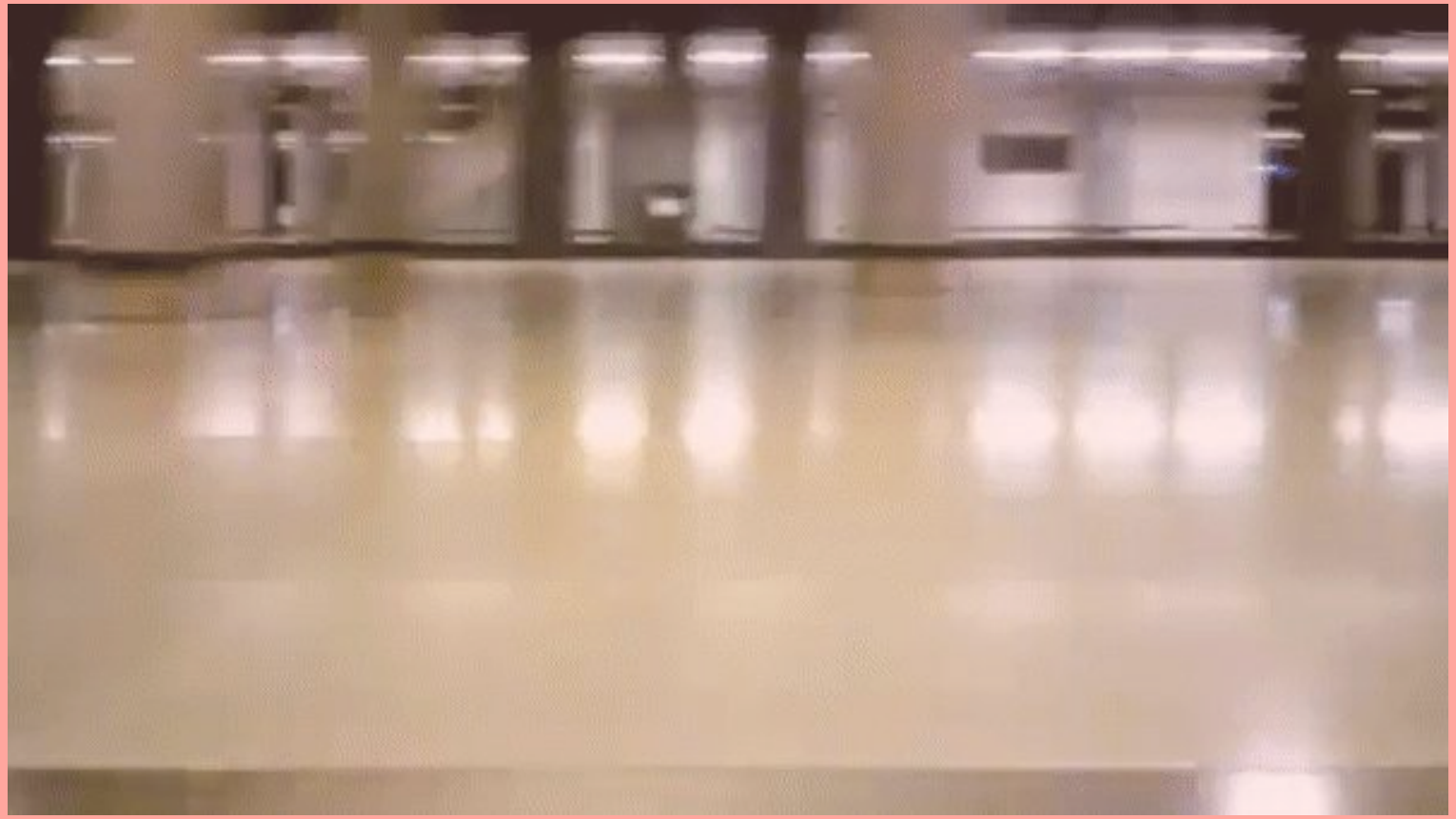
FINAL PROJECT: KALEIDOSCOPE

Zhexian Zhang | William Koh | Dhanya Janaki

INTRODUCTION

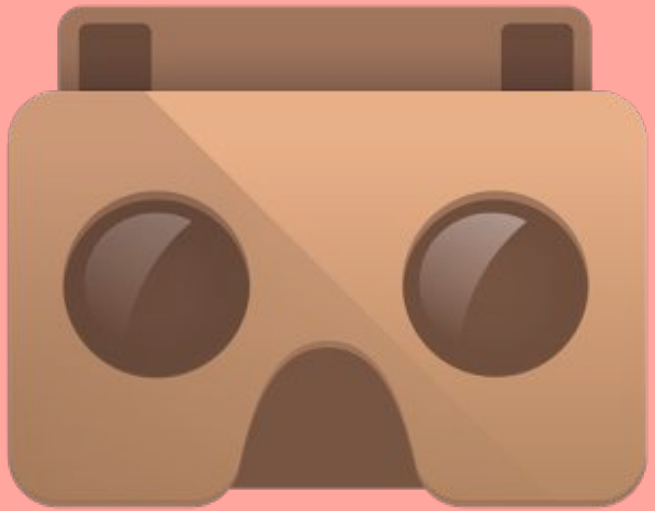
RELATED WORK



