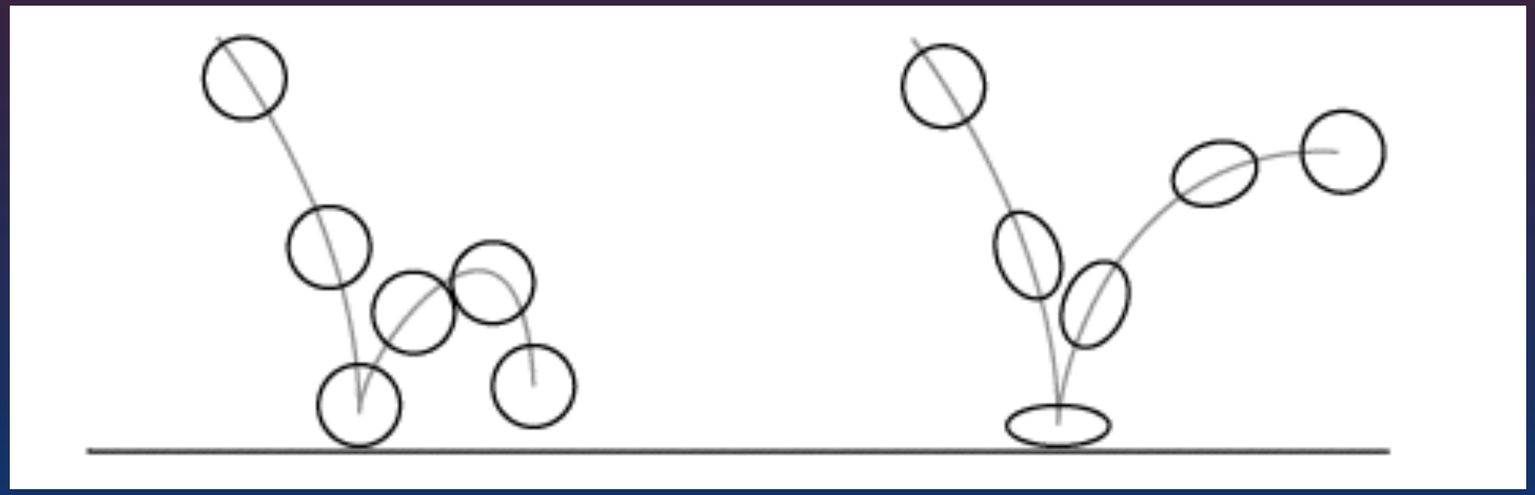
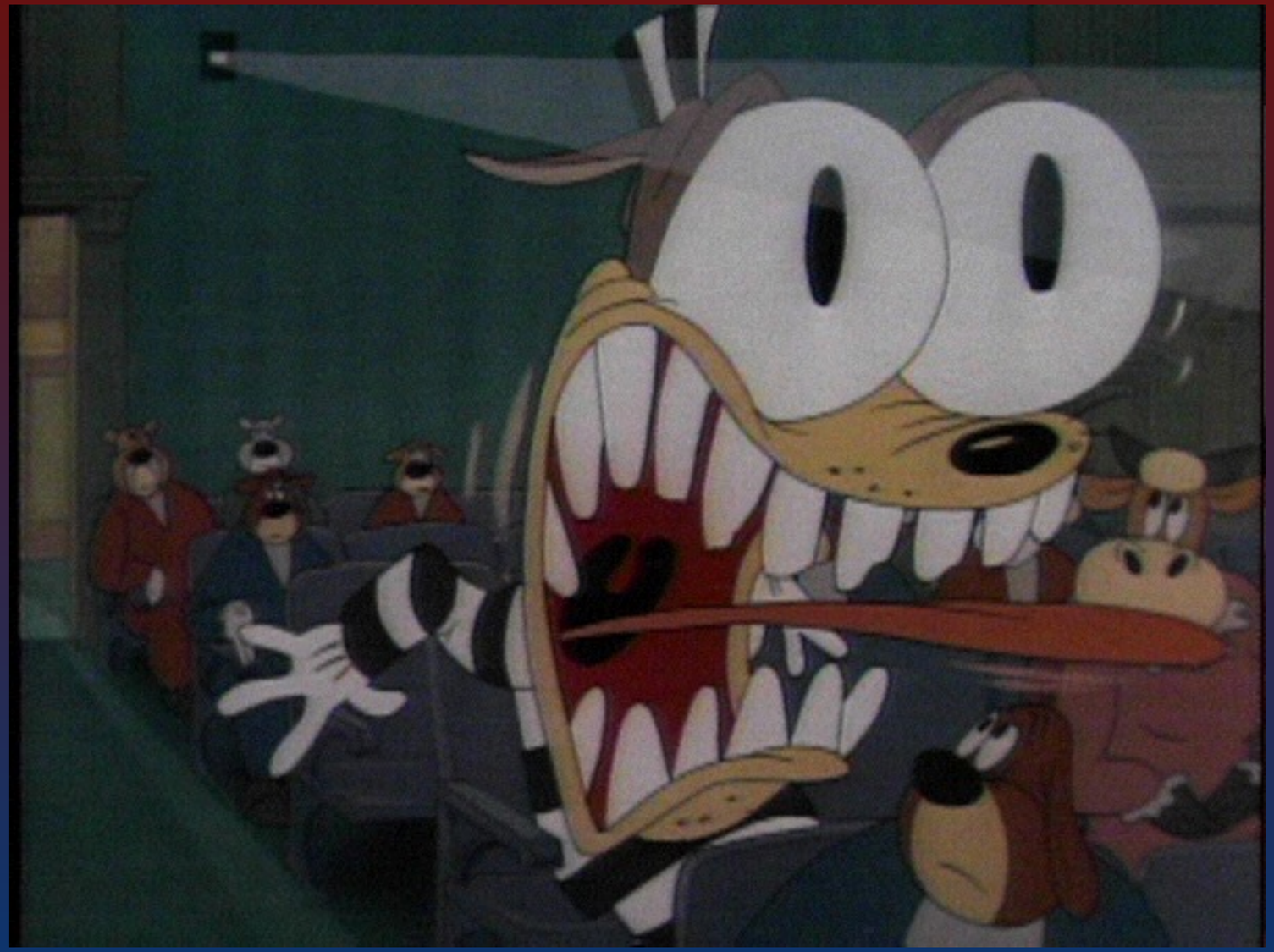
Effects and animation.

