葉惟欣
F74109016

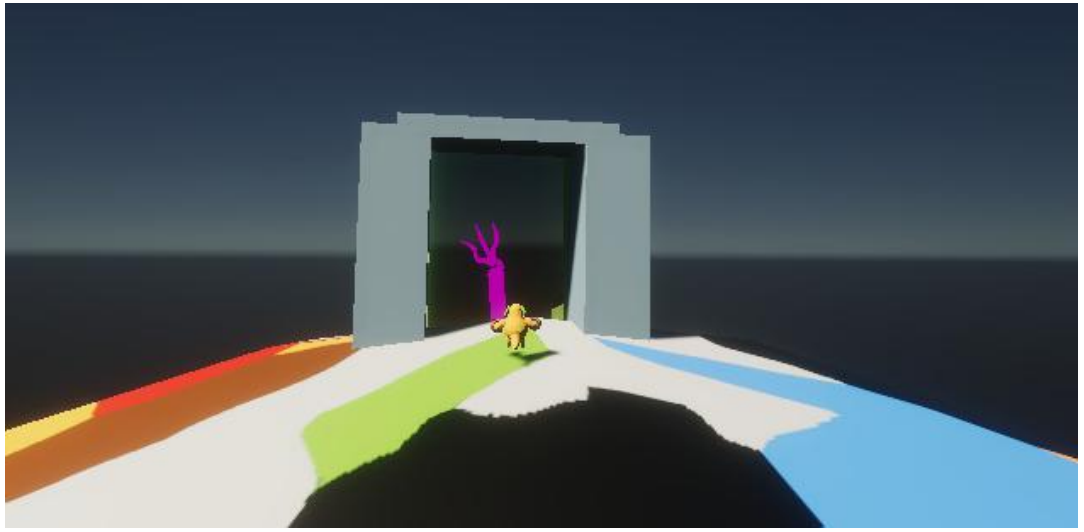
傳送門

MouseLook.cs	控制旋轉角度透過玩家滑鼠
PlayerMovement.cs	控制玩家移動，由up,down,left,right， 與經過傳送門後的位置瞬移
PortalCamara.cs	傳送門的相機旋轉角度隨著玩家的旋轉角度同時改變
PortalTeleporter.cs	計算傳送前玩家的自身旋轉角度& 玩家面相傳送門的角度(距離向量)
PortalTextureSetup.cs	設定傳送門render的材質
ScreenCutoutShader.shader	顯示在傳送門上的shader

