Unity講座
～ブロック崩し②～

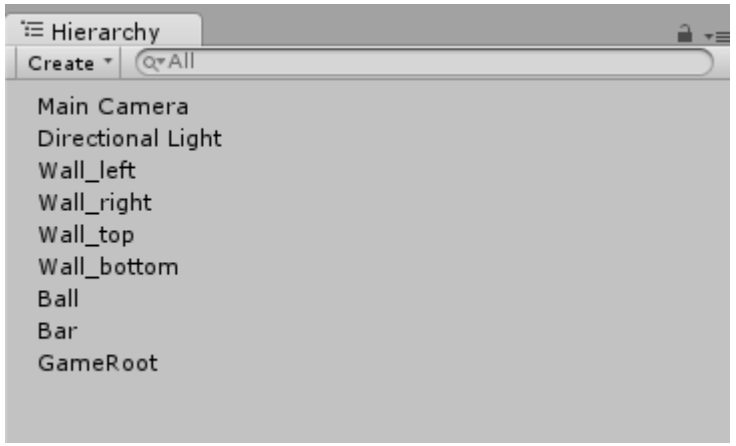
3年　　海苔　威

1. 初めに

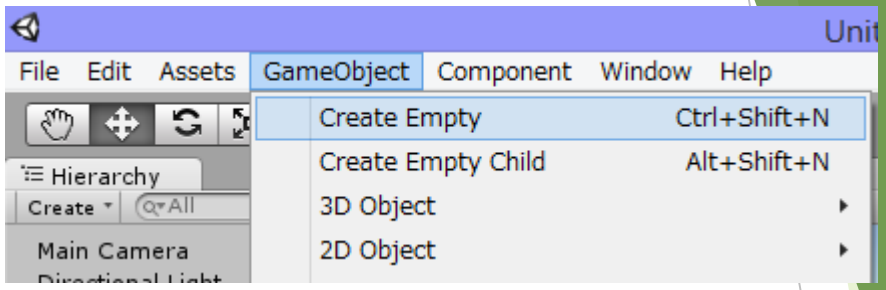
1-1. オブジェクトをまとめる

- いきなり作業もアレなんで
- Hierarcyをきれいにしましょう

- Unityを起動　→　前回のプロジェクトを開く
- Hierarchyを見る
- いろいろ増えて見づらい

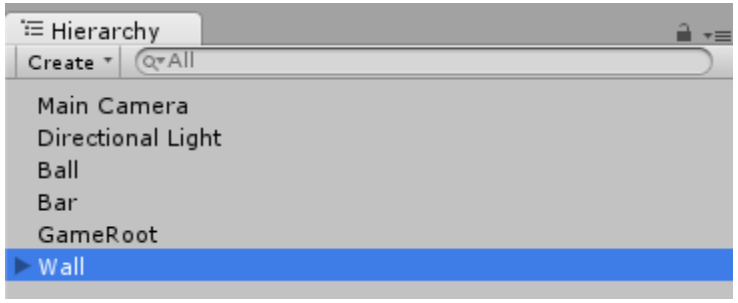


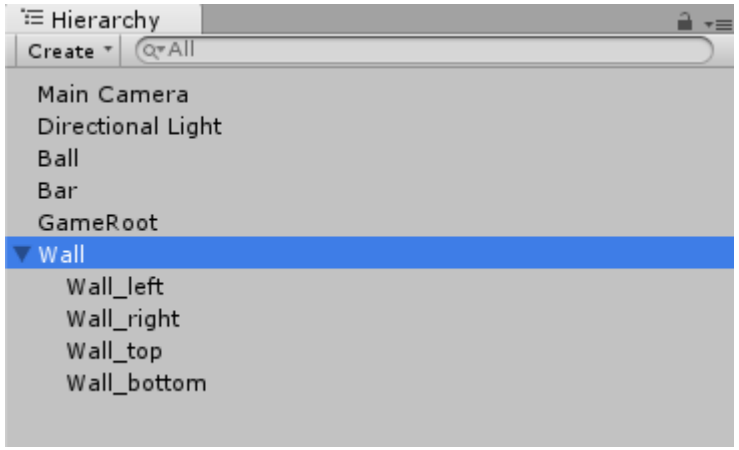
- 「Game Object」→「Create Empty」
- 名前は「Wall」



- 壁を作るオブジェクトを1つにまとめよう
- 作った「Wall」に「Wall_right」「Wall_left」「Wall_top」「Wall_bottom」を入れる

- Hierarchyがすっきり





- Wallを開けば4つのオブジェクトが見える

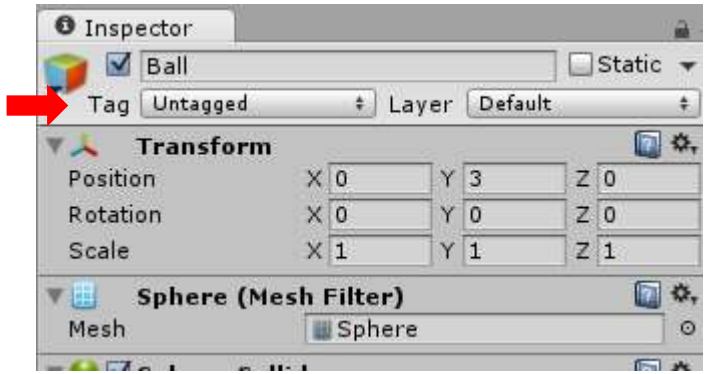
- こんな風にまとめられるものはまとめると良い！

2. スコア処理

2-1. タグを使う

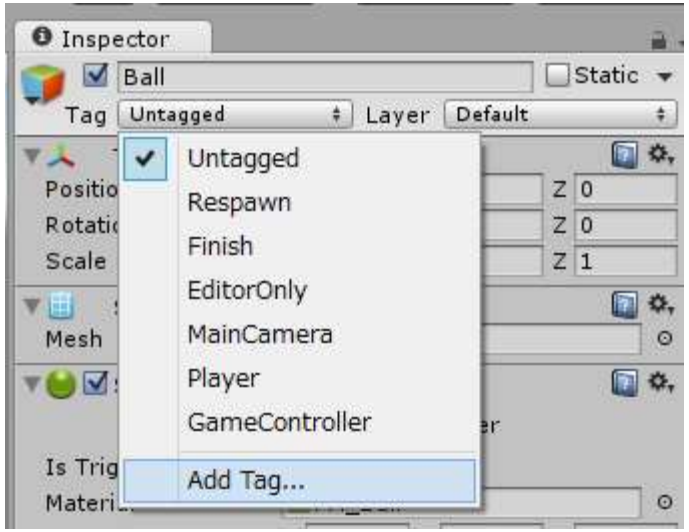
- というわけで本題へ
- 前回ブロックに「Block01」などの名前を付けましたが、これ以外にもブロックを特定する方法があります