Part 1.




paintings-art-picture.com



Mapping value ranges.

```
float mapValue(float value, float srcMin, float srcMax, float dstMin, float dstMax) {
    float retVal = dstMin + ((value - srcMin)/(srcMax-srcMin) * (dstMax-dstMin));
    if(retVal < dstMin) {
        retVal = dstMin;
    }
    if(retVal > dstMax) {
        retVal = dstMax;
    }
    return retVal;
}
```

Tweening.

In-be-tweening.


4
6
8
18
20
1
23
24
24
TAIL
23


0.0
1.0

Linear interpolation.

```
float lerp(float from, float to, float time) {
    return (1.0-time)*from + time*to;
}
```

```
float lerp(float v0, float v1, float t) {
    return (1.0-t)*v0 + t*v1;
}

void update() {
    animationTime = animationTime + elapsed;
    float animationValue = mapValue(animationTime, animationStart,
animationEnd, 0.0, 1.0);
    DrawCircle(lerp(0.0, 1.0, animationValue), 0.0);
}
```

0.0

1.0

Easing in.

```
float easeIn(float from, float to, float time) {
    float tVal = time*time*time*time*time;
    return (1.0f-tVal)*from + tVal*to;
}
```

0.0

1.0

Easing out.

```
float easeOut(float from, float to, float time) {
    float oneMinusT = 1.0f-time;
    float tVal =  1.0f - (oneMinusT * oneMinusT * oneMinusT *
oneMinusT * oneMinusT);
    return (1.0f-tVal)*from + tVal*to;
}
```

0.0

1.0

Easing in and out.

```
float easeInOut(float from, float to, float time) {
    float tVal;
    if(time > 0.5) {
        float oneMinusT = 1.0f-((0.5f-time)*-2.0f);
        tVal =  1.0f - ((oneMinusT * oneMinusT * oneMinusT * oneMinusT *
oneMinusT) * 0.5f);
    } else {
        time *= 2.0;
        tVal = (time*time*time*time*time)/2.0;
    }
    return (1.0f-tVal)*from + tVal*to;
}
```

0.0

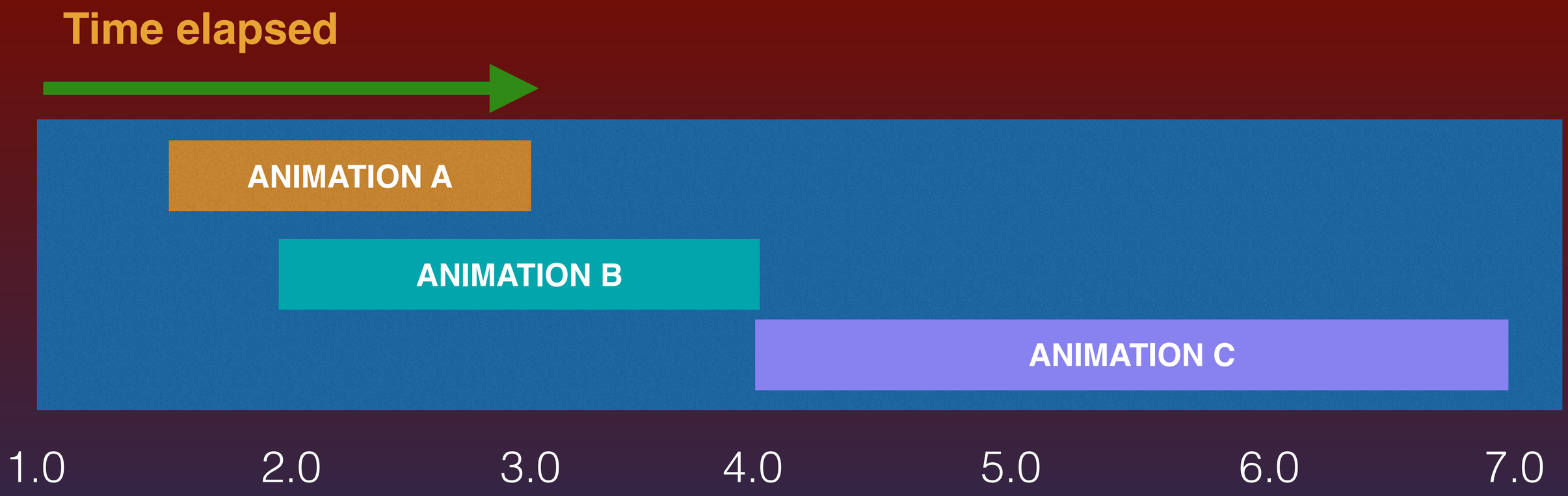
1.0

Overshooting our target.

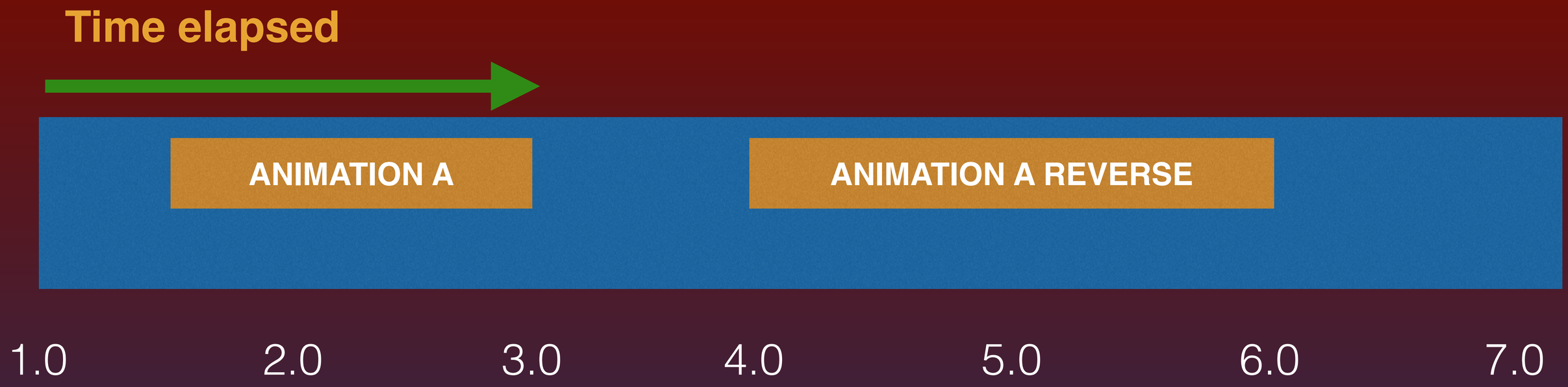
```
float easeOutElastic(float from, float to, float time) {
    float p = 0.3f;
    float s = p/4.0f;
    float diff = (to - from);
    return from + diff + (diff*pow(2.0f,-10.0f*time) * sin((time-s)*(2*PI)/p));
}
```

Mapping animations to time.


