Fragment Shaders

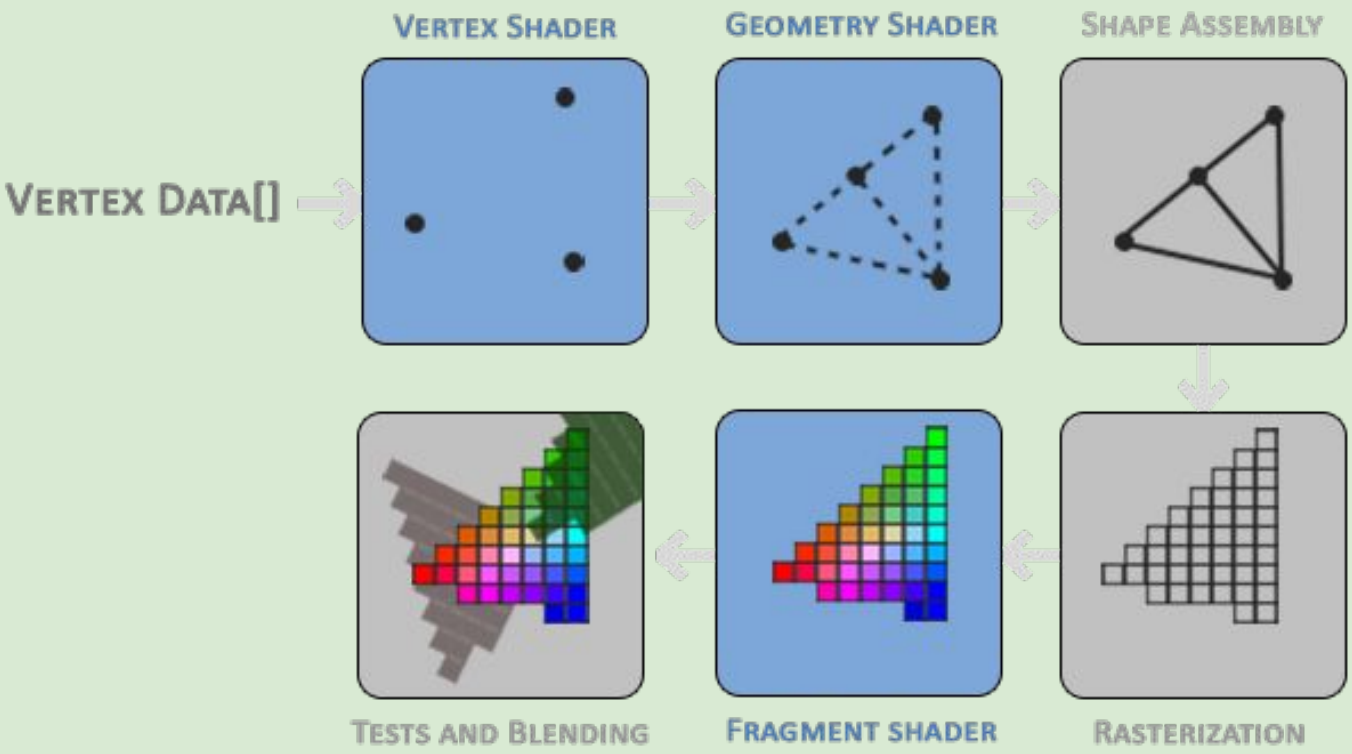
uocsclub.net/codeandcoffee/shaders

1920x1080

= 2,073,600

x 60 FPS

= 124,416,000

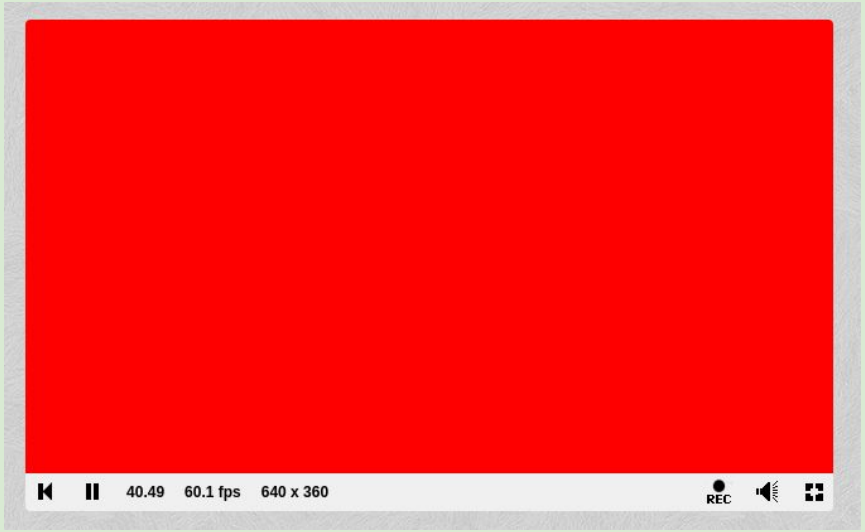

VERTEX SHADER
GEOMETRY SHADER
SHAPE ASSEMBLY
VERTEX DATA[]
TESTS AND BLENDING
FRAGMENT SHADER
RASTERIZATION

Colours!

https://www.shadertoy.com/new

Colours!

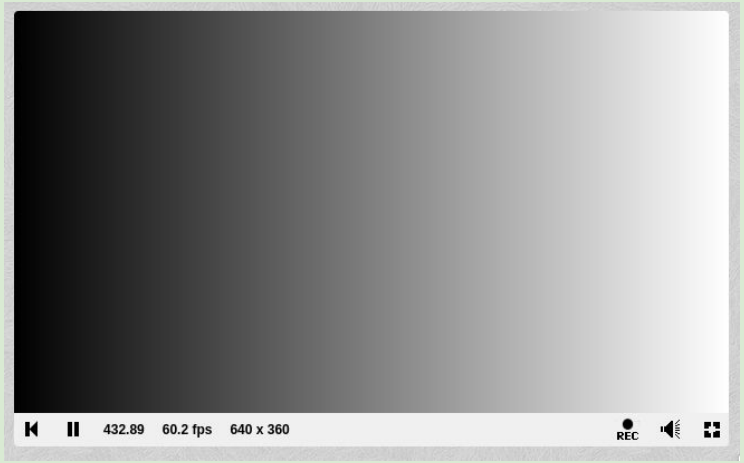
```
void mainImage( out vec4 fragColor, in vec2 fragCoord ) {
    fragColor = vec4(1.0, 0.0, 0.0, 1.0); // try making green or blue
}
```



Colours!