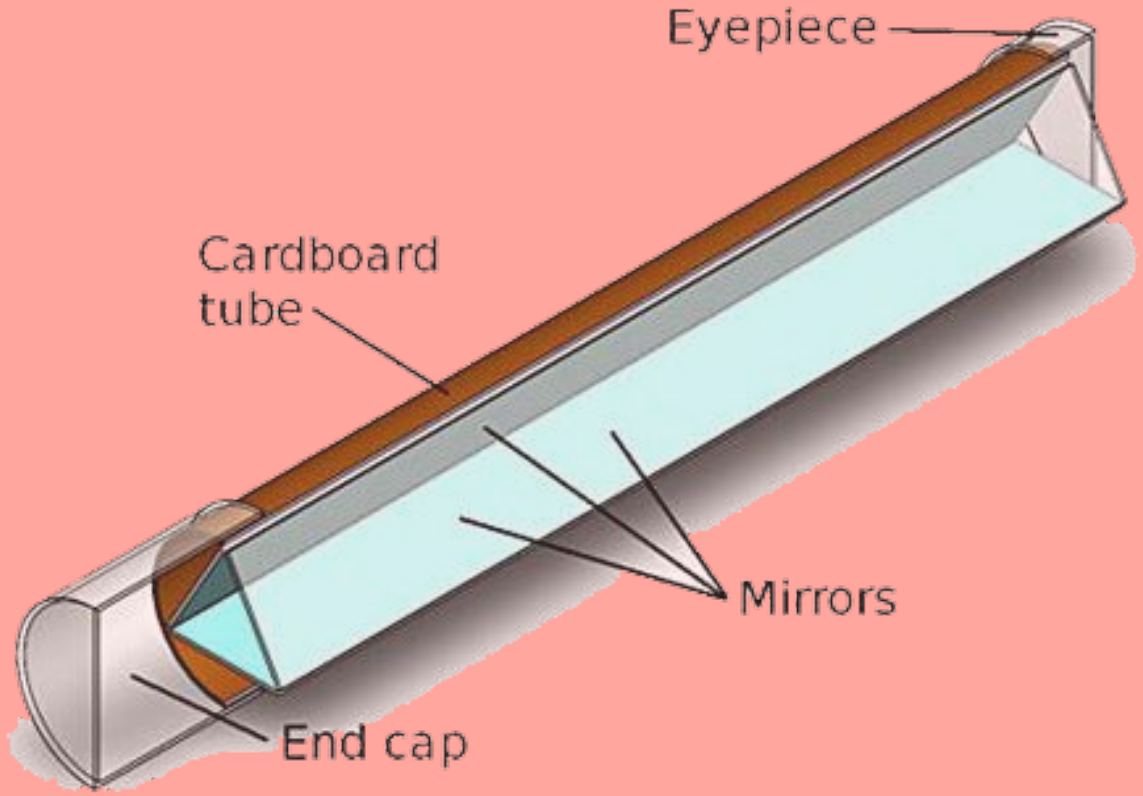
RELATED WORK

OBJECTIVES

- Basic digital kaleidoscope
- Adjustable object, size, color, and mirror angles
- Virtual reality viewing capability