Time elapsed
ANIMATION A
ANIMATION B
ANIMATION C
1.0
2.0
3.0
4.0
5.0
6.0
7.0

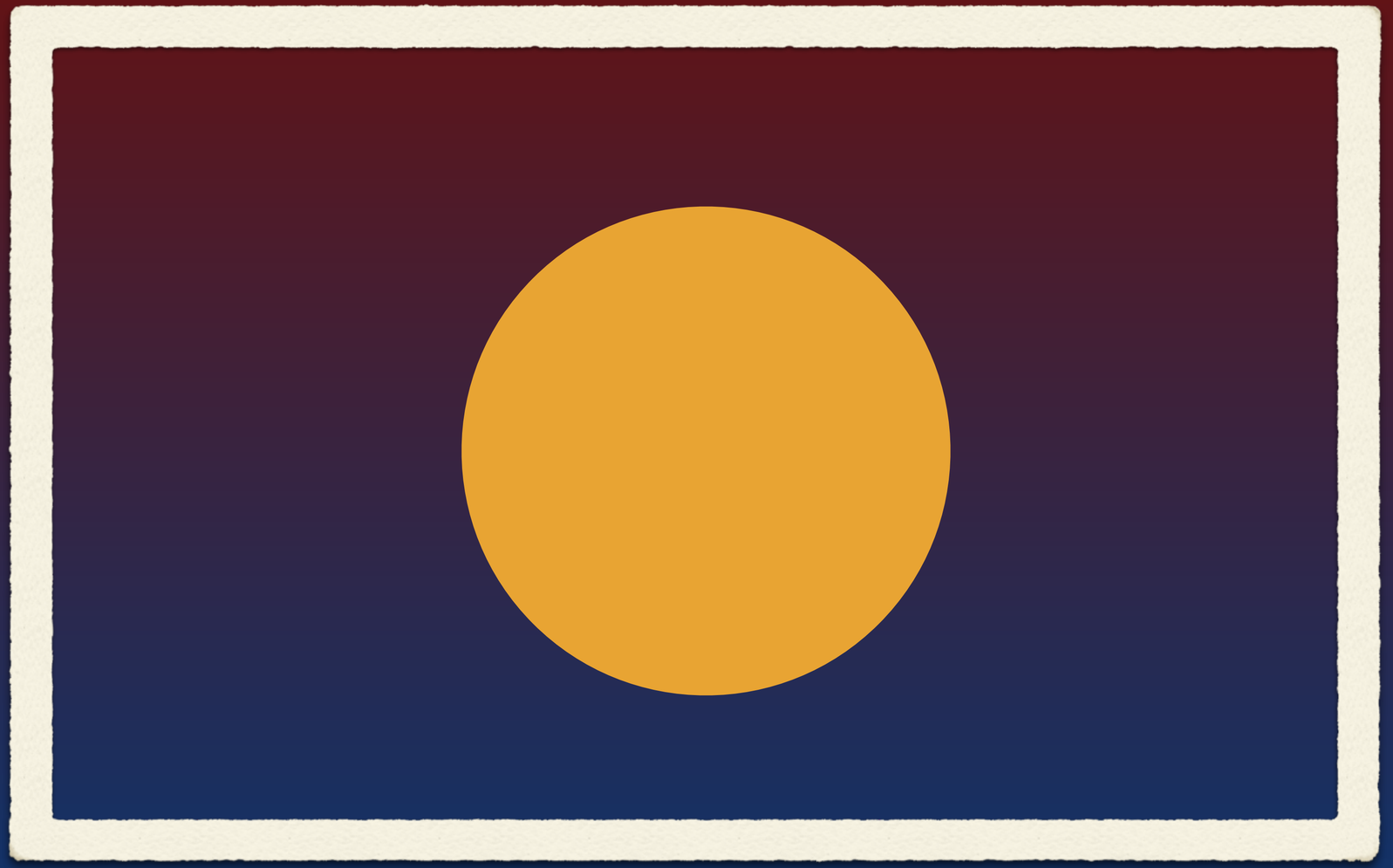


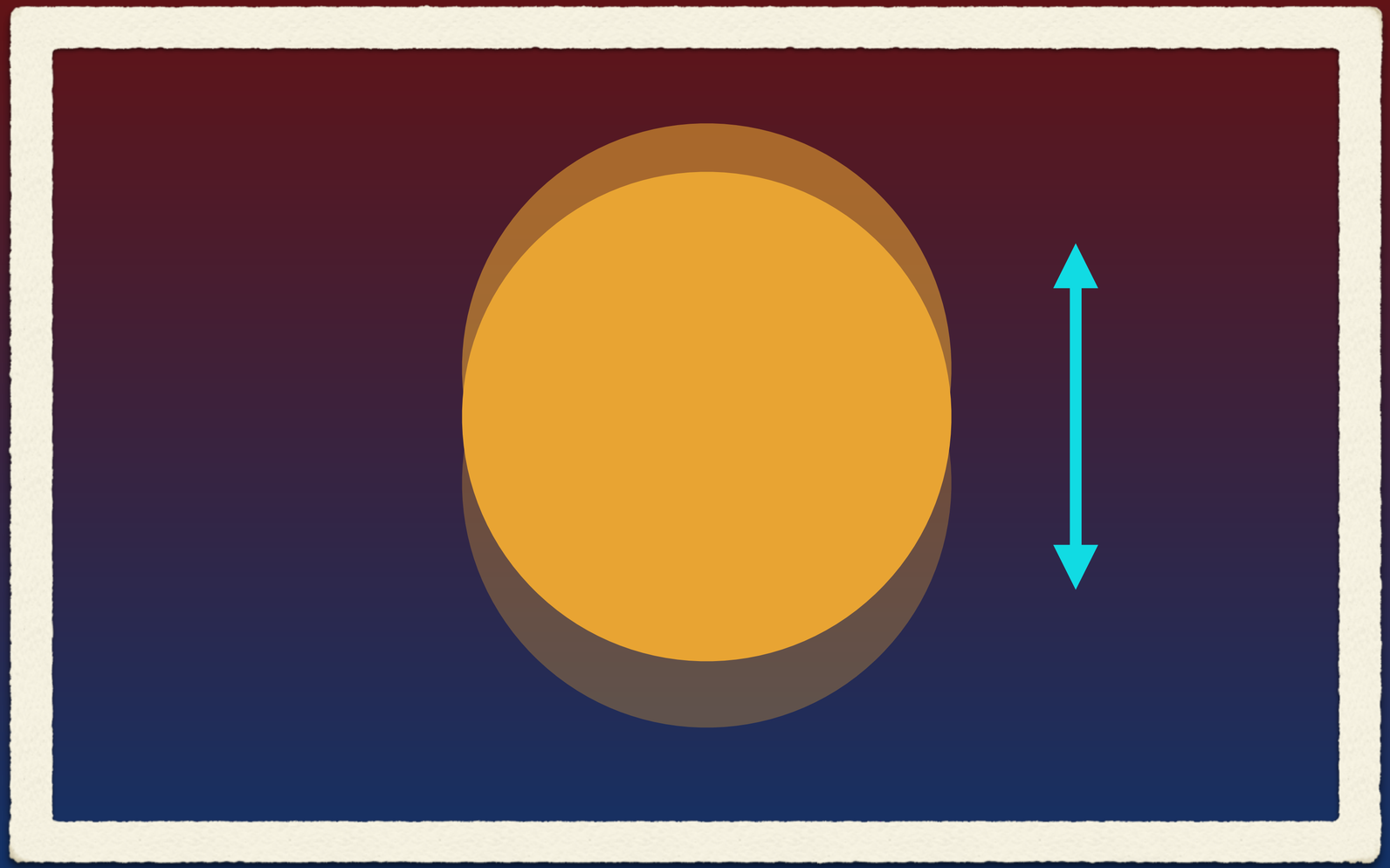

```
float animationAValue = mapValue(timeElapsed, 1.5f, 3.0f, 0.0f, 1.0f);
float animationBValue = mapValue(timeElapsed, 2.0f, 4.0f, 0.0f, 1.0f);
float animationCValue = mapValue(timeElapsed, 4.0f, 7.0f, 0.0f, 1.0f);
```

