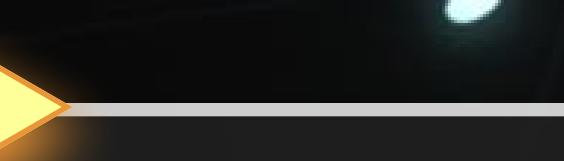


Made With



Made All By Myself



Meta Clash is a roguelike action game with unique mechanics based on physics. Players control a ball to attack their enemies.

<https://flandrescarlet-15532.itch.io/meta-clash>

Demo Link

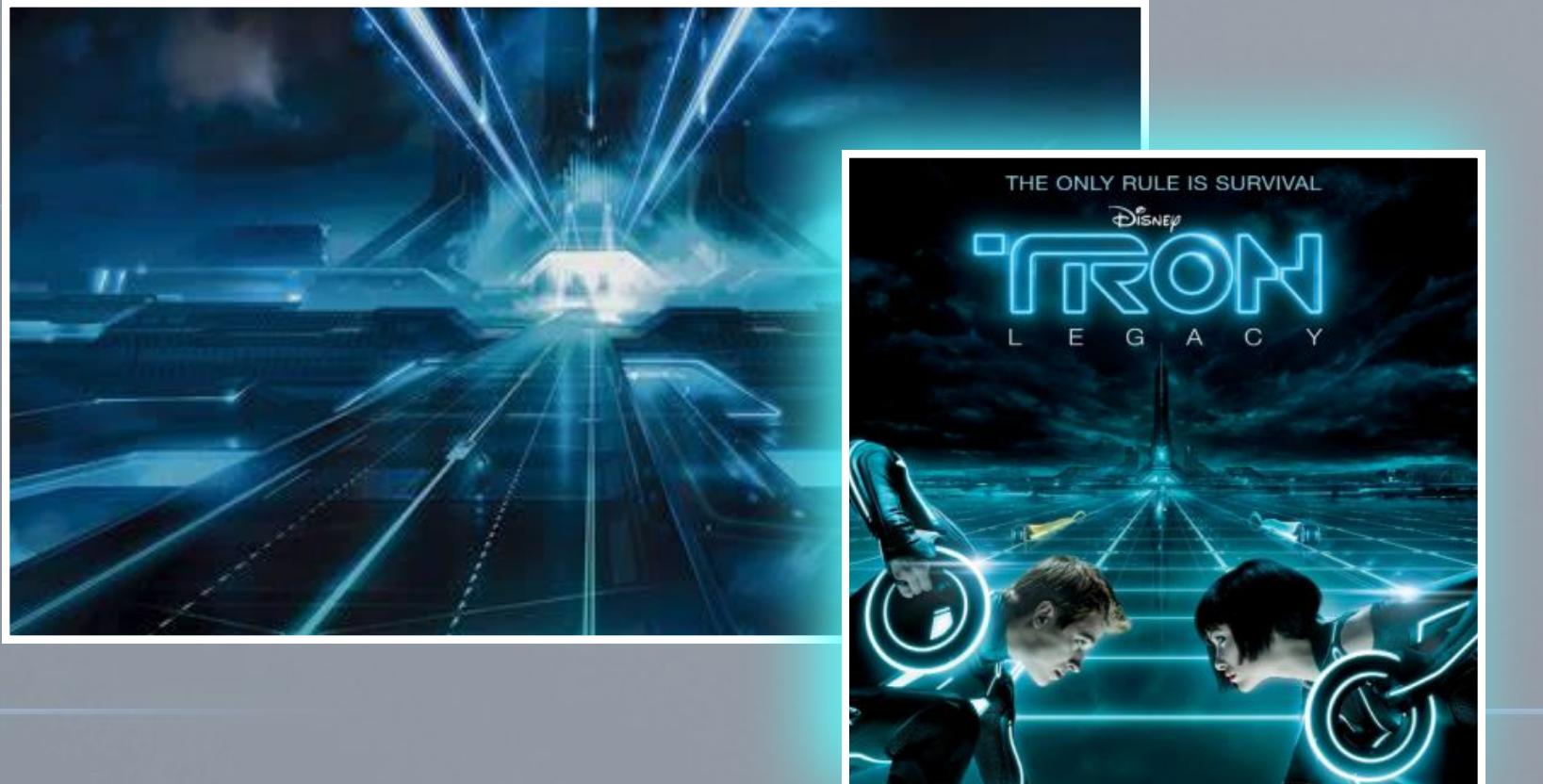
META CLASH

■ Reference



The gameplay was inspired by the Windows game “3D Pinball”. Since there was a lack of games with new mechanics on the market, I decided to combine the core gameplay of “3D Pinball” with action game elements.

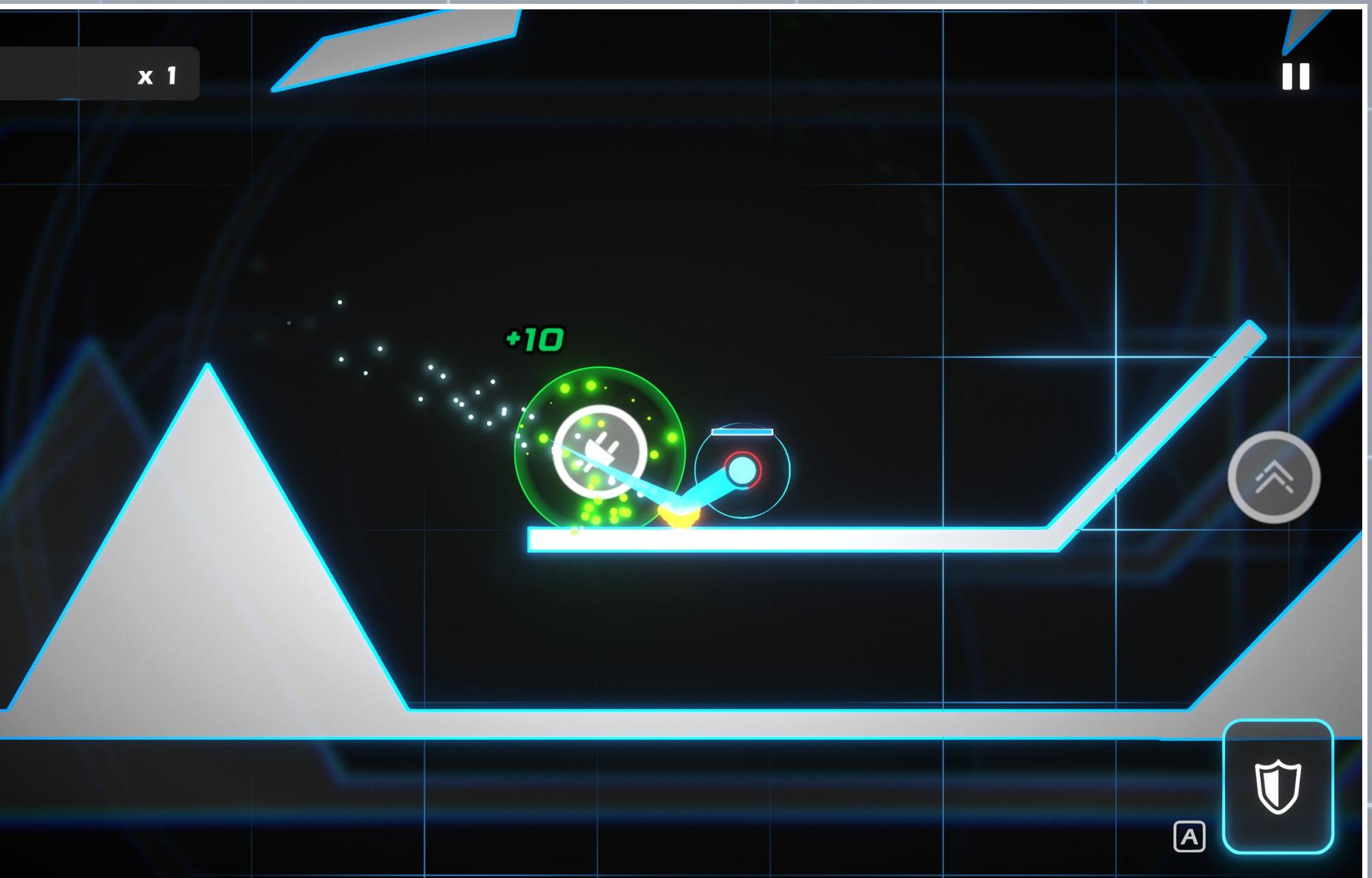
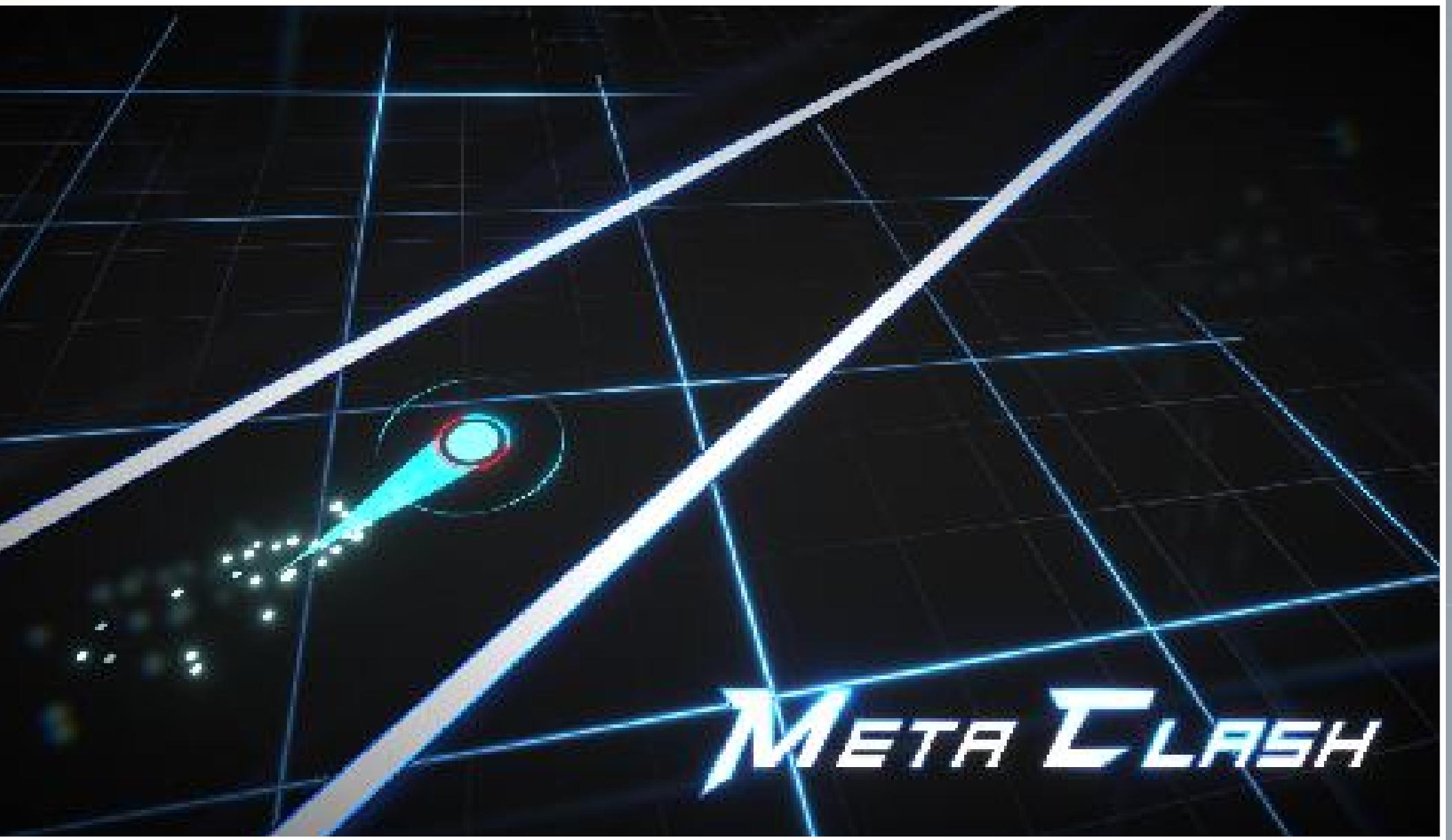
The game's art style is inspired by the movie “Tron: Legacy”, and is set in a futuristic metaverse.



