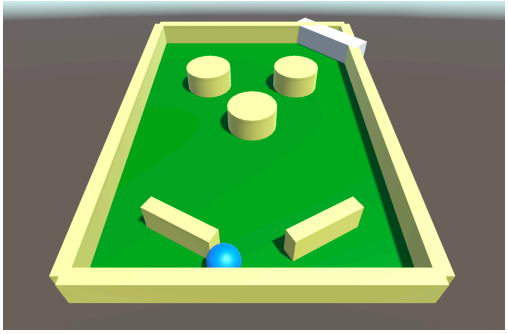
ゲームプログラミング

第7回 水野慎士

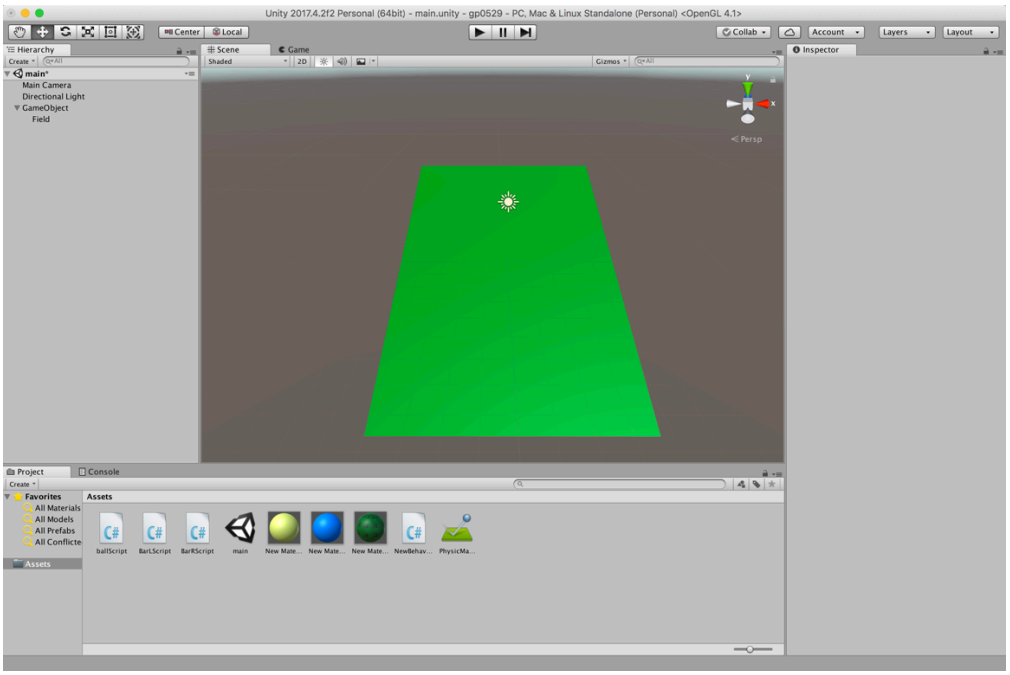
今日の目標

- ピンボールを作ってみる
  - キーボード操作でボール射出
  - キーボード操作でフリッパー操作
  - フリッパーの角度でボールの跳ね返り方向変化
  - ジェットバンパーがボールを弾き飛ばし
  - ボールの動きでエフェクト発生
- 必要な主な要素(スクリプト)
  - キーボード操作の受付（押す／離す）
  - オブジェクトの方向の検出
  - オブジェクトへの力の付与
  - エフェクトの基礎



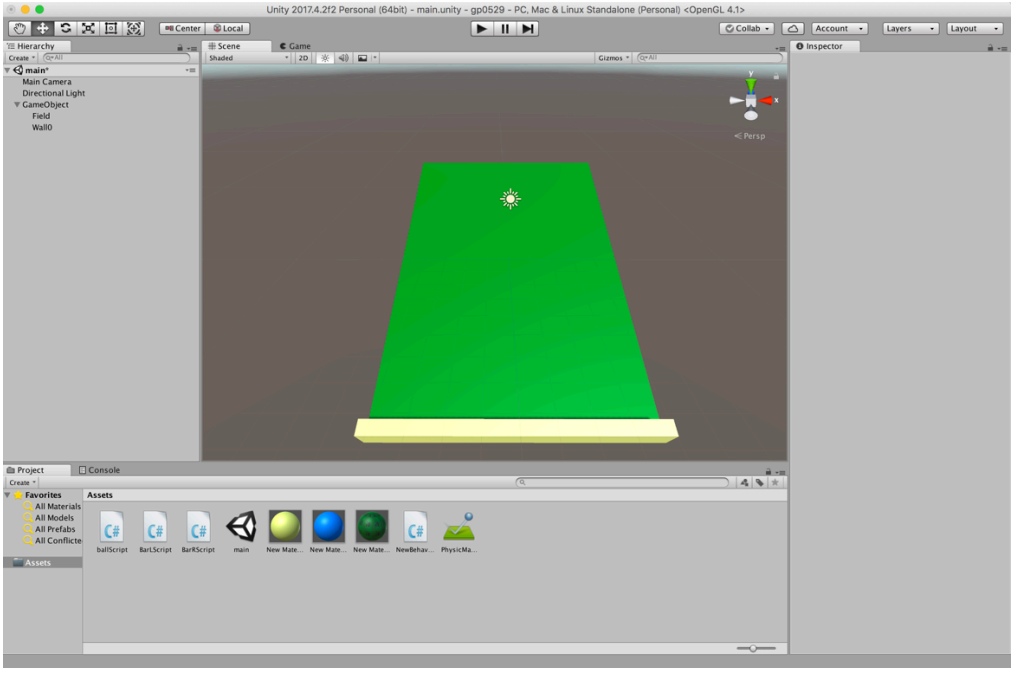
オブジェクト(盤面)の追加

- GameObject → 3D Object → Plane
  - 名前を”Field”に変更
  - scale(1, 1, 1.5)



