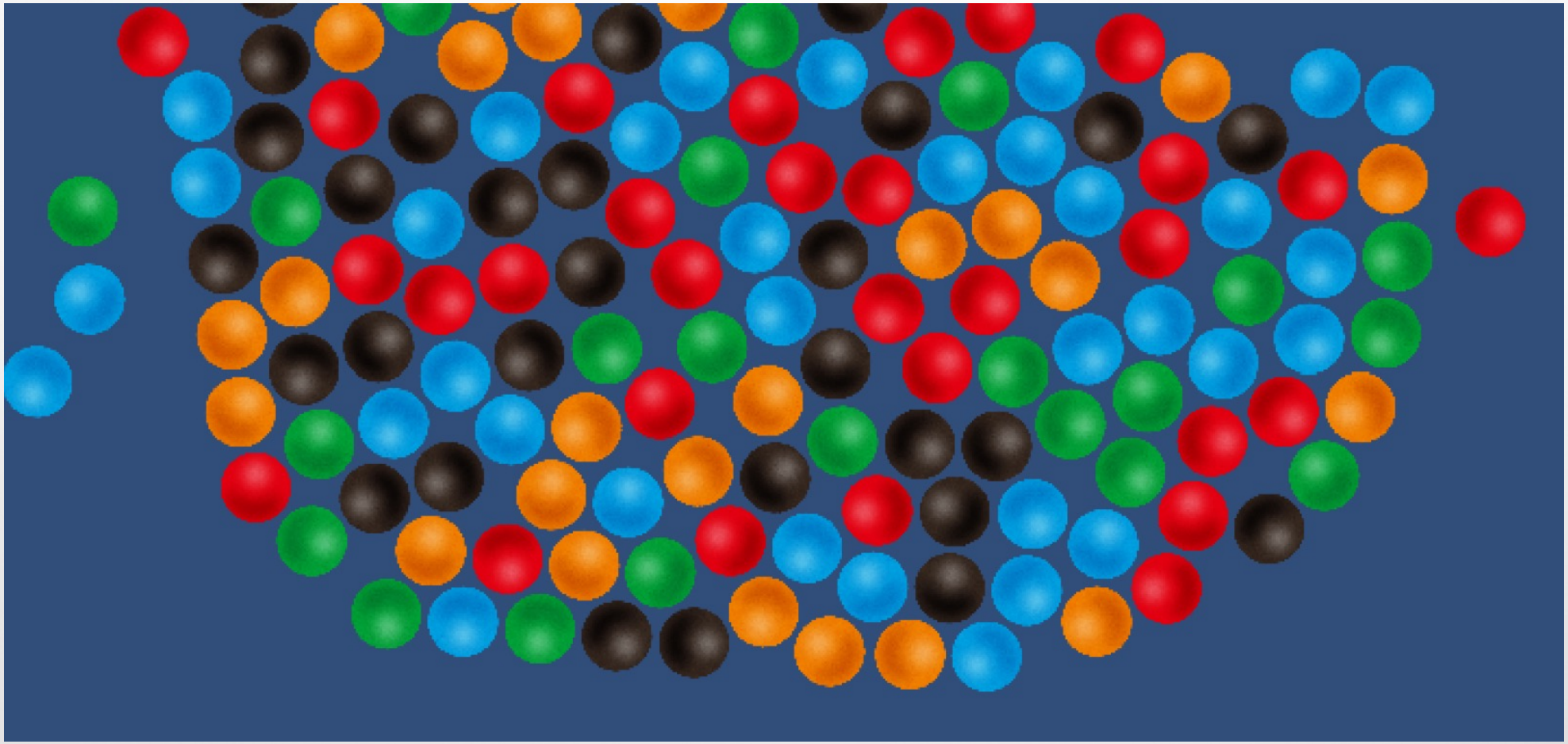
2Dゲーム作成

目次

0．制作するゲームの紹介/覚えて欲しい事
1．ボール生成について
2．ボールのランダム表示、識別化
3．クリック(タッチ)判定
4．ボール削除判定
5．距離制限、種類判定、最低消去数の設定
6．ボールの補給、拡大化