■ Iteration



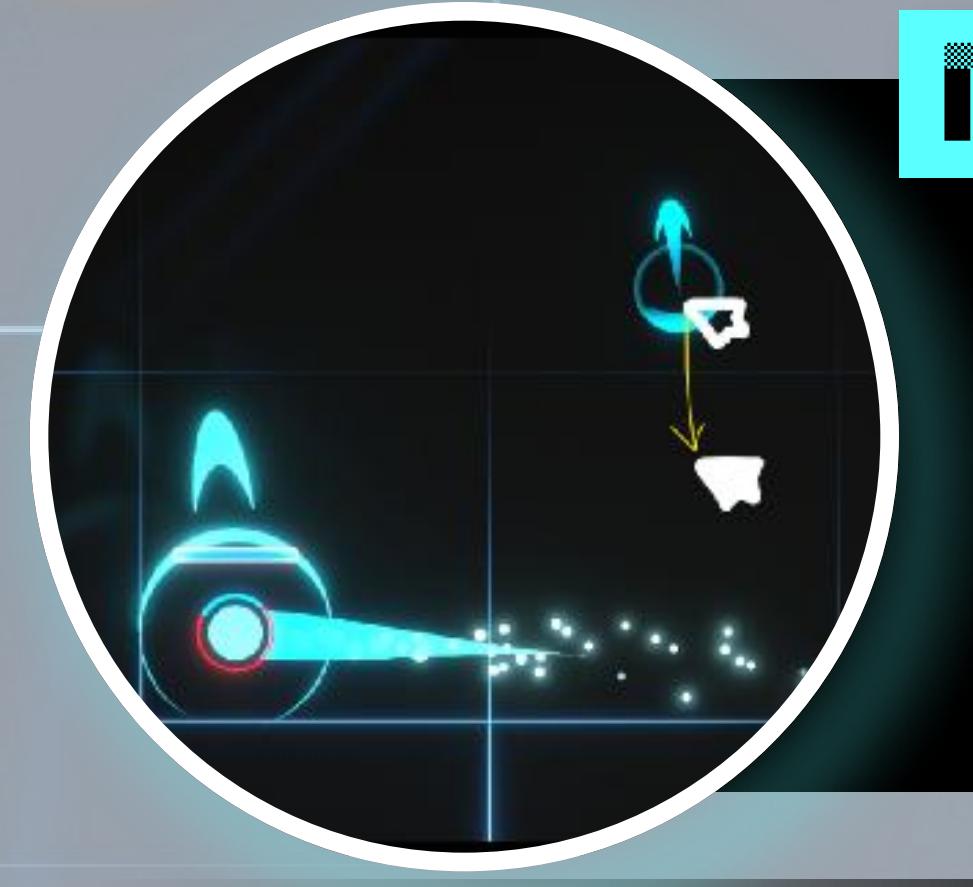
■ Screenshots



Gameplay

MOVE

Drag your mouse/finger at any position (except for places occupied by buttons) to control the direction of your ball.



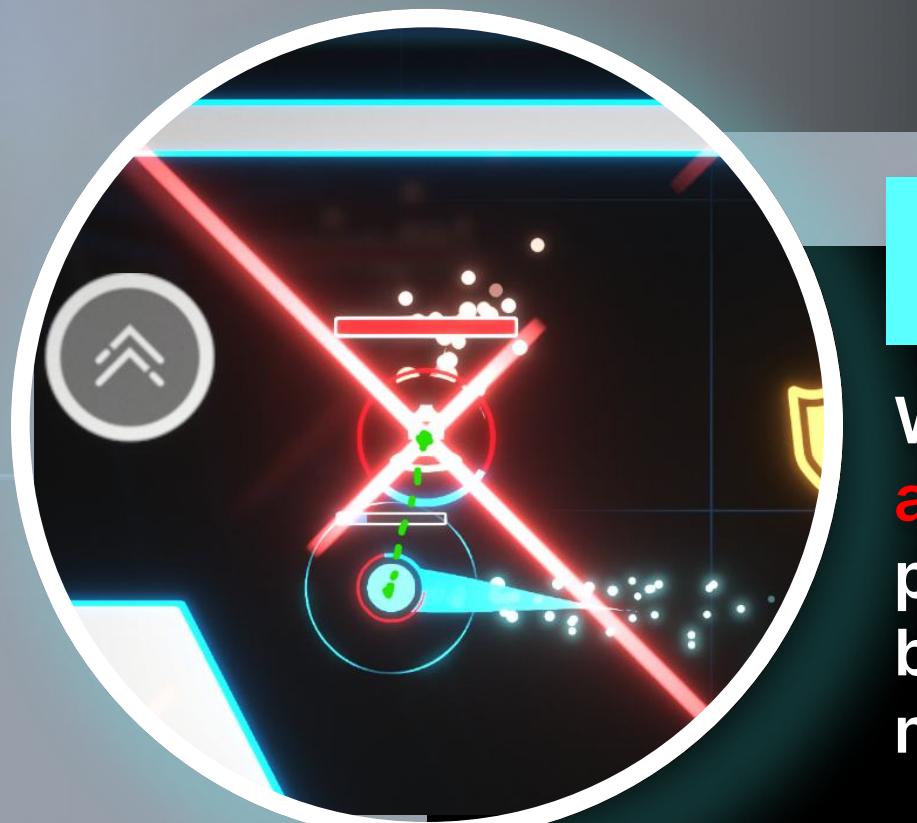
ATTACK

When the **blue** part of your ring **collides** with the **red** part of a enemy, you **deals damage** to it based on your Damage Multiplier and relative speed.



PARRY

When a enemy is **about to hit the player**, an **alert** will appear. The player can use the parry skill by pressing corresponding buttons. Parry deals more damage than normal attack.