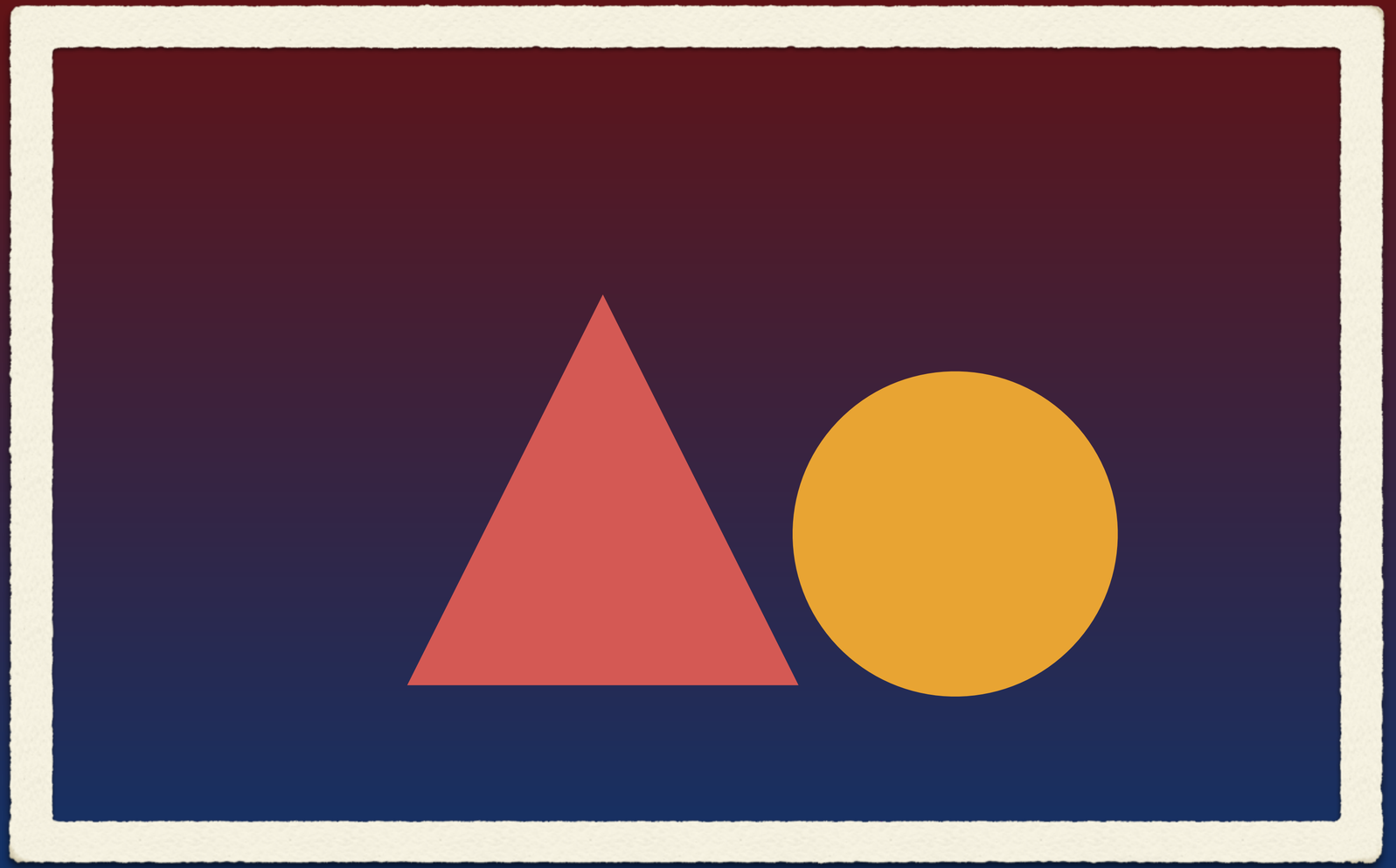


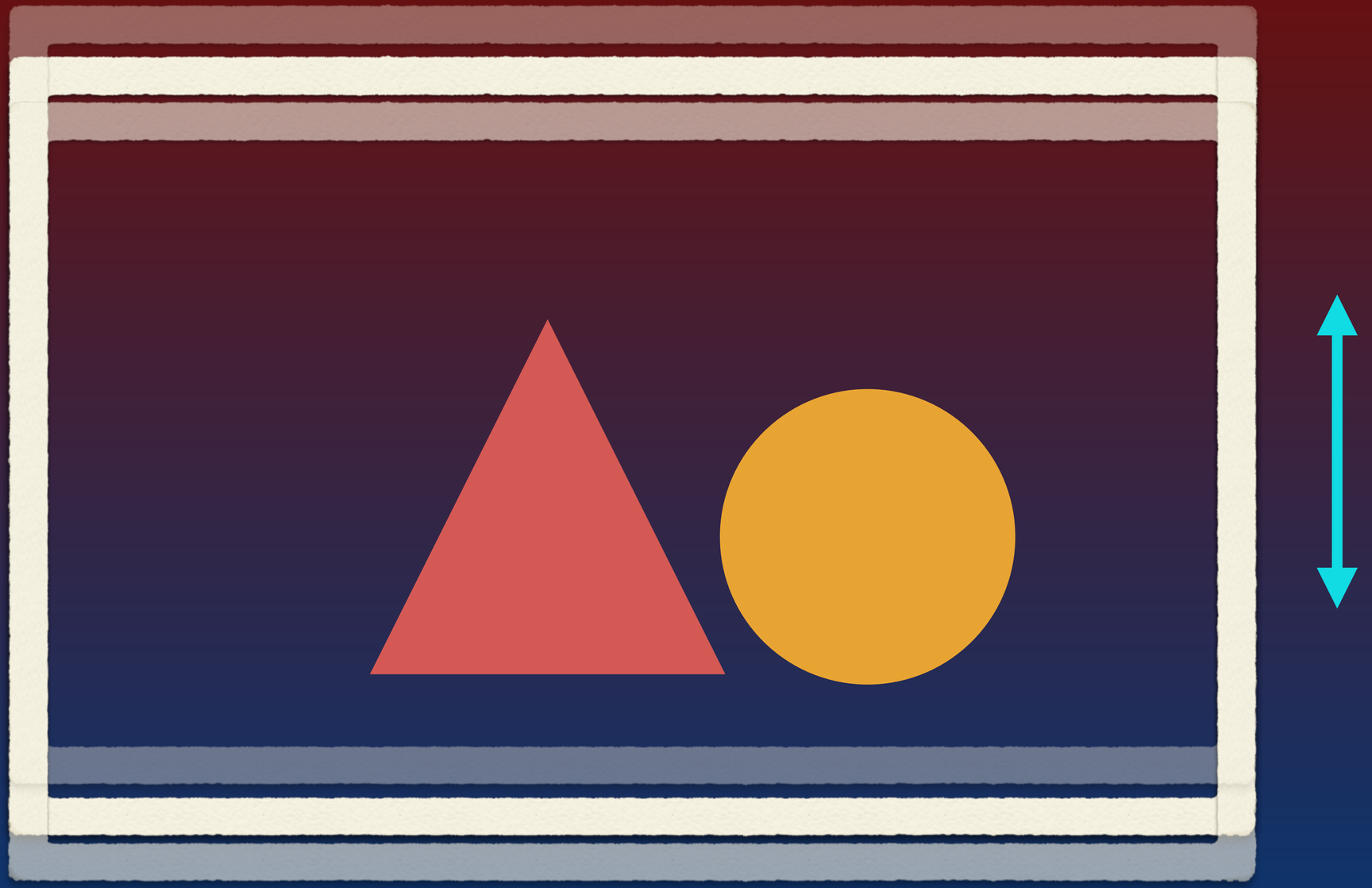

```
float animationAValue;
if(timeElapsed > 4.0) {
    animationValue = mapValue(stageAnimationTime, 6.0, 4.0, 0.0, 1.0);
} else {
    animationValue = mapValue(stageAnimationTime, 1.5, 3.0, 0.0, 1.0);
}
```