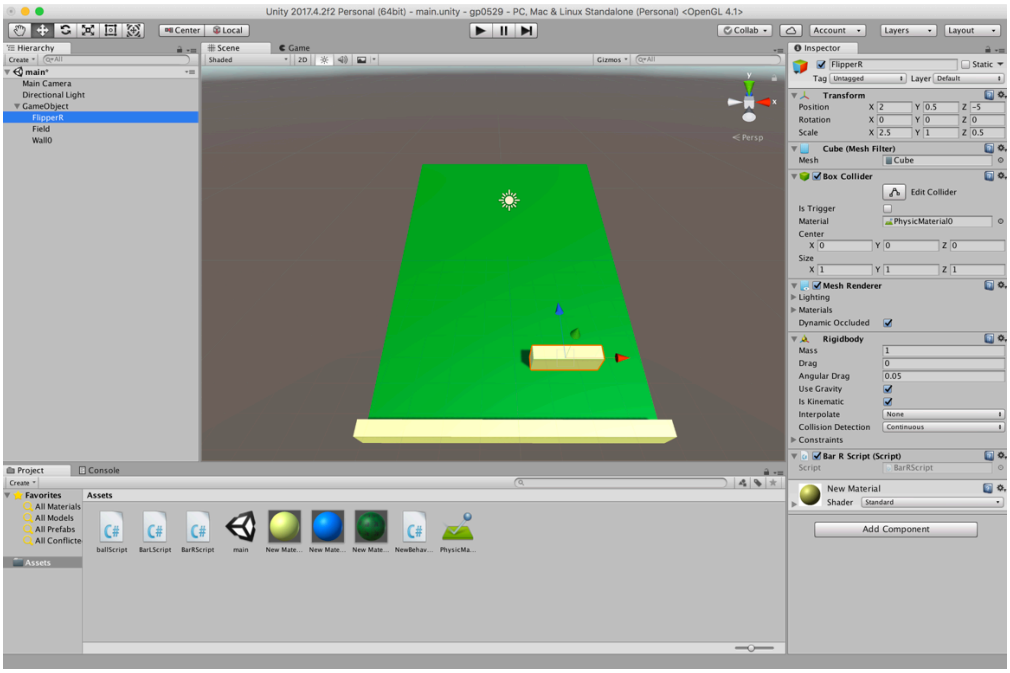
オブジェクト(壁)の追加

- GameObject → 3D Object → Cube
  - 名前を”Wall0”に変更
  - Position(0, 0.5, -7.5), Scale(10, 1, 0.5)



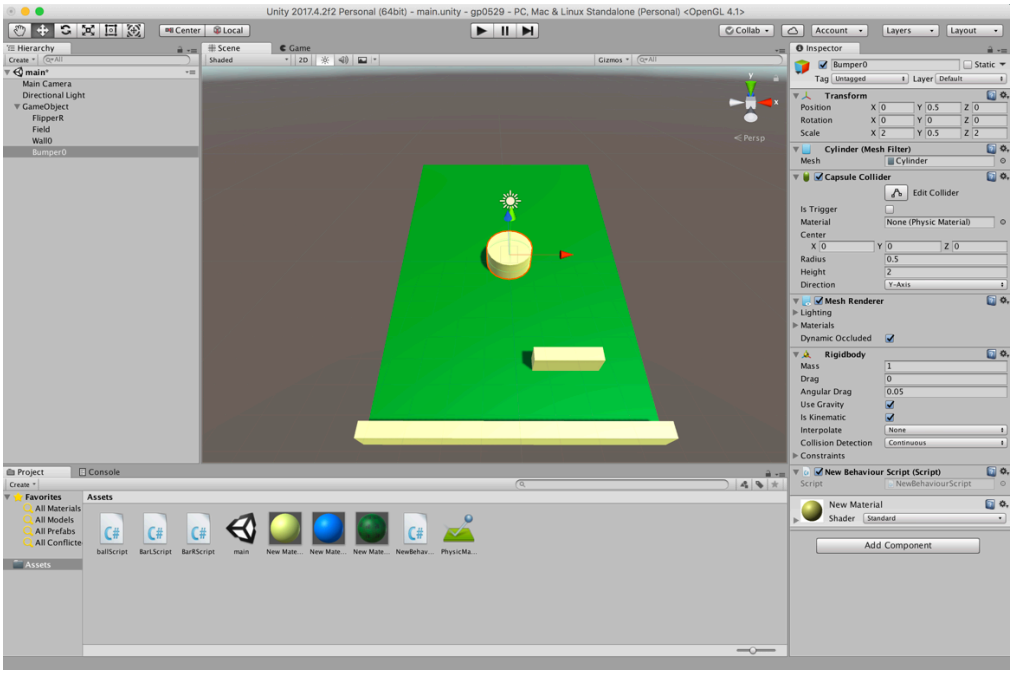
オブジェクト(フリッパー)の追加

- GameObject → 3D Object → Cube
  - 名前を”FlipperR”に変更
  - Position(2, 0.5, -5), Scale(2.5, 1, 0.5)