０．制作するゲームの紹介/覚えて欲しい事



０．制作するゲームの紹介/覚えて欲しい事

＜覚えて欲しいこと＞

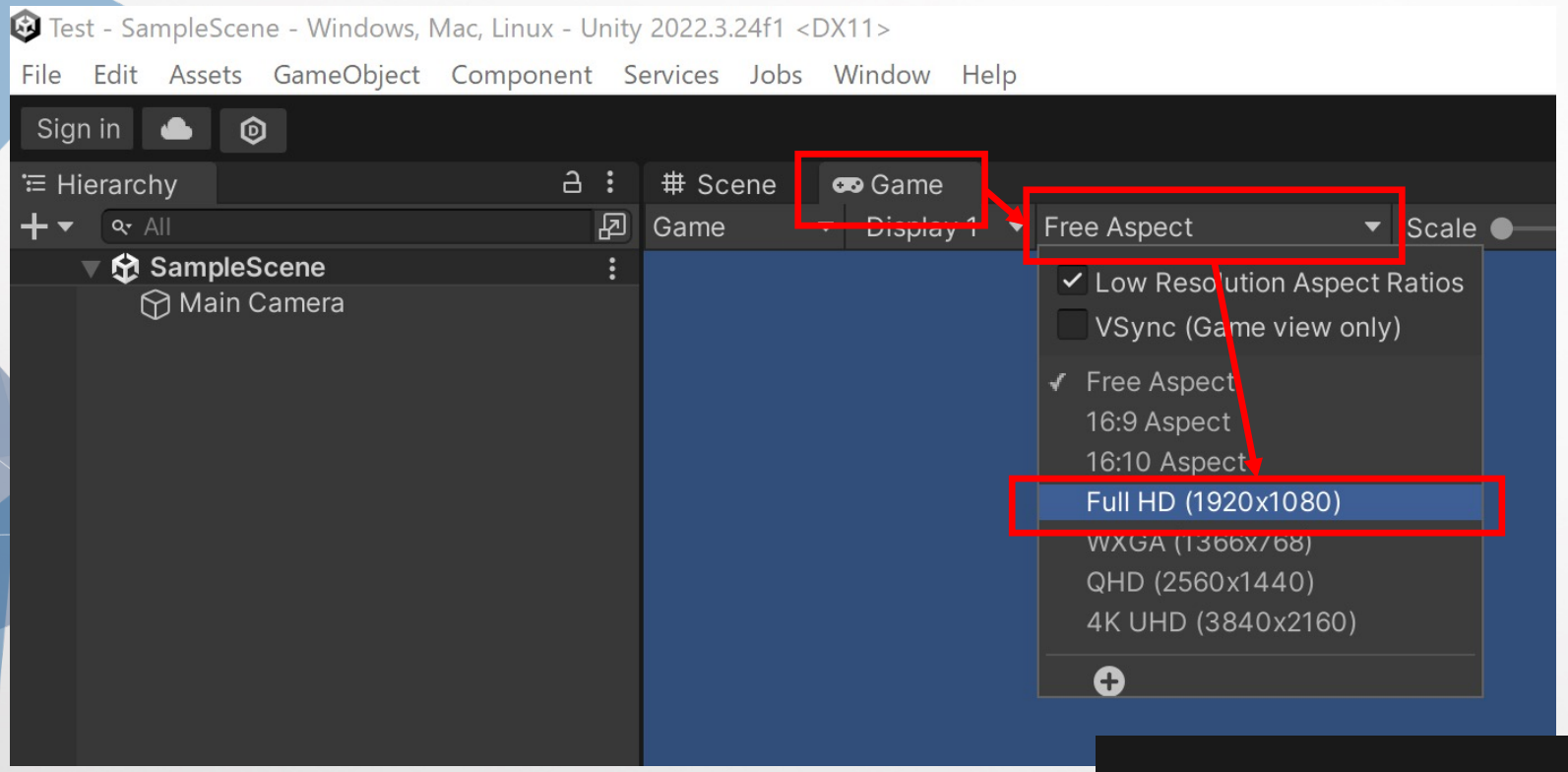
・配列の理解、使い方

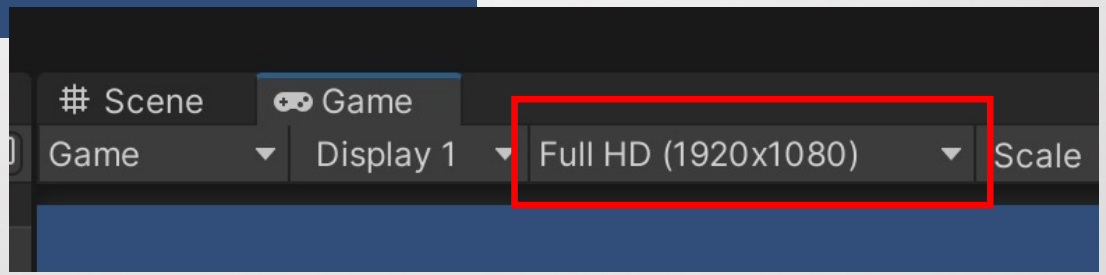