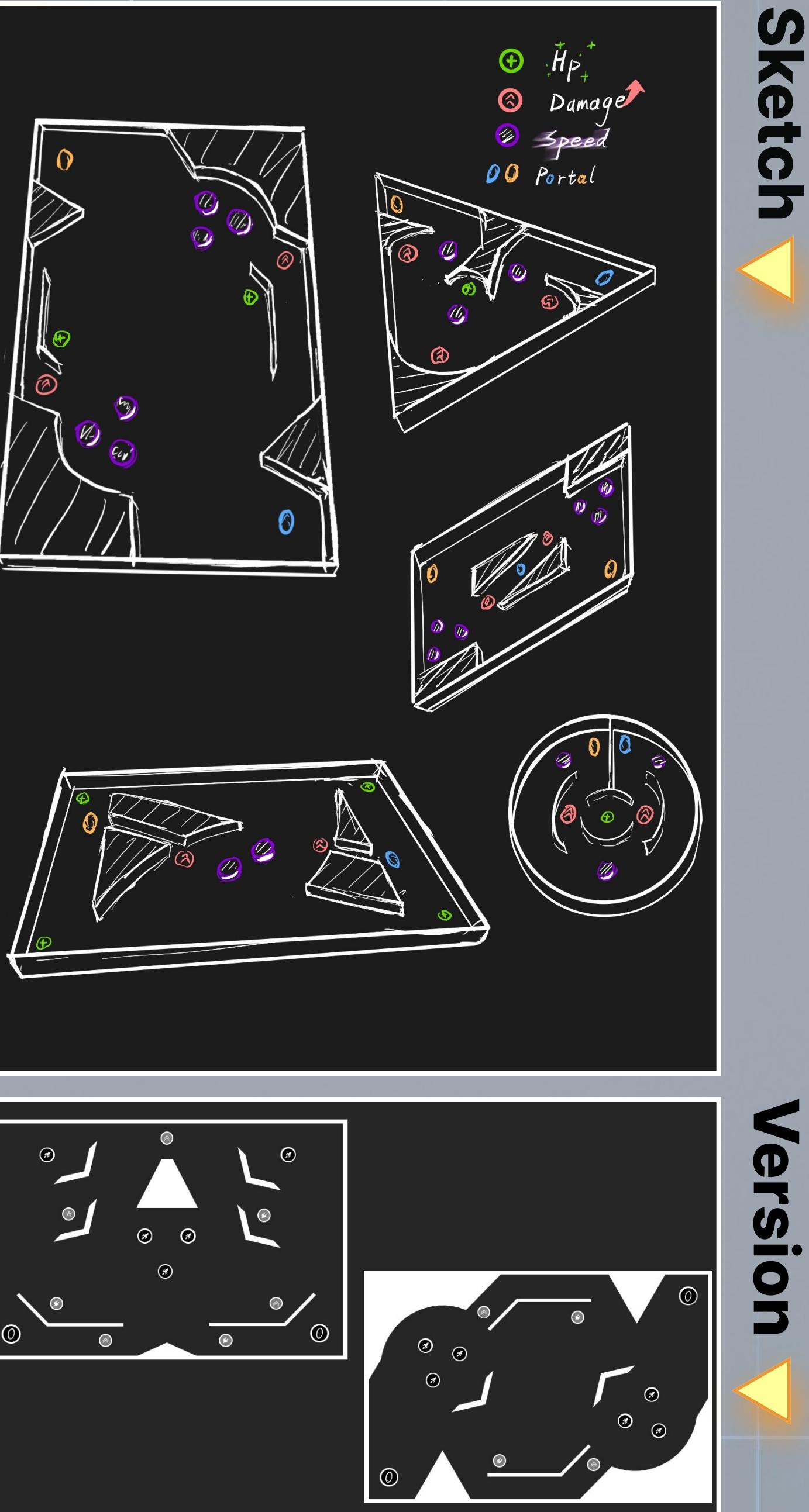


x 1.0

DAMAGE MULTIPLIER

Shown in the top-left corner. Directly affects the damage players cause.

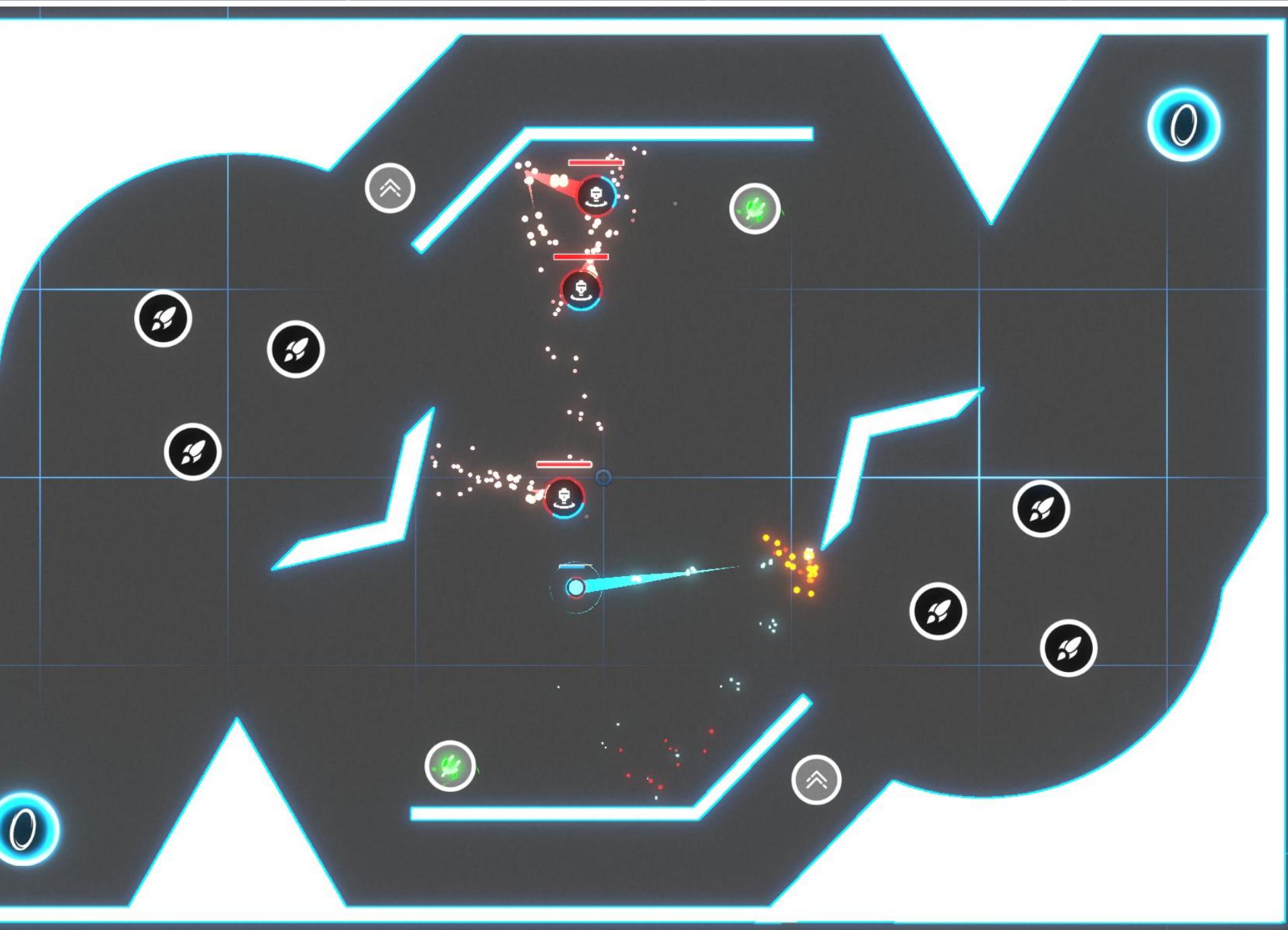
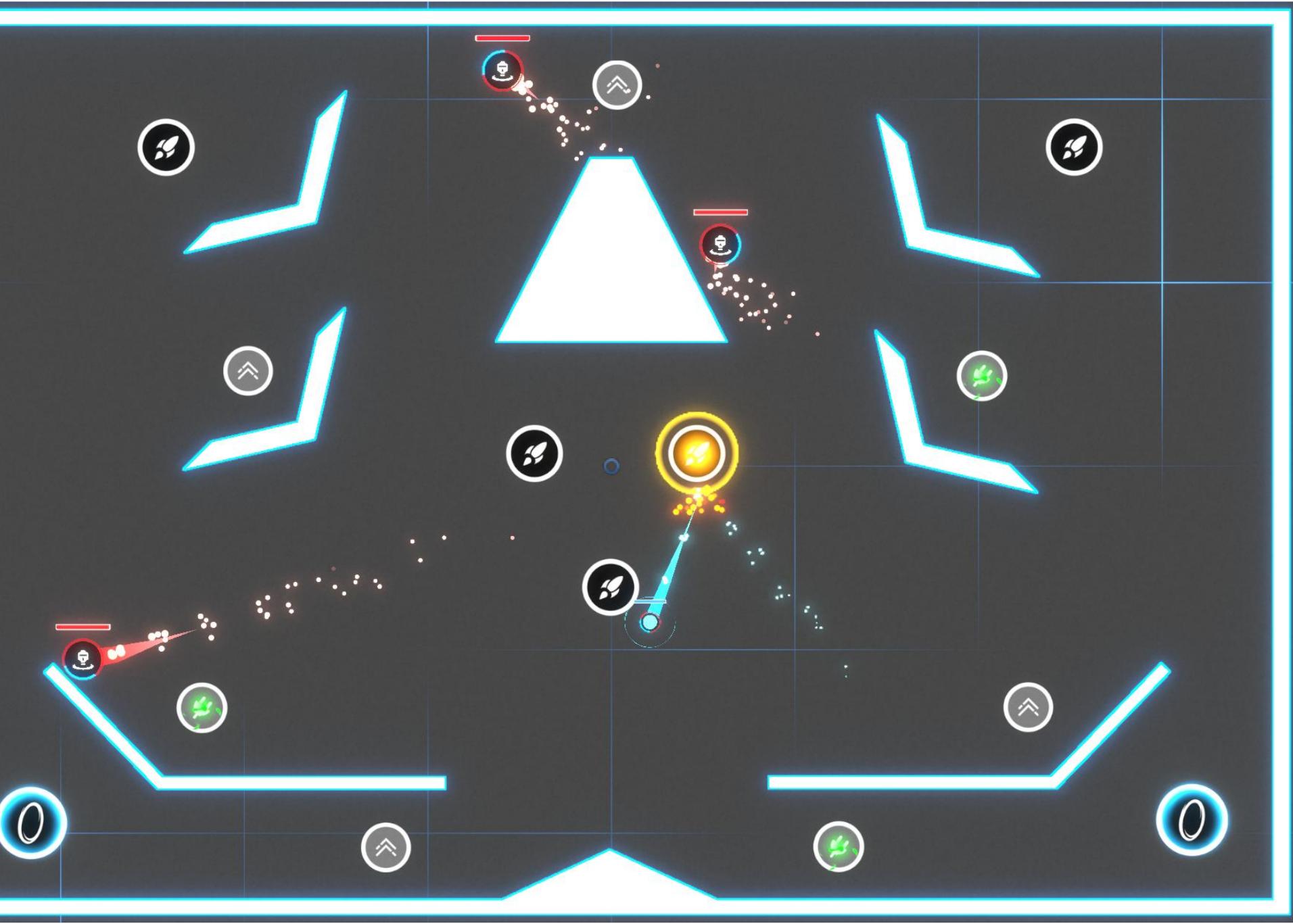
Map Design