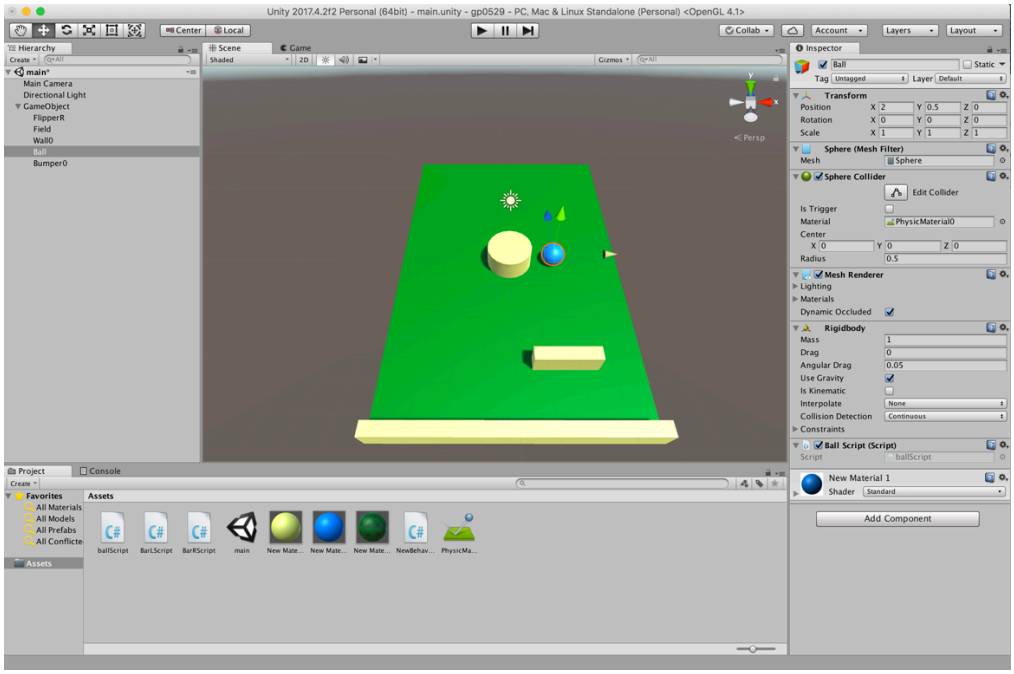
オブジェクト(ジェットバンパー)の追加

- GameObject → 3D Object → Cylinder
  - 名前を”Bumper0”に変更
  - Position(0, 0.5, 0), Scale(2, 0.5, 2)



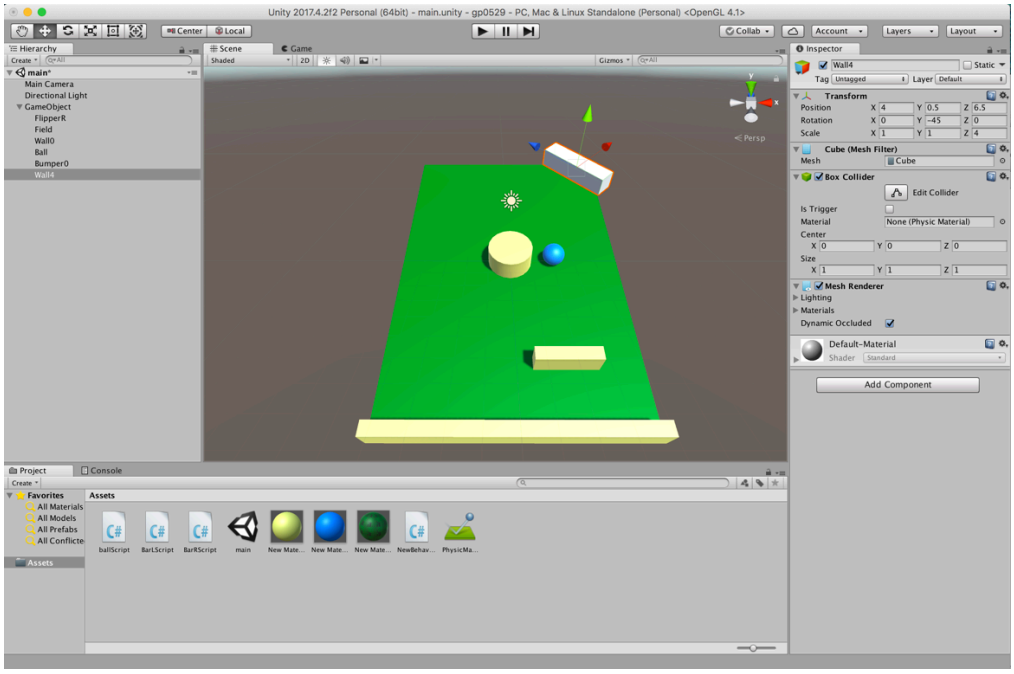
オブジェクト(ボール)の追加

- GameObject → 3D Object → Sphere
  - 名前を”Ball”に変更
  - Position(2, 0.5, 0), Scale(1, 1, 1)



オブジェクト(コーナー)の追加

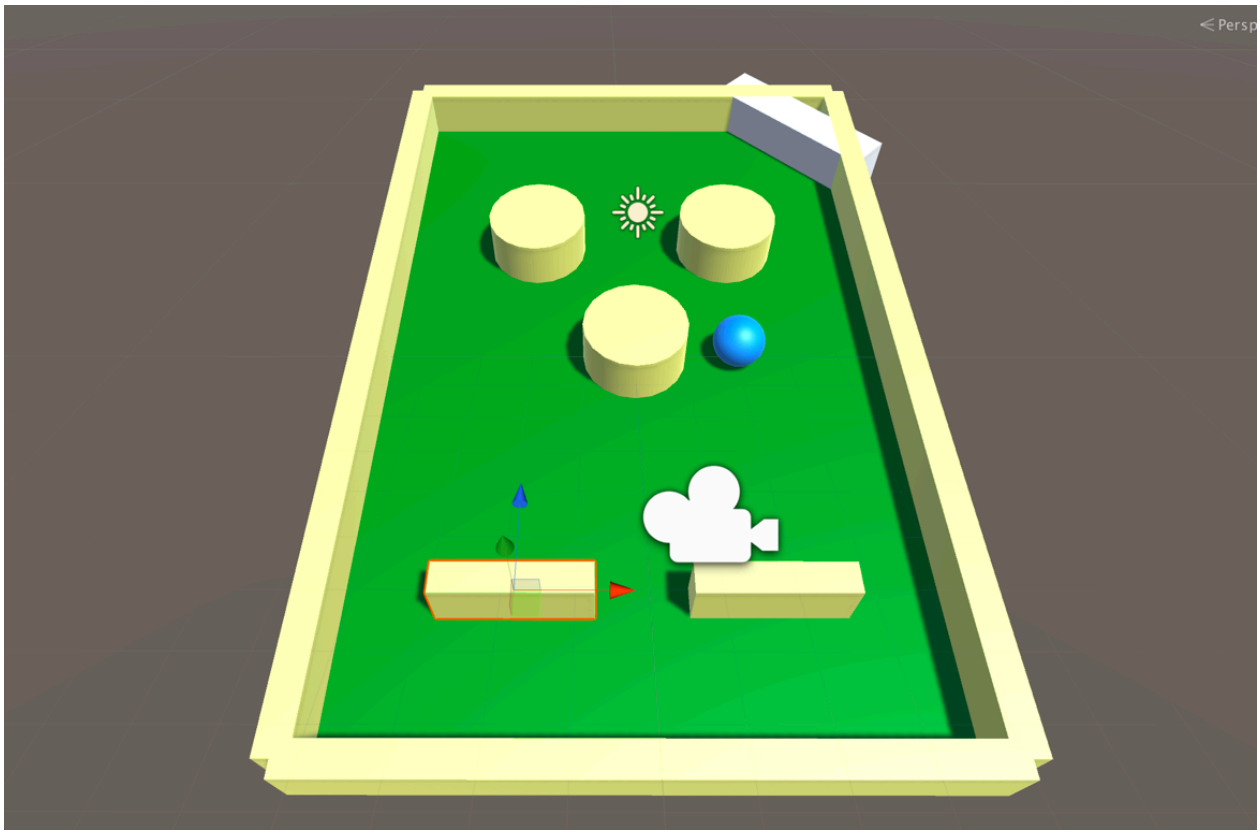
- GameObject → 3D Object → Cube
  - 名前を”Wall4”に変更
  - Position(4, 0.5, 6.5), Scale(1, 1, 4)