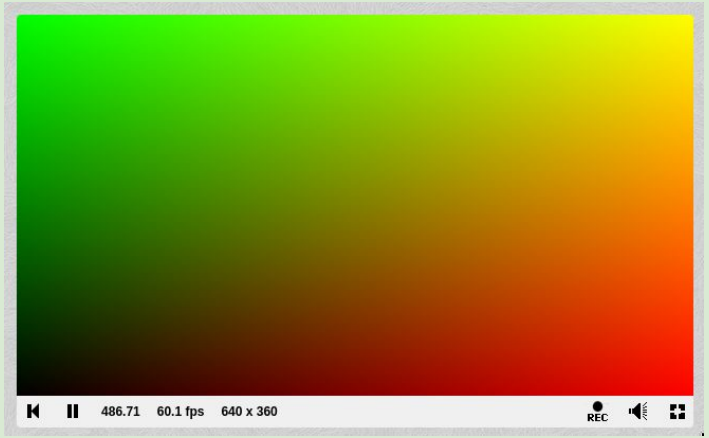
- Colours are a vec4 of (Red, Green, Blue, Alpha)
- We can use screen coordinates to choose a colour

```
void mainImage( out vec4 fragColor, in vec2 fragCoord )
{
    // Normalized pixel coordinates (from 0 to 1)
    vec2 uv = fragCoord/iResolution.xy;


    fragColor = vec4(uv.x, uv.x, uv.x, 1.0);
}
```

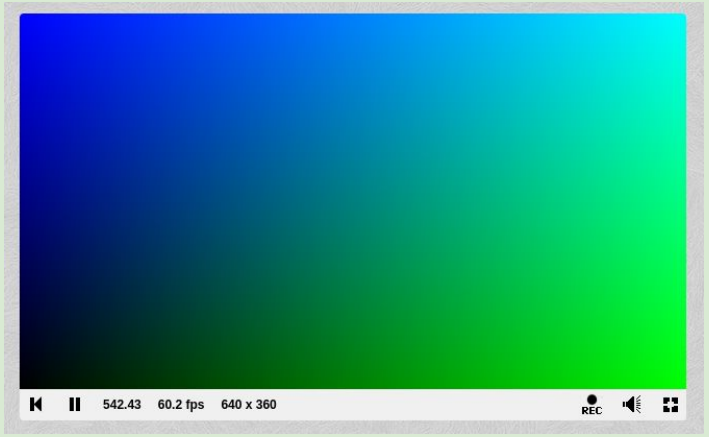


Colours!

fragColor = vec4(uv, 0.0, 1.0);

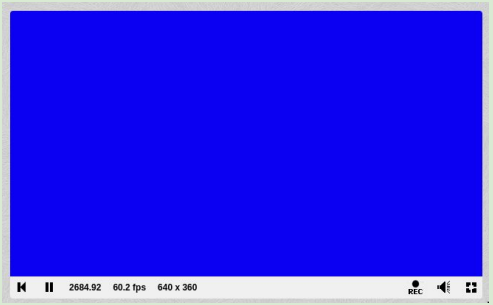


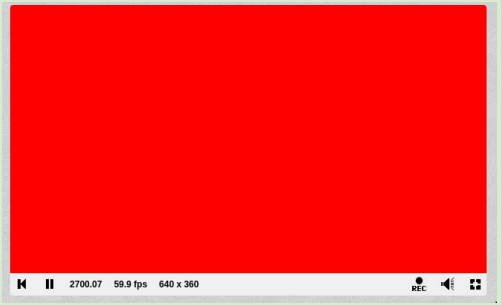
fragColor = vec4(0.0, uv, 1.0);



Colours!

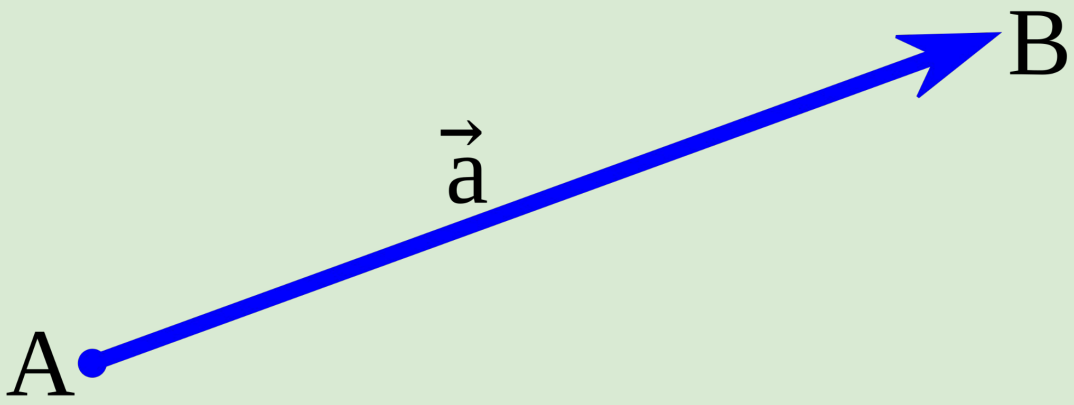
```
vec3 red = vec3(1., 0., 0.);

vec3 blue = vec3(0., 0., 1.);

fragColor = vec4(mix(red, blue, (sin(iTime)+1.)/2.), 1.0);
```





Vectors!