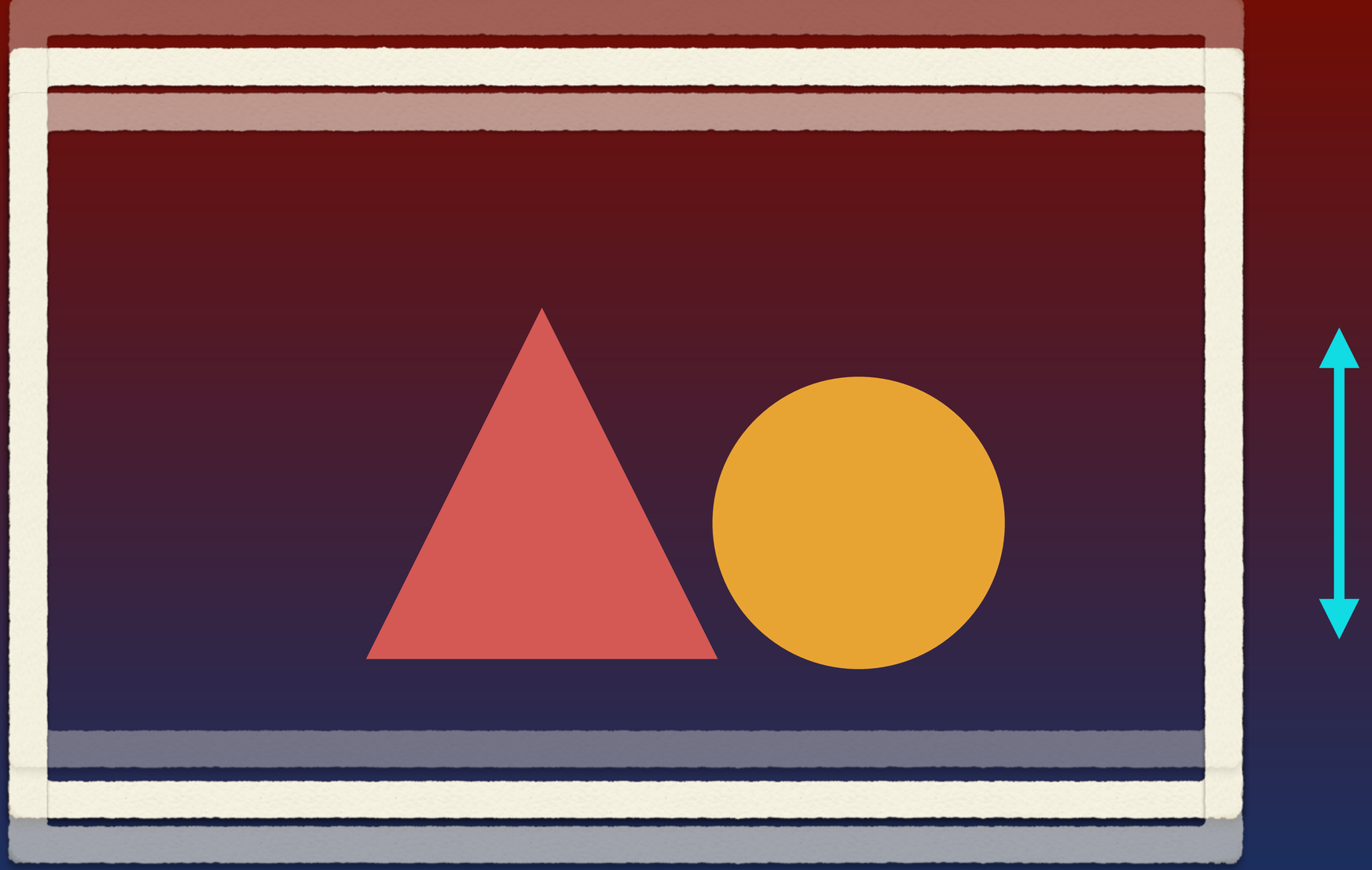
Screen Shake!











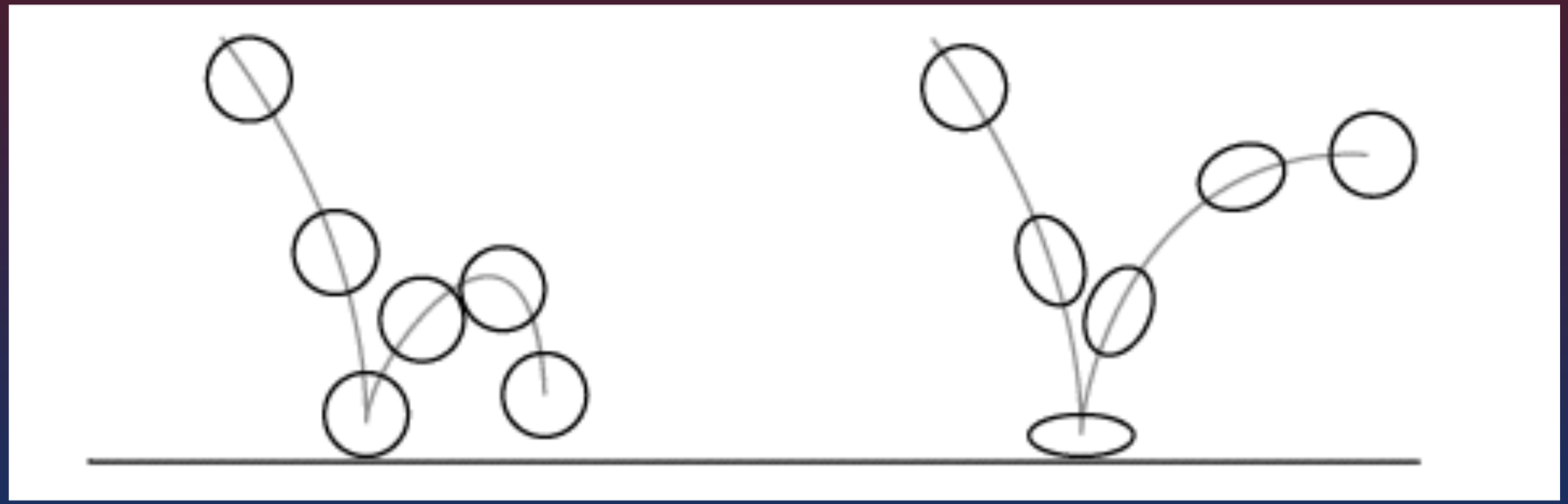
```
screenShakeValue += elapsed;

glTranslatef(0.0f, sin(screenShakeValue * screenShakeSpeed)* screenShakeIntensity, 0.0f);
```

You can shake it sideways or both ways!

THERE IS NO WRONG WAY TO SHAKE THE SCREEN.

Squash and stretch.



Squashing.

Fast movement on the X axis or impact on the Y axis..

Stretching.

Fast movement on the Y axis or impact on the X axis..

Thomas was Alone.

https://www.youtube.com/watch?v=22WW4_BxpR8#t=467

Map velocity on an axis to scale on that axis and inverse of that scale on the other axis.