・Listの理解、使い方

・座標変換の仕組み

0．前準備

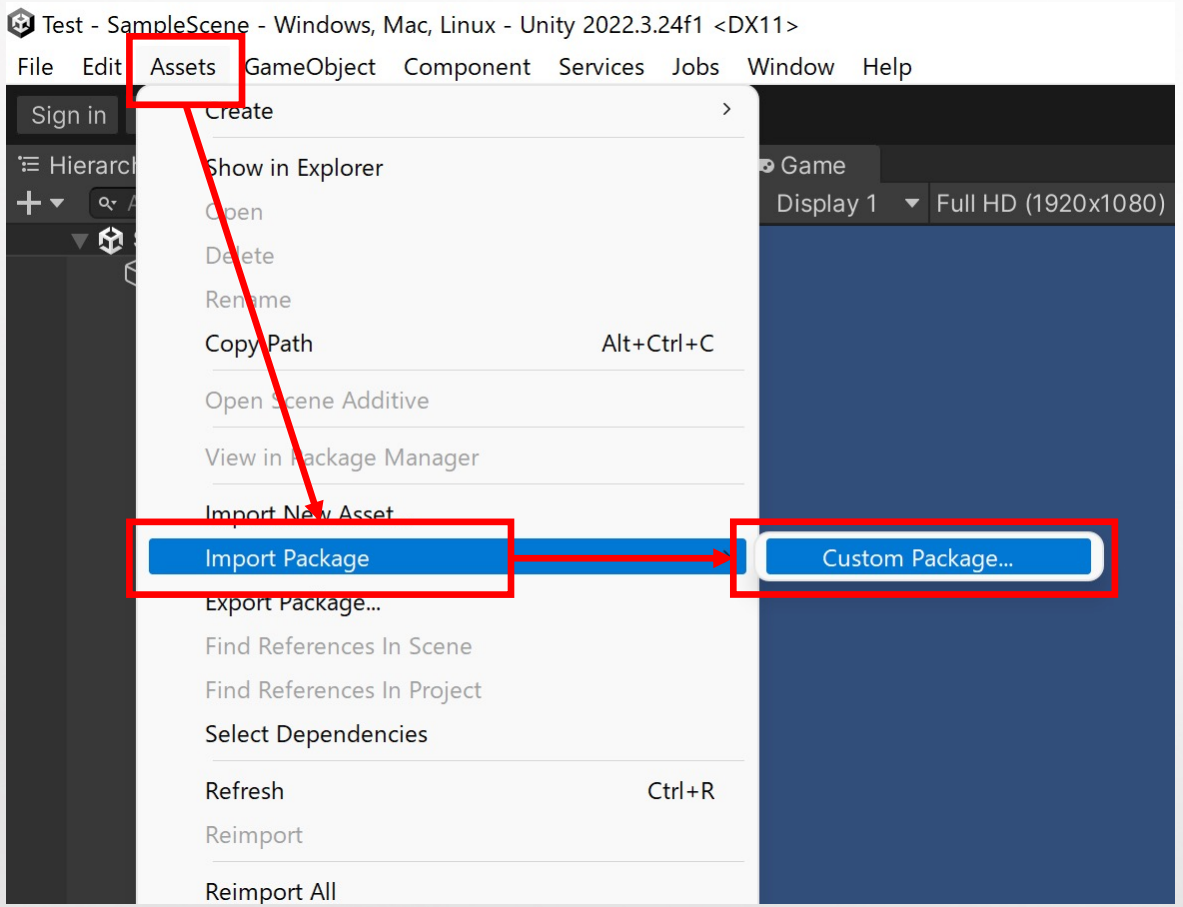
プロジェクト開いた人は、以下の操作をお願いします。(画面の解像度の設定)