Sketch

In-Game Overview

Final Version



Code

-Settings System

```
[CustomEditor(typeof(CustomSlider))]
public class SliderInspector : Editor{
    public override void OnInspectorGUI(){
        CustomSlider slider = (CustomSlider)target;
        slider.UpdateUI();

        if (slider.changeSettings){
            EditorGUILayout.BeginHorizontal();
            EditorGUILayout.LabelField("Corresponding Variable", GUILayout.Width(150));
            var allFields FieldInfo[] = typeof(SettingsData).GetFields();
            List<string> fields = new();
            foreach (var f FieldInfo in allFields){
                if (f.FieldType == typeof(float) || f.FieldType == typeof(int) || f.FieldType == typeof(double)){
                    fields.Add(f.Name);
                }
            }

            var names = new string[fields.Count];
            int i = 0;
            foreach (var field string in fields){
                names[i] = Tools.ToNormalEnglish(field);
                i++;
            }

            int selected = fields.IndexOf(slider.variableName);
            if (selected == -1){
                selected = 0;
            }

            selected = EditorGUILayout.Popup(selected, names);
            slider.variableName = fields[selected];
        }
        EditorGUILayout.EndHorizontal();
    }
}

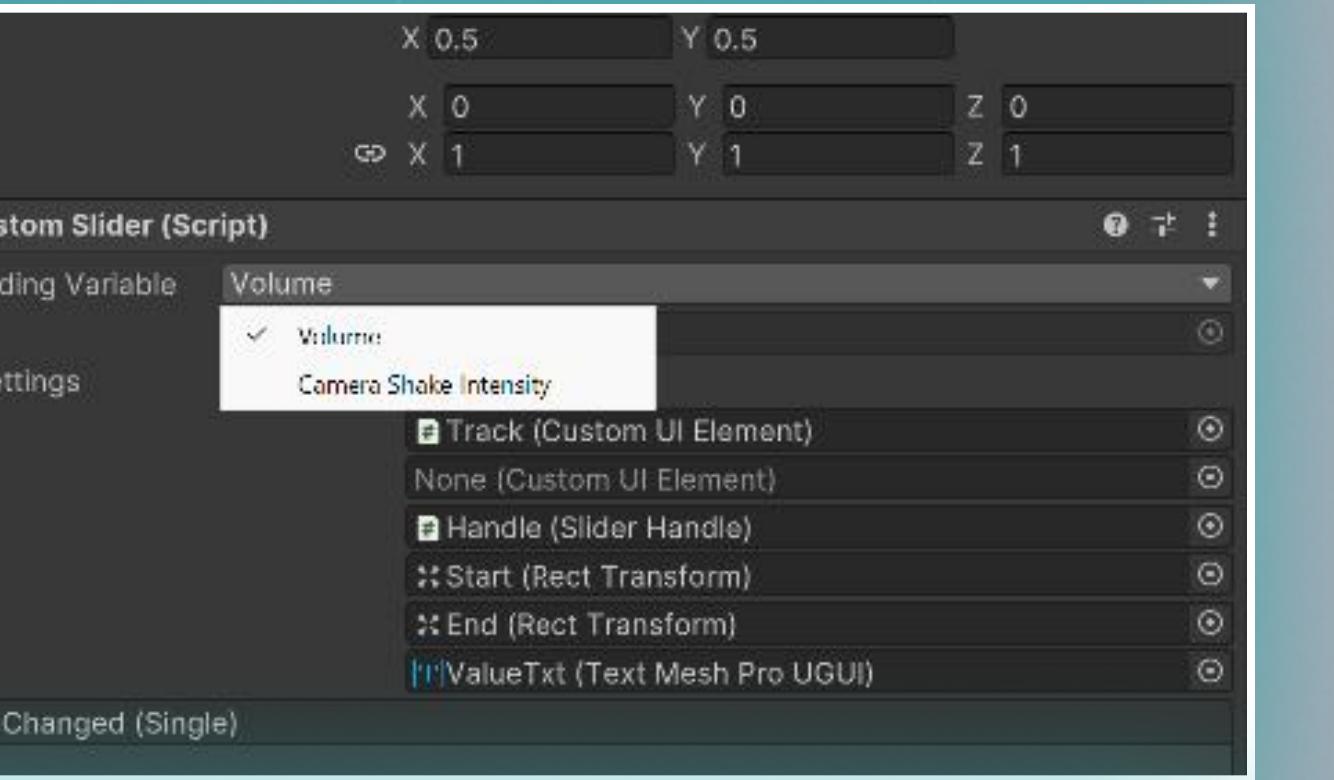
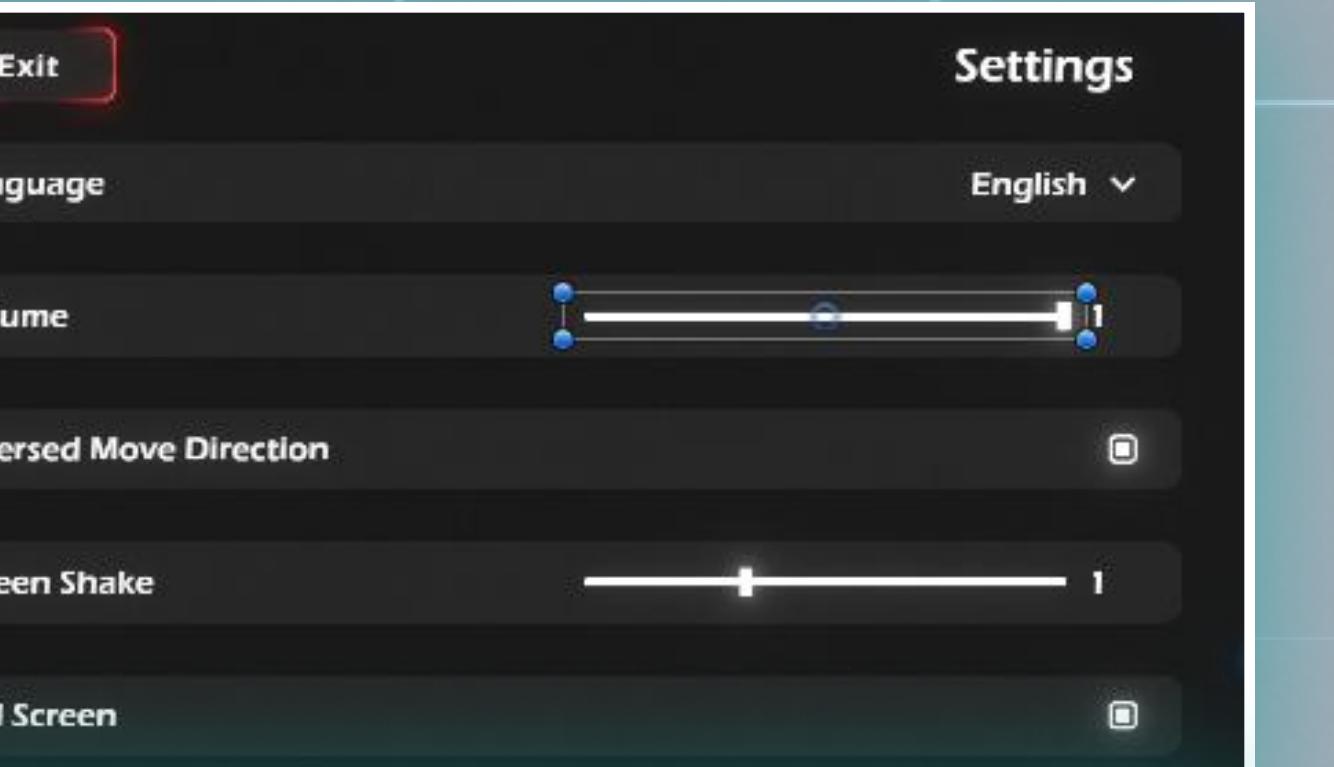
public float value{
    get{ return _value; }
    set{
        float tmp = Mathf.Clamp(value, float)Math.Round(value, roundAccuracy), minValue, maxValue);
        if (tmp != _value){
            SoundSys.PlaySound(name: "SFX/Slider");
            _value = tmp;
            if (onValueChanged != null){
                onValueChanged.Invoke(_value);
            }
        }

        if (variable != null){
            variable.SetValue(obj: Settings.data, _value);
            Settings.SaveSettings();
        }
        UpdateUI();
    }
}
```

```
public class SettingsData : ICloneable{
    public bool inverseMouseDirection;
    public bool fullScreen;
    public float volume;
    public float cameraShakeIntensity;
    public Language language;

    public void Init(){
        inverseMouseDirection = false;
        cameraShakeIntensity = 1.5f;
        language = Application.systemLanguage == SystemLanguage.Chinese ? Language.Chinese : Language.English;
        fullScreen = false;
        volume = 1.0f;
    }

    public object Clone(){
        return base.MemberwiseClone();
    }
}
```



```
public class SettingsData : ICloneable{
    public bool inverseMouseDirection;
    public bool fullScreen;
    public float volume;
    public float cameraShakeIntensity;
    public Language language;

    public void Init(){
        inverseMouseDirection = false;
        cameraShakeIntensity = 1.5f;
        language = Application.systemLanguage == SystemLanguage.Chinese ? Language.Chinese : Language.English;
        fullScreen = false;
        volume = 1.0f;
    }

    public object Clone(){
        return base.MemberwiseClone();
    }
}
```

