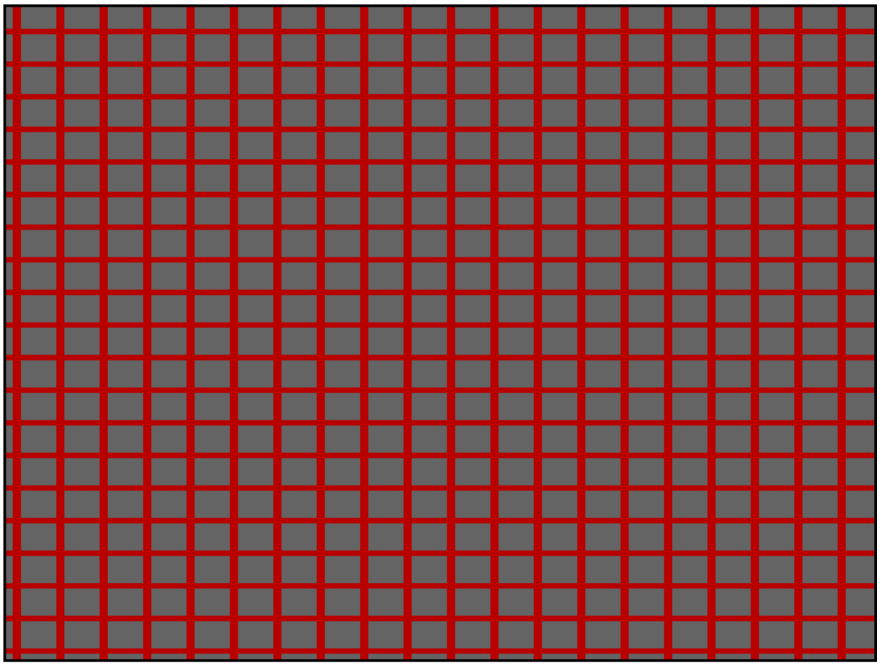
Freddy Gu
The Ohio State University
CSE 5542
08/28/2017

Homework 1

1) For the first backdrop, I wanted to experiment with different patterns and try to create a grid like pattern using the colors from the provided implementation.

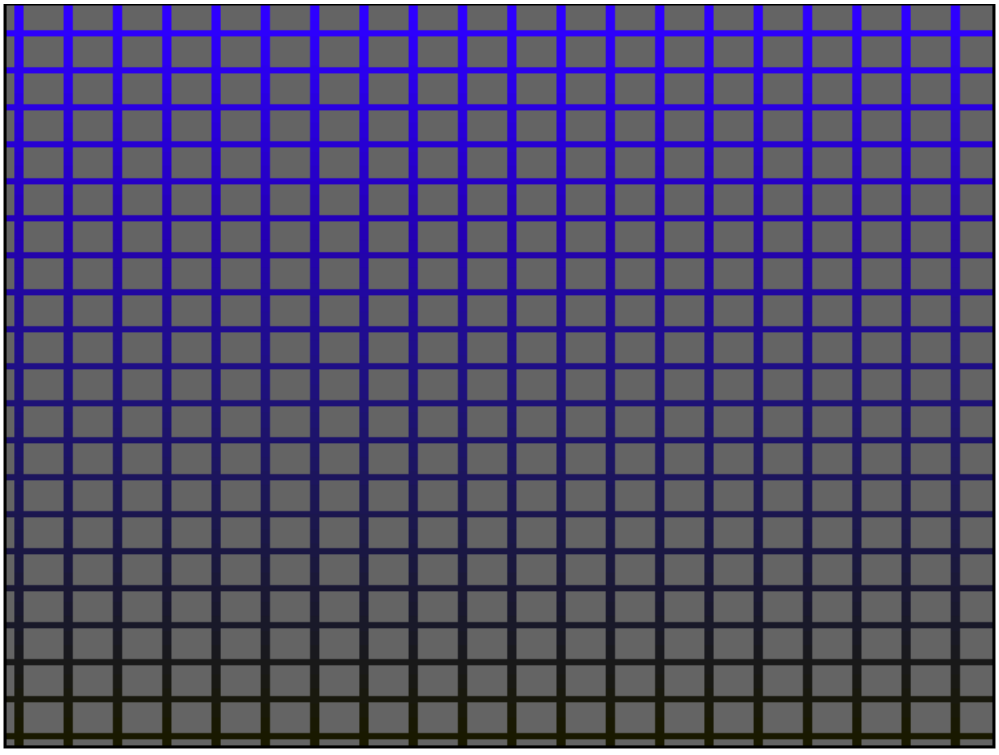
```
precision highp float;
void main(void) {
  float yThreshold;
  float xThreshold;
  const float xScale = 1.0 / 640.0;
  const float yScale = 1.0 / 480.0;
  const vec4 scarlet = vec4(0.733, 0.0, 0.0, 1.0);
  const vec4 grey = vec4(0.4, 0.4, 0.4, 1.0);
  float x = xScale * gl_FragCoord.x;
  float y = yScale * gl_FragCoord.y;
  vec4 color;
  yThreshold = 30.0*(sin(40.0 * 3.1415*x) + 1.0);
  xThreshold = 30.0*(sin(40.0 * 3.1415*y) + 1.0);
  if(yThreshold < 55.0 && xThreshold < 55.0)
    color = grey;
  else
    color = scarlet;
  gl_FragColor = color;
}
```