傳送門1
https://youtu.be/lu8SfiFLmzw

可以以在A星球上相對於傳送門的視野與角度
站在B星球上的東西

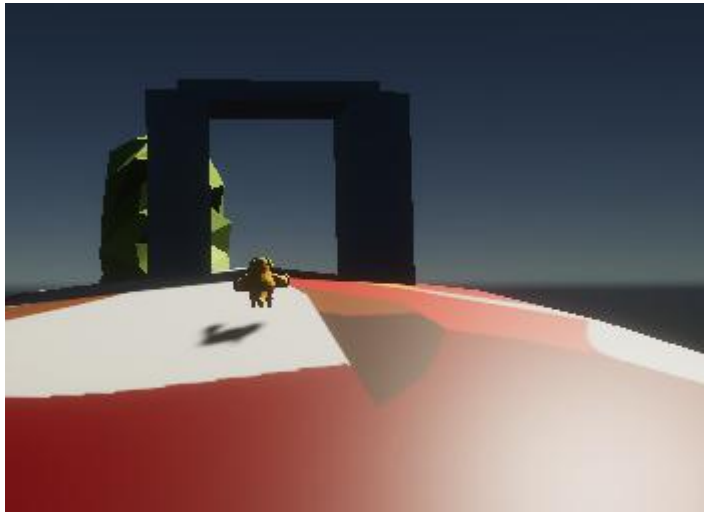
可以以在B星球上相對於傳送門的視野與角度
看相對站在A星球上的東西能看到的東西



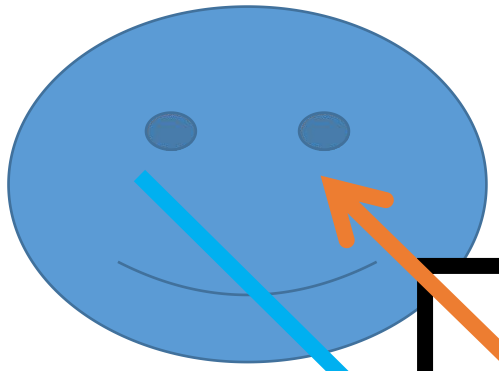
傳送門2
https://youtu.be/D_VF4Y-bNzA

在A星球傳送門的另外一面就不會看到B星球的東西

在B星球傳送門的另外一面就不會看到A星球的東西



傳送門示意圖



X

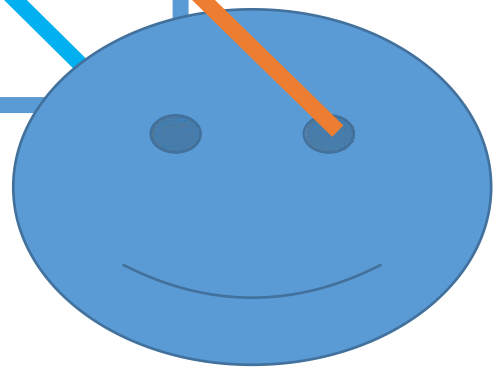
從B世界到A世界行走的方向
且也要從L側才能看到X側的東西

Z

傳送門A

傳送門B

Y



L

從A世界到B世界行走的方向
且也要從X側才能看到L側的東西

PlayerMovement

- 判斷有沒有傳送，如果有傳送就重新給新的座標。
- 在FixedUpdate()做，不能再Update。如果在Update()做畫面會跟不上，就會無法傳送成功。

```
void FixedUpdate()
{


    float x = Input.GetAxis("Horizontal");
    float z = Input.GetAxis("Vertical");

    if (transport == false)
    {
        Vector3 move = transform.right * x + transform.forward * (z);
        controller.Move(move * speed * Time.fixedDeltaTime);

    }
    else if(transport == true && rotate == false)
    {
        controller.transform.position = newposition;
        rotate = true;
    }
    else if(transport == true && rotate == true)
    {
        controller.transform.Rotate(angle_x, angle_y, angle_z);
        rotate = false;
        transport = false;
    }

}
```

PortalCamara

- 計算兩扇門間的旋轉角度差，用來作為玩家對於傳送門的面相方向，才可以在傳送過後相對的方向不會改變。

```
public class PortalCamara : MonoBehaviour
{
    // Start is called before the first frame update
    public Transform player;
    public Transform portal;
    public Transform another_portal;

    // Update is called once per frame

    void Update()
    {



        float angle1 = another_portal.transform.eulerAngles.x - portal.transform.eulerAngles.x;
        float sinx = Mathf.Sin(angle1 * Mathf.Deg2Rad);
        float cosx = Mathf.Cos(angle1 * Mathf.Deg2Rad);

        float angle2 = another_portal.transform.eulerAngles.y - portal.transform.eulerAngles.y;
        float siny = Mathf.Sin(angle2 * Mathf.Deg2Rad);
        float cosy = Mathf.Cos(angle2 * Mathf.Deg2Rad);

        float angle3 = another_portal.transform.eulerAngles.z - portal.transform.eulerAngles.z;
        float sinz = Mathf.Sin(angle3 * Mathf.Deg2Rad);
        float cosz = Mathf.Cos(angle3 * Mathf.Deg2Rad);

        float r11 =  cosy * cosz + siny * sinx * sinz;
        float r12 = -cosy * sinz + siny * sinx * cosz;
        float r13 =  siny * cosx;
        float r21 =  cosx * sinz;
        float r22 =  cosx * cosz;
        float r23 = -sinx;
        float r31 = -siny * cosz + cosy * sinx * sinz;
        float r32 =  siny * sinz + cosy * sinx * cosz;
        float r33 =  cosy * cosx;

        Vector3 vector1 = (player.GetChild(0).position - another_portal.position);

        Vector3 vectorNew = new Vector3(vector1.x * r11 + vector1.y * r12 + vector1.z * r13,
                                        vector1.x * r21 + vector1.y * r22 + vector1.z * r23,
                                        vector1.x * r31 + vector1.y * r32 + vector1.z * r33);

        transform.position = portal.position + vectorNew;

        Vector3 newCameraDircetion = new Vector3(player.GetChild(0).forward.x * r11 + player.GetChild(0).forward.y * r12 + player.GetChild(0).forward.z * r13,
                                                 player.GetChild(0).forward.x * r21 + player.GetChild(0).forward.y * r22 + player.GetChild(0).forward.z * r23,
                                                 player.GetChild(0).forward.x * r31 + player.GetChild(0).forward.y * r32 + player.GetChild(0).forward.z * r33);

        transform.rotation = Quaternion.LookRotation(newCameraDircetion, Vector3.up);

    }
}
```