“ Wall0” ” FlipperR” ” Bumper0” の複製

- “ Wall0” を複製して” Wall1” ～” Wall3” を作成
  - “ Wall1” :Position:(0, 0.5, 7.5) Scale(10, 1, 0.5)
  - “ Wall2” :Position:(5, 0.5, 0) Scale(0.5, 1, 15)
  - “ Wall3” :Position:(-5, 0.5, 0) Scale(0.5, 1, 15)
- “ FlipperR” を複製して” FlipperL” を作成
  - “ FlipperL” :Position(-2, 0.5, -5) Scale(2.5, 1, 0.5)
- “ Bumper0” を複製して
  ” Bumper1” ” Bumper2” を作成
  - “ Bumper1” :Position(-2, 0.5, 3)
  - “ Bumper2” :Position(2, 0.5, 3)

ピンボール台

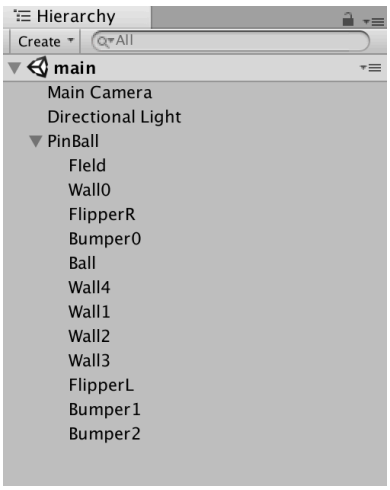


Rigidbodyおよび跳ね返り/摩擦特性の追加

- 作成したすべてのオブジェクトにRigidbody追加
  - Collision Detectionを「Continuous Dynamic」に設定
    - 連続的な衝突判定 → すり抜け防止
  - “ Ball” ：Use Gravity ○, Is Kinematic ×
  - その他： Use Gravity × , Is Kinematic ○
- Assets → Create → Physic Material
  - 名前を”PMaterial0”に変更
  - 作成したすべてのオブジェクトにドラッグして適用
    - Dynamic Friction：0
    - Static Friction：0
    - Bounciness：0.5

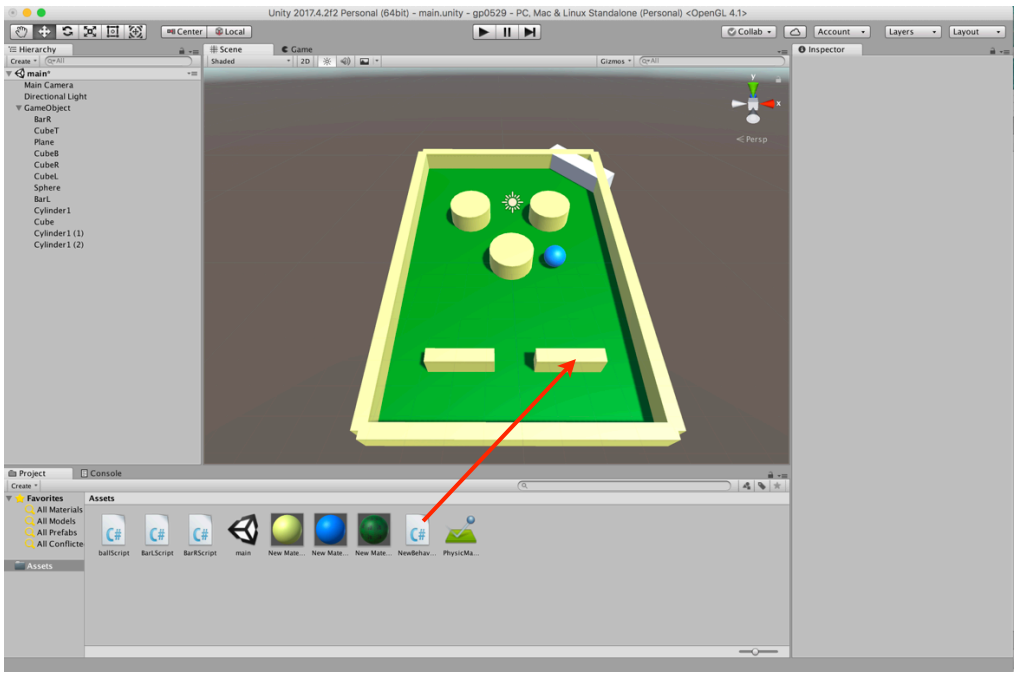
空のオブジェクト”Pinball”追加と親子関係構築

- GameObject → Create Empty
  - 名前を”Pinball”に変更
- 作成したすべてのオブジェクトを”Pinball”にドラッグ
  - 作成したすべてのオブジェクトが”Pinball”の子
- “ Pinball”:Rotation(-20, 0, 0)
  - 全体的に傾く