Freddy Gu
The Ohio State University
CSE 5542
08/28/2017

2) For the second backdrop, I wanted to experiment with directly using the gl_FragCoord vector as a stepper value.

```
precision highp float;
void main(void) {
  float yThreshold;
  float xThreshold;
  const float xScale = 1.0 / 640.0;
  const float yScale = 1.0 / 480.0;
  const vec4 grey = vec4(0.4, 0.4, 0.4, 1.0);
  float x = xScale * gl_FragCoord.x;
  float y = yScale * gl_FragCoord.y;
  vec4 color;
  yThreshold = 30.0*(sin(40.0 * 3.1415*x) + 1.0);
  xThreshold = 30.0*(sin(40.0 * 3.1415*y) + 1.0);
  if(yThreshold < 55.0 && xThreshold < 55.0)
    color = grey;
  else
    color = vec4(0.1, 0.1, gl_FragCoord.y / 480.0, 1.0);
  gl_FragColor = color;
}
```



Freddy Gu
The Ohio State University
CSE 5542
08/28/2017

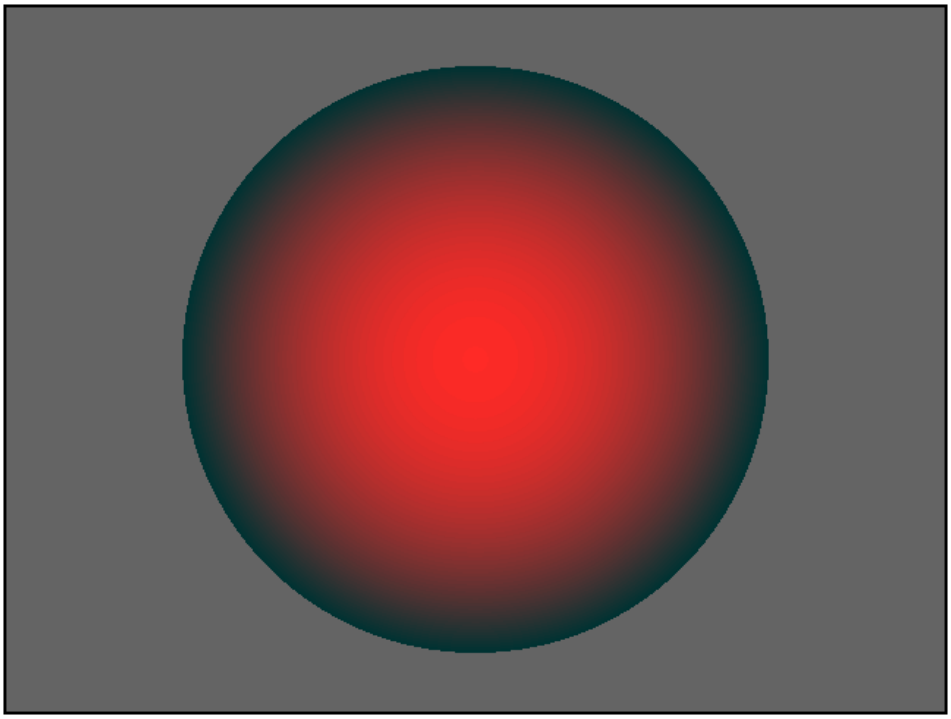
3) For the third background, I wanted to experiment with circular shapes. The shader simply shows scarlet if the position is within a centered circle, and grey when outside.

```
precision highp float;
void main(void) {
  const vec4 scarlet = vec4(0.733, 0.2, 0.2, 1.0);
  const vec4 grey = vec4(0.4, 0.4, 0.4, 1.0);
  float xCenter = 640.0 / 2.0;
  float yCenter = 480.0 / 2.0;
  float xPos = gl_FragCoord.x;
  float yPos = gl_FragCoord.y;
  float r = 150.0;

  float dist = pow(xPos - xCenter, 2.0) + pow(yPos - yCenter, 2.0);

  if (dist < pow(r, 2.0)) {
    gl_FragColor = scarlet;
  } else {
    gl_FragColor = grey;
  }
}
```