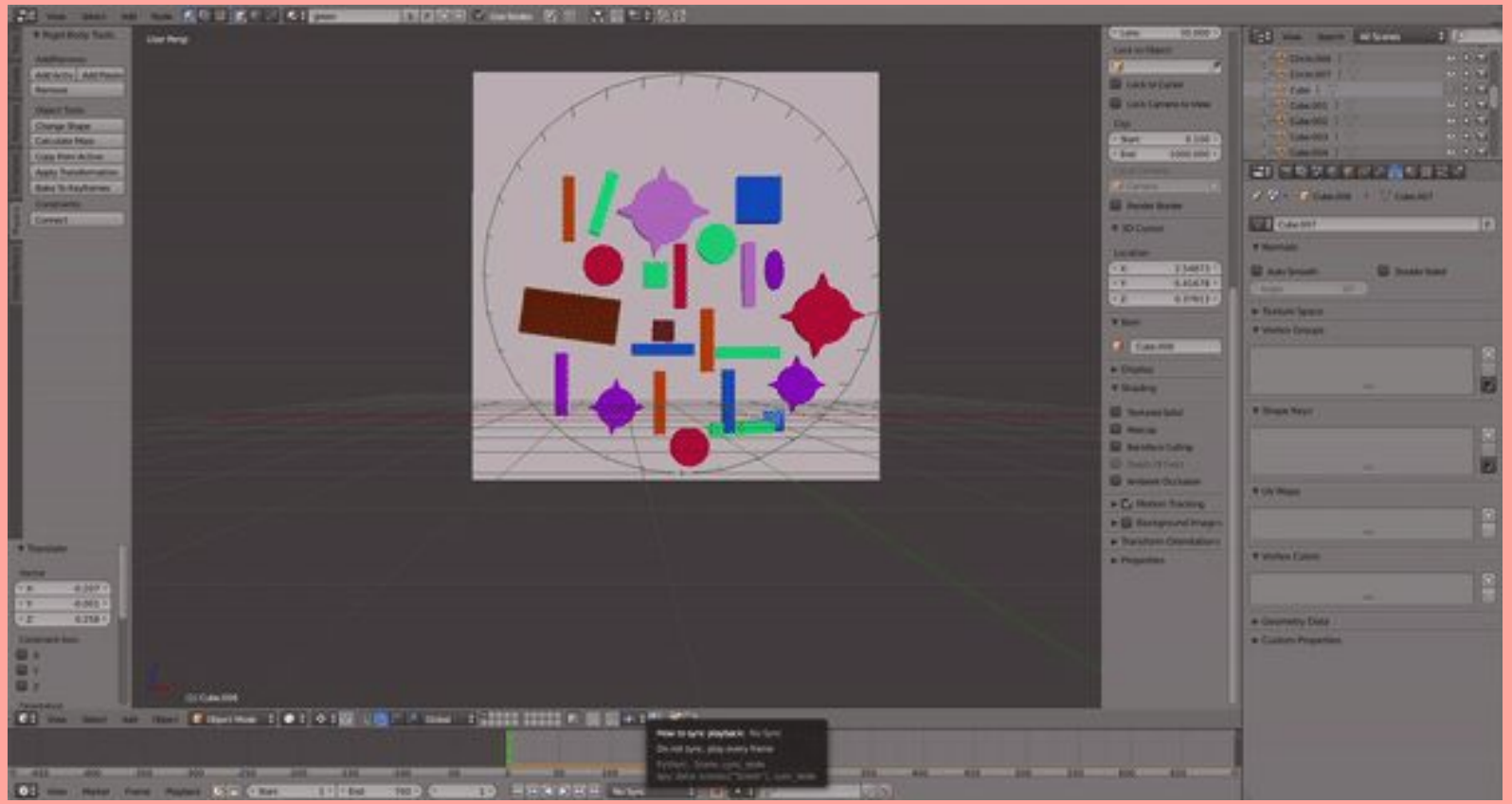


HOW DOES IT WORK

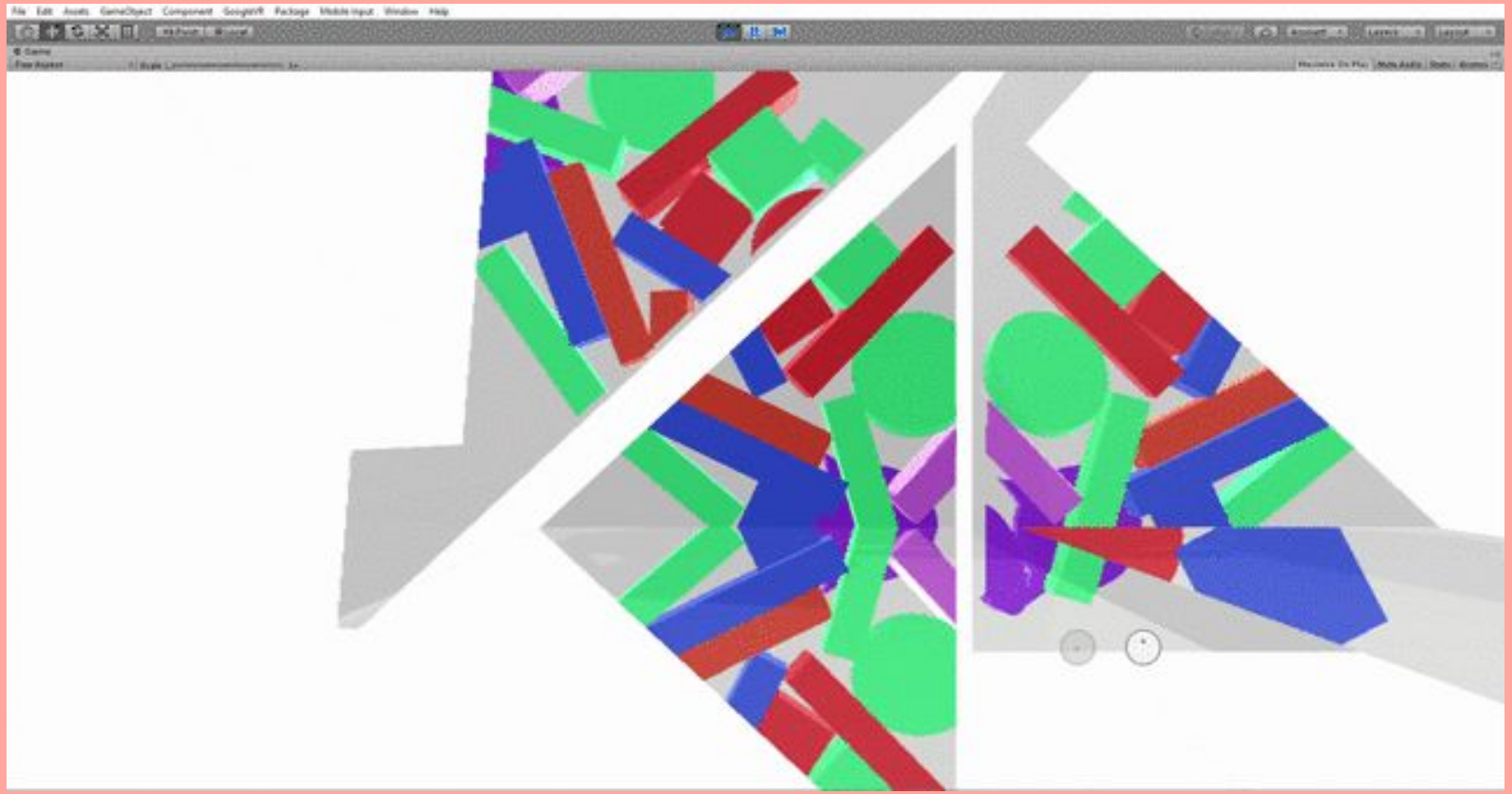


TECHNOLOGIES USED

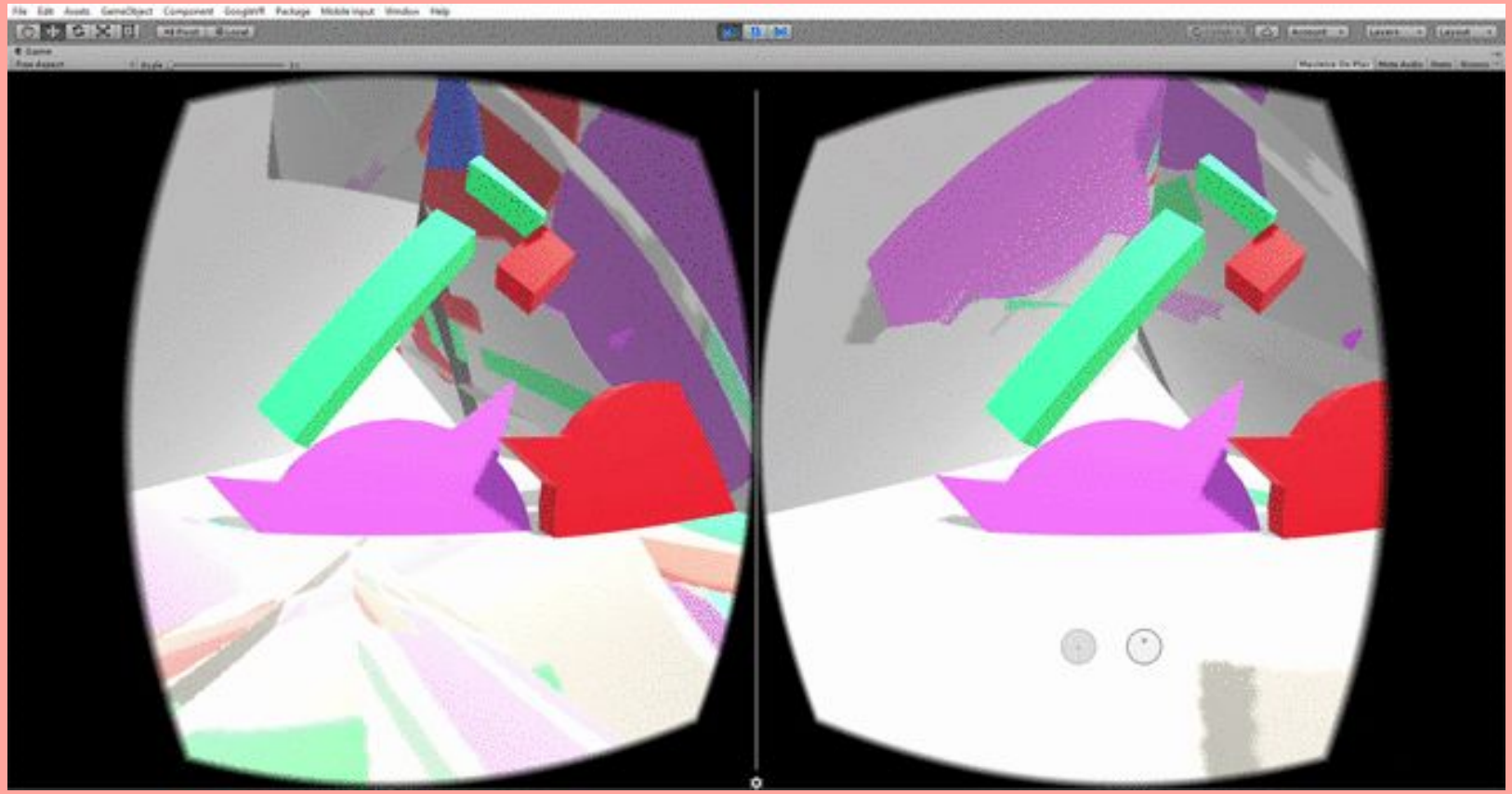
ATTEMPT 1: BLENDER



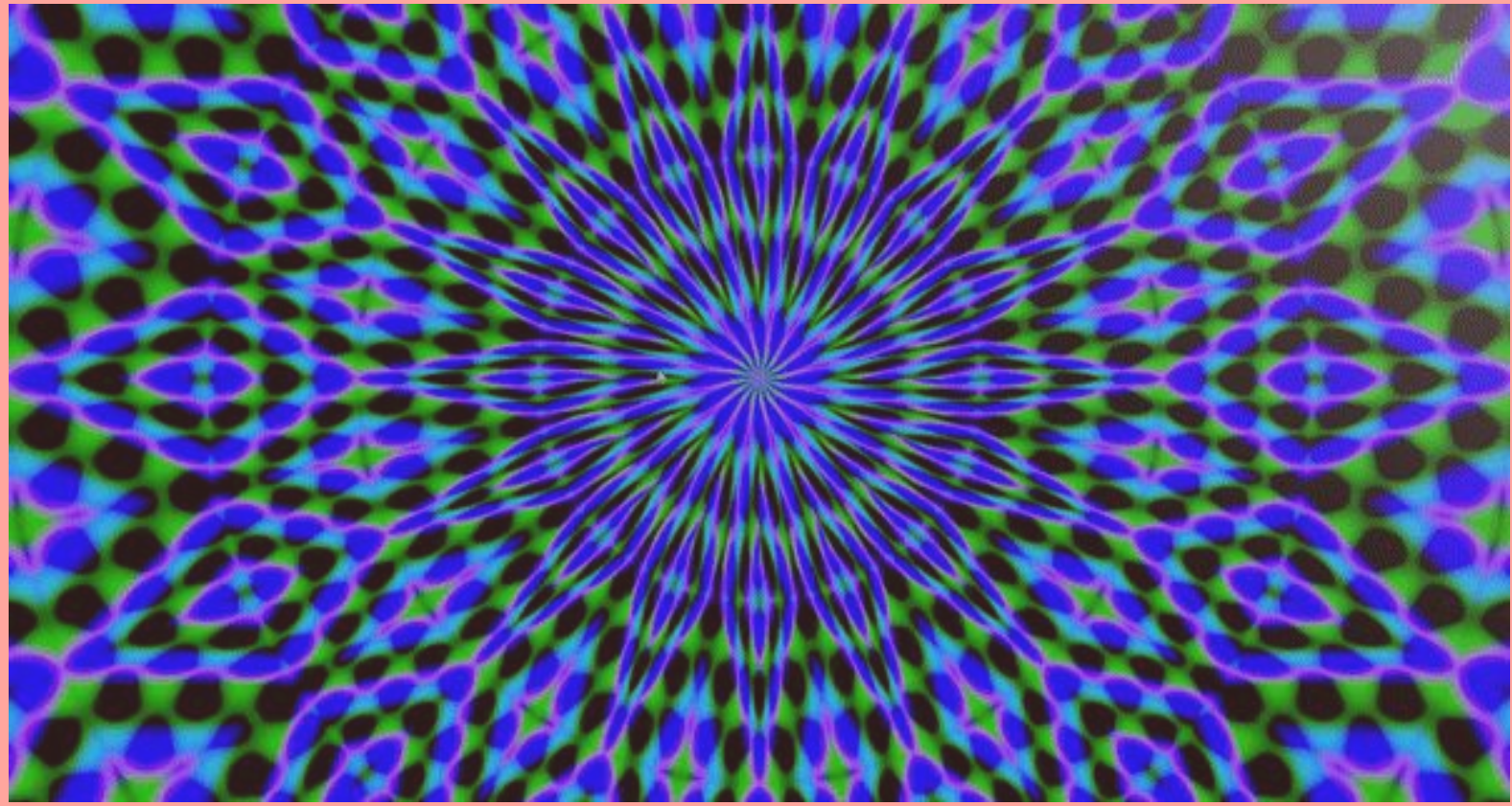
ATTEMPT 1: UNITY



ATTEMPT 1: VR



ATTEMPT 2: BLENDER (AGAIN)



3D VIEW IS NOT WORKING PROPERLY.....
SO USE 2D!

ADDED FEATURES

- Changing the scene- background, color and lighting
- Changing number of mirrors
- VR

HOW IT WORKS

What we need:

- Mirror script
- Mirror shader
- Google VR

MIRROR SCRIPT

OnRenderImage()

- This function allows modification on final image via shader based filters.
- The main purpose of this function is to copy the source texture into destination texture with a shader.

  Graphics.Blit(source, destination, _material);

MIRROR SHADER

Vertex and fragment Shader

- Lighting is not needed and produce exotic image effect.
- Get vertices and convert to polar coordinates.
- Did calculation to get the rotation angle and return to 2D texture then rendered.

GOOGLE VR

- Google VR SDK: **GvrView** handles the VR mode.
- In order to toggle between VR mode and normal mode, a button is used as a trigger.