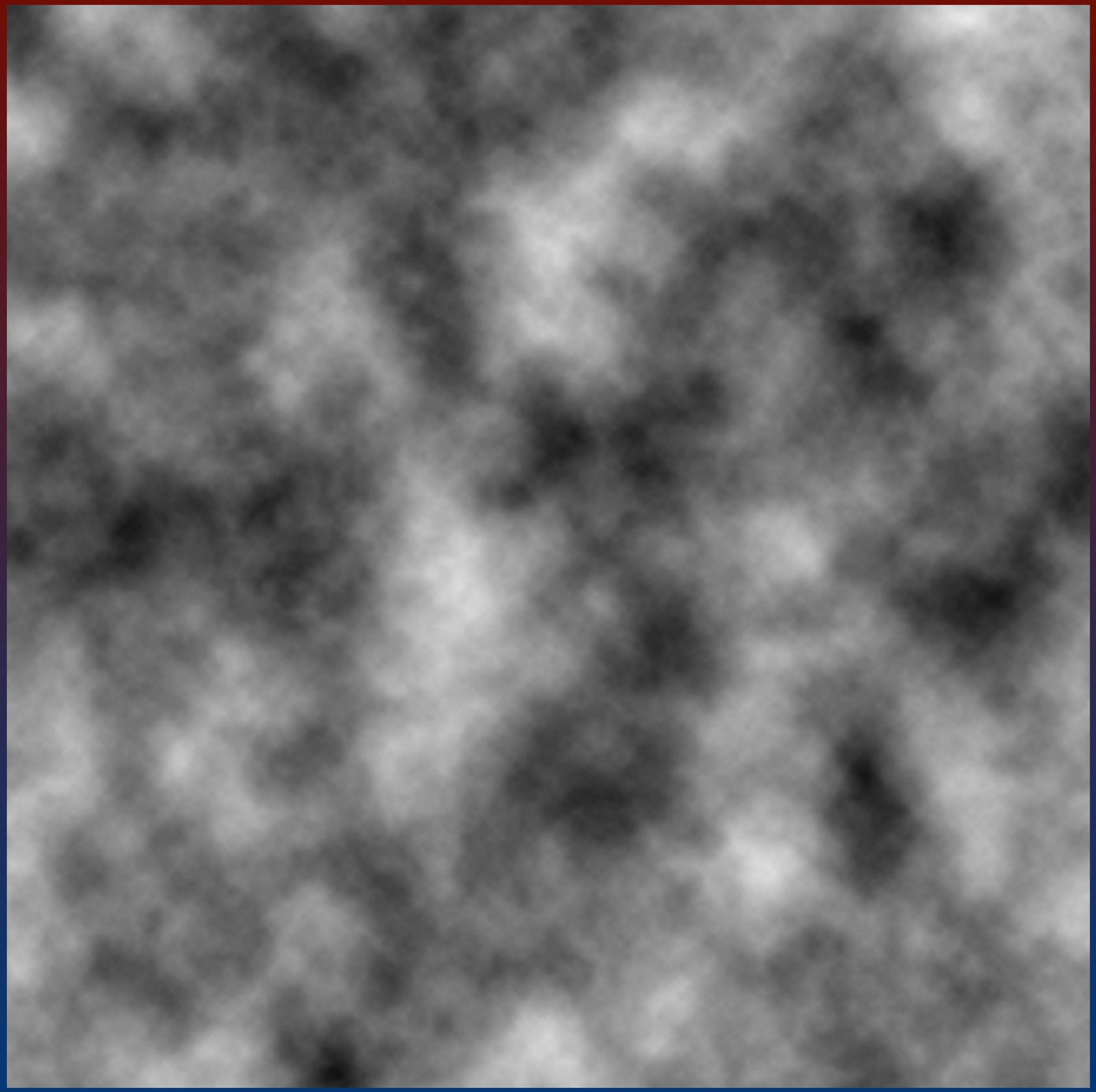
Map velocity on an axis to scale on that axis and inverse of that scale on the other axis.

```
// map Y velocity 0.0 - 5.0 to 1.0 - 1.6 Y scale and 1.0 - 0.8 X scale

