1　演出を作ろう

Text 用のスクリプト

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

public class SmartBallScript : MonoBehaviour {
    float power = 0f; //ボールを打つ時の強さ
    GameObject[] cubes = new GameObject[5]; //障害物の GameObject を保管する
    Vector3[] moves = new Vector3[5]; //位置を取得
    public Text score; ☆

// Use this for initialization
void Start () {
```

演出用のスクリプト

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

public class SmartBallScript : MonoBehaviour {
    float power = 0f; //ボールを打つ時の強さ
    GameObject[] ob_cubes; //障害物の GameObject を保管する
    GameObject[] goals;
    bool flg = true;
    public Text score;
// Use this for initialization
void Start () {
        ob_cubes = GameObject.FindGameObjectsWithTag("ob_cube");
        goals = GameObject.FindGameObjectsWithTag("goal");
        int n = 0;
        foreach(GameObject obj in goals){
            Renderer renderer = obj.GetComponent<Renderer>();
            renderer.material.SetFloat("_Mode", 3f);
            renderer.material.SetInt("_SrcBlend", (int)BlendMode.SrcAlpha);
            renderer.material.SetInt("_DstBlend", (int)BlendMode.OneMinusSrcAlpha);
            renderer.material.SetInt("_ZWrite", 0);
            renderer.material.DisableKeyword("_ALPHATEST_ON");
            renderer.material.EnableKeyword("_ALPHABLEND_ON");
            renderer.material.DisableKeyword("_ALPHAPREMULTIPLY_ON");
            renderer.material.renderQueue = 3000;
            renderer.material.color = new Color(0f, 0.15f * n, 1f - 0.15f * n++, 0.5f);
        }
        foreach(GameObject obj in ob_cubes){
            Vector3 move = obj.transform.position;
            //アニメーションの設定
            AnimationClip clip = new AnimationClip(); //インスタンスの生成
```

```
            clip.legacy = true;
            Keyframe[] keysX = new Keyframe[2]; //keyframe 配列の生成
            keysX[0] = new Keyframe(0f, move.x - 5); //アニメーション開始の keyframe
            keysX[1] = new Keyframe(1f, move.x + 3); //アニメーション終了の keyframe
            AnimationCurve curveX = new AnimationCurve(keysX); //アニメーションを変化させ
る値を設定
            clip.SetCurve("", typeof(Transform), "localPosition.x", curveX); //アニメー
ションを設定
            clip.wrapMode = WrapMode.PingPong; //アニメーションを設定
            Keyframe[] keysY = new Keyframe[2];
            keysY[0] = new Keyframe(0f, move.y);
            keysY[1] = new Keyframe(1f, move.y);
            AnimationCurve curveY = new AnimationCurve(keysY);
            clip.SetCurve("", typeof(Transform), "localPosition.y", curveY);
            Keyframe[] keysZ = new Keyframe[2];
            keysZ[0] = new Keyframe(0f, move.z);
            keysZ[1] = new Keyframe(1f, move.z);
            AnimationCurve curveZ = new AnimationCurve(keysZ);
            clip.SetCurve("", typeof(Transform), "localPosition.z", curveZ);
            Animation animation = obj.GetComponent<Animation>();
            animation.AddClip(clip, "clip1");
            animation.Play("clip1");
        }
    }

// Update is called once per frame
void Update () {
        Rigidbody rigidbody = GetComponent<Rigidbody>();
        Renderer renderer = GetComponent<Renderer>();

        MoveCube();

        rigidbody.AddForce(0f, 0f, -1f);
        //ボールを打ち出す
        if (flg)
        {
            if (Input.GetKey(KeyCode.Space))
            {
                power += 0.01f;
                if (power > 1f)
                {
                    power = 0.25f;
                }
                renderer.material.color = new Color(1f, power, 0f);
            }
        }
        if (Input.GetKeyUp(KeyCode.Space)){
                rigidbody.AddForce(new Vector3(0f, 0f, power * 3000f));
                power = 0f;
                renderer.material.color = Color.red;
            flg = false;
        }
```

```
}
    //障害物を回転させる
    void MoveCube(){
        foreach(GameObject obj in ob_cubes){
            obj.transform.Rotate(new Vector3(0f, 1f, 0f));
        }
    }

    void OnCollisionEnter(Collision collision){
        if (collision.gameObject.tag == "ob_cube"){
            Behaviour b = (Behaviour)collision.gameObject.GetComponent("Halo");
            b.enabled = true;
        }
    }

    void OnCollisionExit(Collision collision){
        Rigidbody rigidbody = GetComponent<Rigidbody>();
        if(collision.gameObject.tag == "ob_cube"){
            Behaviour b = (Behaviour)collision.gameObject.GetComponent("Halo");
            b.enabled = false;
            Vector3 v = rigidbody.velocity;
            if(v.magnitude < 15){
                v *= 2.0f;
                if(v.magnitude < 5){
                    v *= 2.0f;
                }
                rigidbody.velocity = v;
            }
        }
    if(collision.gameObject.tag == "ob_wall"){
            Vector3 v = rigidbody.velocity;
            if(v.magnitude < 15){
                v *= 2.0f;
                if(v.magnitude < 5){
                    v *= 2.0f;
                }
                rigidbody.velocity = v;
            }
        }
    }

    void OnTriggerEnter(Collider collider){
        Rigidbody rigidbody = GetComponent<Rigidbody>();
        if(collider.gameObject.tag == "goal"){
            rigidbody.velocity = Vector3.zero;
            rigidbody.angularVelocity = Vector3.zero;
            int n = 1;
            foreach(GameObject obj in goals){
                if(obj == collider.gameObject){
                    score.text = "point:" + (n * 100);
                    ParticleSystem ps = collider.gameObject.GetComponent<ParticleSystem>();
                    ps.Play();
                }
```

```
                n++;
            }
        }
    }

}
```