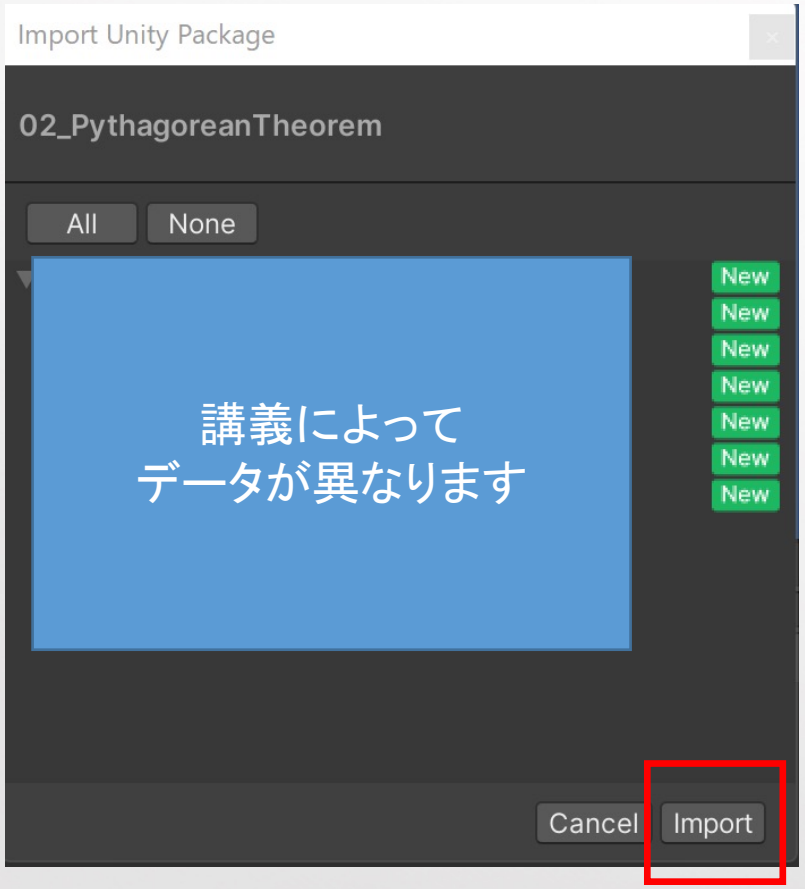
０．前準備

次に、unitypackageというパッケージファイル(データが詰め込まれたファイル)を読み込みます。



０．前準備

ファイルを指定し、以下の画面が出たらImportを押すと、ファイルが展開されます。



１．ボール生成について

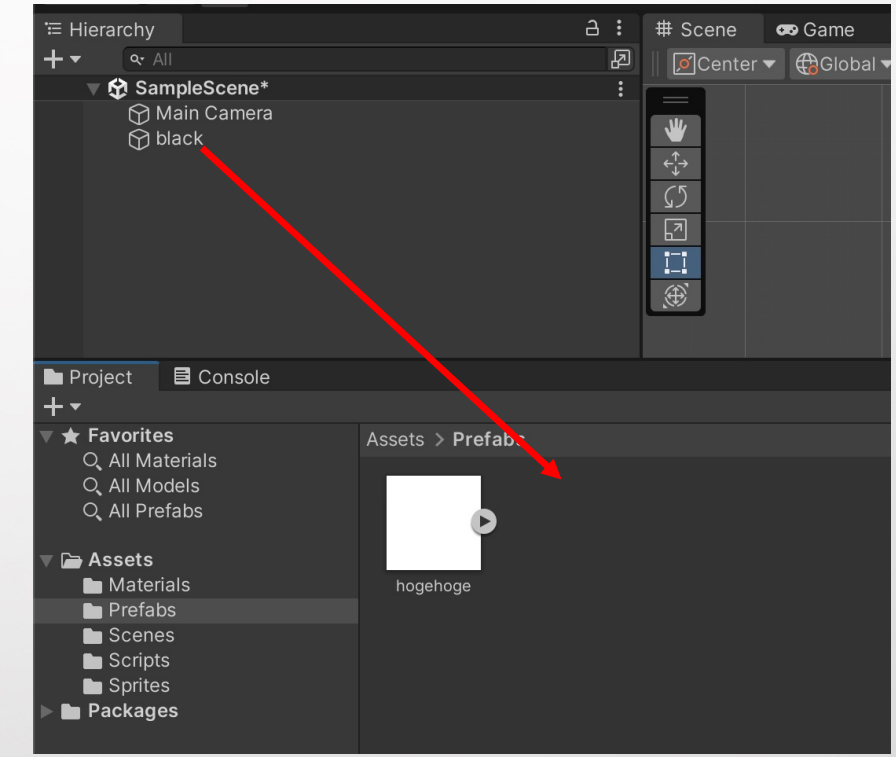
