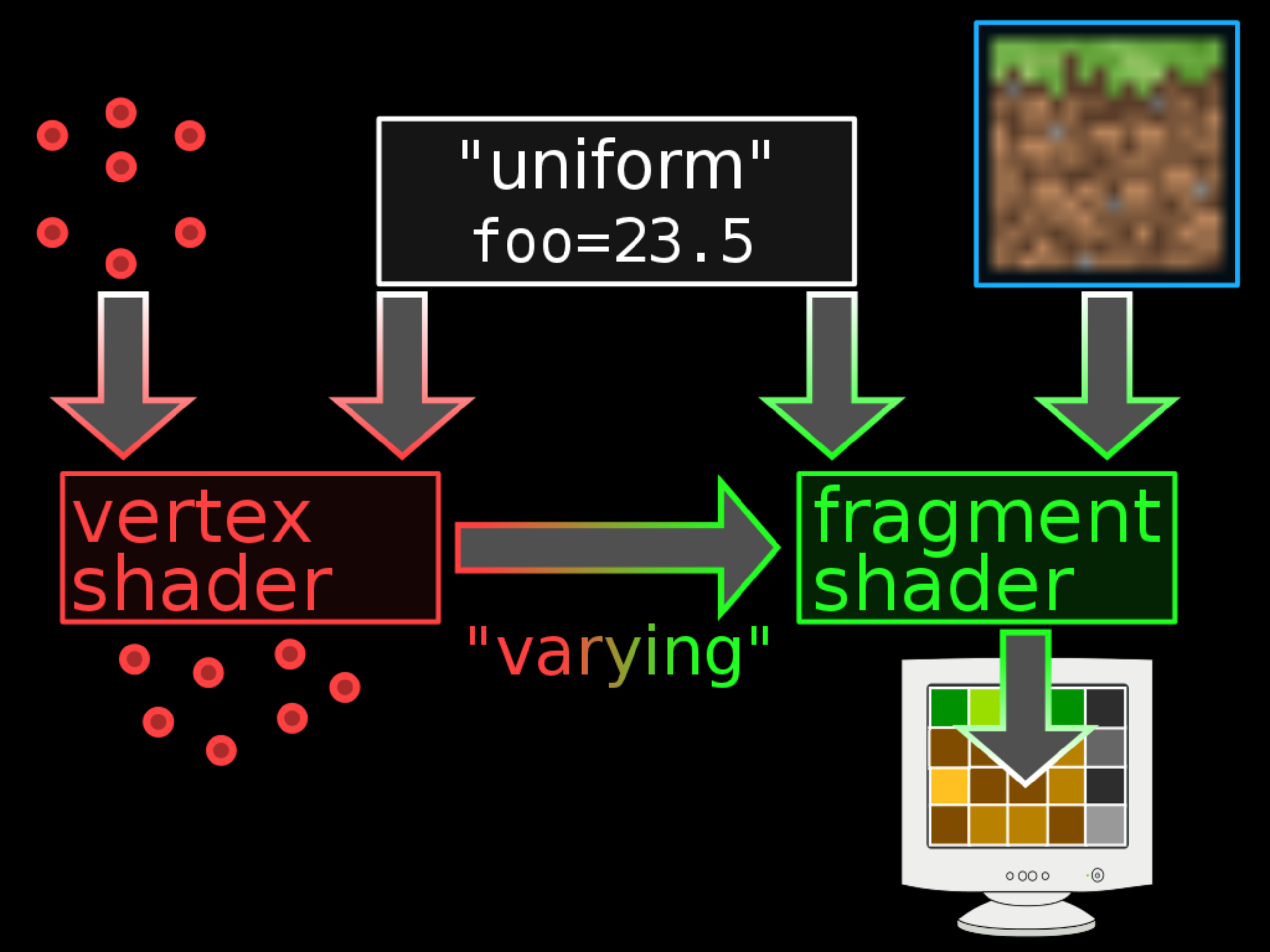
```
void main()
{
  bla  la = bla();
  bla  blah(bla)*bla;
  bla = blah(bla);
  blah();

  gl_FragColor = bla;
}
```


vertex
shader


fragment
shader


vertex shader
"varying"
fragment shader


"uniform"
foo=23.5
vertex shader
"varying"
fragment shader

GLSL Syntax

types:

```
  float vec2 vec3 vec4 mat2 mat3 ...
```

operations:

```
  + - * / dot cross length distance reflect ...
```

```
vec3 myFunction (float foo)
{
  vec3 pos = vec3(1.0, 2.0, 3.0) * foo;
  pos.x += 5;
  pos = normalize(pos);
  return cross(pos,vec3(1.0, 0.0, 0.0));
}
```

```
varying vec2 texCoord;

void main ()
{
  texCoord = gl_MultiTexCoord0.st;
  vec4 pos = gl_ModelViewMatrix * gl_Vertex;

  gl_Position = gl_ProjectionMatrix * pos;
}
```

```
uniform sampler2D tex;
varying vec2 texCoord;
void main ()
{
  gl_FragColor = texture2D(tex, texCoord));
}
```

```
varying vec2 texCoord;
uniform float worldTime;
void main ()
{
  texCoord = gl_MultiTexCoord0.st;
  vec4 pos = gl_ModelViewMatrix * gl_Vertex;
  pos.y += sin(pos.z + worldTime); //wheeee!
  gl_Position = gl_ProjectionMatrix * pos;
}
```

```
uniform sampler2D tex;
varying vec2 texCoord;
void main ()
{
  gl_FragColor = texture2D(tex, texCoord));
}
```