- Hierarchyの「Ball」を選択し、Inspectorを見る

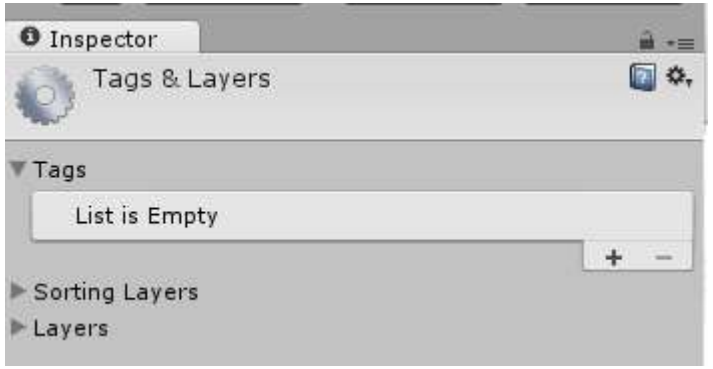


- これがタグ

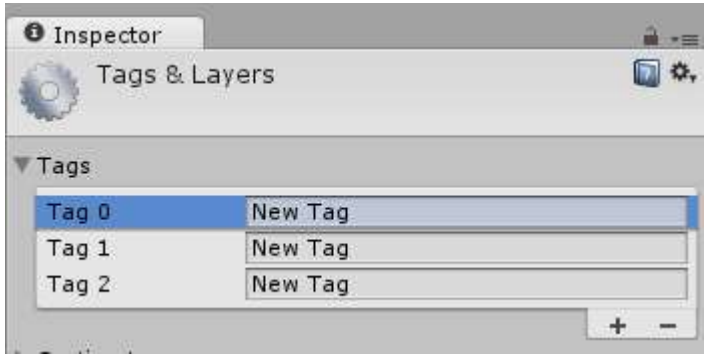
- Untaggedを選び、「Add Tag...」をクリック



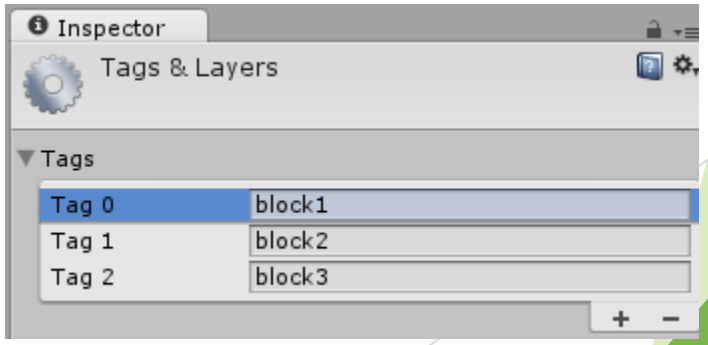
- ↓みたいな画面が出る



- +を3回押す



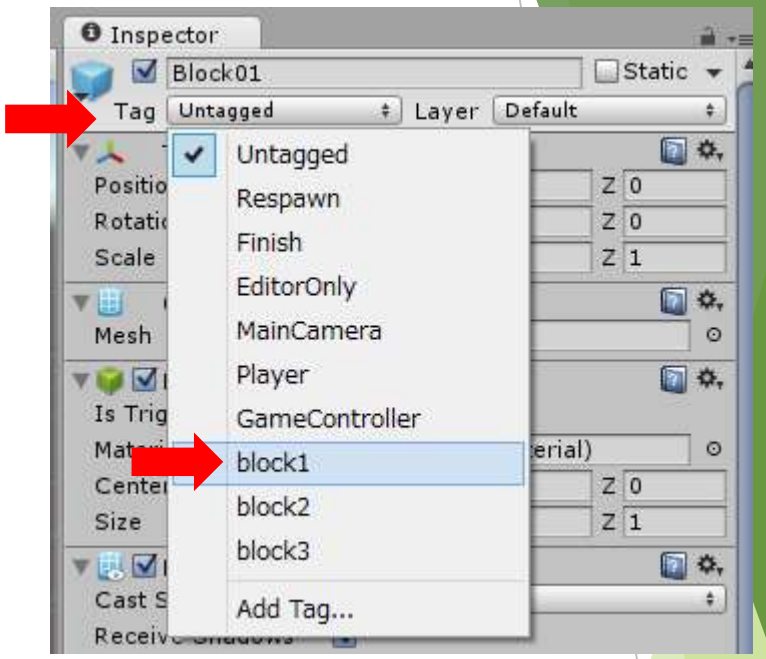
- それぞれ「block1」「block2」「block3」と書く



- タグが出来たのでAssetsにあるブロックで設定する

- 「Block01」のInspectorのTagを「block1」にする
- 同様に「Block02」は「block2」
  「Block03」は「block3」とタグ付けする

- タグ付けの確認をする
- 「S02_Block」を開く



- プログラムを書く

```
using UnityEngine;
using System.Collections;

public class S02_Block : MonoBehaviour {

    void OnCollisionEnter(Collision other)
    {
        switch (this.gameObject.tag)
        {
            case "block1":
                Debug.Log("ブロック1です");
                break;
            case "block2":
                Debug.Log("ブロック2です");
                break;
            case "block3":
                Debug.Log("ブロック3です");
                break;
        }
        GameObject.Destroy(this.gameObject);
    }
}
```

- Switch(条件)

  case（条件1）①

  case（条件2）②

  これで条件が条件1の場合①を

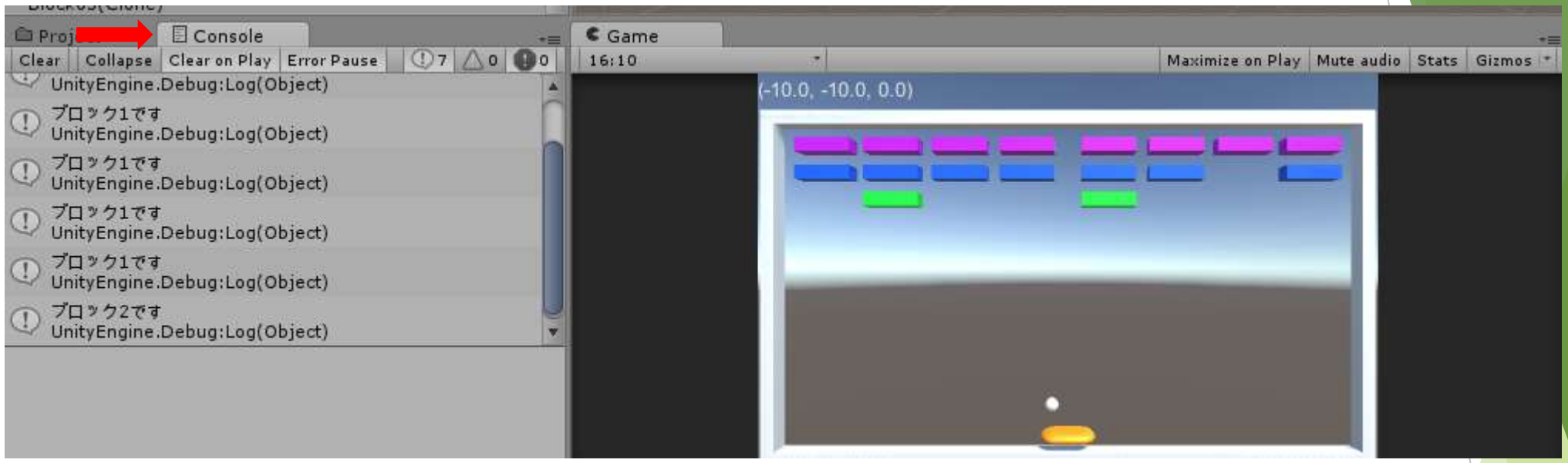
  条件が条件2の場合②を実行する

- つまり衝突した時の物体のタグが

  「block1」のときはコンソールに「ブロック1です」と表示する

- ゲームを実行

- こんな感じでコンソールに文章が表示される



- コンソールが無い人は「Window」→「Console」で開ける

- ここまで出来たらタグ付け完了

2-2. スコア表示部分

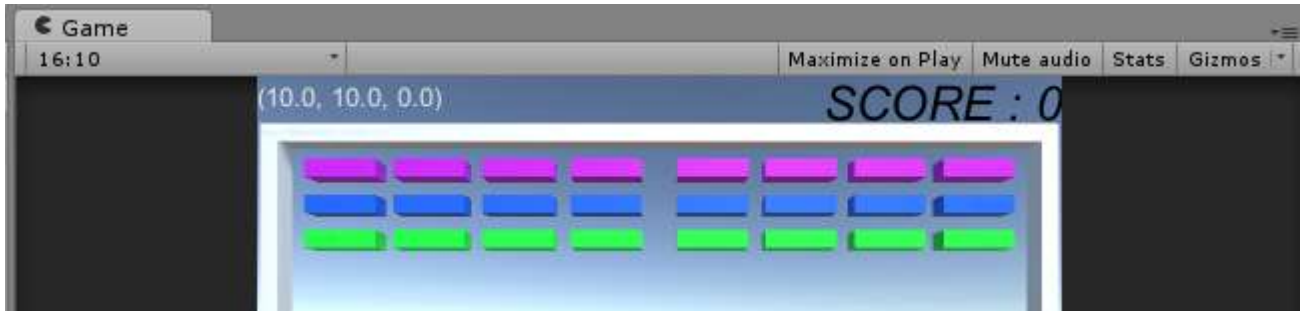
- スコア加算は置いといて、まず表示部分を作ります
- 「Assets」→「Create」→「C# Script」
- 名前は「S02_Score」
- 「GameRoot」にドラッグ＆ドロップ
- ↓のプログラムを書く

```
using UnityEngine;
using System.Collections;

public class S02_Score : MonoBehaviour {

    private int score = 0;
    public GUIStyle gui_score;

    void OnGUI() {
        GUI.Label(new Rect(0, 0, screen.width, 30), "SCORE : " + score, gui_score);
    }
}
```

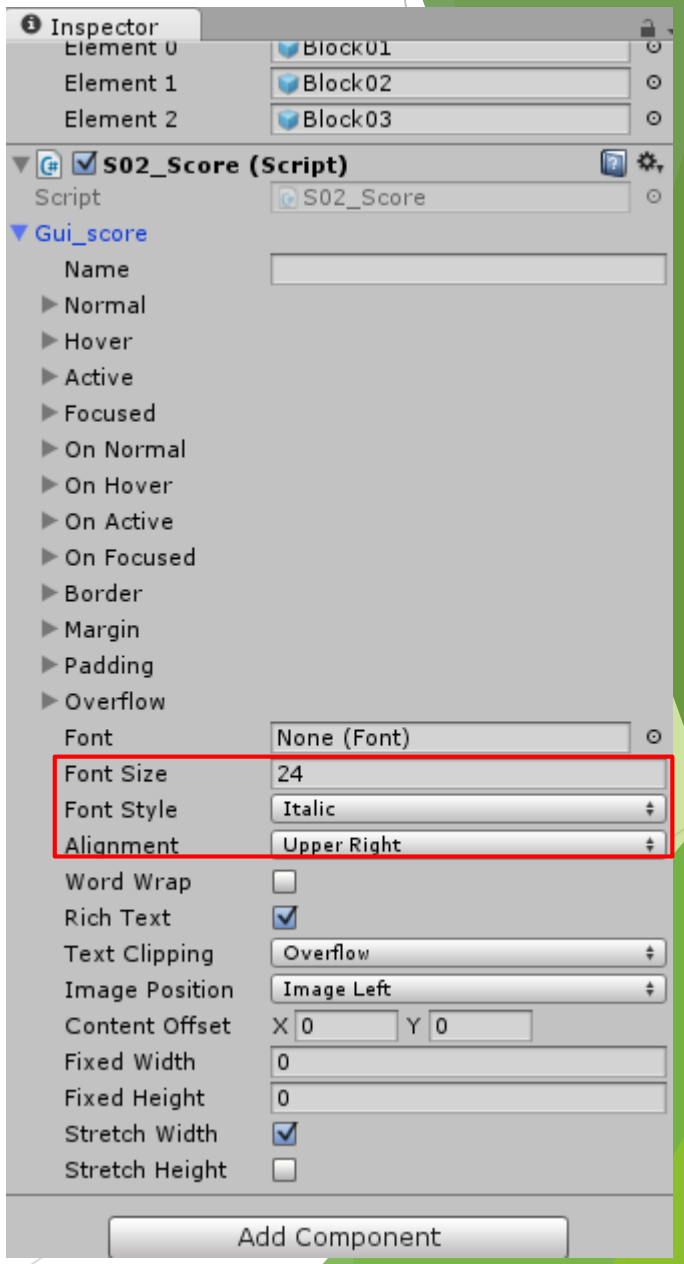
- GUIをいじる「GUIStyle」という変数を定義
- 「SCORE：（実際の数値）」という文を表示させる
- この時、gui_scoreを渡すことでguiをいじれるようになる
- GameRootのInspectorを見てみよう

