E09: Static

So this is a fun effect. Some nice static like you see on a TV screen. Since there is no `rand()` function to use in a shader we'll use a texture full of random pixels instead. Time to dive into the code.

```
uniform vec2 u_NoiseTextureSize;

void main(){
  gl_Position = cc_Position;
  cc_FragColor = clamp(cc_Color, 0.0, 1.0);
  cc_FragTexCoord1 = cc_TexCoord1;

  vec2 screen01 = (0.5*gl_Position.xy/gl_Position.w + 0.5);
  cc_FragTexCoord2 = screen01*cc_ViewSizeInPixels/u_NoiseTextureSiz
e;
  cc_FragTexCoord2 += cc_Random01.xy;
}
```

Another vertex shader! Woo? Boo? It starts out the same way as the default vertex shader, but what do those last three lines do?

```
vec2 screen01 = (0.5*gl_Position.xy/gl_Position.w + 0.5);
```

In OpenGL, the `gl_Position` coordinates go from (-1, -1) for the bottom left corner of the screen to (1, 1) for the top right. This is called *clip coordinates*. This

isn't quite the same as texture coordinates so we need to do a little math to put them into the [0, 1] range. So what's up with dividing by `gl_Position.w`? Well... It's complicated, and mostly required if you use 3D effects. *"Perspective divide"* is the term to Google for if you want to know more.

```
cc_FragTexCoord2 = screen01*cc_ViewSizeInPixels/u_NoiseTexture
Size;
```

Now that we have a [0, 1] value for the vertex relative to the screen, we can multiply that be the screen size to convert it to pixels. Fortunately, Cocos2D provides a builtin uniform for that, `cc_ViewSiveInPixels`. Since the texture was set to repeat in the Objective-C code, if we divide the pixel coordinate by the noise texture's size We now have a texture coordinate where the pixels in the texture will always line up with the screen. This might be a lot to take in, but it's a common pattern in special effect shaders and definitely something you'll want to learn.

```
cc_FragTexCoord2 += cc_Random01.xy;
```

This last line makes the static animate. While you don't get a `rand()` function in shaders, Cocos2D provides a builtin `vec4` uniform that it fills with random numbers in the range [0, 1] each frame. We can just add this as an offset to the texture coordinate to make the static jitter around randomly. Neat. Onto the fragment shader.

```
uniform sampler2D u_NoiseTexture;

uniform float u_NoiseAmount;

void main(){
  vec4 textureColor = texture2D(cc_MainTexture, cc_FragTexCoord1);
  vec4 noiseColor = texture2D(u_NoiseTexture, cc_FragTexCoord2);

  gl_FragColor = cc_FragColor*textureColor;
  gl_FragColor.rgb += (textureColor.a*u_NoiseAmount)*(2.0*noiseColo
r.rgb - 1.0);
}
```

Nothing fancy here. Note that the noise value is converted to the [-1, 1] range and premultiplied by the texture's alpha.

Excercises:

- Try making the static grayscale instead of colored.
- Try making the static scroll instead of jumping around randomly.