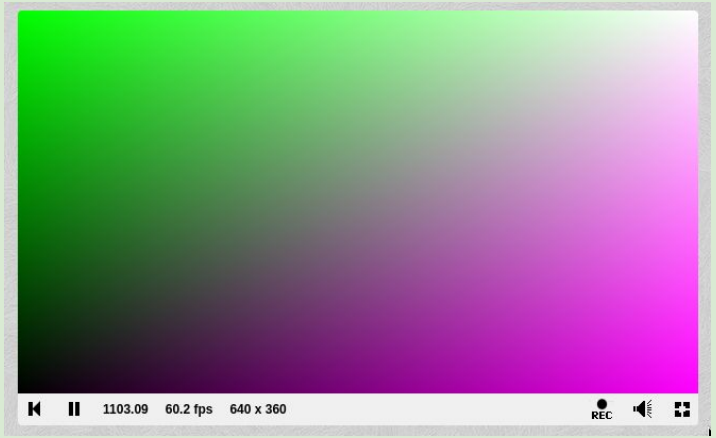
- vec2, vec3, vec4
- Length
- Vector constructors
- Swizzling
- Fract, max, min



Vectors examples

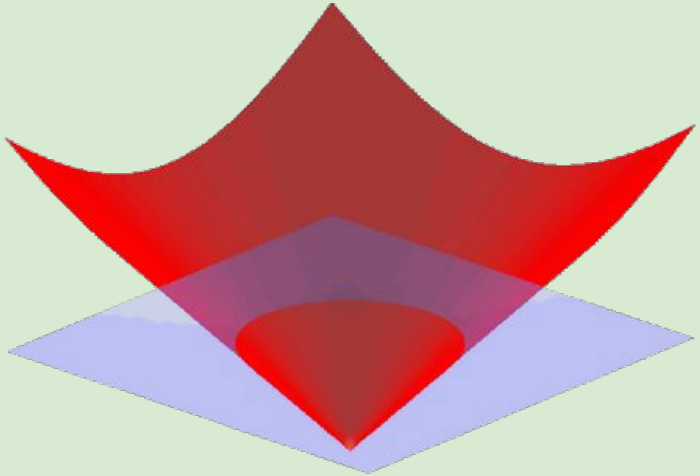
```
float col = length(uv);

fragColor = vec4(col, col, col, 1.0);
```

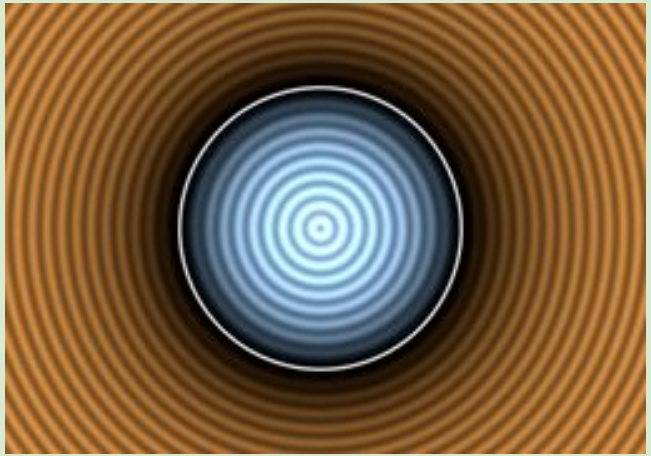

```
fragColor = vec4(uv.xyx, 1.0);
```

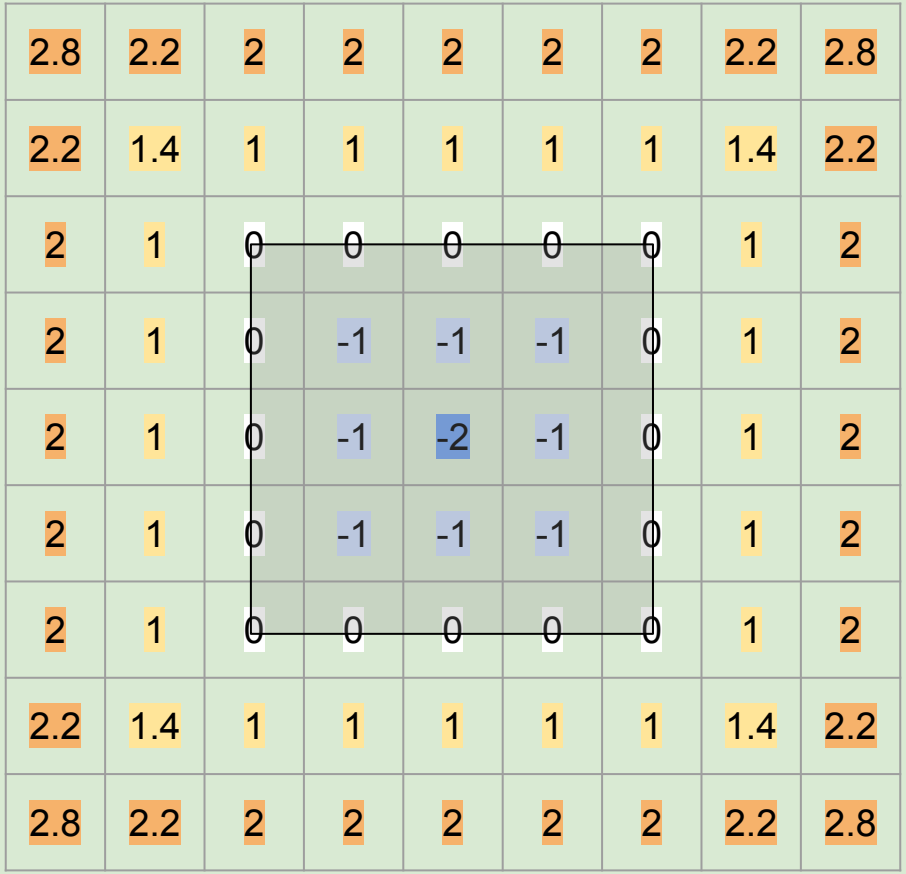


SDF

- Signed distance field! (not the Super Dimension Fortress)
- Every point encodes its distance from a shape
- https://iquilezles.org/articles/distfunctions2d/ have fun!