Using **Reflection** to get all the variables in the settings that match the component, and add a custom **Popup** to set the corresponding settings item of an UI with just two clicks.

```
public class BuffSys{
    public Dictionary<string, List<Buff>> stringPairs;
    public Dictionary<BuffTriggerEvent, UnityEvent<object[]>> events;
    public Unit unit;

    #region 1用法
    public BuffSys(Unit u){
        unit = u;
        stringPairs = new Dictionary<string, List<Buff>>();
        events = new Dictionary<BuffTriggerEvent, UnityEvent<object[]>>();
    }
    #endregion

    #region 2用法
    public void AddBuff(Buff b){
        b.unit = unit;
        if (stringPairs.ContainsKey(b.name)){
            if (b.cumulateMode == BuffCumulateMode.AddLayer){
                if (b.layerLimit == 0 || b.layerLimit > stringPairs[b.name].Count){
                    stringPairs[b.name].Add(b);
                    //allBuffs[b.triggerEvent].Add(b);
                }
            }
            else if (b.cumulateMode == BuffCumulateMode.Refresh){
                if (b.lastingType == BuffLastingType.Lasting){
                    if (b.duration > stringPairs[b.name][0].durationLeft){
                        stringPairs[b.name][0].durationLeft = b.duration;
                    }
                }
                else if (b.lastingType == BuffLastingType.Triggered){
                    if (b.maxTriggerCount > stringPairs[b.name][0].maxTriggerCount){
                        stringPairs[b.name][0].maxTriggerCount = b.maxTriggerCount;
                    }
                }
                else{
                    stringPairs.Add(b.name, new List<Buff>(){ b });
                }
            }
            else{
                stringPairs.Add(b.name, new List<Buff>(){ b });
            }
        }
        if (b.triggerEvent == BuffTriggerEvent.Immediate){
            b.Trigger();
        }
    }
}
```

```
if (name == "Speed Up"){
    int value = 2;
    if (rank == 'A'){
        value = 3;
    }
    else if (rank == 'S'){
        value = 5;
    }

    return new Buff(out b, name, displayName: LocalizeTxt(params: "Speed Up", "加速"), BuffCumulateMode.AddLayer, BuffLastingType.Permanent, BuffTriggerEvent.MatchStarted, actionTriggered: () => {
        var u UnitEntity = b.unit.entity;
        u.rb.linearVelocity += u.direction * value;
    }, describe: LocalizeTxt(params: $"Increase the initial speed by {value} m/s.", $"提高 {value} m/s 初始速度"));
}

if (name == "Extra Damage"){
    int value = 25;
    if (rank == 'A'){
        value = 50;
    }
    else if (rank == 'S'){
        value = 100;
    }

    return new Buff(out b, name, displayName: LocalizeTxt(params: "Extra Damage", "伤害加深"), BuffCumulateMode.AddLayer, BuffLastingType.Permanent, BuffTriggerEvent.Immediate, actionTriggered: () => {
        b.unit.percentBuff[UnitAttribute.DamageMult] += value / 100f;
    }, actionDeleted: () => {
        b.unit.percentBuff[UnitAttribute.DamageMult] -= value / 100f;
    }, describe: LocalizeTxt(params: $"Increase {value}% damage multiplier (Multiplied).", $"提高 {value}% 伤害倍率 (乘算)"));
}
```