PortalTeleporter

- 透過玩家的位置跟傳送門的位置 距離向量，乘上兩個傳送門之間的角度差(三維)

- 計算兩扇門間的旋轉角度差，用來作為玩家對於傳送門的面相方向，才可以在傳送過後相對的方向不會改變。

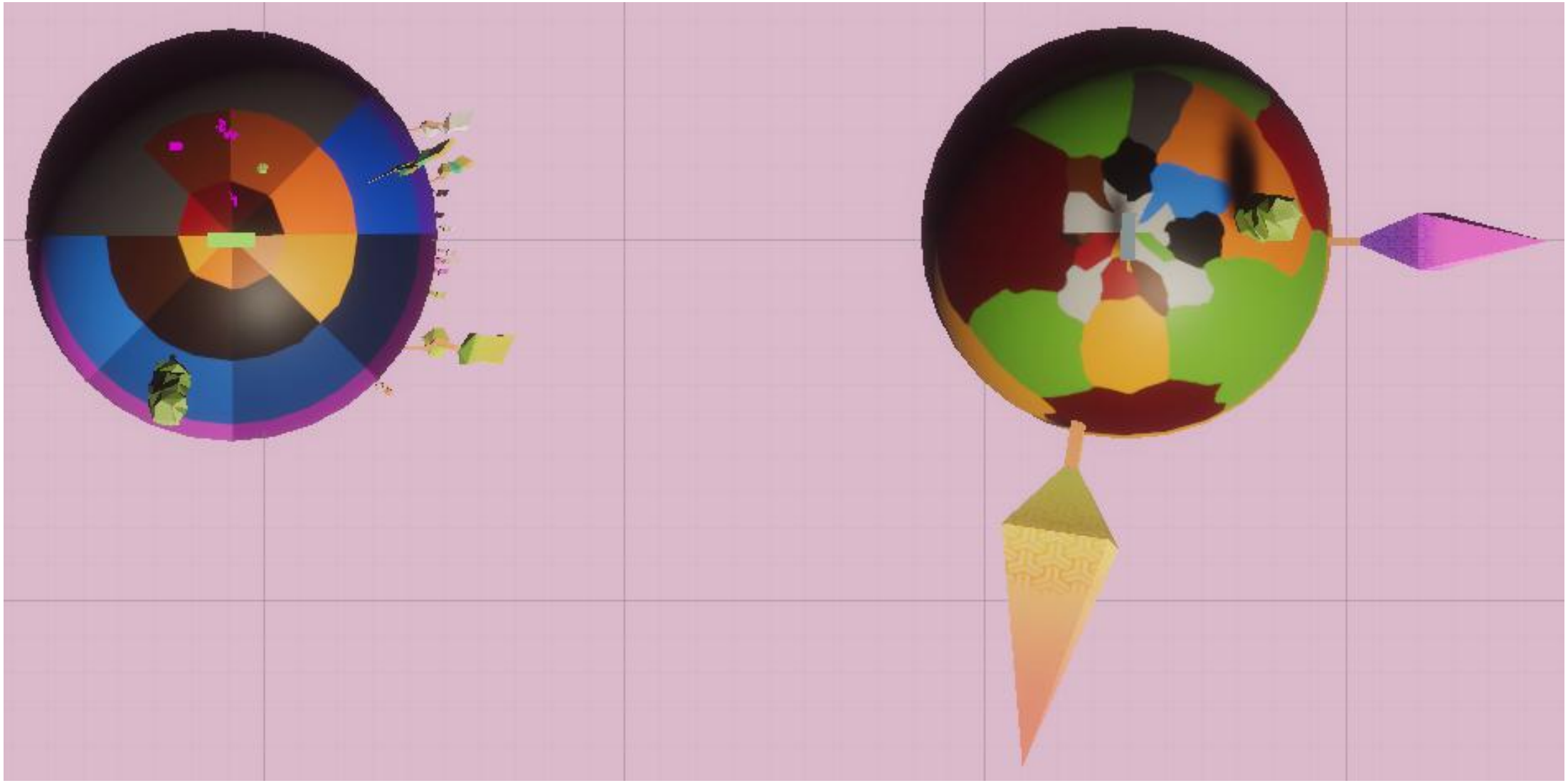
```
void OnTriggerEnter(Collider other)
{

    Vector3 portalToPlayer = player.transform.position - transform.transform.position;
    float dotProduct = Vector3.Dot(transform.transform.forward, portalToPlayer);
   Debug.Log(dotProduct);
   Debug.Log(portalToPlayer);
   if (other.tag == "Player" && dotProduct>0.0 && player.GetComponent<PlayerMovement>().transport == false )
    {
        float angle_x = transform.transform.eulerAngles.x - another_portal.transform.eulerAngles.x;
        float sin_x = Mathf.Sin(angle_x * Mathf.Deg2Rad);
        float cos_x = Mathf.Cos(angle_x * Mathf.Deg2Rad);

        float angle_y = transform.transform.eulerAngles.y - another_portal.transform.eulerAngles.y;
        float sin_y = Mathf.Sin(angle_y * Mathf.Deg2Rad);
        float cos_y = Mathf.Cos(angle_y * Mathf.Deg2Rad);

        float angle_z = transform.transform.eulerAngles.z - another_portal.transform.eulerAngles.z;
        float sin_z = Mathf.Sin(angle_z * Mathf.Deg2Rad);
        float cos_z = Mathf.Cos(angle_z * Mathf.Deg2Rad);