以下の手順で画面上でボールを落とすところまで進めてください。

１：ボールの画像選択、画面に配置(画像はblackでお願いします)

２：Rigidbody2D、Circle Colider2Dを付与

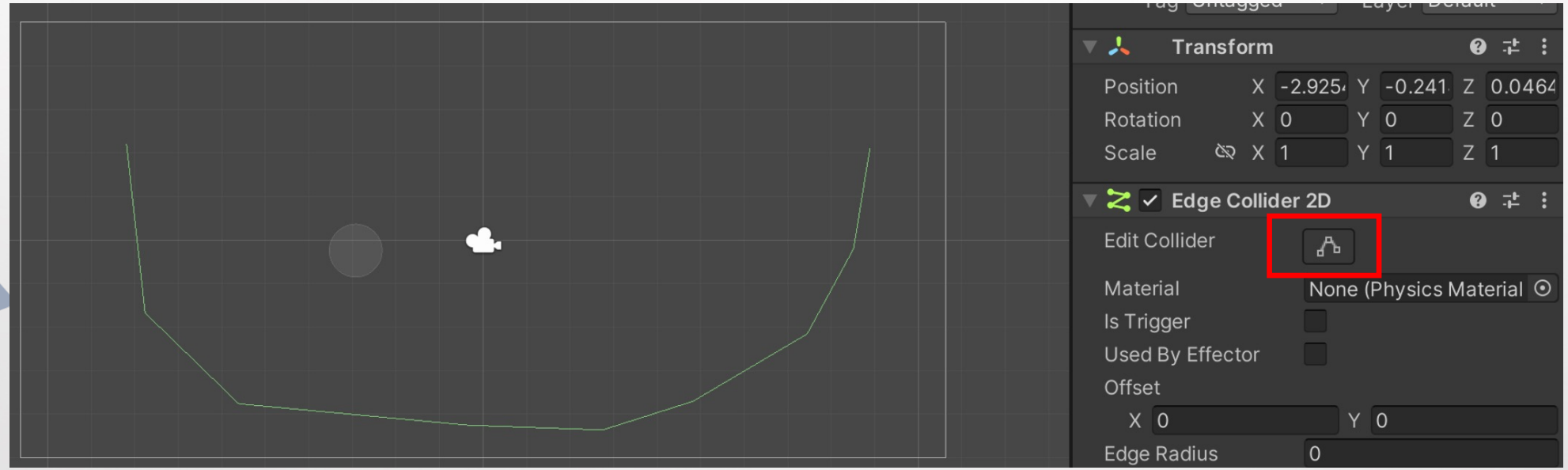
３：実行した場合、ボールが落ちるか確認する

４：ボールのPrefab化
(Prefabフォルダに格納)