- Inspector内に「GUI_score」という項目が増えている
- 赤枠内を変更

- ゲームを実行
- 上の右端にスコアが表示される



- Normalの中で色を変えれるので好みでどうぞ



2-3. スコア加算関数

- 衝突時にスコアを加算するための関数を用意
- 「S02_Score」をいじる

Miscellaneous Files　S02_Score

```
using UnityEngine;
using System.Collections;

public class S02_Score : MonoBehaviour {

    private int score = 0;
    public GUIStyle gui_score;

    public void Additional_score(int value)
    {
        score += value;
    }

    void OnGUI() {
        GUI.Label(new Rect(0, 0, Screen.width - 10, 30), "SCORE : " + score, gui_score);
    }
}
```

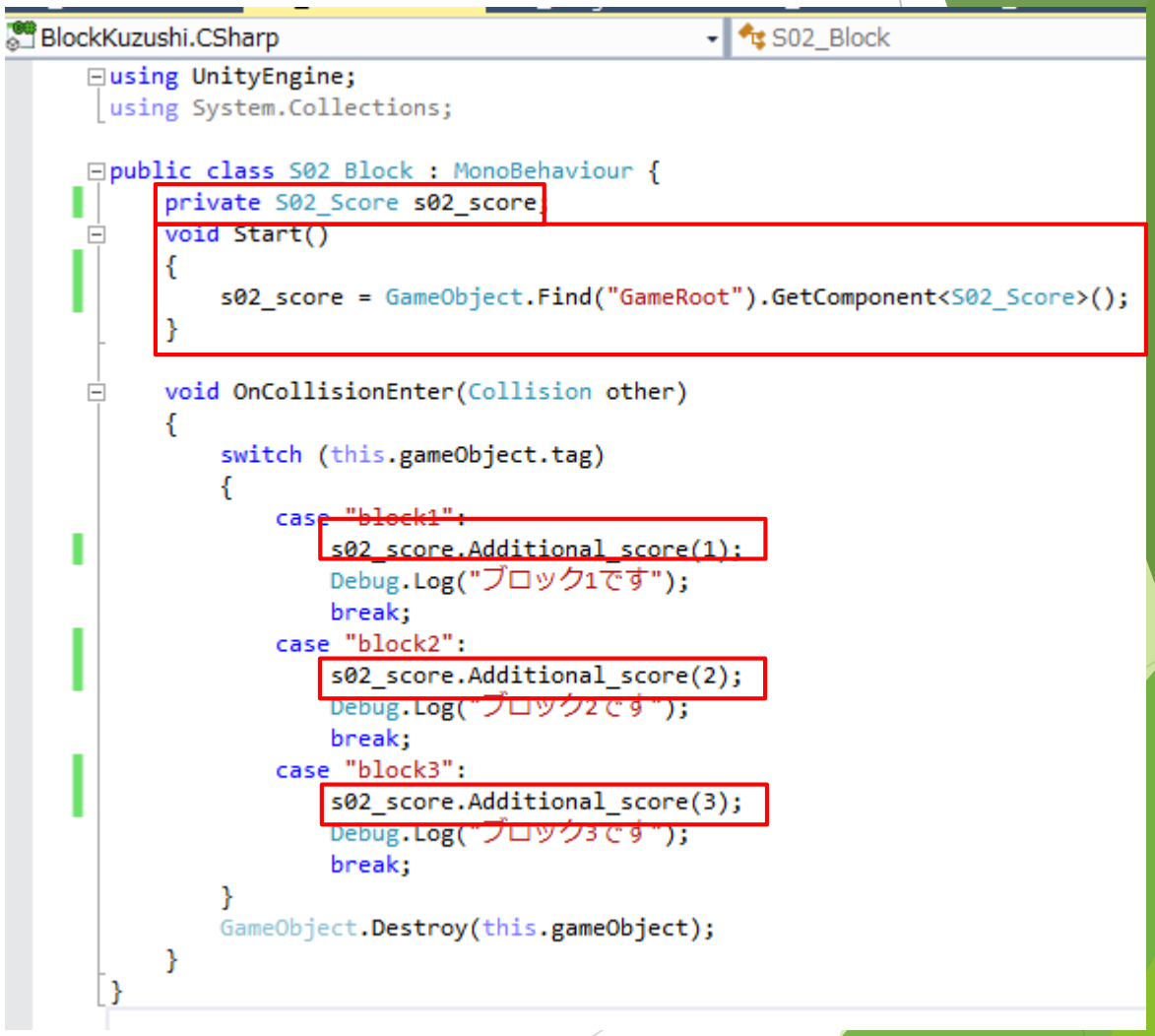
- 赤枠を追加、変更

- Additional_scoreはvalueという整数値を受け取り、socreに加算する
- しかし、まだ関数を作っただけで呼ばれない
- どこから呼ぶのか？
- →衝突時に呼び出す！！

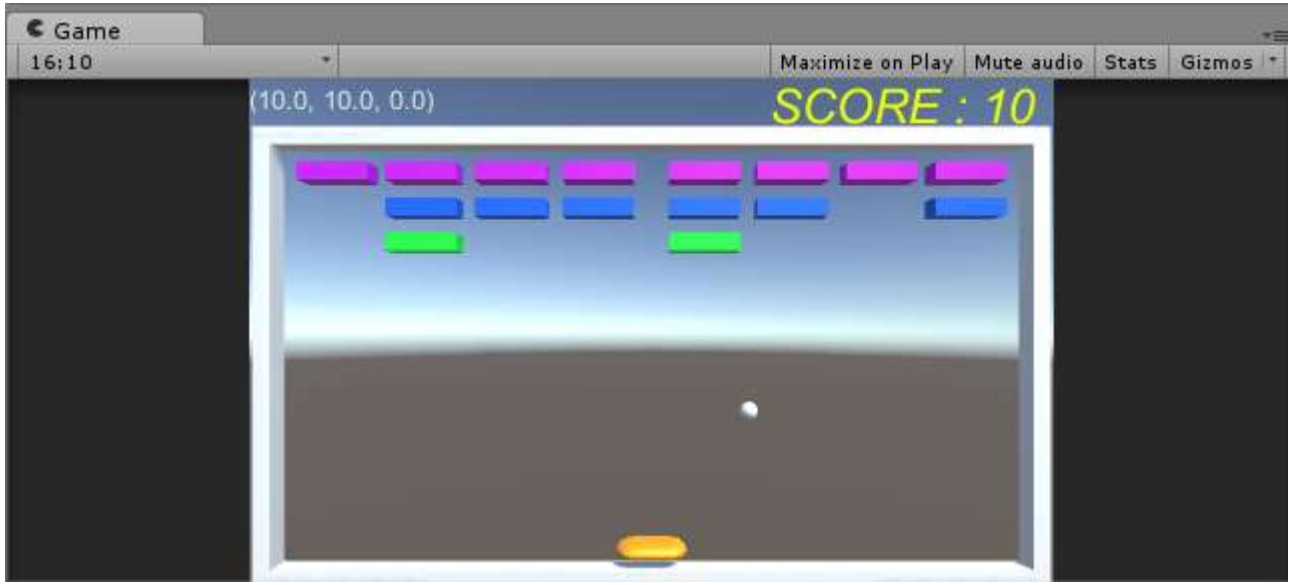
- というわけで「S02_Block」を開く
- 赤枠を変更

- Additional_scoreを使うために
- 変数s02_scoreを用意
- Start関数でs02_scoreを定義
- それぞれのブロックに衝突したとき

ブロック1は1、ブロック2は2、
ブロック3は3を引き渡す



- ゲーム実行
- こんな感じになるはず



- 上画面だとブロック1が4つとブロック2が2つでスコアは10点みたいな
- スコア処理はおけ

3. ブロックの詳細設定

3-1. ブロックに体力をつける

- よくある何度かぶつけて壊れるブロックを作る
- 「S02_Block」を開く
- まずはStart関数を書き換え

- Int でhpを定義
- Start関数でブロック2だけhpを2に設定
  ブロック1とブロック3はhpが1

```
using UnityEngine;
using System.Collections;

public class S02_Block : MonoBehaviour {
    private S02_Score s02_score;
    private int hp;

    void Start()
    {
        s02_score = GameObject.Find("GameRoot").GetComponent<S02_Score>();

        switch (tag)
        {
            case "block2":
                hp = 2;
                break;
            default:
                hp = 1;
                break;
        }
    }
```