scale_y = mapValue(fabs(velocity_y), 0.0, 5.0, 1.0, 1.6);
scale_x = mapValue(fabs(velocity_y), 5.0, 0.0, 0.8, 1.0);
```

Perlin noise.

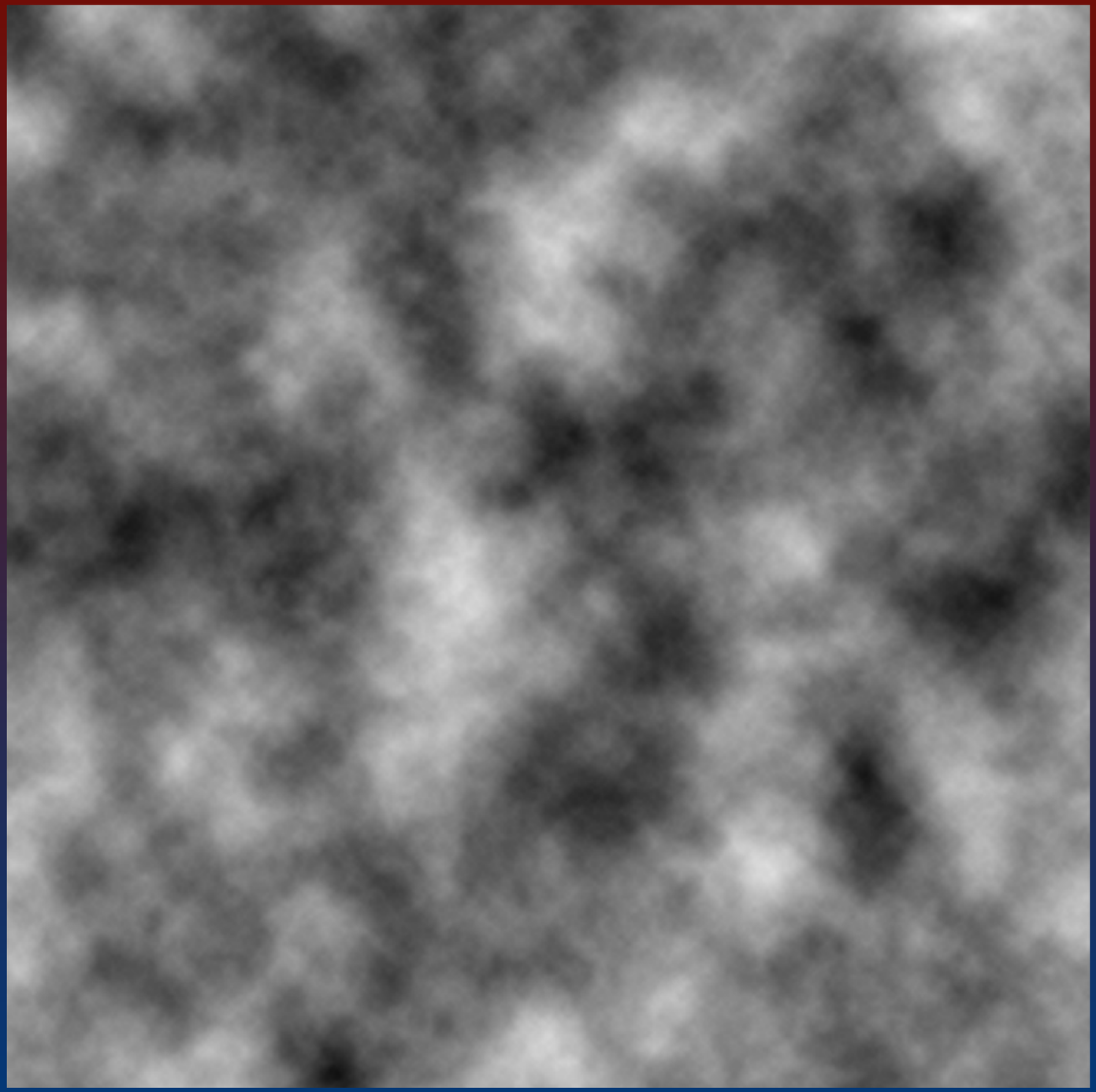
http://mrl.nyu.edu/~perlin/doc/oscar.html