GOOGLE VR

```
void isVR()
{
    if(cam[0].isActiveAndEnabled == false)
    {
        gvr.VRModeEnabled = true;
        //cam[0].gameObject.SetActive(true);
        cam[0].gameObject.GetComponent<Camera>().enabled = true;
        cam[1].gameObject.SetActive(false);

    }
    else if(cam[0].isActiveAndEnabled == true)
    {
        gvr.VRModeEnabled = false;
        cam[0].gameObject.GetComponent<Camera>().enabled = false;
        cam[1].gameObject.SetActive(true);
    }
}
```

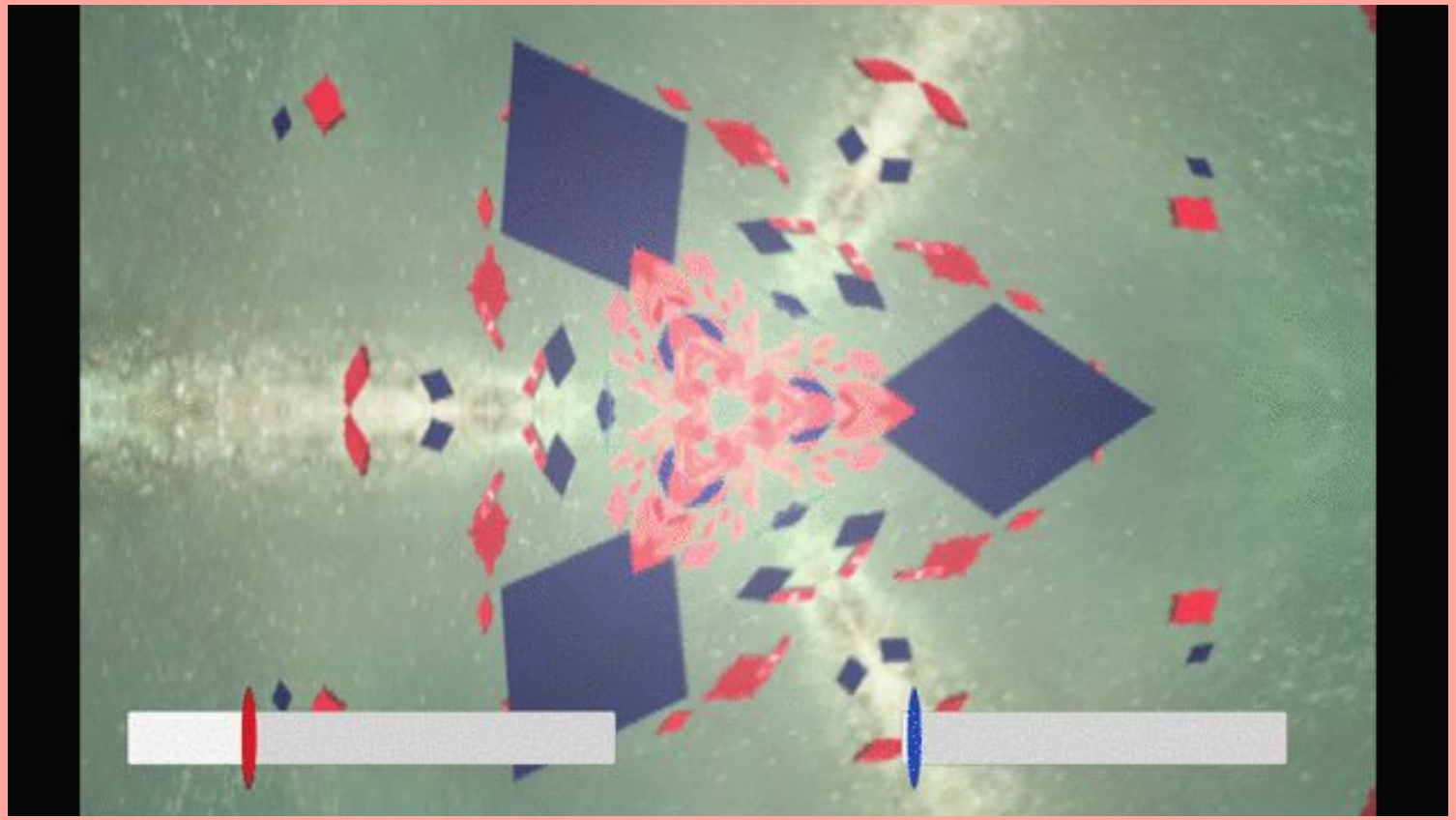
CHANGE SCENE: DOUBLE TAP

```
if (Input.GetMouseButtonDown(0))
{
    _buttonDownPhaseStart = Time.time;
}

if (Input.GetMouseButtonUp(0))
{

    if (Time.time - _buttonDownPhaseStart < 0.2f)
    {
        Debug.Log("double click");
        _buttonDownPhaseStart = -1;
```