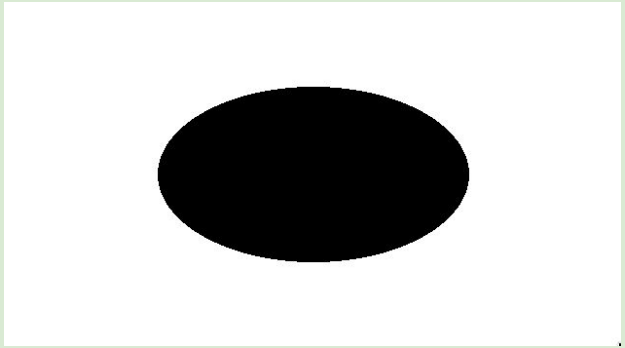





2.8 2.2 2 2 2 2 2 2.2 2.8
2.2 1.4 1 1 1 1 1 1.4 2.2
2 1 0 0 0 0 0 1 2
2 1 0 -1 -1 -1 0 1 2
2 1 0 -1 -2 -1 0 1 2
2 1 0 -1 -1 -1 0 1 2
2 1 0 0 0 0 0 1 2
2.2 1.4 1 1 1 1 1 1.4 2.2
2.8 2.2 2 2 2 2 2 2.2 2.8

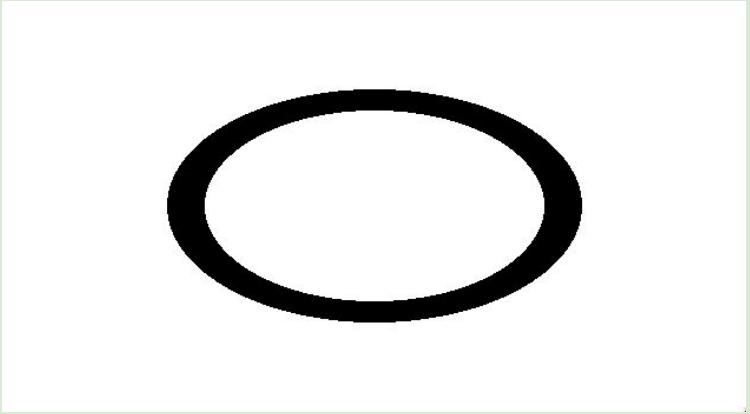
Circle SDF

```
void mainImage( out vec4 fragColor, in vec2 fragCoord ) {
    // -1 to 1
    vec2 uv = fragCoord/iResolution.xy * 2. - 1.;


    float r = .5;
    float sdf = length(uv) - r;
    float c = step(0., sdf); // try smoothstep(smoothstep(-0.1, 0.1, sdf);
    fragColor = vec4(c, c, c, 1.);
}
```



Ring SDF

```
void mainImage( out vec4 fragColor, in vec2 fragCoord ) {
    // -1 to 1
    vec2 uv = fragCoord/iResolution.xy * 2. - 1.;


    float r = .5;
    float halfThickness = .05;
    float sdf = abs(length(uv) - r) - halfThickness;
    float c = step(0., sdf); // try smoothstep(smoothstep(-0.1, 0.1, sdf);
    fragColor = vec4(c, c, c, 1.);
}
```

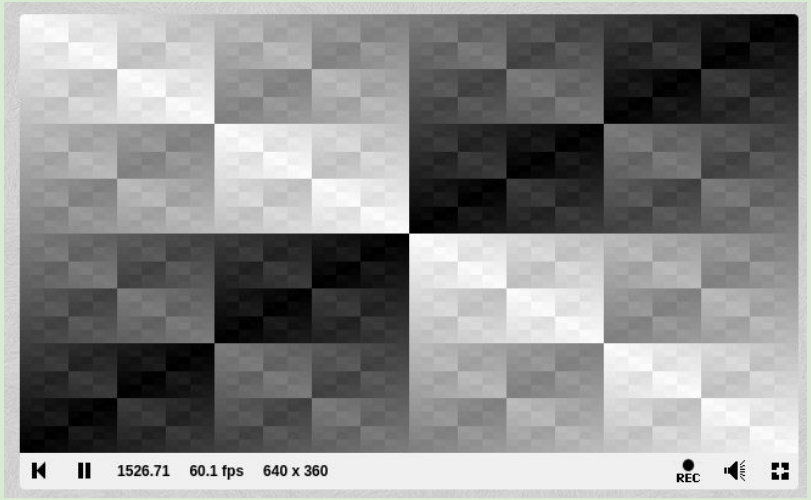


Box SDF

```
void mainImage( out vec4 fragColor, in vec2 fragCoord ) {
    // -1 to 1
    vec2 uv = fragCoord/iResolution.xy * 2. - 1.;


    float r = .5;
    vec2 d = abs(uv) - r;
    float sdf = length(max(d, 0.)) + min(max(d.x, d.y), 0.);
    float c = step(0., sdf); // what if we vary the cutoff?
    fragColor = vec4(c, c, c, 1.);
}
```

Visualizing bit ops

```
ivec2 i = ivec2(uv * 64.);

float col = float(i.x ^ i.y) / 64.;

fragColor = vec4(col, col, col, 1.0);
```



x ^ x = 0 -> bottom-left to top-right (along y=x)

x ^ 0 = x -> bottom line (along y=0)

x ^ 1 = not x -> top line (along y=1)

How fast to escape?

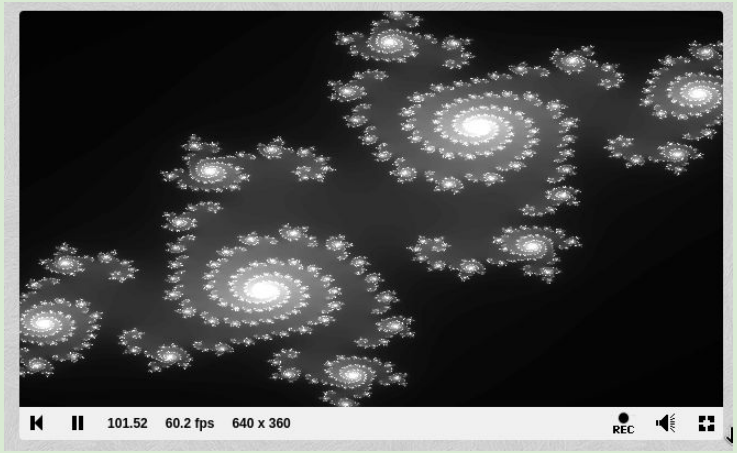
```
#define ITERATIONS 100


vec2 sq(vec2 p) {
    return vec2(p.x*p.x - p.y*p.y, 2.*p.x*p.y);
}
```

How fast to escape?