１．ボール生成について

ボールの受け皿を作成します。
空のゲームオブジェクトを作成し、Edge Colider2Dを付与、Edit Colider
より自由な形でコライダーを作成していく。実行してボールが乗ることを確認。



出来たら、再生ボタンでコライダーが有効になっているか確認してください。

１．ボール生成について

Scriptsのフォルダ内にあるGameManagerを開いてください。
Startから呼ばれているSpawnsメソッドがあると思います。
Spawnsのメソッドからコルーチンで呼んでいるCreateBallの中で以下の処理を追記してください。

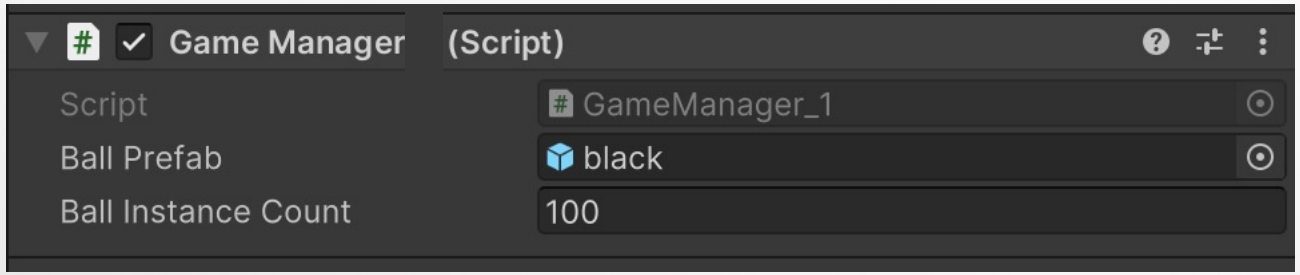
```
//ボールを0.04秒ごとに生成する。作成する数は引数のcount分。
//生成する際、X軸は+0.2から-0.2の範囲、Y軸は8固定とする
//角度の指定が出てきた場合は、Quaternion.identity(回転しない)でOK
```

yield return nullは消して作業をお願いします。

```
//↓これは消してください(何か戻り値を返さないとエラーになるため)
yield return null;
```

２．ボールのランダム表示、識別化

出来た人はヒエラルキー上で空のGameObjectを生成、GameManagerと命名してください。
BallPrefabには、先ほど作成したblackのPrefabを設定お願いします。
BallInstanceCountはボールの個数なので、お好きな値で。