Freddy Gu
The Ohio State University
CSE 5542
08/28/2017

4) For the fourth background, I wanted to experiment with circular gradients. In doing so I got a fragment that looked like a 2D representation of a red sphere.

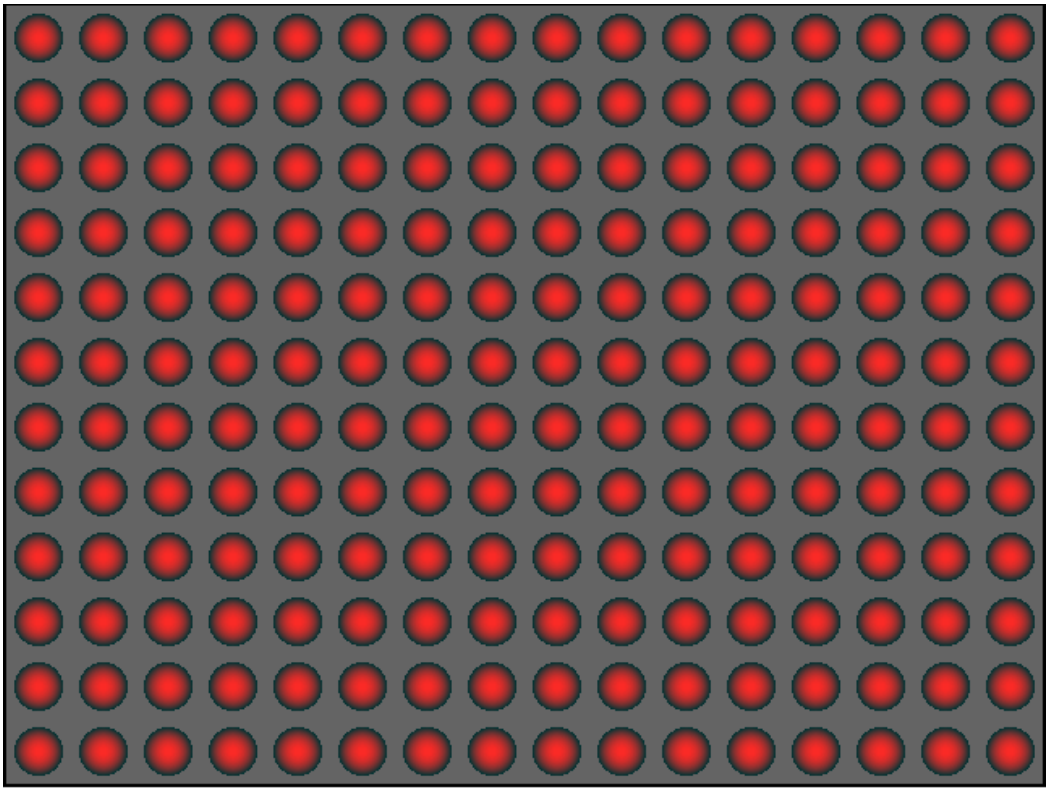
```
precision highp float;
void main(void) {
  const vec4 scarlet = vec4(0.733, 0.2, 0.2, 1.0);
  const vec4 grey = vec4(0.4, 0.4, 0.4, 1.0);
  float xCenter = 640.0 / 2.0;
  float yCenter = 480.0 / 2.0;
  float xPos = gl_FragCoord.x;
  float yPos = gl_FragCoord.y;
  float r = 200.0;

  float dist = pow(xPos - xCenter, 2.0) + pow(yPos - yCenter, 2.0);

  if (dist < pow(r, 2.0)) {
    gl_FragColor = vec4(1.0 - (dist / pow(r, 2.0)), 0.2, 0.2, 1.0);
  } else {
    gl_FragColor = grey;
  }
}
```