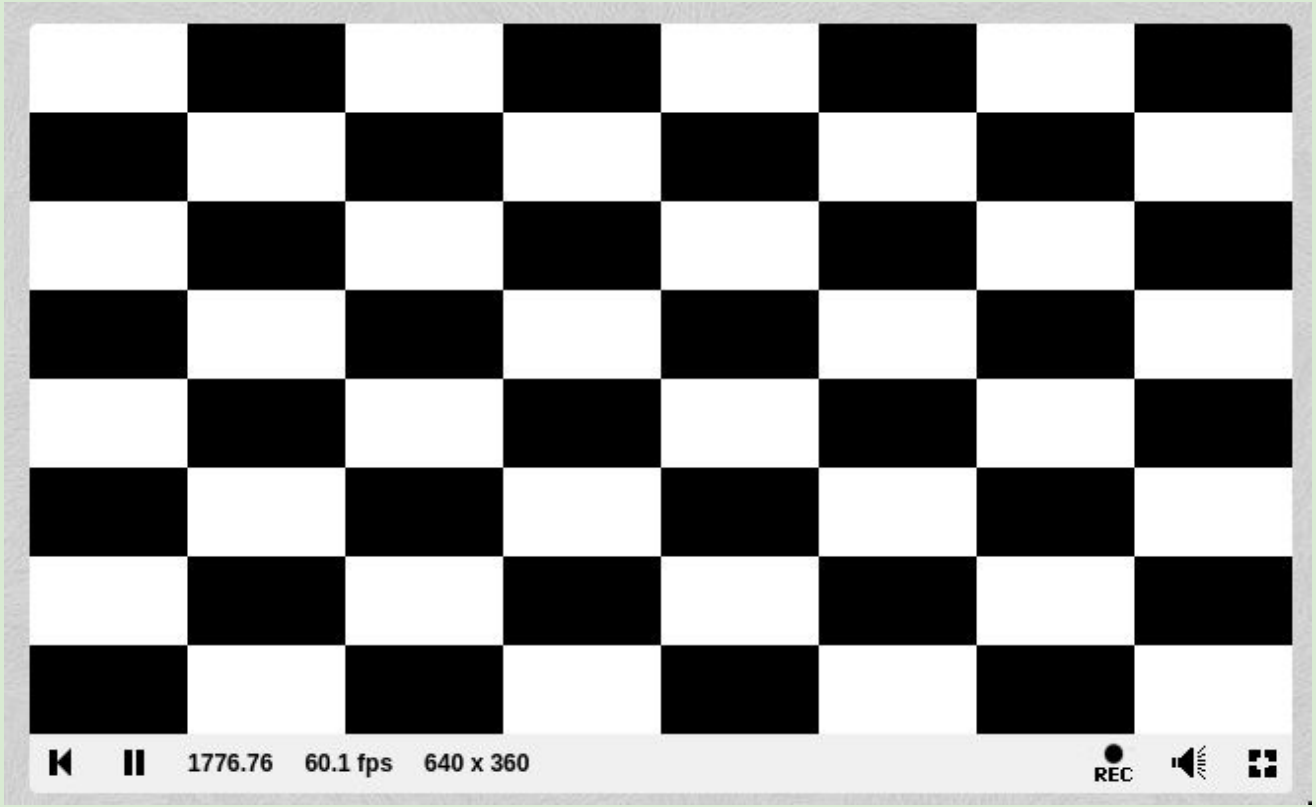
```
float escape(vec2 p, vec2 mouse) {
    for (int i = 0; i < ITERATIONS; i++) {
        if (length(p) > 2.) return float(i) / float(ITERATIONS);
        p = sq(p) + mouse;
    }
    return 1.;
}
```

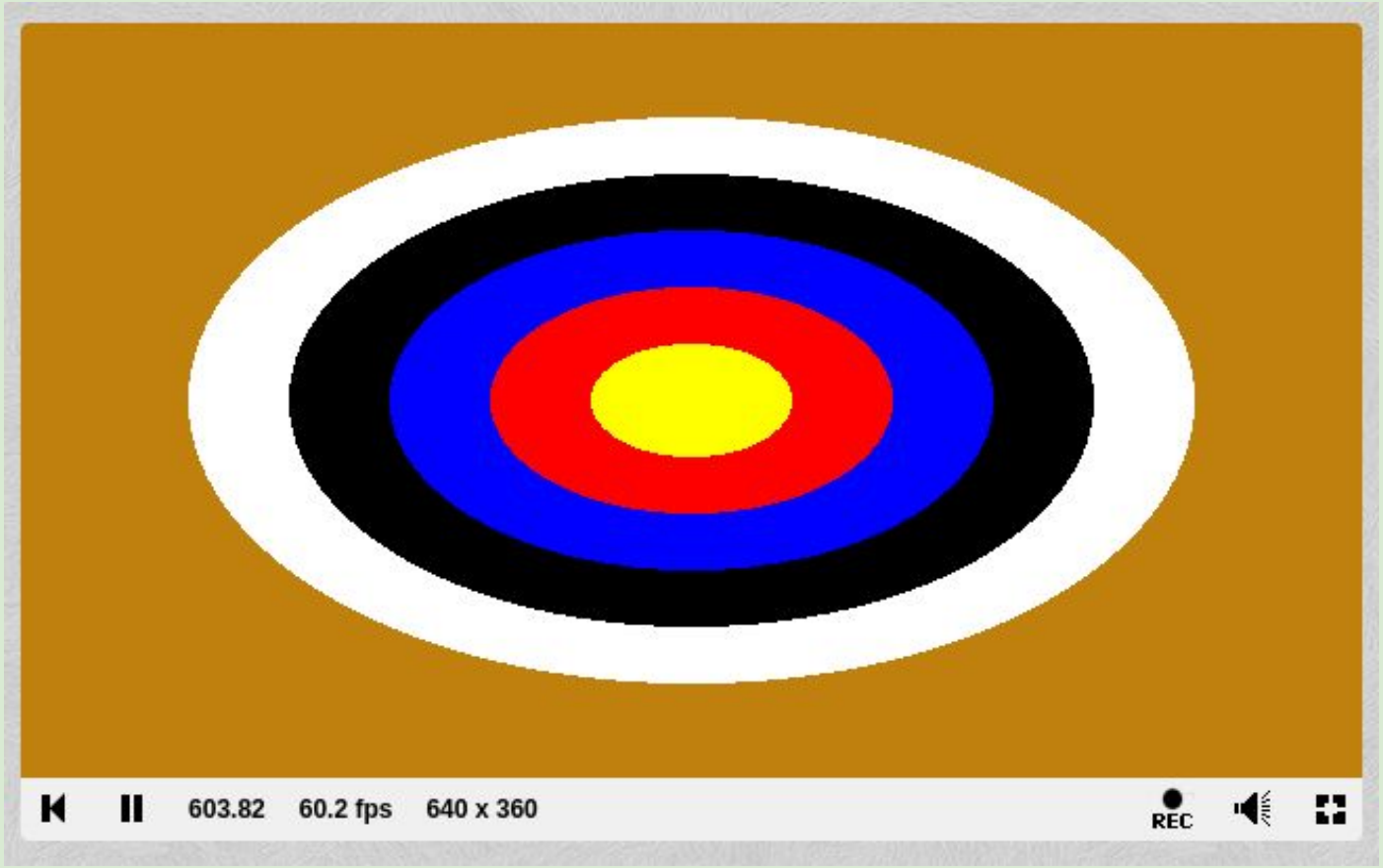
How fast to escape?