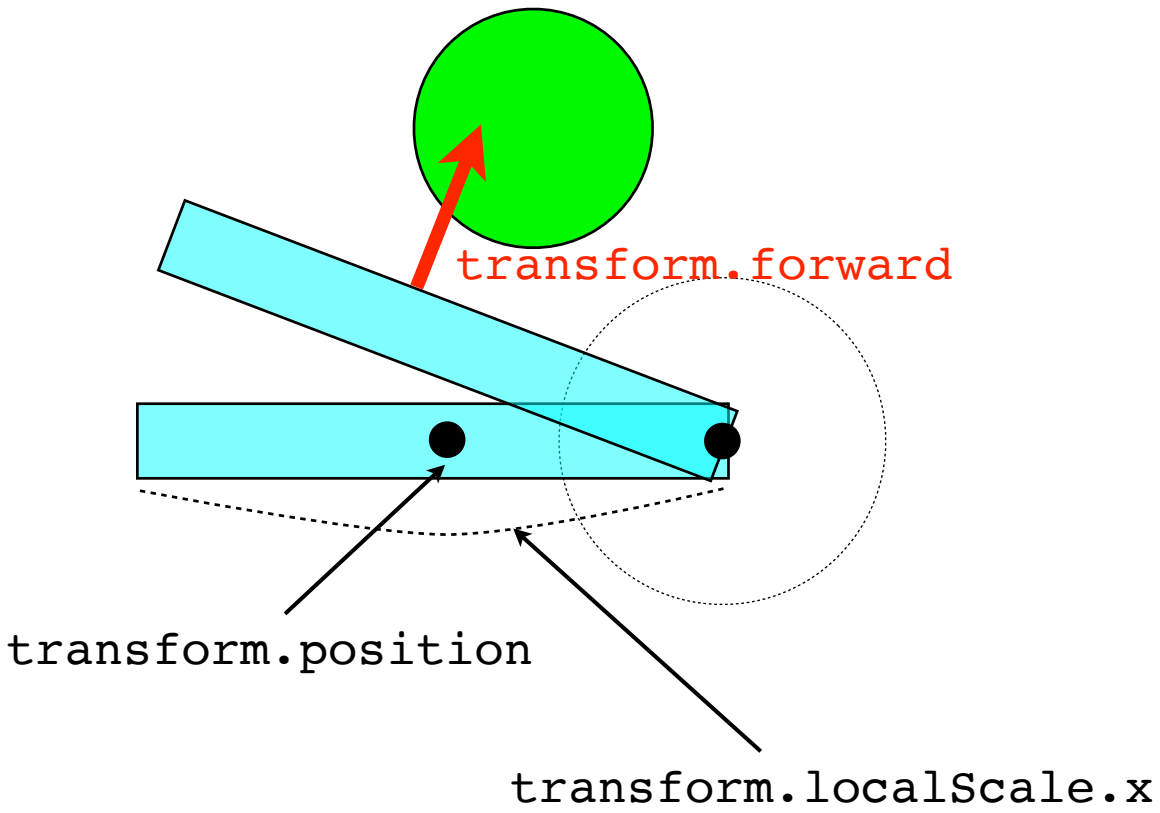


スクリプトの作成とオブジェクトへの適用(2)

- Assets → Create → C# Script
  - 名前を”FlipperR_Control”に変更
  - オブジェクト”FlipperR”にドラッグして適用



フリッパーの動き



スクリプトの編集(フリッパー右操作(1))

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

public class FlipperR_Control : MonoBehaviour {
    Vector3 p, up;
    int flg;

    // Use this for initialization
    void Start () {
        //"FlipperR"の中心位置
        p = transform.position;
        //pを"FlipperR"の横幅の半分だけ右に移動
        p.x += transform.localScale.x * 0.5f;
        //"FlipperR"の上向き方向
        up = transform.up;
    }
```

スクリプトの編集(フリッパー右操作(2))

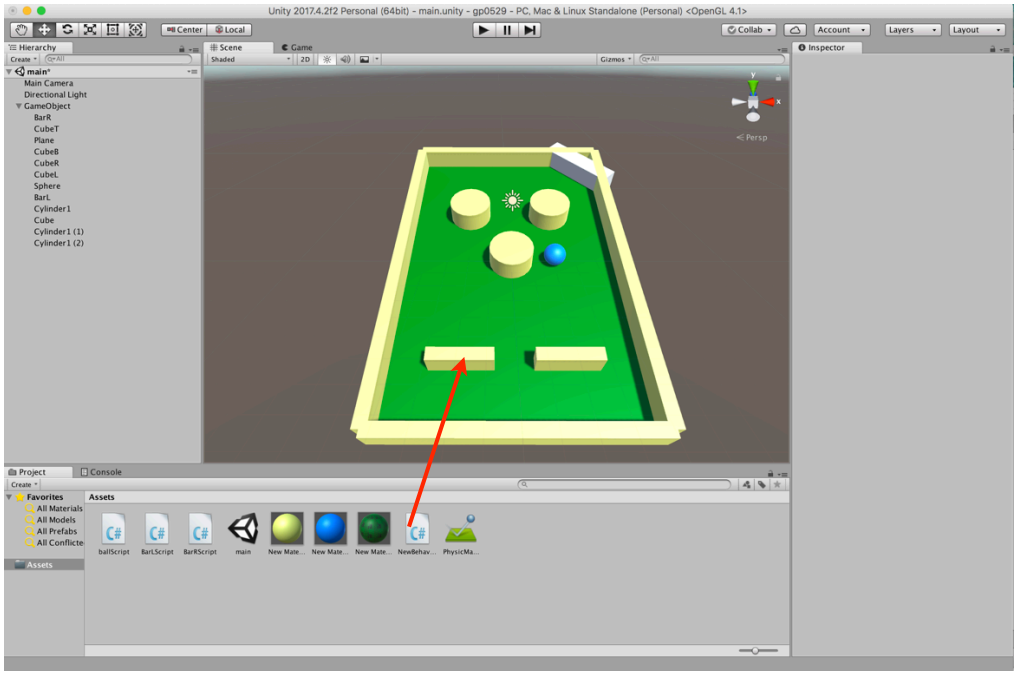
```
    // Update is called once per frame
    void Update () {
        flg = 0;
        //[M]キーが押されている場合
        if (Input.GetKey (KeyCode.M)) {
            //角度が範囲内
            if (transform.localEulerAngles.y < 30 ||
                transform.localEulerAngles.y > 320) {
                //pを中心，upを回転軸にして10度回転
                transform.RotateAround (p, up, 10);
                //flgを1に変更
                flg = 1;
            }
        //角度が範囲内
        } else if (transform.localEulerAngles.y < 40 ||
                   transform.localEulerAngles.y > 330) {
            //pを中心，upを回転軸にして-2度回転
            transform.RotateAround (p, up, -2);
        }
    }
```

スクリプトの編集(フリッパー右操作(3))