Freddy Gu
The Ohio State University
CSE 5542
08/28/2017

5) For the fifth background, I experimented with tiling the same fragment across the entire viewport. This was done relatively simply by using the mod function to find the distance of each point within its inner area from its center.

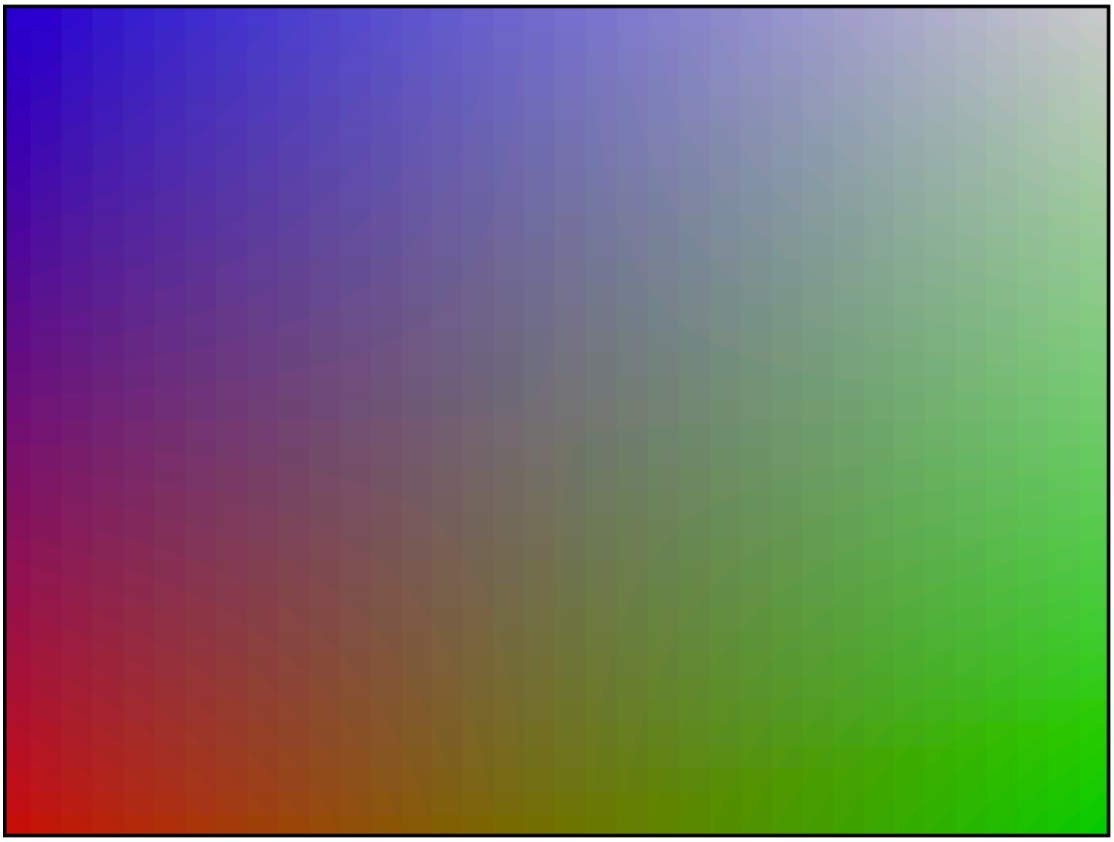
```
precision highp float;
void main(void) {
  const vec4 scarlet = vec4(0.733, 0.2, 0.2, 1.0);
  const vec4 grey = vec4(0.4, 0.4, 0.4, 1.0);
  float xCenter = 40.0 / 2.0;
  float yCenter = 40.0 / 2.0;
  float xPos = mod(gl_FragCoord.x, 40.0);
  float yPos = mod(gl_FragCoord.y, 40.0);
  float r = 15.0;

  float dist = pow(xPos - xCenter, 2.0) + pow(yPos - yCenter, 2.0);

  if (dist < pow(r, 2.0)) {
    gl_FragColor = vec4(1.0 - (dist / pow(r, 2.0)), 0.2, 0.2, 1.0);
  } else {
    gl_FragColor = grey;
  }
}
```