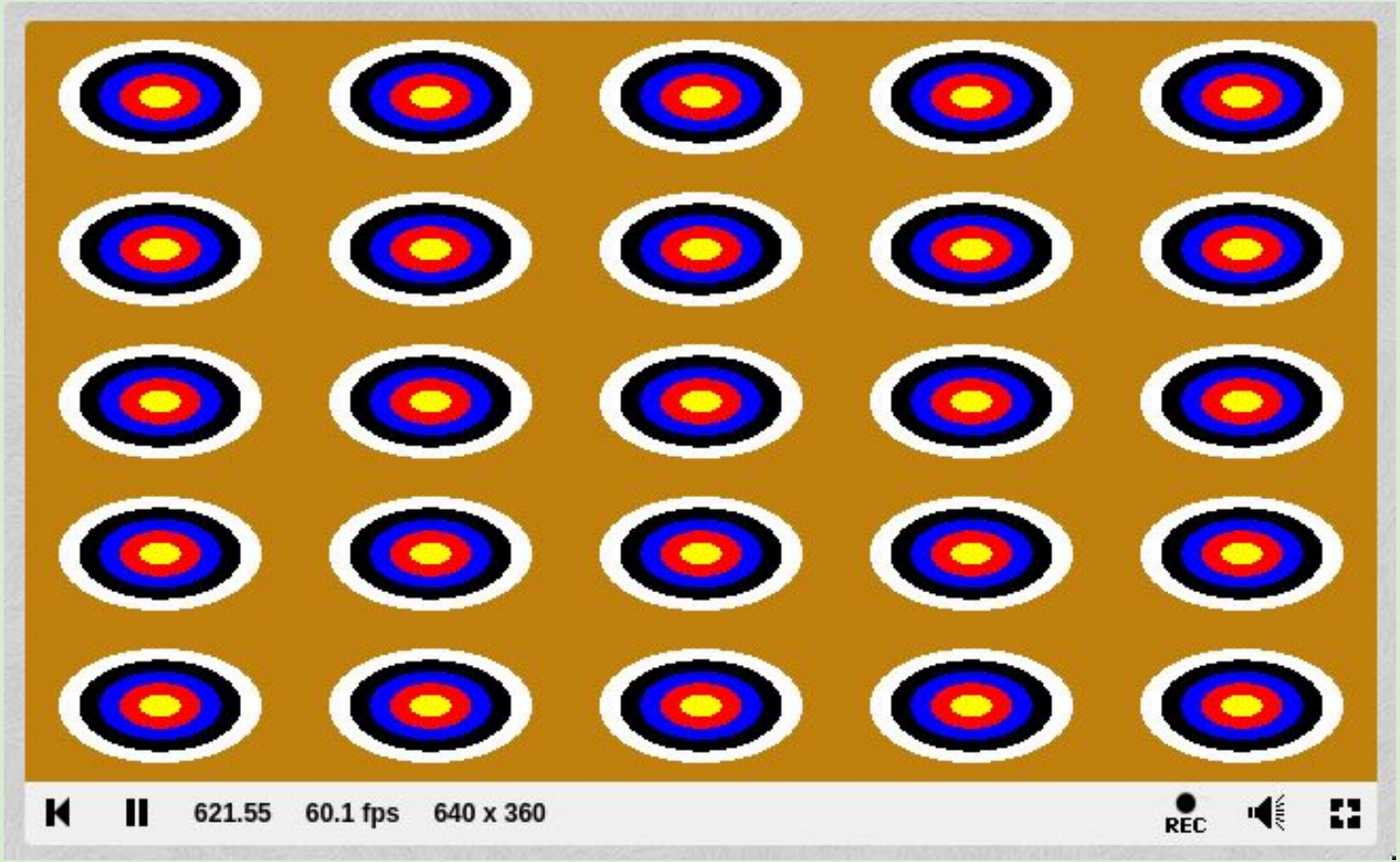
```
void mainImage( out vec4 fragColor, in vec2 fragCoord)  {
    // -1 to 1
    vec2 uv = fragCoord/iResolution.xy * 2. - 1.;
    vec2 mouse = iMouse.xy/iResolution.xy * 2. - 1.;


    float c = escape(uv, mouse);
    fragColor = vec4(c, c, c, 1.0);
}
```