        float r11 =  cos_y * cos_z + sin_y * sin_x * sin_z;
        float r12 = -cos_y * sin_z + sin_y * sin_x * cos_z;
        float r13 =  sin_y * cos_x;
        float r21 =  cos_x * sin_z;
        float r22 =  cos_x * cos_z;
        float r23 = -sin_x;
        float r31 = -sin_y * cos_z + cos_y * sin_x * sin_z;
        float r32 =  sin_y * sin_z + cos_y * sin_x * cos_z;
        float r33 =  cos_y * cos_x;

        Vector3 vectorNew = new Vector3(portalToPlayer.x * r11 + portalToPlayer.y * r12 + portalToPlayer.z * r13,
                                        -(portalToPlayer.x * r21 + portalToPlayer.y * r22 + portalToPlayer.z * r23),
                                        portalToPlayer.x * r31 + portalToPlayer.y * r32 + portalToPlayer.z * r33);

       player.GetComponent<PlayerMovement>().newposition = another_portal.position + vectorNew;
       player.GetComponent<PlayerMovement>().angle_x = angle_x;
       player.GetComponent<PlayerMovement>().angle_y = angle_y;
       player.GetComponent<PlayerMovement>().angle_z = angle_z;
       player.GetComponent<PlayerMovement>().transport = true;
    }
}
```

PortalTextureSetup

- 傳送門上顯示另一個相機在相對位置上的攝影畫面，(好像有另一個人站在另外一個傳送門那邊一樣)