Freddy Gu
The Ohio State University
CSE 5542
08/28/2017

6) For the sixth background, I wanted to explore more with gradients. I made a four-cornered gradient with four different base colors using the mix functions.

```
precision highp float;
void main(void) {
  const vec4 redBase = vec4(0.8, 0.1, 0.1, 1.0);
  const vec4 greenBase = vec4(0.1, 0.8, 0.1, 1.0);
  const vec4 blueBase = vec4(0.1, 0.1, 0.8, 1.0);
  const vec4 whiteBase = vec4(0.8, 0.8, 0.8, 1.0);
  float xPos = gl_FragCoord.x;
  float yPos = gl_FragCoord.y;

  vec4 xBase1 = mix( redBase, greenBase, xPos / 640.0 );
  vec4 xBase2 = mix( blueBase, whiteBase, xPos / 640.0 );

  gl_FragColor = mix ( xBase1, xBase2, yPos / 480.0 );
}
```

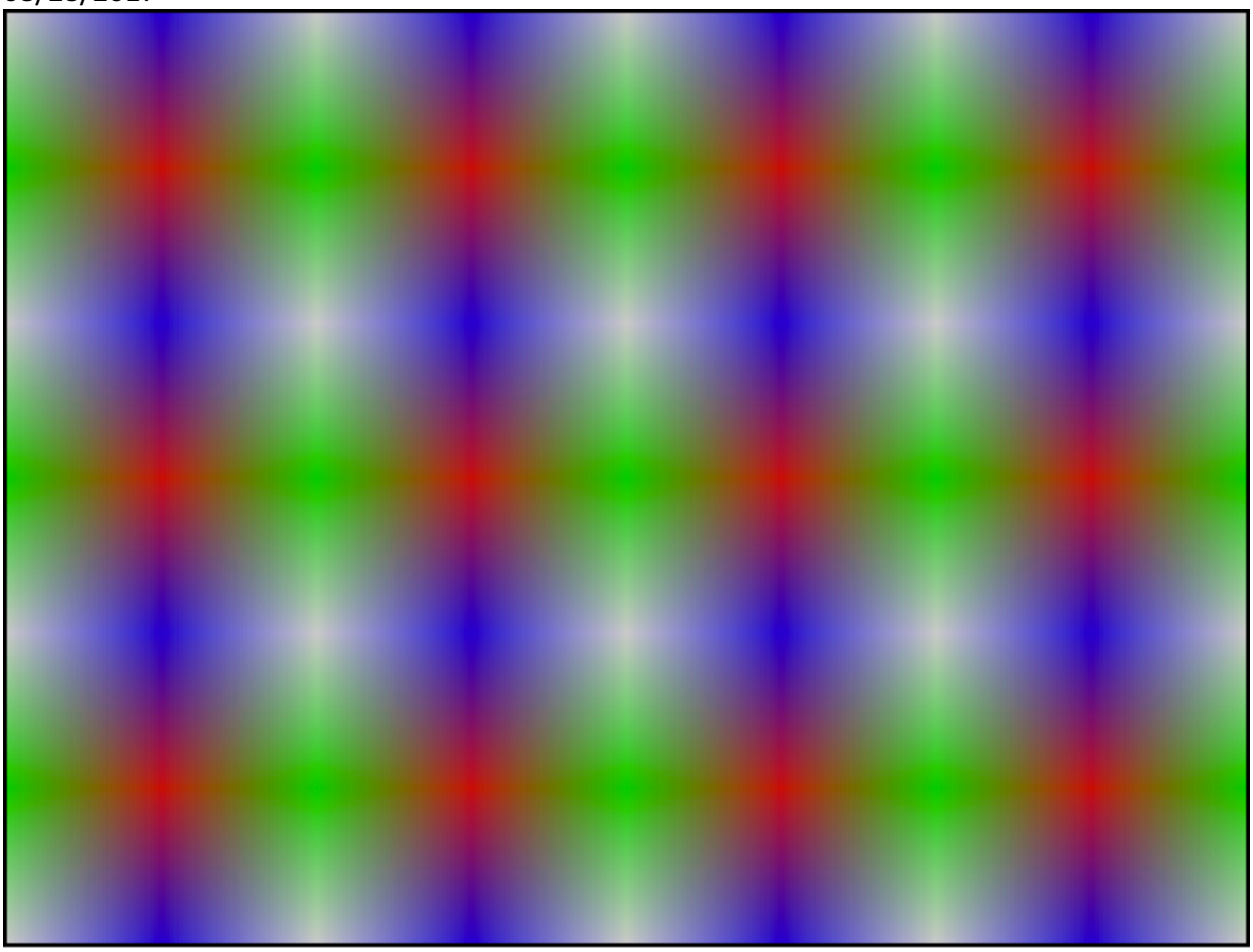


Freddy Gu
The Ohio State University
CSE 5542
08/28/2017

7) For the last background, I wanted to have different segment areas for the same gradient pattern from the sixth background. Additionally, I wanted the colors in each corner of the inner areas to match with their neighbors, so I added some logic to flip the mix statements depending on which inner area they were located in. The result is a background with smooth transitions between the colors between each area.