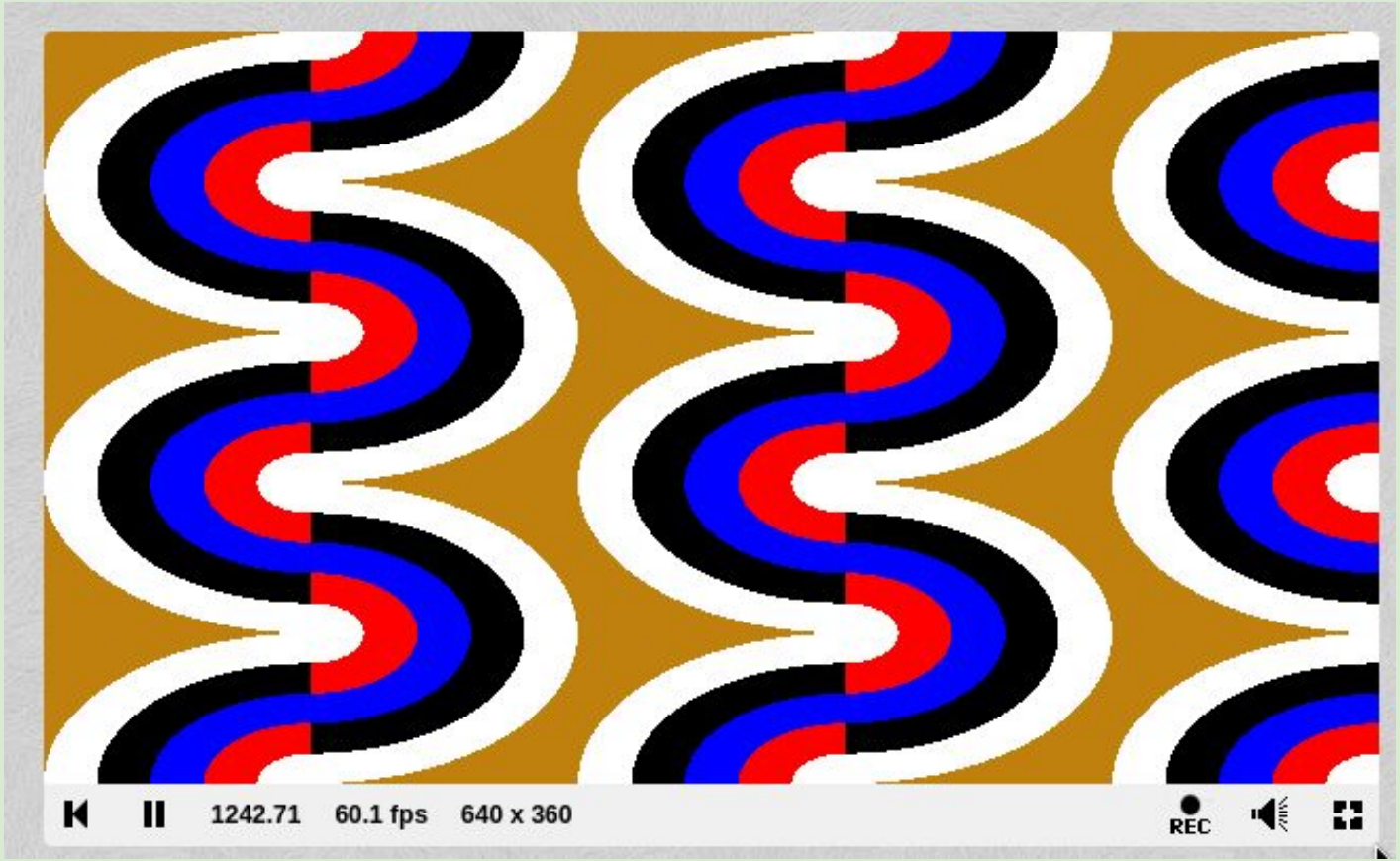


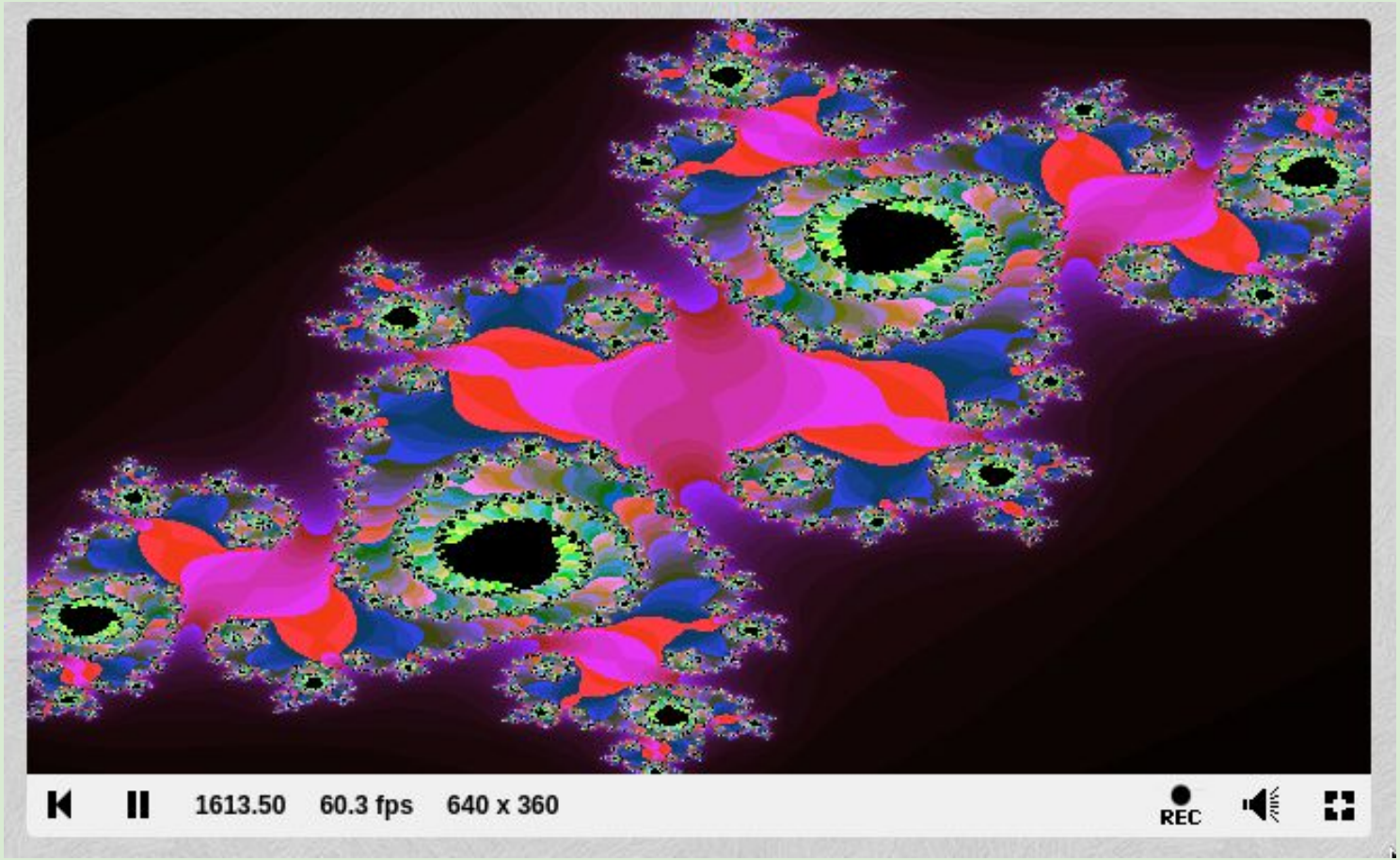
Your turn!