- 次はOnCollisionEnterの中身

```
    void OnCollisionEnter(Collision other)
    {
        hp--;
        if (hp == 0)
        {
            switch (this.gameObject.tag)
            {
                case "block1":
                    s02_score.Additional_score(1);
                    Debug.Log("ブロック1です");
                    break;
                case "block2":
                    s02_score.Additional_score(2);
                    Debug.Log("ブロック2です");
                    break;
                case "block3":
                    s02_score.Additional_score(3);
                    Debug.Log("ブロック3です");
                    break;
            }
            GameObject.Destroy(this.gameObject);
        }
    }
}
```

- 衝突時にhp-1をしてhpが0の場合に処理を行う
- これでゲームを実行すると真ん中の列のブロックだけ2回衝突しないと壊れない

- 2回衝突しないと壊れないが、今が何回目かわからない
- →衝突ごとに色を変えよう！
- 衝突ごとに色が薄くなるようにする
- 「S02_Block」を開いて
- ifにelse文を追加して記述

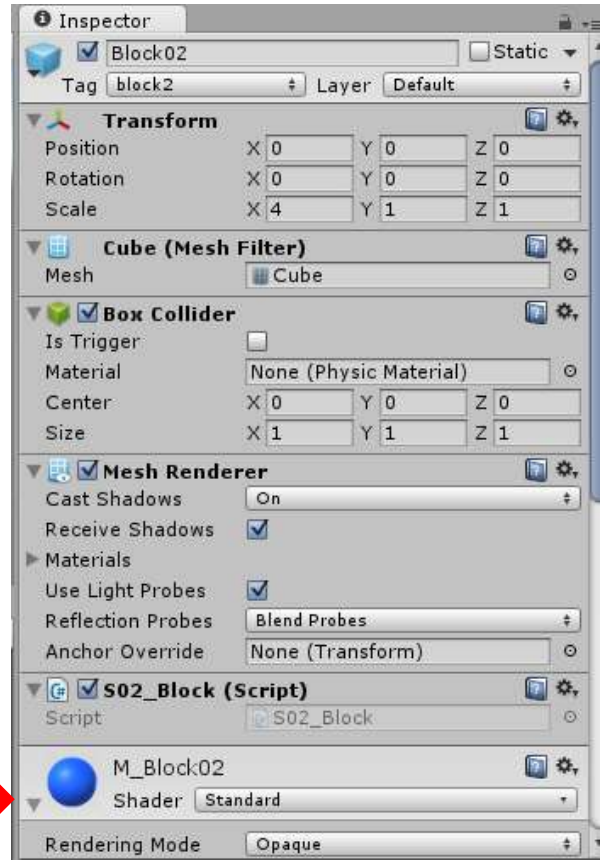
```
            }
            GameObject.Destroy(this.gameObject);
        }
        else
        {
            GetComponent<Renderer>().material.color *= new Color(1.0f, 1.0f, 1.0f, 0.66f);
        }
    }
}
```

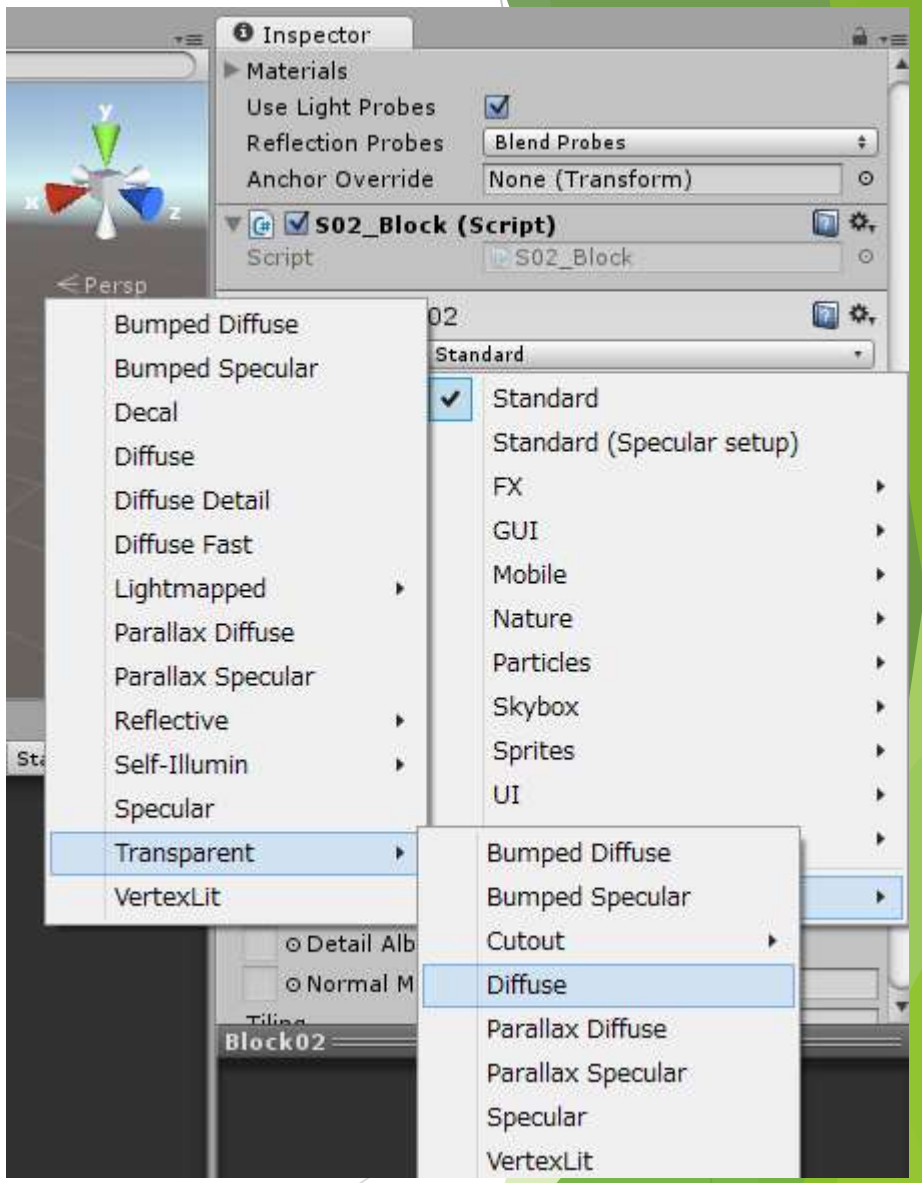
- もし衝突時に-1して、hpが0でなければ色の(r, g, b)はそのままで濃さを0.66倍する

- hpが2だとつまらないので3にしとこう

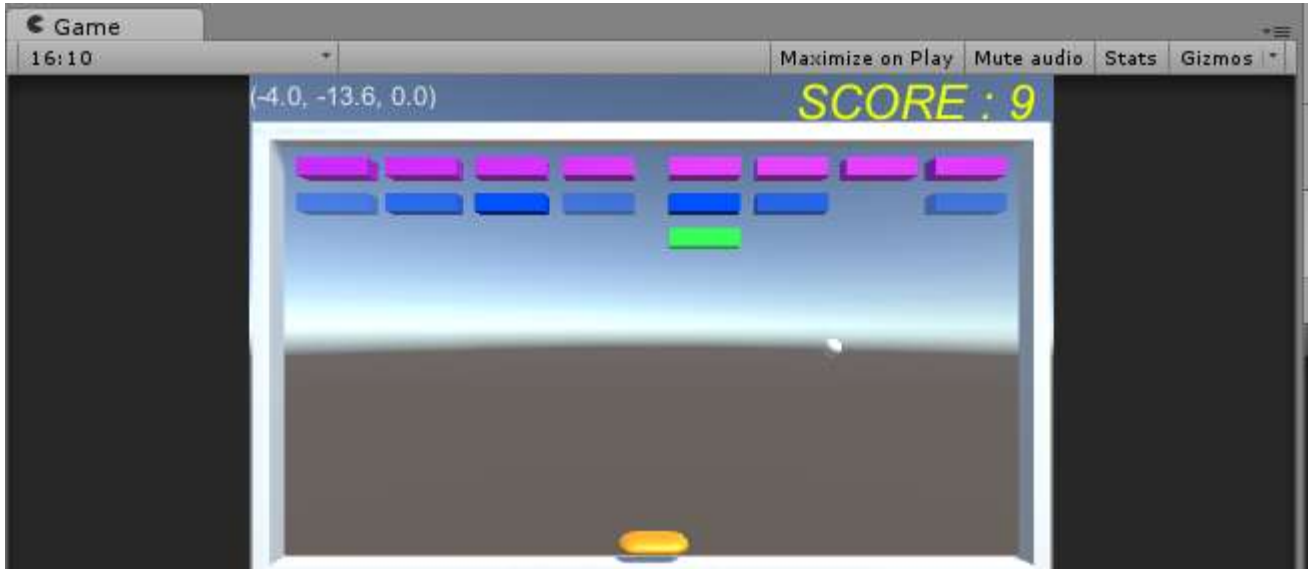
```
void Start()
{
    s02_score = GameObject.Find("GameRoot").GetComponent<S02_Score>();

    switch (tag)
    {
        case "block2":
            hp = 3;
            break:
```

- プログラムは以上で保存
- Block02の設定をいじる
- AssetsのBlock02のInspectorを見る
- M_Blockの「Shader」をクリック
- 「Regacy Shaders」→「Transparent」
  →「Diffuse」を選択