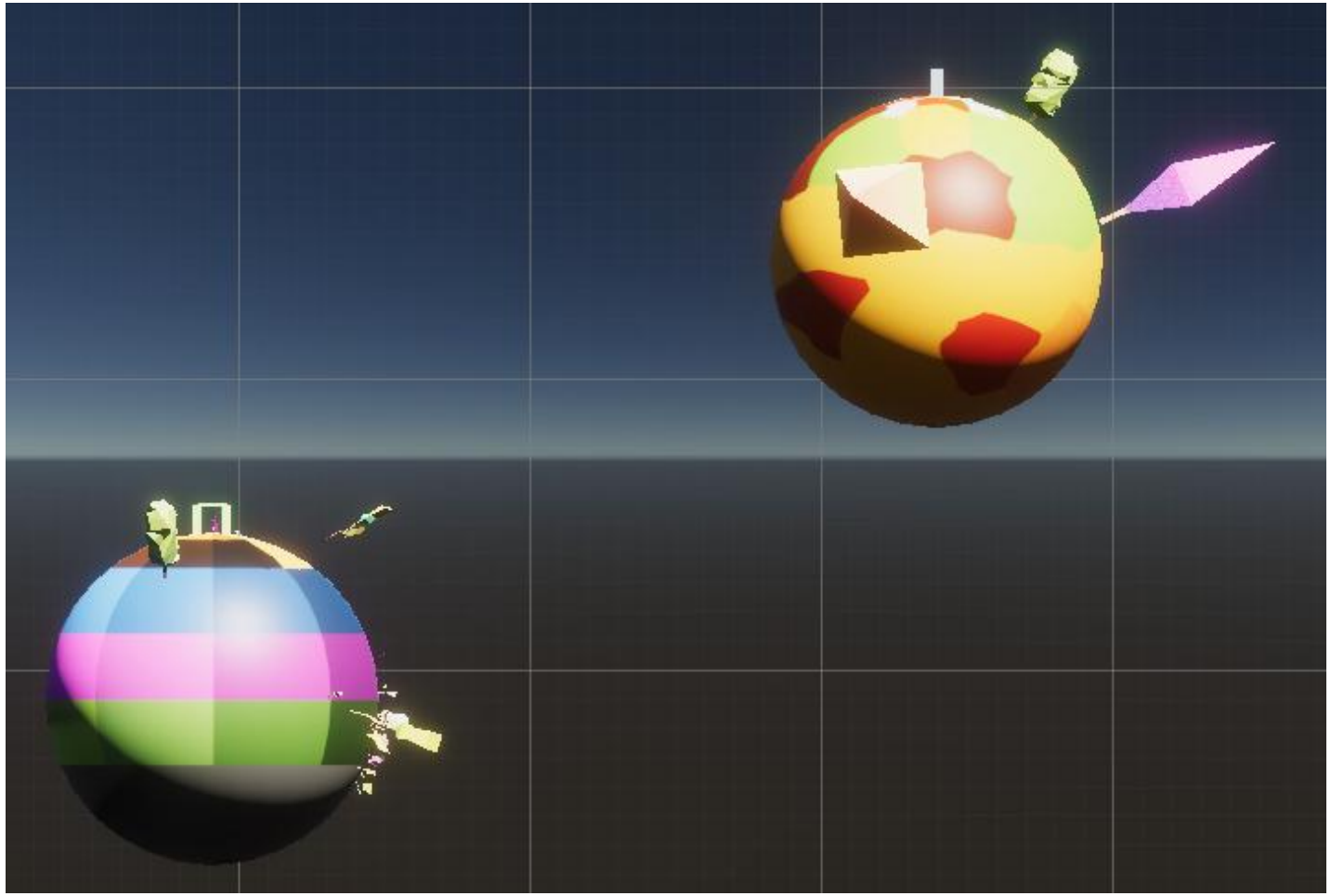
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PortalTextureSetup : MonoBehaviour
{
    public Camera cameraB;
    public Camera cameraA;
    public Material cameraMatB;
    public Material cameraMatA;
    void Start()
    {
        if (cameraA.targetTexture != null)
        {
            cameraA.targetTexture.Release();
        }
        cameraA.targetTexture = new RenderTexture(Screen.width, Screen.height, 24);
        cameraMatA.mainTexture = cameraA.targetTexture;

        if (cameraB.targetTexture != null)
        {
            cameraB.targetTexture.Release();
        }
        cameraB.targetTexture = new RenderTexture(Screen.width,Screen.height,24);
        cameraMatB.mainTexture = cameraB.targetTexture;
    }
}
```

星球重力



星球重力：優點

- **優點一**：如果用星球做為每個場景，就不用作開放世界的邊界了
- **優點二**：傳送門的作法目前跟重力沒關係，將玩家傳送到另一個星球後才又重新計算重力



Code- 星球重力

計算球體的重力

設定玩家現在的重力來源
(ex: A星球 or B星球)

控制玩家的重力的改變(隨著在球體走動上的重力方向改變)

FauxGravityAttractor

算出球體的重力