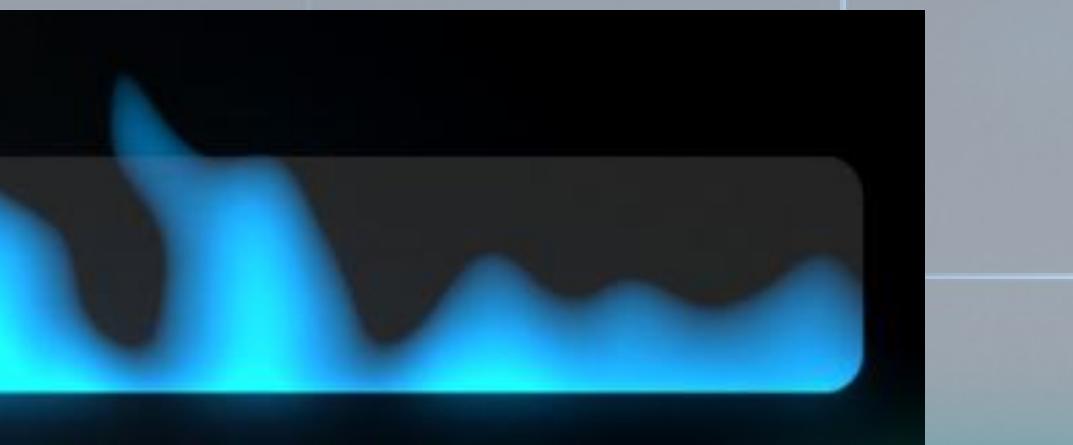


Code

-Buff System

Technical Art

-2D Fire



```
Properties{
    _TexA ("TexA", 2D) = "white" {}
    _TexB ("TexB", 2D) = "white" {}
    _Color ("Color", Color) = (1,1,1,1)
    _Emission ("Emission", Float) = 1
    _Transform ("Transform", Range(0,1)) = 0
    _Fill("Fill (1 for true)", Int) = 1
    _EdgeWidth ("EdgeWidth", Float) = 0
    [HideInInspector]
    _Stencil ("Stencil", Int) = 0
}

SubShader{
    Tags{
        "Queue" = "Transparent"
        "RenderType" = "Sprite"
    }
    LOD 100

    ZWrite off
    Cull off
    Blend SrcAlpha OneMinusSrcAlpha

    Stencil{
        Ref [_StencilRef]
        Comp LEqual
    }

    Pass{
        CGPROGRAM
        #pragma vertex vert
        #pragma fragment frag

        #include "UnityCG.cginc"
        #include "Common/CommonShaderMethods.cg"

        struct appdata{
            float4 vertex : POSITION;
            float2 uv : TEXCOORD0;
            float4 color : COLOR;
        }
    }
}
```

```
ct v2f{
    float2 uv : TEXCOORD0;
    float4 vertex : SV_POSITION;
    float4 color : COLOR;

    float2 _MainTex;
    t4 _MainTex_ST;
    t _Emission, _Edge;
    t4 _Color;
    t2 _Scale, _Speed;

vert(appdata v){
    v2f o;
    o.vertex = UnityObjectToClipPos(v.vertex);
    o.uv = TRANSFORM_TEX(v.uv, _MainTex);
    o.color = v.color;
    return o;
}

t4 frag(v2f i) : SV_Target{
    float h = (1 - pow(i.uv.y, 8.0)) - (1.5 - _Edge);
    float2 uv = (i.uv + _Time * _Speed * -1) * (1 / _Scale);
    float fire = GradientNoise(uv, scale 10) * SimpleNoise(uv, scale 10);

    float a = tex2D(_MainTex, i.uv).a * clamp(fire + h, 0, 1);
    float4 color = _Color * (_Emission + 1);
    return float4(color.rgb, a) * i.color;
}

G
```

```
CommonShader\methods.cg ^
1  float2 unity_gradientNoise_dir(float2 p)
2  {
3      p = p % 289;
4      float x = (34 * p.x + 1) * p.y;
5      x = (34 * x + 1) * x % 289;
6      x = frac(x / 41) * 2 - 1;
7      return normalize(float2(x, y));
8  }
9
10 float unity_gradientNoise(float2 p)
11 {
12     float2 ip = floor(p);
13     float2 fp = frac(p);
14     float d00 = dot(unity_gradiantNoise_dir(ip), ip);
15     float d01 = dot(unity_gradiantNoise_dir(ip + float2(0, 1)), ip + float2(0, 1));
16     float d10 = dot(unity_gradiantNoise_dir(ip + float2(1, 0)), ip + float2(1, 0));
17     float d11 = dot(unity_gradiantNoise_dir(ip + float2(1, 1)), ip + float2(1, 1));
18     float fp2 = fp * fp * fp * (fp * (fp + 1) * (fp + 2));
19     return lerp(lerp(d00, d01, fp.x), lerp(d10, d11, fp.x), fp2);
20 }
21 float GradientNoise(float2 UV, float Scale)
22 {
23     return unity_gradientNoise(UV * Scale);
24 }
25 inline float unity_noise_randomize(float2 uv)
26 {
27     return frac(sin(dot(uv, float2(12.9898, 78.233))) * 43758.5453);
28 }
29 inline float unity_noise_interpolate(float a, float b, float t)
30 {
31     return (1.0 - t)*a + (t*b);
32 }
33 inline float unity_noise_interpolate3d(float a, float b, float c, float t)
34 {
35     return (1.0 - t)*a + (t*(1.0 - t)*b) + (t*t)*c;
36 }
37 inline float unity_valueNoise(float2 uv)
38 {
39     float2 i = floor(uv);
40     float2 f = frac(uv);
41     float c0 = unity_noise_randomize(i);
42     float c1 = unity_noise_randomize(i + float2(1, 0));
43     float c2 = unity_noise_randomize(i + float2(0, 1));
44     float c3 = unity_noise_randomize(i + float2(1, 1));
45     float r0 = unity_noise_randomize(f);
46     float r1 = unity_noise_randomize(f + float2(0, 1));
47     float r2 = unity_noise_randomize(f + float2(1, 0));
48     float r3 = unity_noise_randomize(f + float2(1, 1));
49     float bottomOfGrid = unity_noise_randomize(i);
50     float topOfGrid = unity_noise_randomize(i + float2(1, 1));
51     float t = unity_noise_interpolate3d(r0, r1, r2, r3, f.y);
52     float freq = pow(2.0, float2(i.x, i.y));
53     float amp = pow(0.5, float2(freq.x, freq.y));
54     float t += unity_valueNoise(freq);
55     freq = pow(2.0, float2(freq.x, freq.y));
56     amp = pow(0.5, float2(freq.x, freq.y));
57     t += unity_valueNoise(freq);
58 }
```

Similar to the Glitter effect, I found that Shader Graphs don't support `_Stencil`, so I wrote a shader myself instead of using shader graphs. The Simple noise and Gradient noise are methods stored in a .cg file, and the codes are from Unity's shader graph document.

Simple Noise Node | Shader Graph | 10.5.1 (unity3d.com)

Gradient Noise Node | Shader Graph | 10.5.1 (unity3d.com)

Technical Art

Custom Full Screen Post-Processing & Timeline

ed a custom **Render Feature**, a
tom **Render Pass** to make a **Volume**,
used a **Shader** to implement a
tom full screen effect. I then used
ty's **Timeline** system to implement
“Judgement Cut End” effect using
ty's Timeline.

```
c class JCERenderFeature : ScriptableRendererFeature {
    public Shader m_Shader;
    public RenderPassEvent _event;
    private static readonly int thresholdID = Shader.PropertyToID(name: "_StripeThreshold");
    private static readonly int distortionID = Shader.PropertyToID(name: "_SpaceDistortion");
    private static readonly int emissionID = Shader.PropertyToID(name: "_Emission");
    private static readonly int inverseID = Shader.PropertyToID(name: "_InverseSingleColor");
    public Material m_Material;
    public List<string> cameraTags;
    public bool active;

    private JCEPass m_RenderPass = null;

    个引用
    public override void AddRenderPasses(ScriptableRenderer renderer, ref RenderingData renderingData) {
        VolumeStack v = VolumeManager.instance.stack;
        JCEVolume JCE = v.GetComponent<JCEVolume>();
        active = true;
        if (JCE.activated.value) {
            if (cameraTags.Contains(renderingData.cameraData.camera.tag)) {
                float threshold = JCE.stripesThreshold.value;
                float distortion = JCE.spaceDistortion.value;
                float emission = JCE.emission.value;
                int inverse = JCE.inverseAndSingleColor.value ? 1 : 0;
                m_Material.SetFloat(thresholdID, threshold);
                m_Material.SetFloat(distortionID, distortion);
                m_Material.SetFloat(emissionID, emission);
                m_Material.SetInt(inverseID, inverse);
                renderer.EnqueuePass(m_RenderPass);
            }
        }
        active = false;
    }

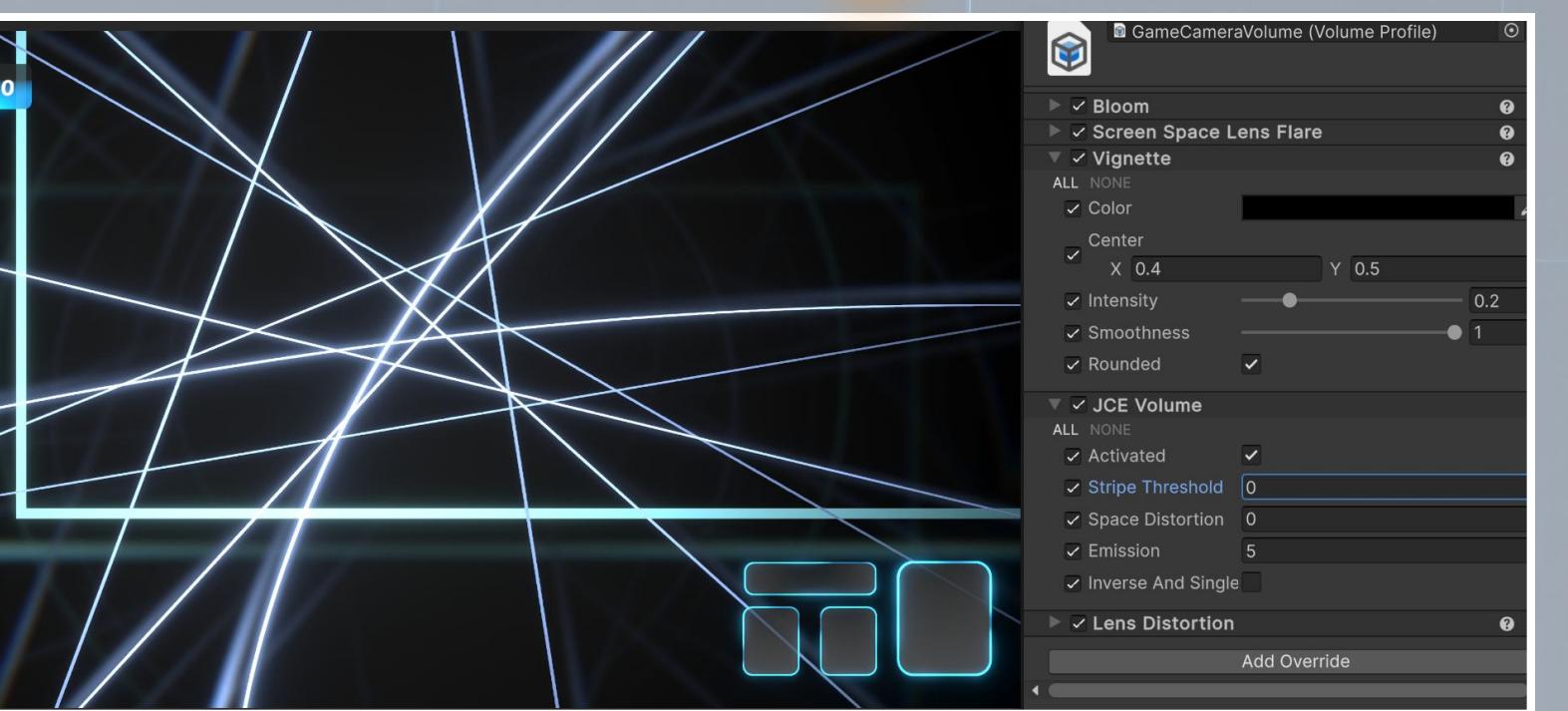
    个引用
    public override void SetupRenderPasses(ScriptableRenderer renderer, in RenderingData renderingData) {
        if (active) {
            m_RenderPass.ConfigureInput(ScriptableRenderPassInput.Color);
            m_RenderPass.SetTarget(renderer.cameraColorTargetHandle);
        }
    }

    个引用
    public override void Create() {
        m_RenderPass = new JCEPass(m_Material, _event);
```