- ゲーム実行



- 真ん中のブロックだけ衝突の度に薄くなって3回目の衝突で破壊
- スコアも破壊時のみ加算される

- できました？

3-2. ブロックを差し替える

- ブロックが破壊されたら違うブロックがもう1回出てくる
- →面白くない？

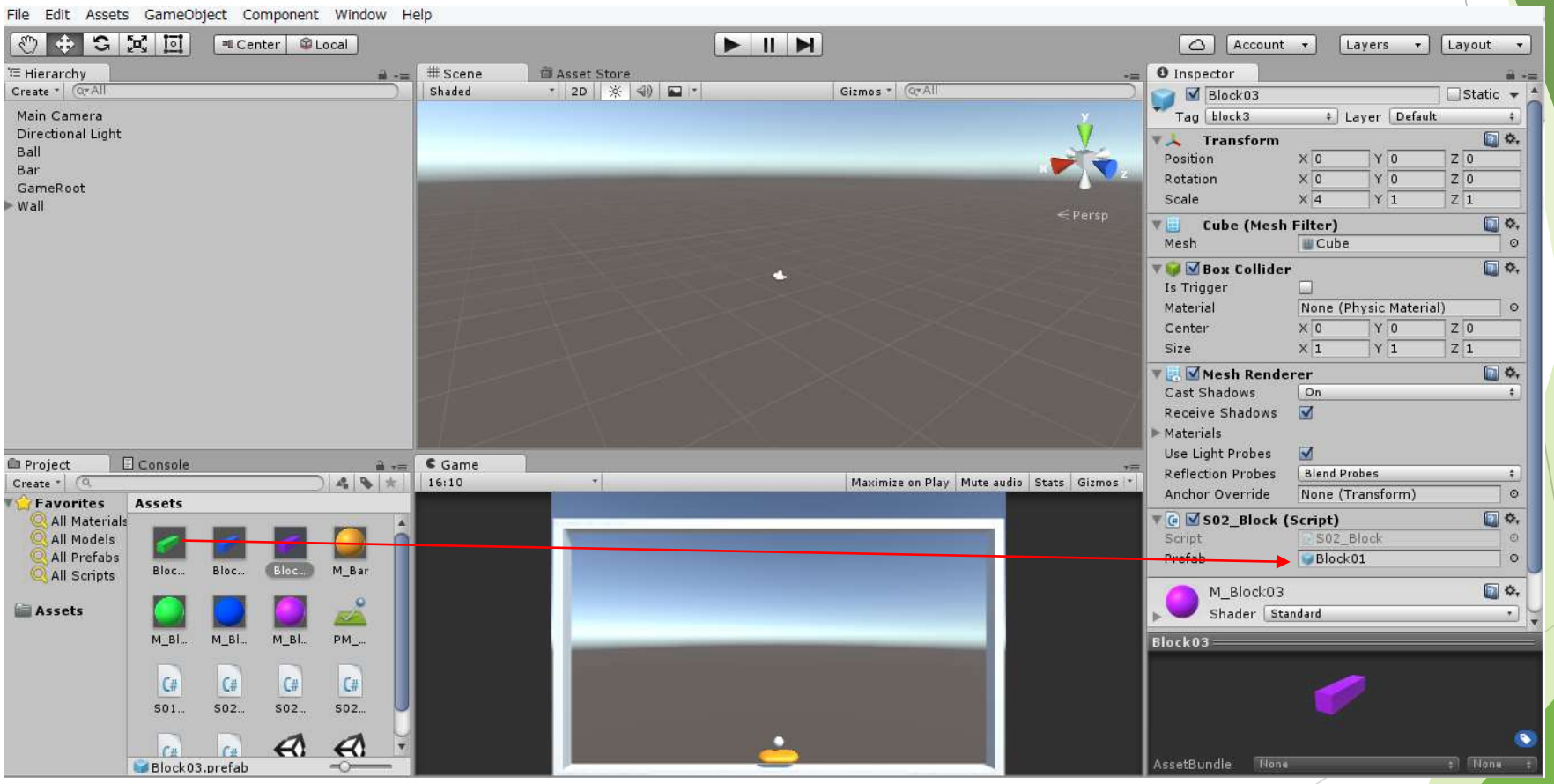
- 「S02_Block」を開く
- 赤枠2か所追加

```
public class S02_Block : MonoBehaviour {
    private S02_Score s02_score;
    private int hp;
    public GameObject prefab;
```

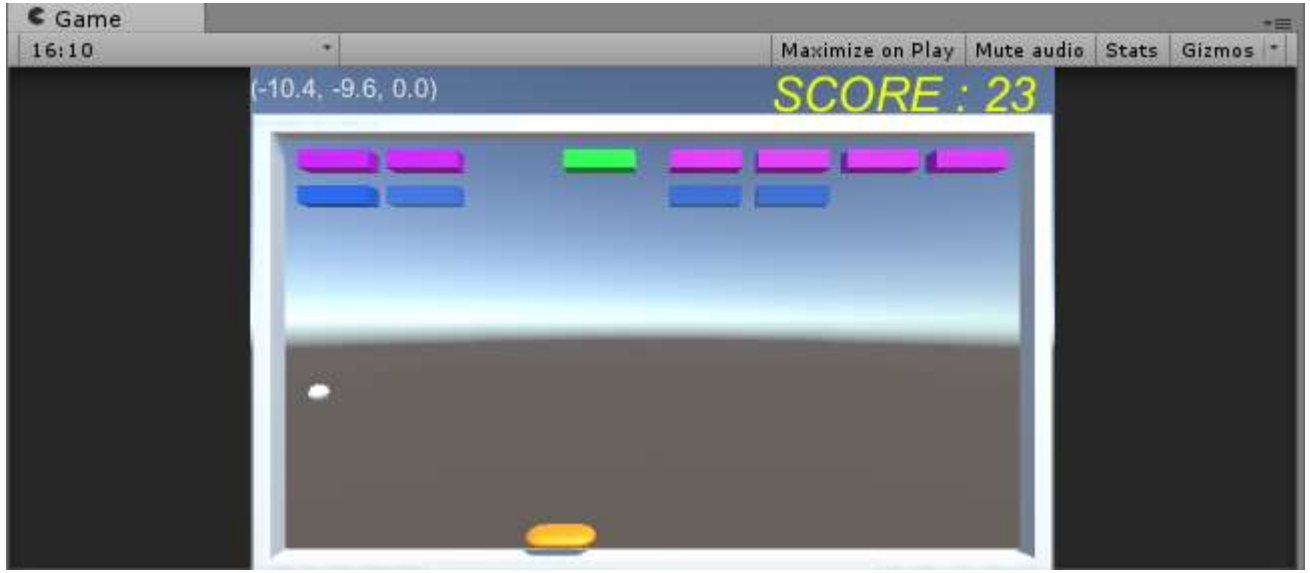
- Block03の破壊時、prefabを同じ場所に配置する
- プログラムは以上
- AssetsのBlock03のInspectorを開く

```
case "block3":
    s02_score.Additional_score(3);
    Instantiate(prefab, transform.position, Quaternion.identity);
    Debug.Log("ブロック3です");
    break;
```

- M_Block03に「prefab」という項目が追加されている
- ここにAssetsにある「Block01」を追加する
- これでBlock03破壊時のprefab = Block01となる



- ゲーム実行

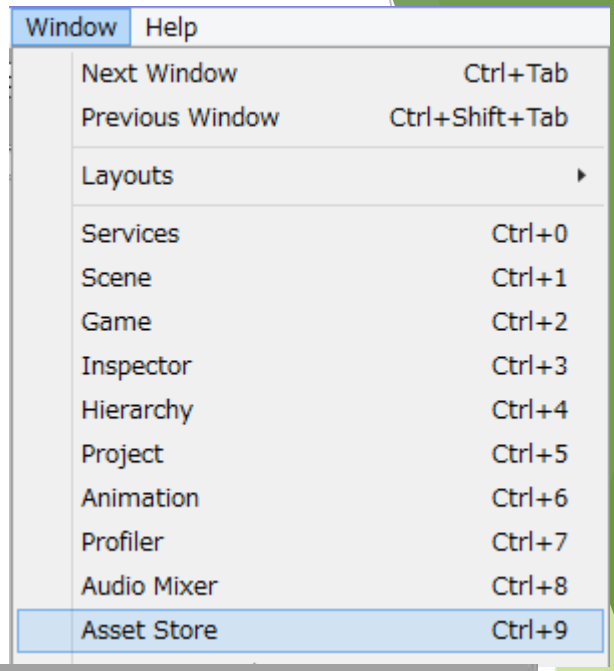


- こんな感じで紫が破壊されると緑が出てくる

4. 効果音を付ける

4-1. アセットストア

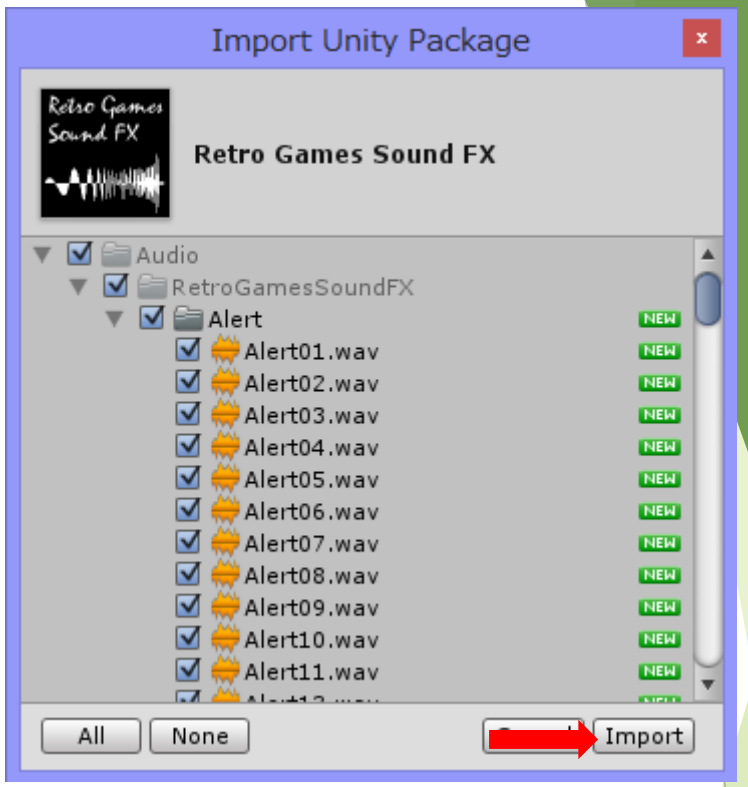
- アセットストアは最強
- Unityで使用できる色々なデータが無料、有料で置いてある
- メニューの「Window」→「Asset Store」
- 画面がある人は直接
- 右の画面が出る