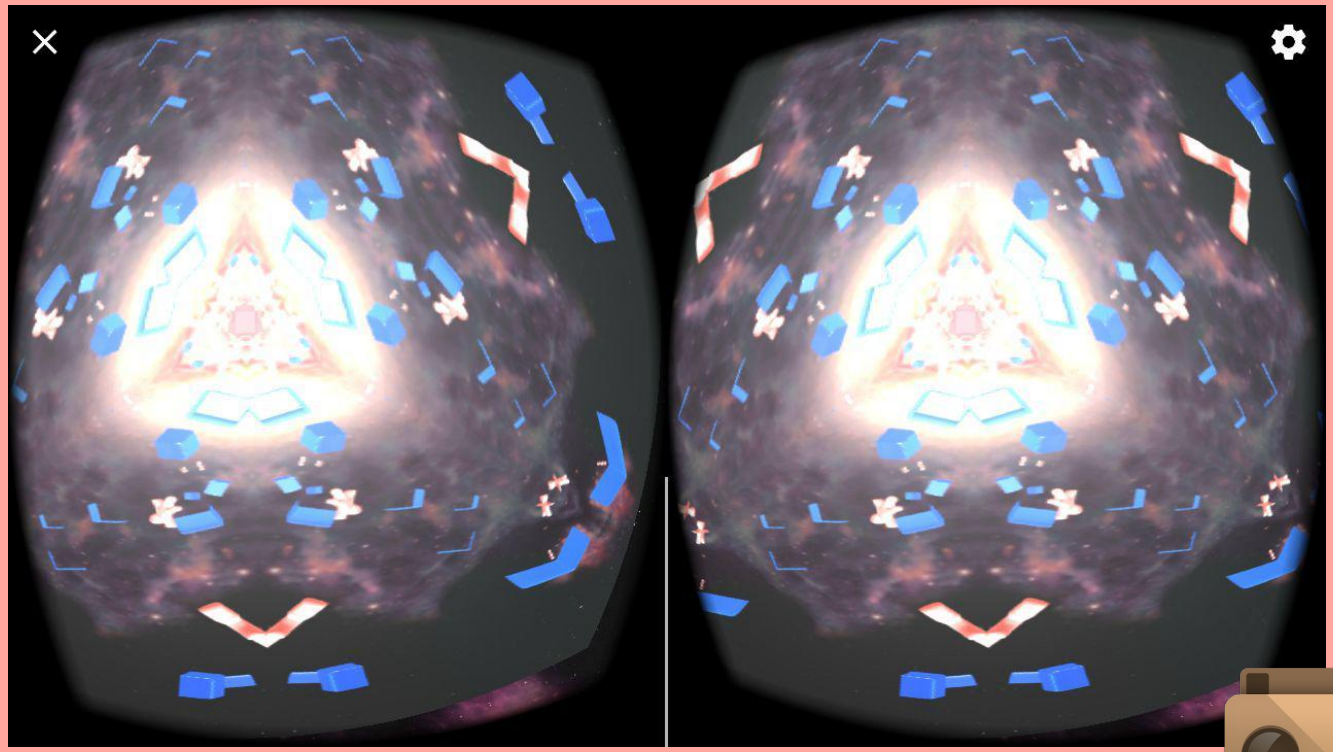
MANUAL ROTATION: DRAGGING

```
     if (Input.GetMouseButton(0))
        {

            // get mouse position on x-axis on the screen and transform the lens
rotation
            yaw += speedH * Input.GetAxis("Mouse X");
           // get mouse position on y-axis on the screen and transform the lens
rotation
            pitch -= speedV * Input.GetAxis("Mouse Y");

            // rotation as Euler Angle in degree.
            transform.eulerAngles = new Vector3(0.0f, 0.0f, pitch);
        }
```

FINAL VERSION



FINAL VERSION



FUTURE WORK

- Limitation of Google Cardboard: limited interaction
- Alternatives: HTC Vive (handles)

DEMO: VR

THE ASK AND THE ANSWER