```
    //衝突を検出した場合
    void OnCollisionEnter(Collision collision) {
        //"FlipperR"の前方向
        Vector3 v = transform.forward;

        if (flg==1)   //flgが1，つまりキーが押されている場合
            //衝突検出対象(ボール)に対して"FlipperR"の前方向に力を付与
            collision.gameObject.
            GetComponent<Rigidbody> ().AddForce (v * 1000);
    }
}
```

スクリプトの作成とオブジェクトへの適用(2)

- Assets → Create → C# Script
  - 名前を”FlipperL_Control”に変更
  - オブジェクト”FlipperL”にドラッグして適用



スクリプトの編集(フリッパー左操作(1))

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

public class FlipperL_Control : MonoBehaviour {
    Vector3 p, up;
    int flg;

    // Use this for initialization
    void Start () {
        //"FlipperL"の中心位置
        p = transform.position;
        //pを"FlipperL"の横幅の半分だけ右に移動
        p.x -= transform.localScale.x * 0.5f;
        //"FlipperL"の上向き方向
        up = transform.up;
    }
```

スクリプトの編集(フリッパー左操作(2))

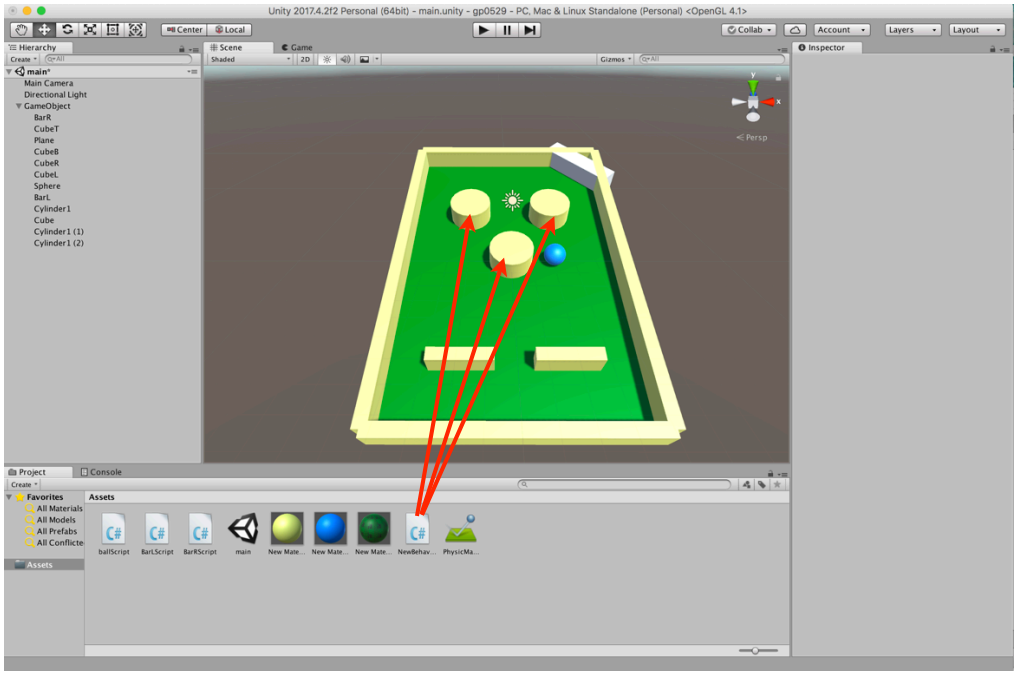
```
    // Update is called once per frame
    void Update () {
        flg = 0;
        //[Z]キーが押されている場合
        if (Input.GetKey (KeyCode.Z)) {
            //角度が範囲内
            if (transform.localEulerAngles.y > 330 ||
                transform.localEulerAngles.y < 40) {
                //pを中心，upを回転軸にして-10度回転
                transform.RotateAround (p, up, -10);
                //flgを1に変更
                flg = 1;
            }
        //角度が範囲内
        } else if (transform.localEulerAngles.y > 320 ||
                   transform.localEulerAngles.y < 30) {
            //pを中心，upを回転軸にして2度回転
            transform.RotateAround (p, up, 2);
        }
    }
```

スクリプトの編集(フリッパー左操作(3))