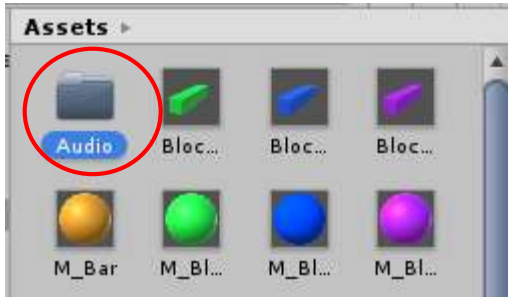
- 検索欄に「retro games」と入力し検索
- 「Retro Games Sound FX」をクリック
- インポートをクリック

- ダウンロード終了を待つ

- ダウンロードが終わると→の画面が出る
- ここでチェックが入っているものをインポートする
- 今回はこのままで「Import」をクリック
- 待つ



- 終わるとAssetsにフォルダが追加されている



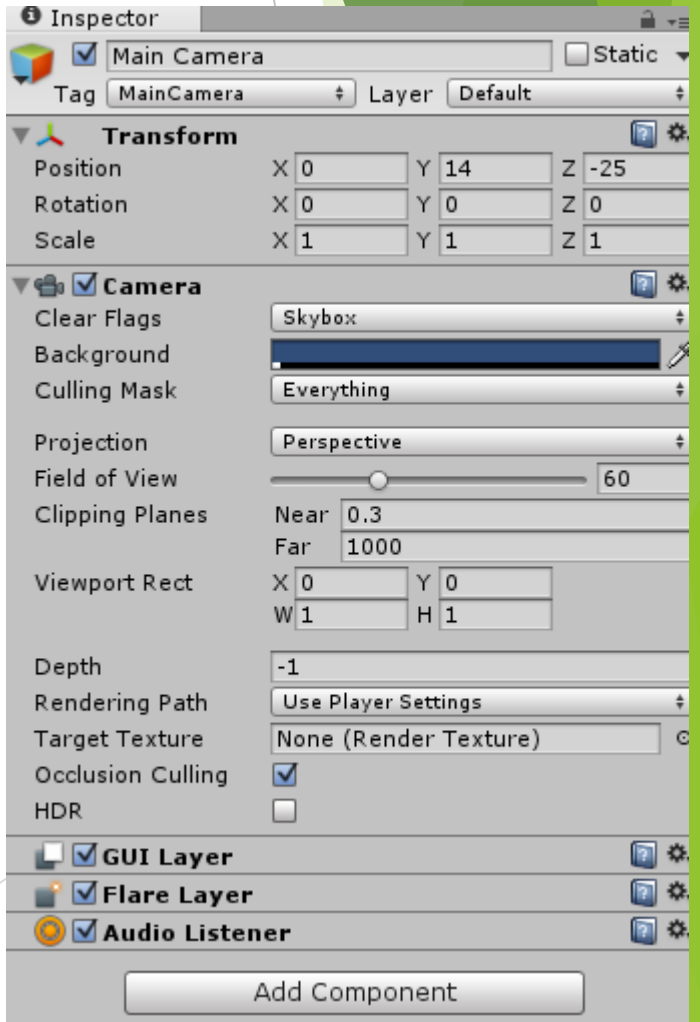
- ここにインポートしたファイルが入っている

- これで音のデータは手に入ったので実際に鳴らす

4-2. 衝突時に音を鳴らす

- Main CameraのInspectorを見ると一番下に「Audio Listener」というコンポーネントがある
- こいつがシーン中で発生した音を拾う
- 「Audio Listener」は1シーンにつき1つしか置けない（初期状態ではMain Cameraの中）ため、他のオブジェクトに着けたい場合main Cameraにある「Audio Listener」は削除する

- 音を鳴らすためのプログラム書きます
- 「S02_Ball」を開く



- 以下のように付け足す

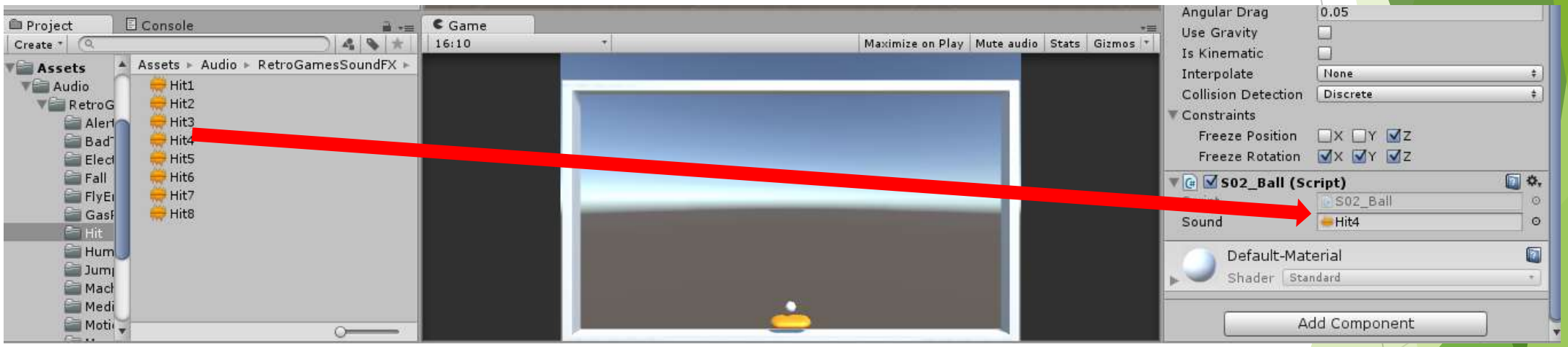
BlockKuzushi.CSharp　S02_Ball

```
using UnityEngine;
using System.Collections;

public class S02_Ball : MonoBehaviour {
    private AudioSource audioSource;
    public AudioClip sound;

    void Start () {
        audioSource = gameObject.AddComponent<AudioSource>();
        audioSource.clip = sound;
        audioSource.loop = false;

        transform.GetComponent<Rigidbody>().velocity = new Vector3(10.0f, 10.0f, 0.0f);
    }

    void OnCollisionEnter()
    {
        audioSource.Play();
    }

    void OnGUI()
    {
        GUI.Label(new Rect(0,0,200,20),""+transform.GetComponent<Rigidbody>().velocity);
    }
}
```

- SudioSourceは音のデータとループするかの情報を持つ
- AudioClipは使用する音をInspectorで設定するために使う
- OnCollisionEnterにより衝突時に音が鳴る

- 上書き保存

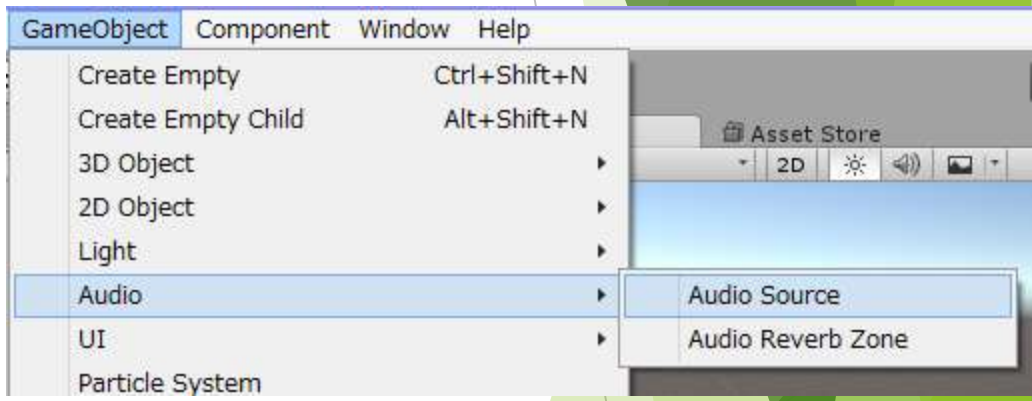
- BallのInspectorを開く
- 右のSoundという項目が追加されているはず
- 「Assets」→「Audio」→「RetroGamesSoundFX」
  →「Hit」→「Hit4」
- の「Hit4」をBallの「Sound」までドラッグ＆ドロップ