```
public void Attract(Transform body) {
    Vector3 gravityUp = (body.position - transform.position).normalized;
    Vector3 bodyUp = body.up;
    body.GetComponent<Rigidbody>().AddForce(gravityUp * gravity);

    Quaternion targetRotation = Quaternion.FromToRotation(bodyUp, gravityUp) * body.rotation;
    body.rotation = Quaternion.Slerp(body.rotation, targetRotation, 50 * Time.deltaTime); //球面线性差值
}
```

FauxGravityBody

update 傳送過後
給予重量的星體

```
public FauxGravityAttractor attractor;
private Transform myTransform;
// Start is called before the first frame update
void Start()
{
    transform.GetComponent<Rigidbody>().constraints = RigidbodyConstraints.FreezeRotation;
    transform.GetComponent<Rigidbody>().useGravity = false;
    myTransform = transform;
}

// Update is called once per frame
void Update()
{
    attractor.Attract(myTransform);
}
```

PlayerController

玩家的重力

```
private Animator animator;
public float moveSpeed = 15;
private Vector3 moveDir;
// 在圓圓的世界
void Start()
{
    animator = GetComponent<Animator>();
}

// 在圓圓的世界
void Update()
{
    moveDir = new Vector3(Input.GetAxisRaw("Horizontal"), 0, Input.GetAxisRaw("Vertical")).normalized;
}
// 在圓圓的世界
private void FixedUpdate()
{
    transform.GetComponent<Rigidbody>().MovePosition(transform.GetComponent<Rigidbody>().position +
                                                     transform.TransformDirection(moveDir) * moveSpeed * Time.deltaTime);
}
```

(隨著玩家跳起，物體的shader會消失)

https://youtube.com/shorts/psBMFnie6fY?feature=share

Code

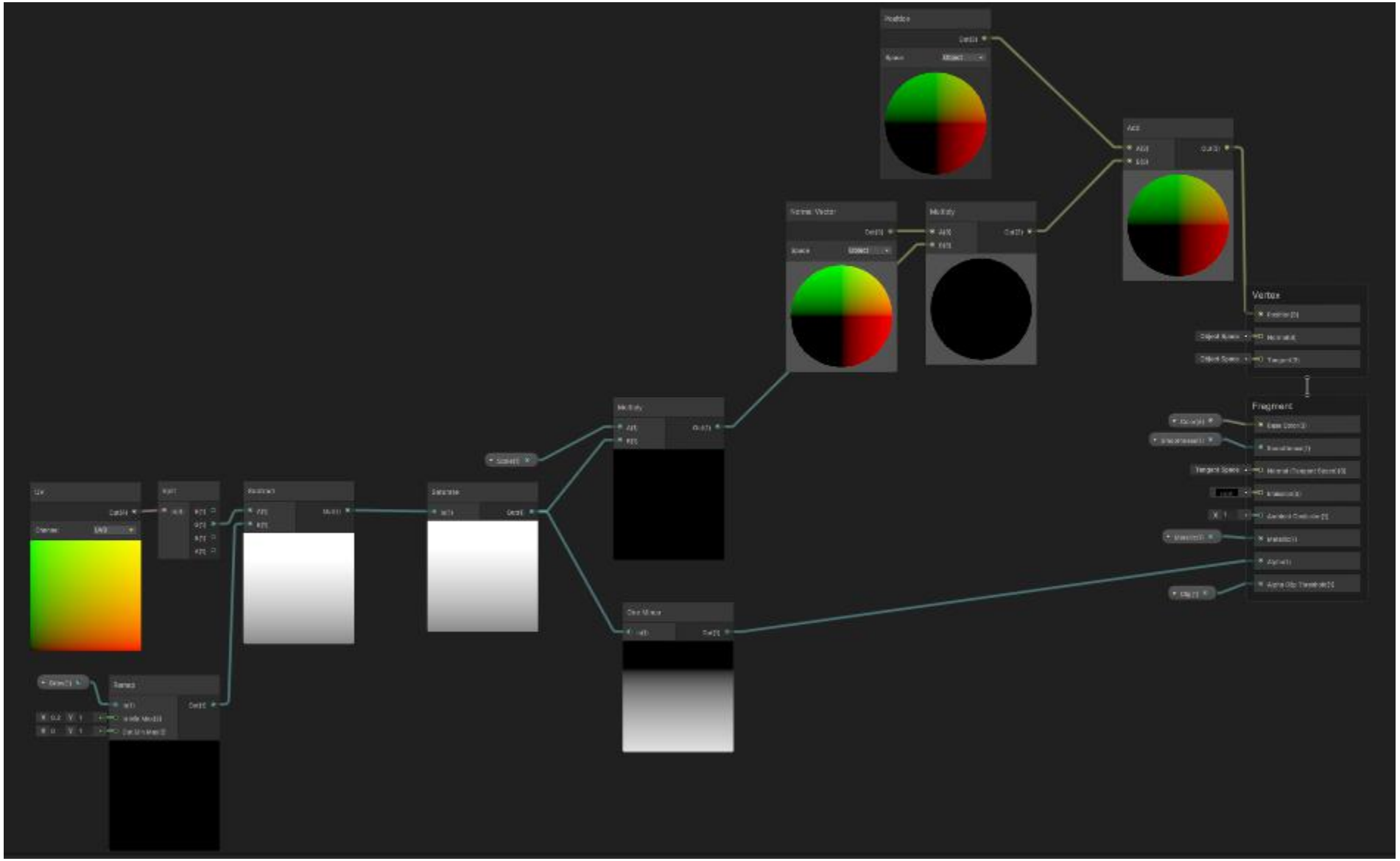
物體shader會消失

玩家跳起後物體就會慢慢隱形

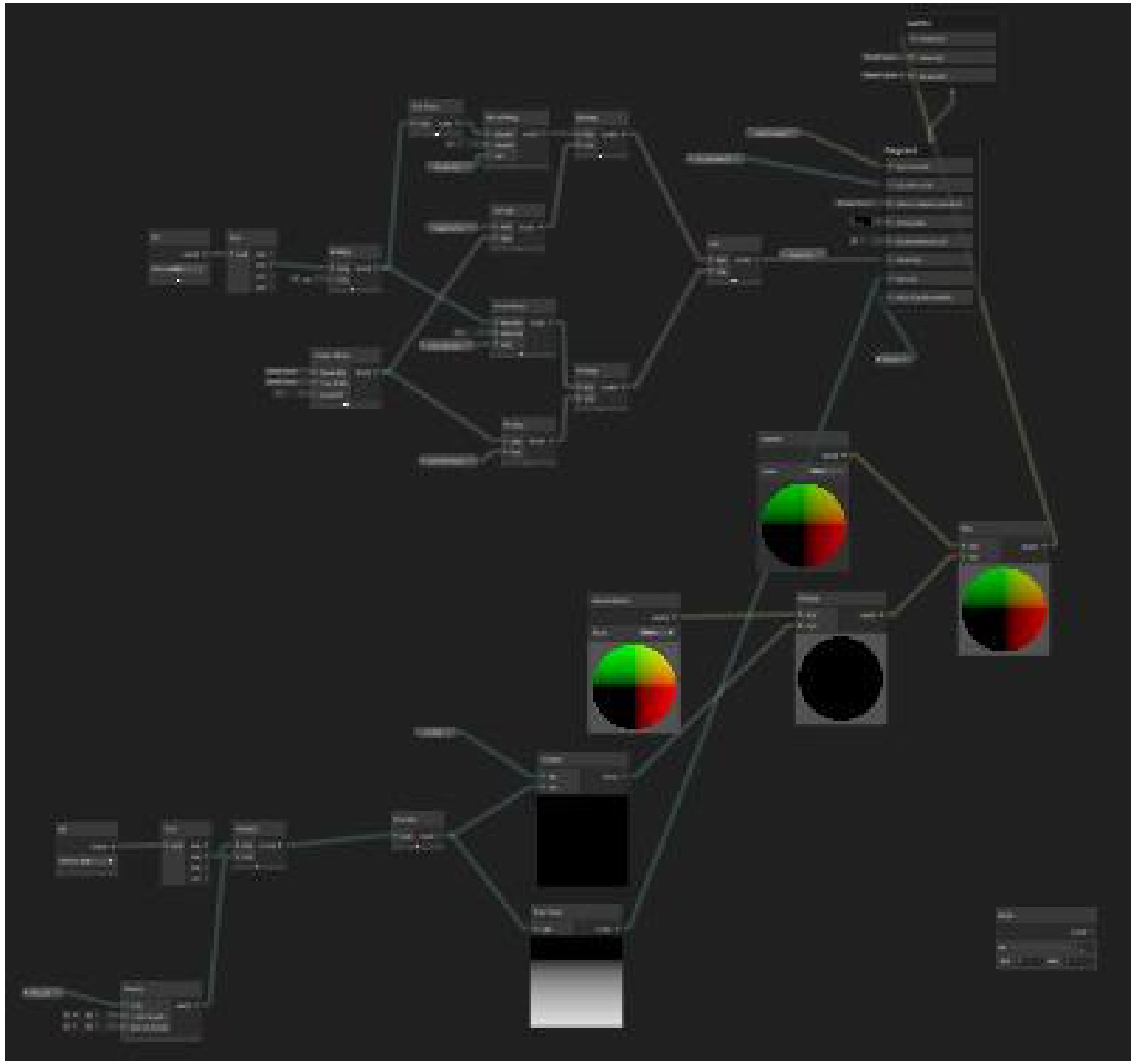
```
public List<MeshRenderer> growVinesMeshes;
public float timeToGrow = 5;
public float refreshRate = 0.05f;
[Range(0, 1)]
public float minGrow = 0.2f;
[Range(0,1)]
public float maxGrow = 0.97f;