Custom Render Feature

Video: https://youtu.be/7urWfuJ2_mM

Be Threshold = 0



Joe Threshold = 0.2

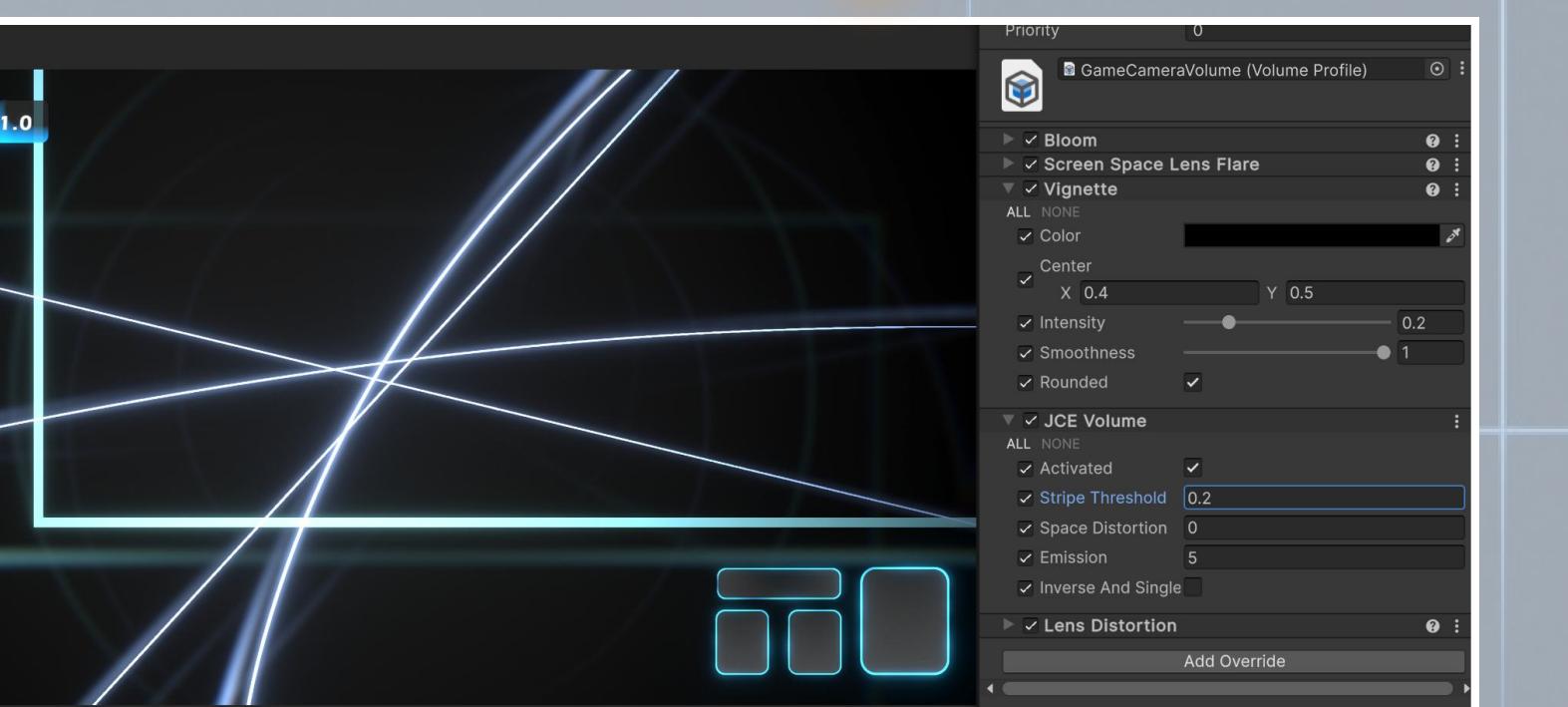
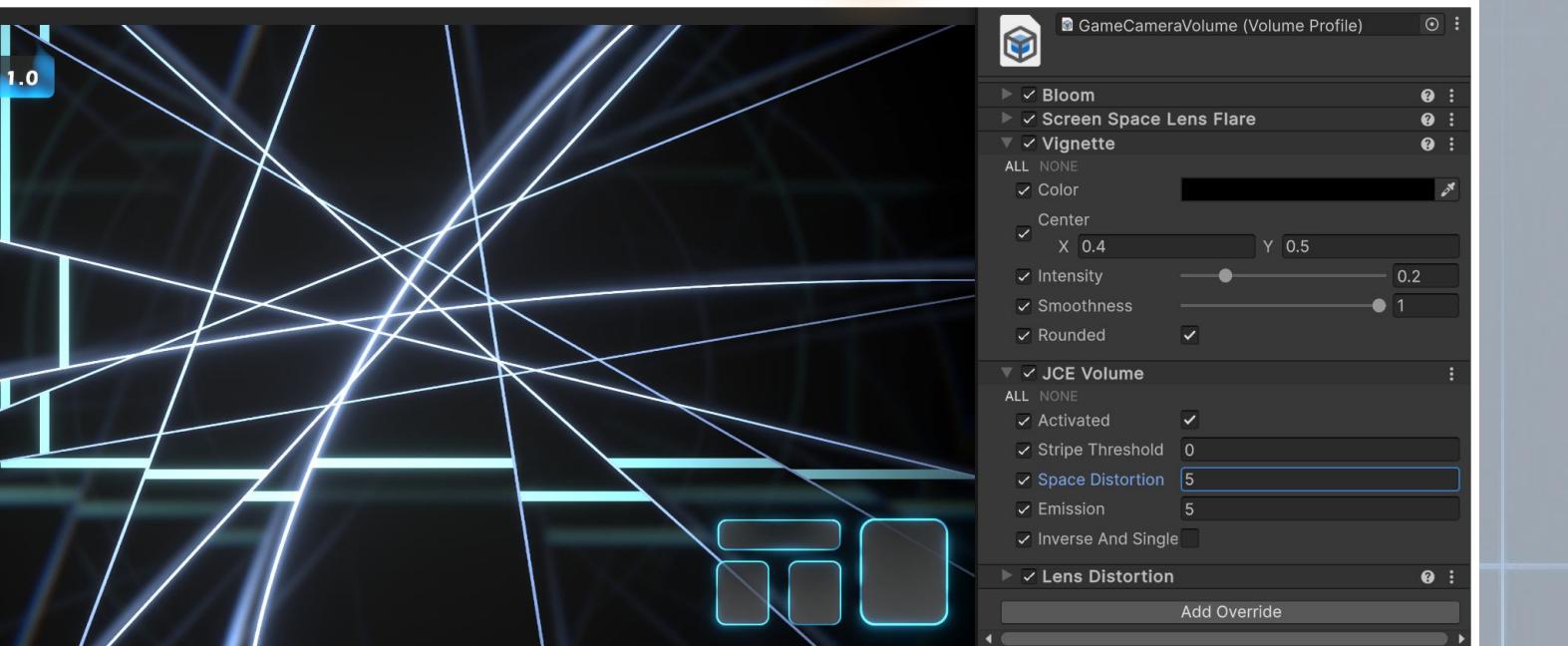
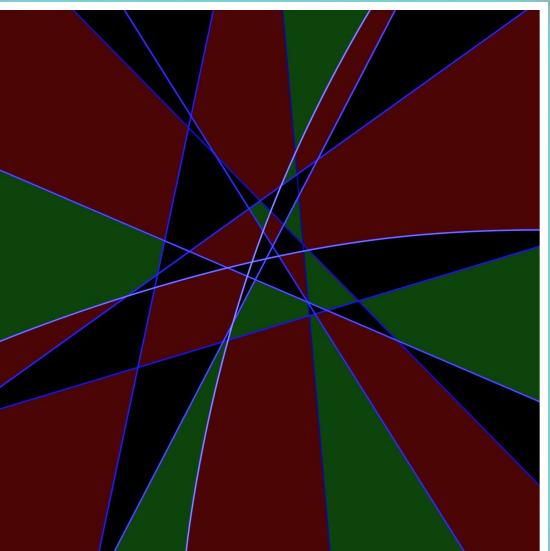