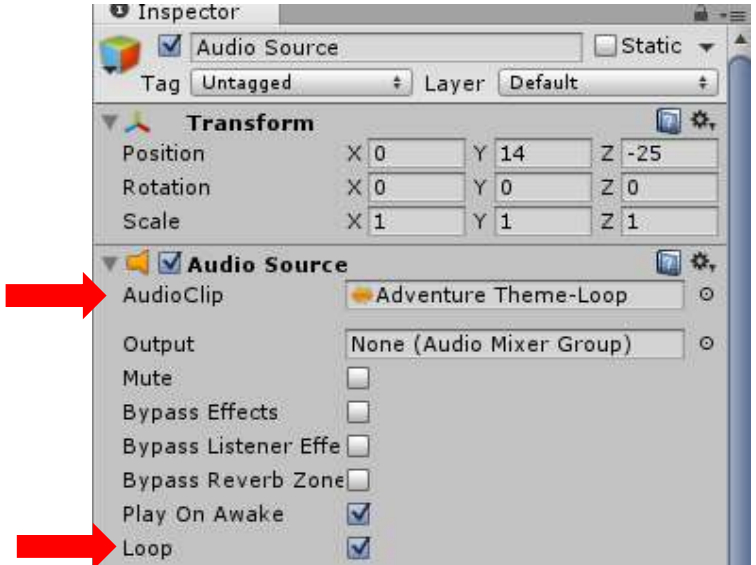


- ゲーム実行
- 音が出てますか？

4-3. BGMを付ける

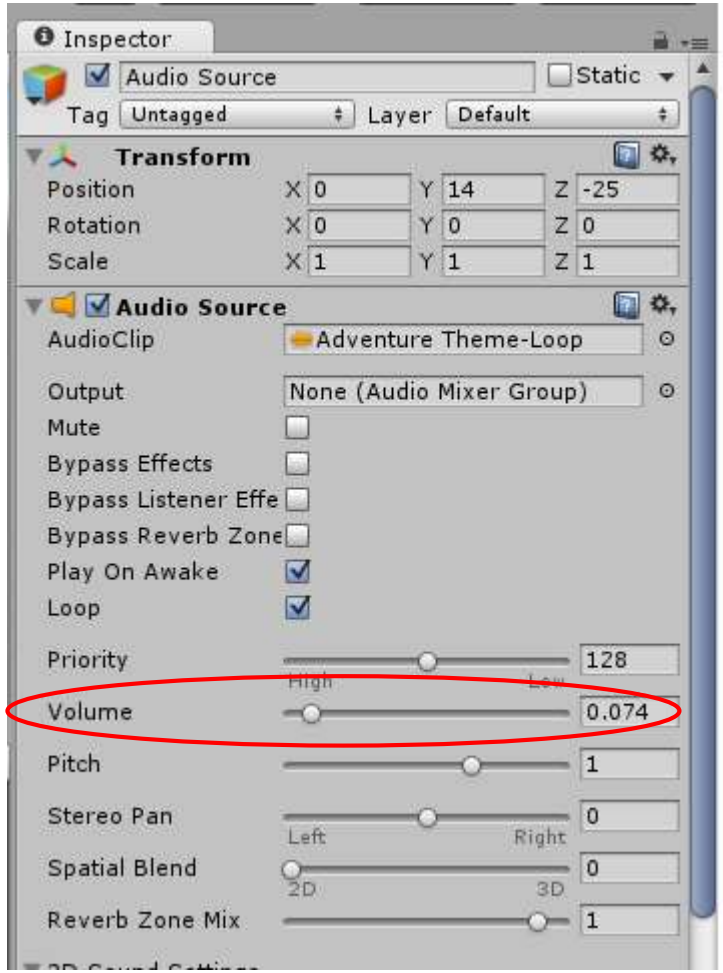
- BGMは衝突時に鳴る！みたいな条件が無いのでやり方を変えます
- 「GameObject」→「Audio」→「Audio Source」
- Audio SourceのInspectorを見る
- 「Audio Clip」に使いたいBGMをドラッグ＆ドロップし「Loop」にチェックを入れる





- ゲーム実行
- 設定したBGMが聞こえればおけ
- 音量が大きすぎる場合はInspectorの Volumeを下げよう
- 他にも値をいじれるので気になるところは 修正すればいいんじゃないかな？

- BGMを付けたい人は付けてください
- 別につけなくてもいいです



5. ゲームクリアの処理

5-1. ゲームクリアの条件設定

- ブロック崩しのゲームクリアは？

  →全てのブロックを壊す

- 判定方法は色々…
- 今回は、ブロックが1つ破壊されたときシーン上にブロックがあるか調べる

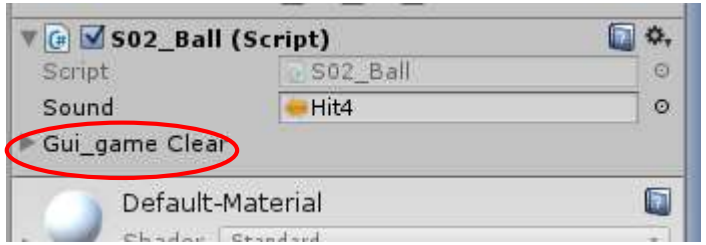
5-2. ブロックの有無を調べる

- 前回設定したタグを使って調べる
- 「S02_Ball」を開く

```
    void OnCollisionEnter()
    {
        audioSource.Play();
        checkAllBlocks();
    }

    bool gameClear;
    void checkAllBlocks()
    {
        if ((GameObject.FindWithTag("block1") == null) && (GameObject.FindWithTag("block2") == null) && (GameObject.FindWithTag("block3") == null))
        {
            gameClear = true;
            transform.GetComponent<Rigidbody>().velocity = Vector3.zero;
        }
    }

    public GUIStyle gui_gameClear;
    void OnGUI()
    {
        GUI.Label(new Rect(0,0,200,20),""+transform.GetComponent<Rigidbody>().velocity);

        if (gameClear)
        {
            GUI.Label(new Rect(0, 0, Screen.width, Screen.height), "GameClear", gui_gameClear);
        }
    }
}
```

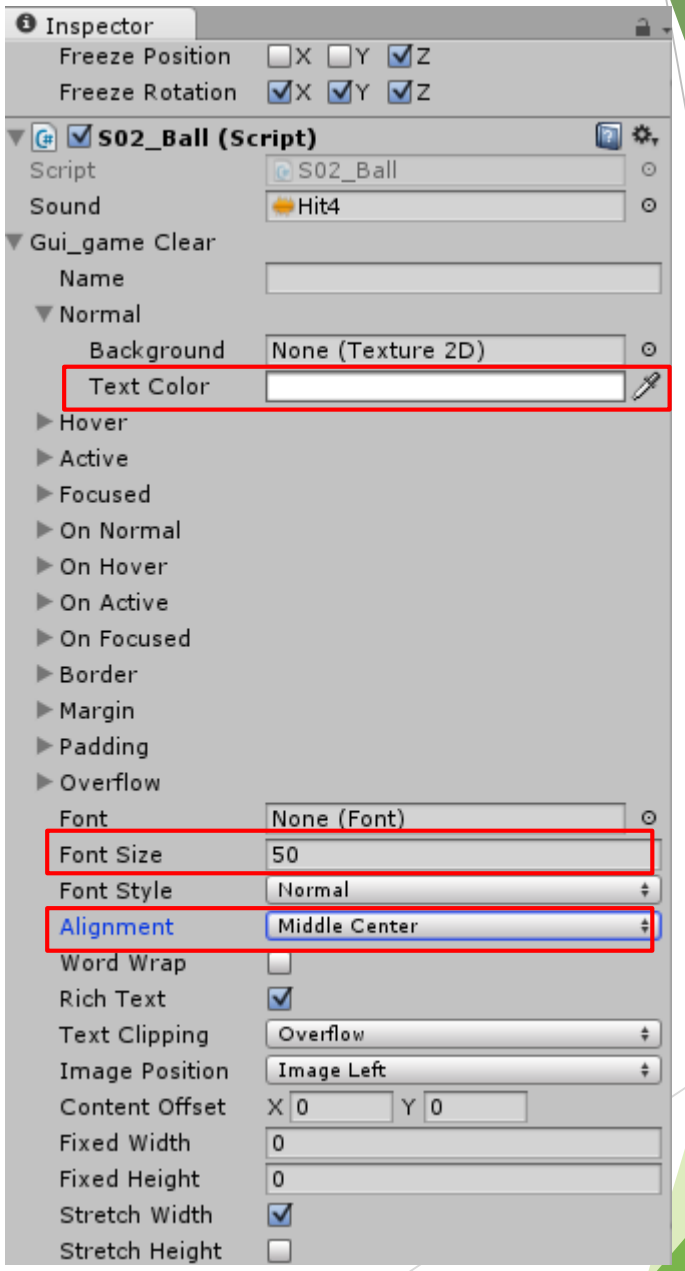
- 衝突時にcheckAllBlocksという関数を呼ぶ
- checkAllBlocksはFindWithTag(“タグ名”)という関数を使う。
  これはゲームオブジェクトに”タグ名”というタグがある場合trueを、ない場合falseを返す。
  つまり衝突時に「block1」「block2」「block3」のタグを持つオブジェクトが1つも存在しない場合ifの条件を満たす。
- ifの中身　…　ゲームクリアのフラグgameClearをtrueにする。
  ボールの速度を0にする。（停止させる）