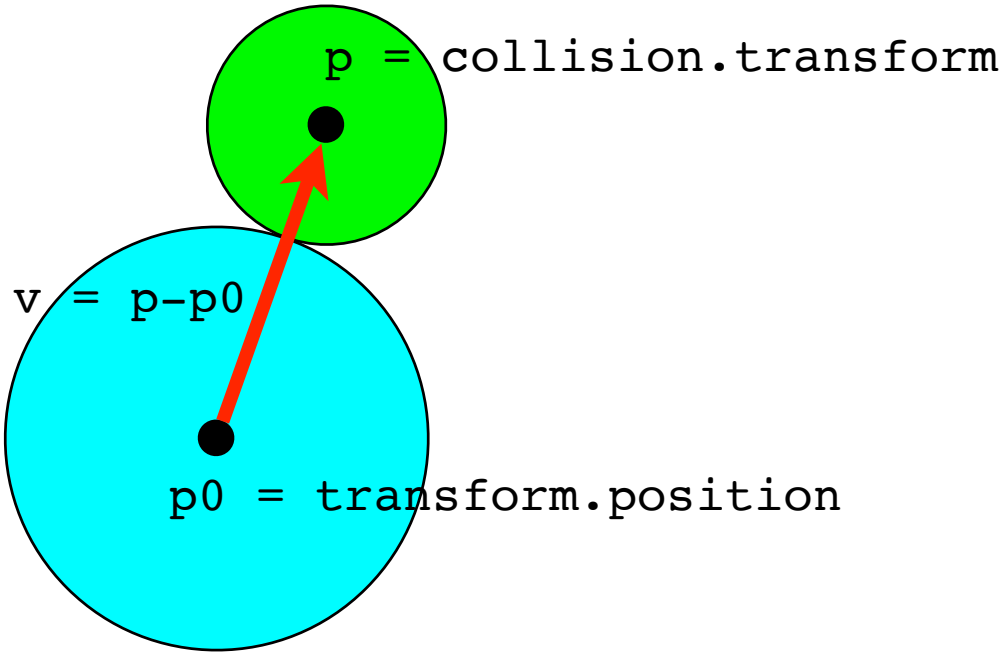
```
    //衝突を検出した場合
    void OnCollisionEnter(Collision collision) {
        //"FlipperL"の前方向
        Vector3 v = transform.forward;

        if (flg==1)  //flgが1，つまりキーが押されている場合
            //衝突検出対象(ボール)に対して"FlipperL"の前方向に力を付与
            collision.gameObject.
            GetComponent<Rigidbody> ().AddForce (v * 1000);
    }
}
```

スクリプトの作成とオブジェクトへの適用(3)

- Assets → Create → C# Script
  - 名前を”Bumper_Control”に変更
  - オブジェクト”Bumper0～2”にドラッグして適用



ジェットバンパーの機能

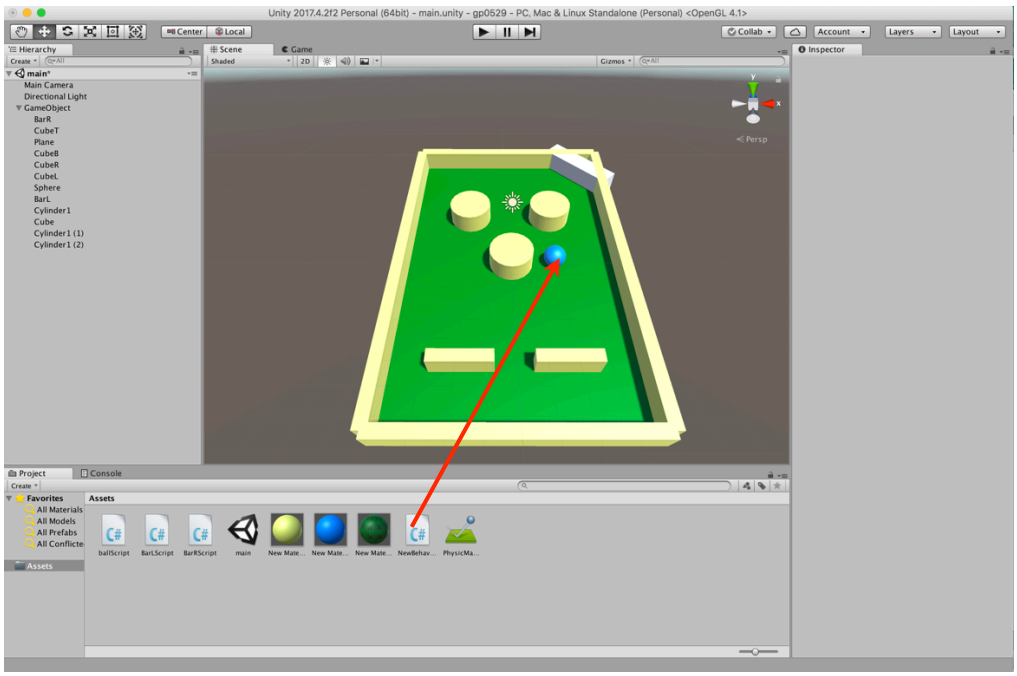


スクリプトの編集(ジェットバンパー)

```
. . . . . .

public class Bumper_Control : MonoBehaviour {
    . . . . . .

    //衝突を検出した場合
    void OnCollisionEnter(Collision collision) {
        //p0はジェットバンパーの位置
        Vector3 p0 = transform.position;
        //pは衝突検出対象(ボール)の位置
        Vector3 p = collision.transform.position;
        //vはp0からpへ向かうベクトル
        Vector3 v = p-p0;
        //衝突検出対象(ボール)にv方向の力を付与
        collision.gameObject.
        GetComponent<Rigidbody>().AddForce (v*200);
    }
}
```

スクリプトの作成とオブジェクトへの適用(4)

- Assets → Create → C# Script
  - 名前を”Ball_Control”に変更
  - オブジェクト”Ball”にドラッグして適用



スクリプトの編集(ボール)

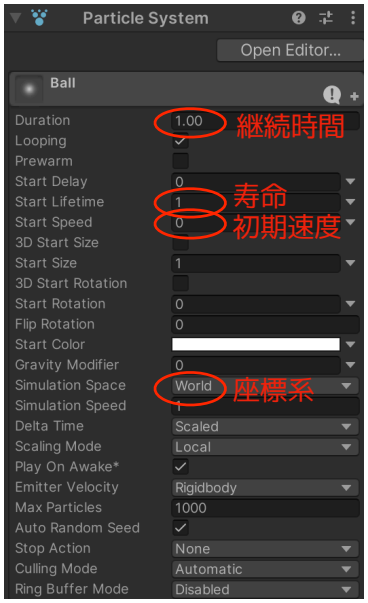
```
public class Ball_Control : MonoBehaviour {
    float pow;  //パワー用
    GameObject obj;  //ゲームオブジェクト格納用

    // Use this for initialization
    void Start () {
        pow = 0f;  //初期値
        //”Field”という名のゲームオブジェクト発見
        obj = GameObject.Find(“Field”);
    }

    . . . . . .

}
```