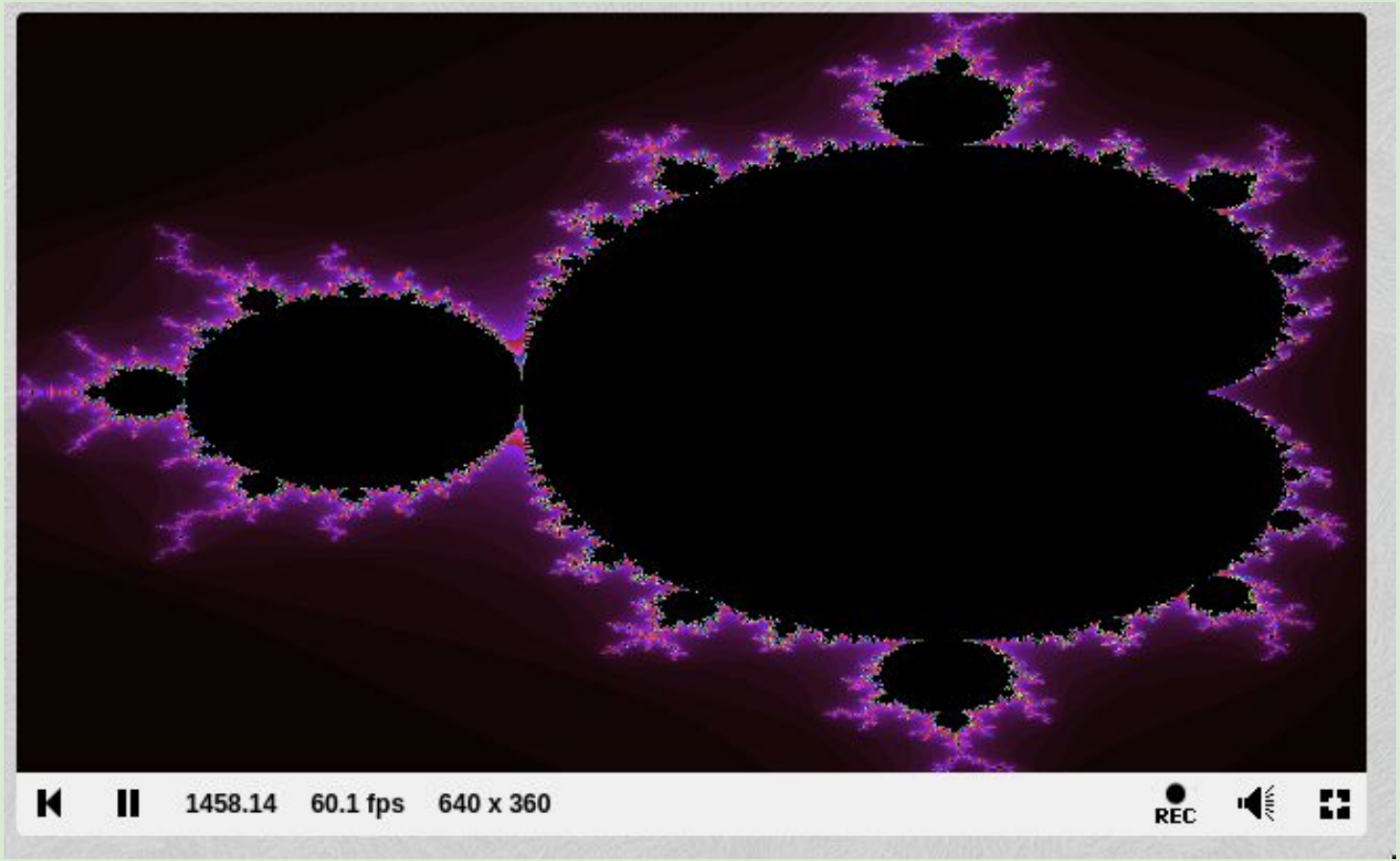


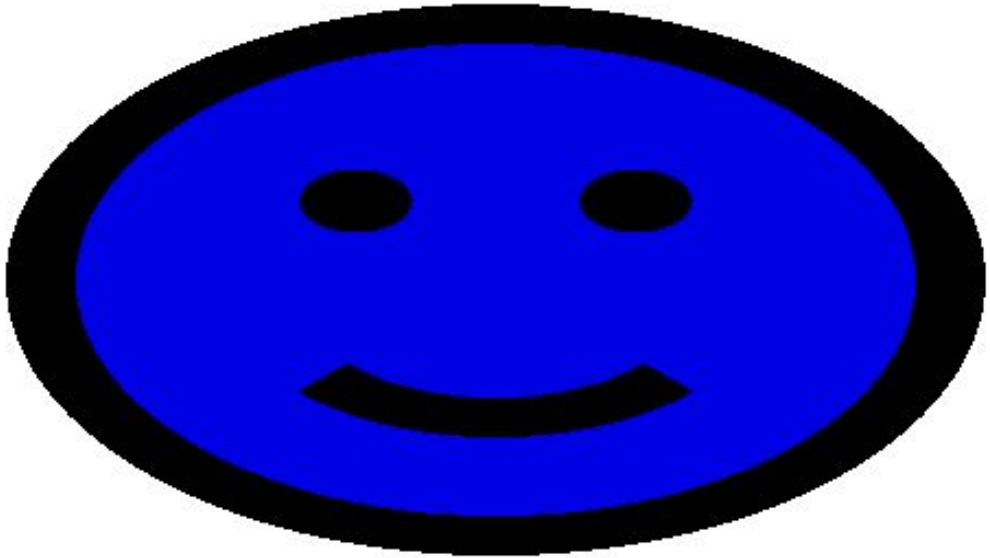

603.82
60.2 fps
640 x 360
REC


621.55
60.1 fps
640 x 360
REC


1242.71
60.1 fps
640 x 360
REC


1613.50
60.3 fps
640 x 360
REC


1458.14
60.1 fps
640 x 360
REC



891.63 60.2 fps 640 x 360

REC

BE CREATIVE!!!