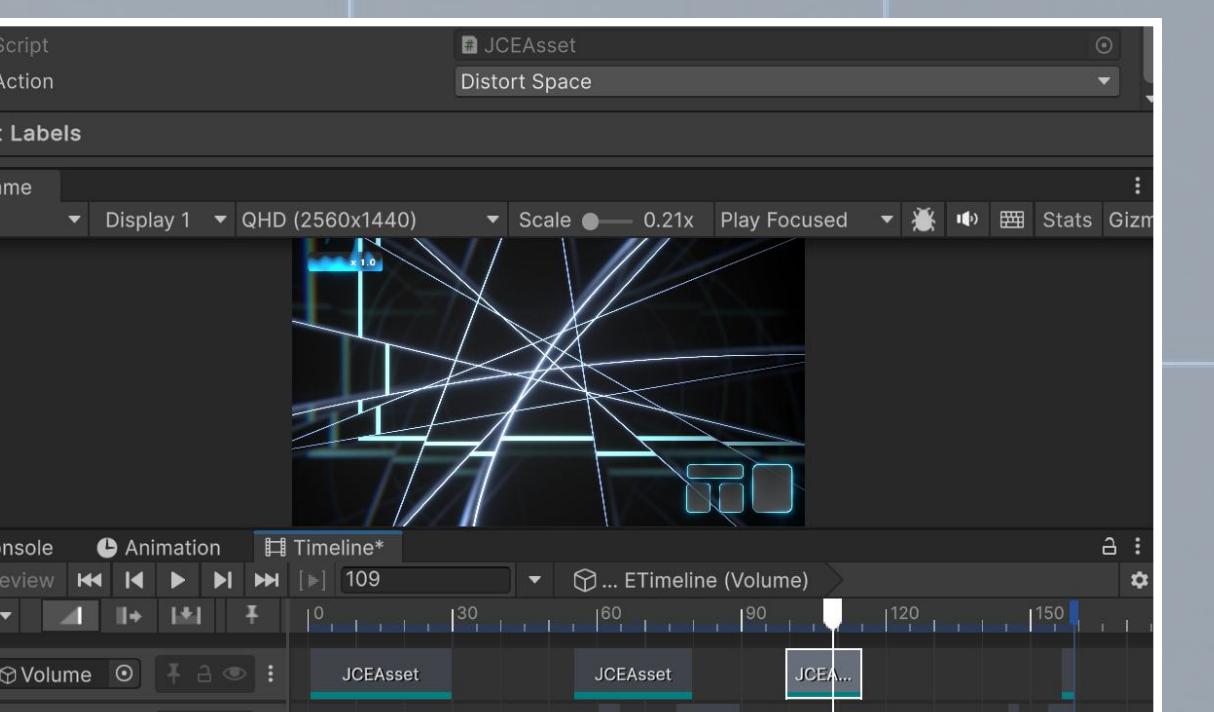
Image Distortion = 5



Main Texture



Timeline



A screenshot of the Unity Editor interface. At the top, there's a toolbar with various icons like 'Script', 'Action', and 'Labels'. Below it is a header bar with the text '# JCEAsset' and 'Pre Distort'. The main workspace shows a scene with a bright, glowing energy effect composed of many white and blue streaks against a dark background. To the right of the scene view are buttons for 'Scale' (set to 0.21x), 'Play Focused', and other scene controls. Below the scene view is a timeline at the bottom of the screen. The timeline has several keyframes labeled 'JCEAsset' at various time points (0, 30, 60, 90, 120, 150). The current frame is set to 73. On the far left, there are buttons for 'Console', 'Animation', and 'Timeline*', with 'Timeline*' being the active tab. At the very bottom, there are buttons for 'Volume' and other scene controls.

Technical Art

- 2D SDF Texture Generation with Compute Shaders

I wrote a **compute shader** using the “two pass” method to generate **Signed Distance Field** texture for icons. Then, I wrote a **Shader** to make the transition animation of the **Pause/Continue** icon.

▼ Compute Shader

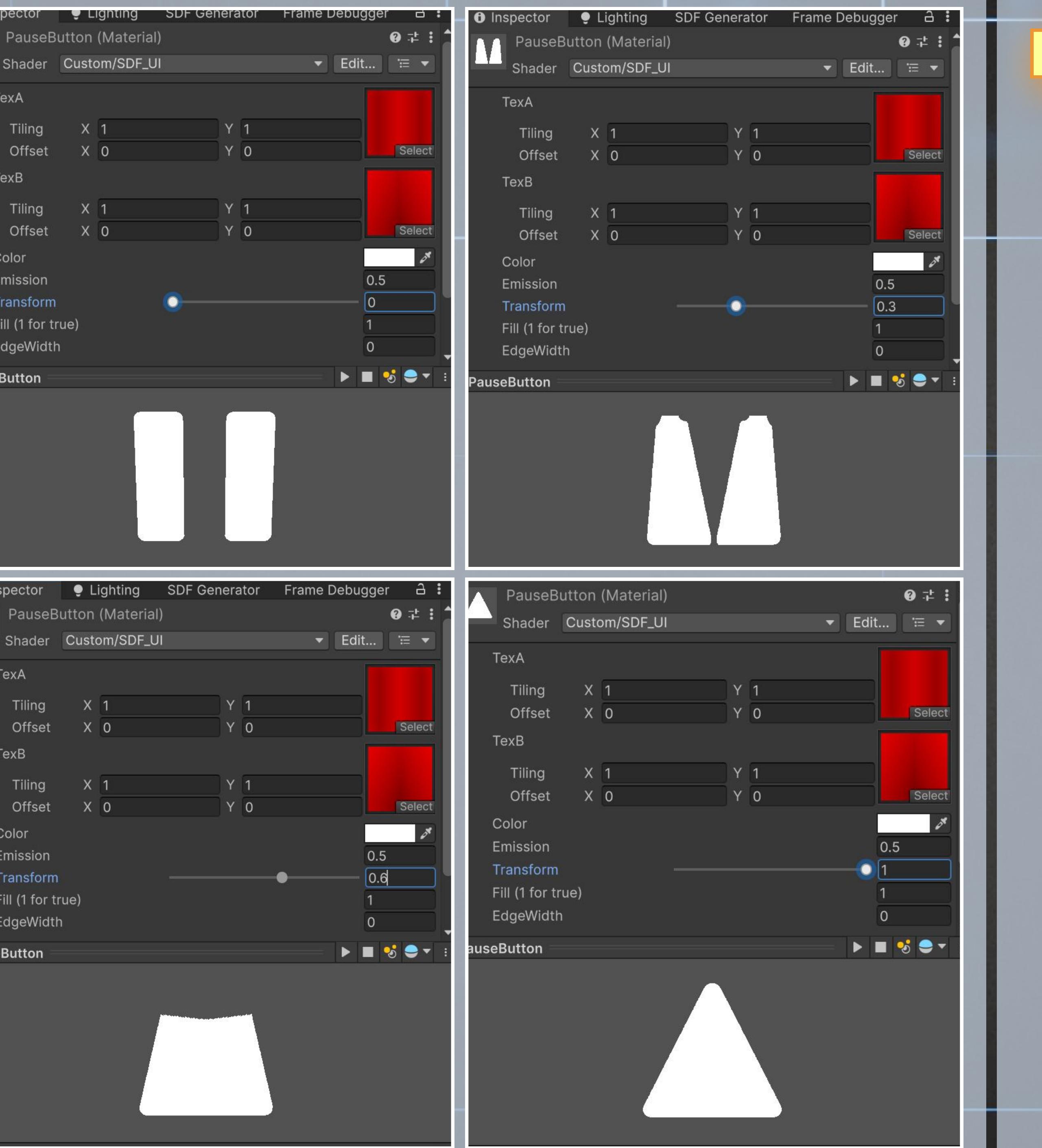
```
#pragma kernel StepA
#pragma kernel StepB

Texture2D<float> _InputTex;
RWTexture2D<float> _DistanceList;
RWTexture2D<float> _OutTex;
float2 _InputTexSize;

[numthreads(16,16,1)]
void StepA(int3 id : SV_DispatchThreadID){
    float size = min(_InputTexSize.x, _InputTexSize.y);
    int delta = 0;
    bool isWhite = _InputTex[id.xy] > 0.5;
    int maxDist = max(id.x, _InputTexSize.x - id.x);
    while (delta < maxDist){
        if (_InputTex[int2(id.x + delta, id.y)] > 0.5 == isWhite && _InputTex[int2(id.x - delta, id.y)] > 0.5 == isWhite){
            delta++;
        } else{
            break;
        }
    }
    if (delta == maxDist){
        delta = size * 100;
    }
    _DistanceList[id.xy] = (isWhite ? -1.0 : 1.0) * delta * delta;
}

[numthreads(16,16,1)]
void StepB(int3 id : SV_DispatchThreadID){
    int size = min(_InputTexSize.x, _InputTexSize.y);
    bool isWhite = _DistanceList[id.xy] < 0;
    int shortest = size * size;
    for (int y = 0; y < _InputTexSize.y; y++){
        float d = pow(y - id.y, 2) + abs(_DistanceList[int2(id.x, y)]);
        shortest = min(shortest, d);
    }
    _OutTex[id.xy] = (isWhite ? -1.0 : 1.0) * sqrt(shortest) / size + 0.5;
}
```

▼ Shader for Icon



Summary

In this project, I got a deeper understanding of **shaders** and **VFX Graphs** and learned how to use features such as **Reflection**, **Customized Inspector**, **Timeline**, **Compute Shaders**, **Custom Render Feature** and so on. I've also learned how to support mobile phones and touchscreen.

Based on some playtests, I'm planning to add more maps and buffs to increase variability. Besides, I'm planning to add save/load systems to allow players continue a game even if they have to leave in the middle.