また、もともとヒエラルキーにあるblackは削除してください。

２．ボールのランダム表示、識別化

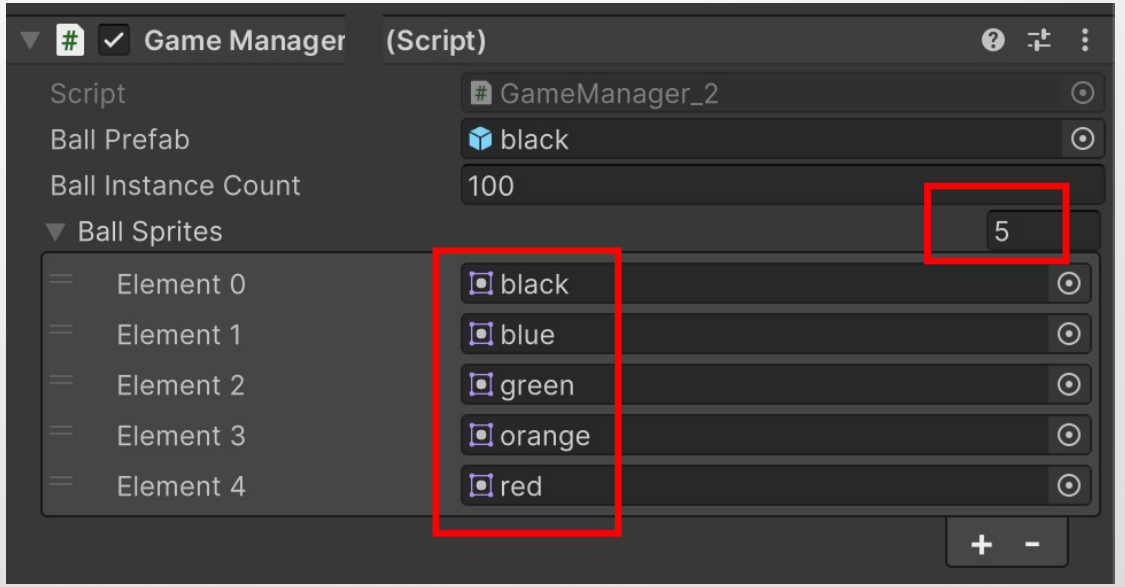
＜ボールのランダム表示＞

スクリプト上での画像の差し替えを実装します、GameManager.csに追記をお願いします。

※自力でボール生成した人は、Instantiateの戻り値を設定してください。

(GameObject ball ＝ の部分です)

出来た人は、BallSpriteのカウントを5に設定し、Spritesのフォルダの画像をアタッチしてください。



２．ボールのランダム表示、識別化

現状、1つ1つのボールにおいて自分が何色なのか識別が出来ていないため、
ボールクラスを作り、そこで番号を識別出来るような仕組みを作成する。
(ボールはランダムで決まるため、GameManagerクラス内で識別番号を
設定出来るようにする)

以下のスクリプトの作成、変更をお願いします。

・Ball.cs(作成後、Prefabのblackにアタッチしてください)
・GameManager.cs

Ball.csみたいに、スクリプト内でStartやUpdateを使わないなら削除しても
大丈夫です！

３．クリック(タッチ)判定

次にクリック判定を実装するので、GameManager.csの変更をお願いします。

＜プログラム解説＞

```
Vector2 mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
RaycastHit2D hit = Physics2D.Raycast(mousePosition, Vector2.zero);
if (hit && hit.collider.GetComponent<Ball>())
{
    Debug.Log("---Ballにhit---");
}
```