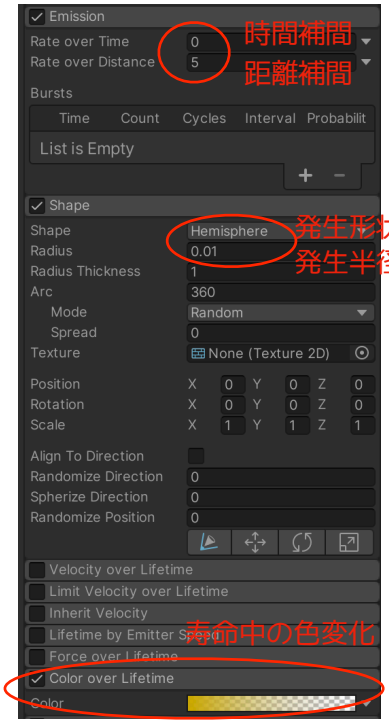
スクリプトの編集(ボール)

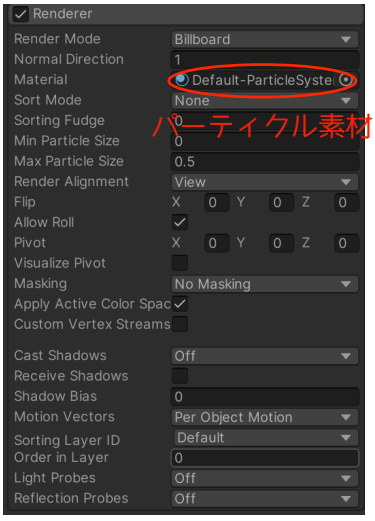
```
public class Ball_Control : MonoBehaviour {

    . . . . . .

    // Update is called once per frame
    void Update () {
        //スペースキーを押したとき
        if (Input.GetKey (KeyCode.Space)) {
            pow += 10f;   //powを加算
        }
        //スペースキーを離したとき
        if (Input.GetKeyUp (KeyCode.Space)) {
            Vector3 v = obj.transform.forward;   //ボール前方向
            GetComponent<Rigidbody>().AddForce (v * pow);
                                                  //ボールに力付与
            pow = 0f;
        }
    }
}
```

ボールへのエフェクト追加

- “ Ball”を選択してから，Inspectorで
  Add Component→ Effects → Particle System





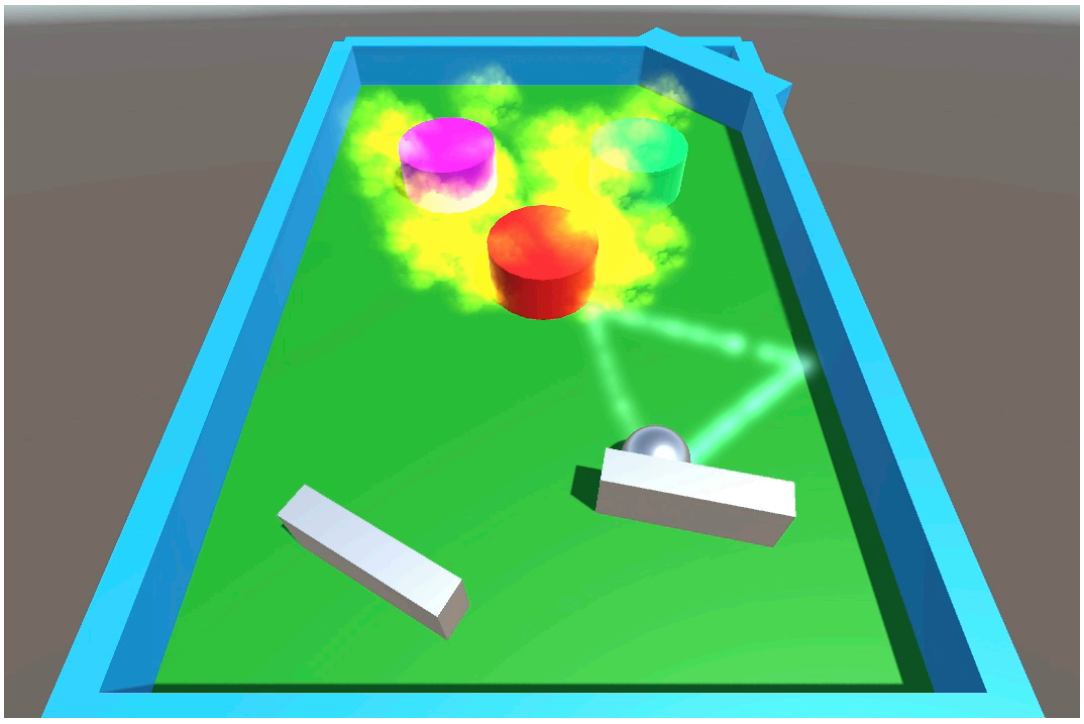


スクリプトの編集(ボール速度でパーティクル発生)

```
public class Ball_Control : MonoBehaviour {

    . . . . . .

   // Update is called once per frame
   void Update () {

      . . . . . .

     //ボール速度が20より大きい場合
     if(GetComponent<Rigidbody>().velocity.magnitude>20){
          //パーティクル再生開始
          GetComponent<ParticleSystem> ().Play ();
     }
}
```

※Play On Awake，Loopingはオフ

エフェクト発生例



課題

- 授業で実施した内容をベースにして，オリジナルのピンボールゲームを作成しなさい．エフェクトやサウンドを適切に使用すること．
  プレイ動画を提出しなさい．