```
precision highp float;
void main(void) {
  const vec4 redBase = vec4(0.8, 0.1, 0.1, 1.0);
  const vec4 greenBase = vec4(0.1, 0.8, 0.1, 1.0);
  const vec4 blueBase = vec4(0.1, 0.1, 0.8, 1.0);
  const vec4 whiteBase = vec4(0.8, 0.8, 0.8, 1.0);
  float xSegment = floor(gl_FragCoord.x / 80.0);
  float ySegment = floor(gl_FragCoord.y / 80.0);
  float xPos =  mod(gl_FragCoord.x, 80.0);
  float yPos = mod(gl_FragCoord.y, 80.0);

  vec4 xBase1;
  vec4 xBase2;
  vec4 yMix;

  if (mod(xSegment, 2.0) == 1.0) {
    xBase1 = mix( redBase, greenBase, xPos / 80.0 );
    xBase2 = mix( blueBase, whiteBase, xPos / 80.0 );
  } else {
    xBase1 = mix( greenBase, redBase, xPos / 80.0 );
    xBase2 = mix( whiteBase, blueBase, xPos / 80.0 );
  }

  if (mod(ySegment, 2.0) == 1.0) {
    yMix = mix ( xBase1, xBase2, yPos / 80.0 );
  } else {
    yMix = mix ( xBase2, xBase1, yPos / 80.0 );
  }

  gl_FragColor = yMix;
}
```

Freddy Gu
The Ohio State University
CSE 5542
08/28/2017



Conclusion) My system is running MacOS Sierra using Google Chrome with an Intel I7 processor and nVidia graphics card. I didn't encounter any issues.