Use PerlinNoise.h and PerlinNoise.cpp in class repository.

noise2 returns a -1.0 to 1.0 noise value for a 2D coordinate.

```
float coord[2] = {0.05f, 0.0};
float val = noise2(coord);
```

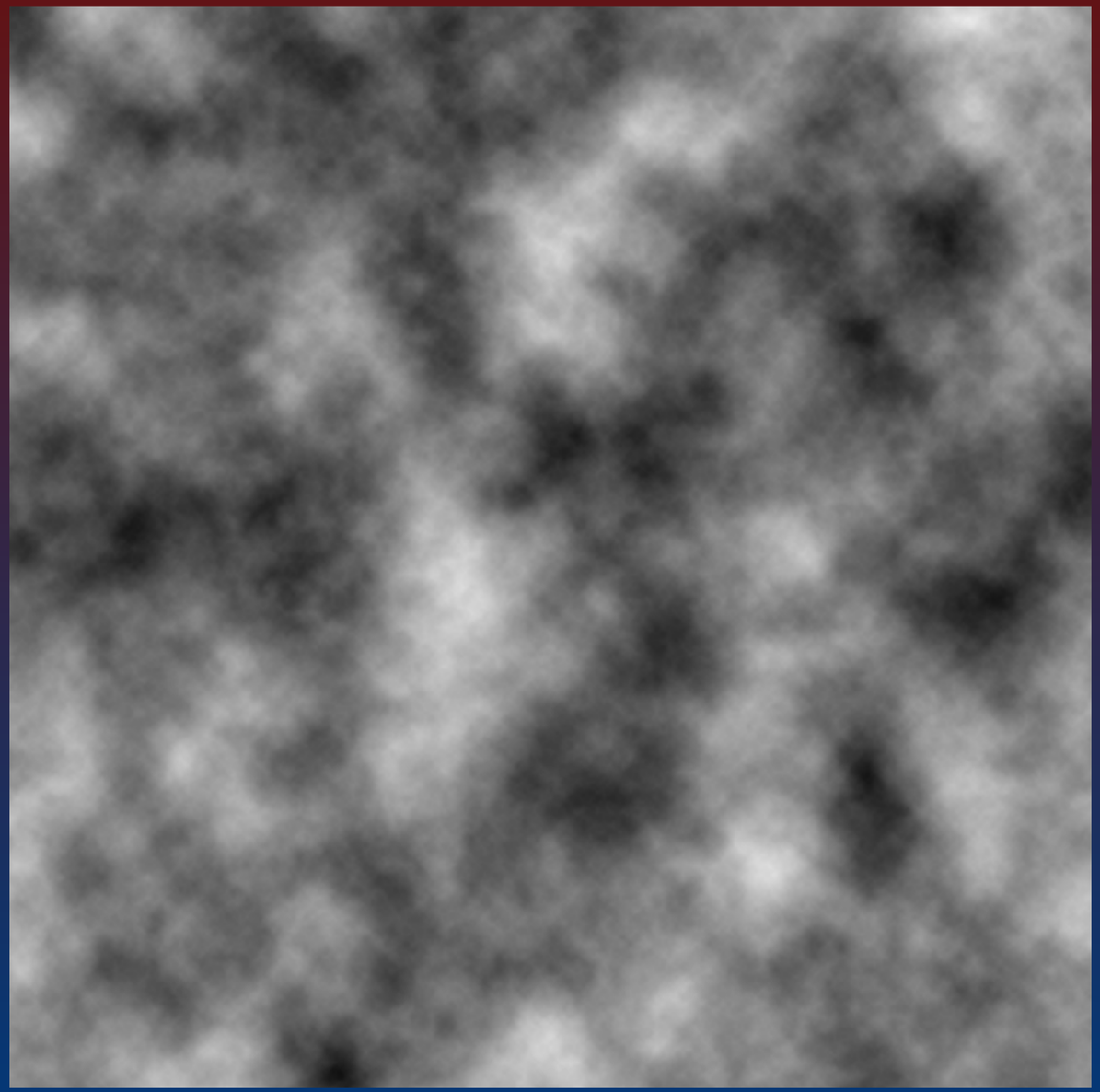


noise1 returns a -1.0 to 1.0 noise value for a 1D coordinate.

```
float val = noise1(0.5f);
```

Using peril noise for natural movement.

Time elapsed



Shaky cam example using 1D noise.

```
perlinValue += elapsed;

glTranslatef(noise1(perlinValue), noise1(perlinValue+ 10.0f), 0.0);
```

Shaky cam example using 2D noise.

```
perlinValue += elapsed;

float coord[2] = {perlinValue, 0.0};
float val = noise2(coord);

coord[1] = 0.5f;
float val2 = noise2(coord);
glTranslatef(val, val2, 0.0);
```

You can use Perlin noise for good looking screen shake too!

Or to make things hover realistically.

Fading in and out.

To fade the screen, you can draw a black rectangle on top of your scene using vertex colors and animate the vertex color alpha.

Don’t forget to enable blending!