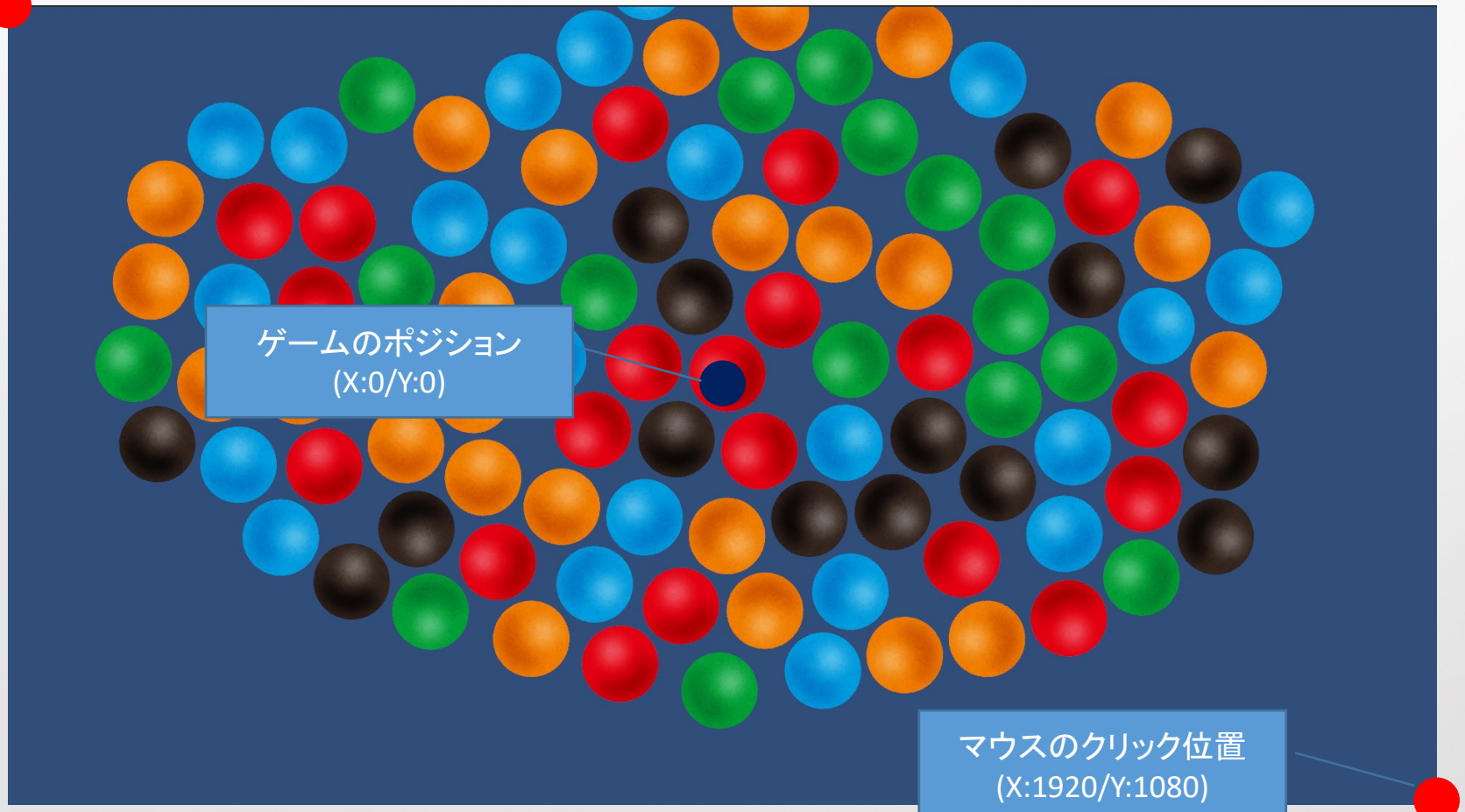
・マウスでクリックした位置を実際のゲーム画面の座標に置き換える
・ゲーム画面の座標に対して、その座標にあるGameObjectを取得する。
・取得したGameObjectに対してBallというコンポーネントが存在するかチェック

3．クリック(タッチ)判定

マウスのクリック位置とゲームのポジションの関係

マウスのクリック位置
(X:0/Y:0)

ここをクリックした場合、
ゲーム上の3D空間の
どこを示しているのか
変換が必要。



４．ボールの削除判定

次に、実際にボールの削除判定を実装します、GameManager.csを編集してください。

＜プログラム解説＞

・OnDragBiginメソッド

→クリックした際のボールを取得する。

・OnDraggingメソッド

→ドラッグ中に触れたボールを取得する。(OnDragBiginとほぼ一緒)

・OnDragEndメソッド

→触れたボールをListに格納、ドラッグ完了後にDestroyする。

この処理があるので、ボールが削除される。

４．ボールの削除判定

＜プログラム解説＞

・AddRemoveBallメソッド

→Contains(既に取得しているGameObjectかを判断)

既に取得していなければListに追加。

→追加したGameObjectはOnDragEndでまとめて消していく。そのために対象のボールを集めている。

現状、今の3点が問題であるため修正していく。

・種類関係なく消せる

・1個でも消せる

・距離関係なく連鎖で消せる

５．距離制限、種類判定、消去数の設定

GameManager.csを編集してください。

＜プログラム解説＞

・距離制限

→Vector2.Distanceで2点の距離を取得

・種類判定

→最後に取得したボールを記録、ドラッグ時には初めに取得したボールと同じIDのものしかとらない。

・消去数

→消すボールはListに格納されているので、格納した個数を参照し、3つ以上だったら消す処理を行う。

６．ボールの補給、拡大化

GameManager内で、以下の処理の作成をお願いします。

・消去した個数分、ボールを生成する。

・選択したボールを拡大して分かりやすくする。