private List<Material> growVinesMaterials = new List<Material>();
private bool fullyGrown;
// Start is called before the first frame update
void Start()
{
    for(int i = 0;i < growVinesMeshes.Count; i++)
    {
        for(int j = 0;j < growVinesMeshes[i].materials.Length; j++)
        {
            if (growVinesMeshes[i].materials[j].HasProperty("Grow_"))
            {
                growVinesMeshes[i].materials[j].SetFloat("Grow_", minGrow);
                growVinesMaterials.Add(growVinesMeshes[i].materials[j]);
            }
        }
    }
}
void Update()
{
    if (Input.GetKeyDown(KeyCode.Space))
    {
        for(int i = 0;i < growVinesMaterials.Count; i++)
        {
            StartCoroutine(GrowVines(growVinesMaterials[i]));
        }
    }
}
```

```
IEnumerator GrowVines(Material mat)
{
    float growValue = mat.GetFloat("Grow_");
    if (!fullyGrown)
    {
        while(growValue < maxGrow)
        {
            growValue += 1 / (timeToGrow / refreshRate);
            mat.SetFloat("Grow_", growValue);
            yield return new WaitForSeconds(refreshRate);
        }
    }
    else
    {
        while (growValue > minGrow)
        {
            growValue -= 1 / (timeToGrow / refreshRate);
            mat.SetFloat("Grow_", growValue);
            yield return new WaitForSeconds(refreshRate);
        }
    }
    if(growValue  >= maxGrow)
    {
        fullyGrown = true;
    }
    else
    {
        fullyGrown = false;
    }
}
```

隱藏shader材質設定



水晶材質



Demo影片 :

傳送門1
https://youtu.be/lu8SfiFLmzw

傳送門2
https://youtu.be/D_VF4Y-bNzA

隱藏物體 :
https://youtube.com/shorts/psBMFnie6fY?feature=share