- OnGUI関数ではgameClearがtrueの場合のみ「GameClear」と画面に表示する
- つまり、通常はボールの速度を左上に表示するだけだがクリア時には「gameClear」を表示するようになる。
- 新たにGUIのスタイルを定義したのでボールのInspectorを見る



- S02_Ballの中のGui_game Clearを見る
- 赤枠を変更
- テキストカラーは自由でいいかも

- これが出来たらゲーム実行といきたいが
  ちゃんと動くか確かめるために全てのブロックが
  破壊されるのを待つのはキツイ！



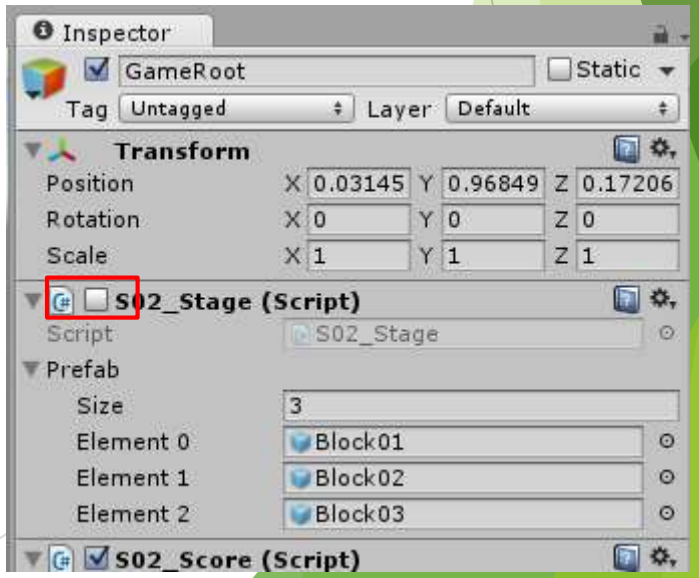
5-3. ラクにデバッグ？

- Hierarchyの「GameRoot」のInspectorを見る
- 「S02_Stage」というスクリプトがある
- こいつがステージのブロックを生成する

  →「S02_Stage」を読まなければブロック生成されない

  →ゲーム実行後1回目の衝突でクリア判定になる！！

- どうすればいいか？
- GameRootのInspector「S02_Stage」のチェックボックスを外す
- これでゲーム実行時に読み込まれなくなる



- ゲーム実行

- こんな感じになりましたか？
  - 最初にブロックが1つも生成されない
  - 1回目の衝突時に「GameClear」が表示される
  - ボールが衝突した場所で停止する

- ちゃんと動いていたらGameRootの「S02_Stage」にチェックを入れ直しましょう

6. ステージ追加

6-1. クリア処理の移動

- ステージを追加する前に…
- ゲームクリア処理がボールで行われるのはおかしい！
- クリア処理のプログラムを「S02_Stage」に移す
- 「S02_Stage」と「S02_Ball」両方書き直す。まずは「S02_Stage」から
- 赤枠変更（次ページに続く）

```
public class S02_Stage : MonoBehaviour {
    public GameObject[] prefab;
    private bool gameClear = false;
    public GUIStyle gui_gameClear;

    void Start()
    {
        blockSetting();

    }
```

- デバッグ用にコメントアウトしておく

```
private void blockSetting()
{

    GameObject block_right1 = GameObject.Instantiate(prefab[0]) as GameObject;

    block_right1.transform.position = new Vector3(3.0f + (0), 19.0f, 0.0f);
    /*for (int i = 0; i < 4; i++)
    {
        GameObject block_right1 = GameObject.Instantiate(prefab[0]) as GameObject;
        GameObject block_right2 = GameObject.Instantiate(prefab[1]) as GameObject;
        GameObject block_right3 = GameObject.Instantiate(prefab[2]) as GameObject;
        GameObject block_left1 = GameObject.Instantiate(prefab[0]) as GameObject;
        GameObject block_left2 = GameObject.Instantiate(prefab[1]) as GameObject;
        GameObject block_left3 = GameObject.Instantiate(prefab[2]) as GameObject;

        block_right1.transform.position = new Vector3(3.0f + (5 * i), 19.0f, 0.0f);
        block_right2.transform.position = new Vector3(3.0f + (5 * i), 21.0f, 0.0f);
        block_right3.transform.position = new Vector3(3.0f + (5 * i), 23.0f, 0.0f);
        block_left1.transform.position = new Vector3(-3.0f - (5 * i), 19.0f, 0.0f);
        block_left2.transform.position = new Vector3(-3.0f - (5 * i), 21.0f, 0.0f);
        block_left3.transform.position = new Vector3(-3.0f - (5 * i), 23.0f, 0.0f);
    }*/
}

public bool checkAllBlocks()
{
    if ((GameObject.FindWithTag("block1") == null) && (GameObject.FindWithTag("block2") == null) && (GameObject.FindWithTag("block3") == null))
    {
        gameClear = true;
    }
    return gameClear;
}

void OnGUI()
{
    if (gameClear)
    {
        GUI.Label(new Rect(0, 0, Screen.width, Screen.height), "GameClear", gui_gameClear);
    }
}
```

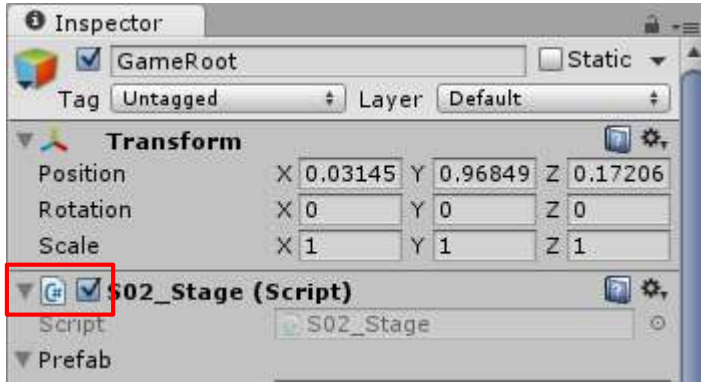
18.0f　にしてください

- 次は「S02_Ball」

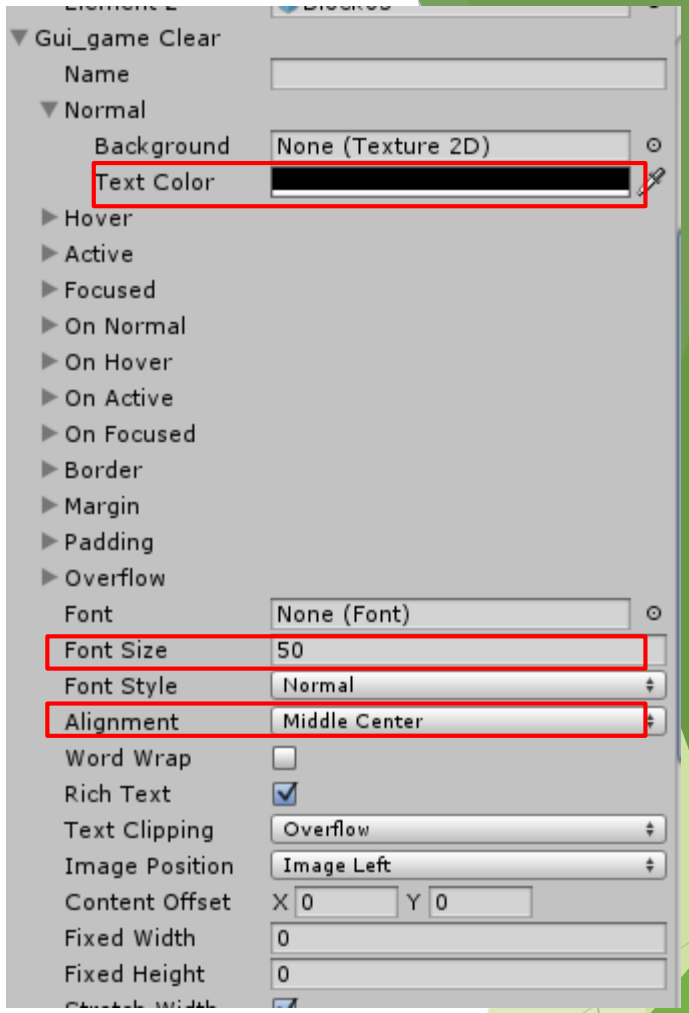
```
using UnityEngine;
using System.Collections;

public class S02_Ball : MonoBehaviour {
    private AudioSource audioSource;
    public AudioClip sound;
    private S02_Stage s02_Stage;

    void Start () {
        audioSource = gameObject.AddComponent<AudioSource>();
        audioSource.clip = sound;
        audioSource.loop = false;
        s02_Stage = GameObject.Find("GameRoot").GetComponent<S02_Stage>();

        transform.GetComponent<Rigidbody>().velocity = new Vector3(10.0f, 10.0f, 0.0f);
    }

    void OnCollisionEnter()
    {
        audioSource.Play();
        if (s02_Stage.checkAllBlocks())
        {
            transform.GetComponent<Rigidbody>().velocity = Vector3.zero;
        }
    }

    void OnGUI()
    {
        GUI.Label(new Rect(0, 0, 200,20),""+transform.GetComponent<Rigidbody>().velocity);
    }
}
```

- GUIをいじる部分がBallからGameRootに移るので再設定
- GameRootのInspectorを開き赤枠をいじる

- 5でチェックボックスが空のままの人は入れ直す

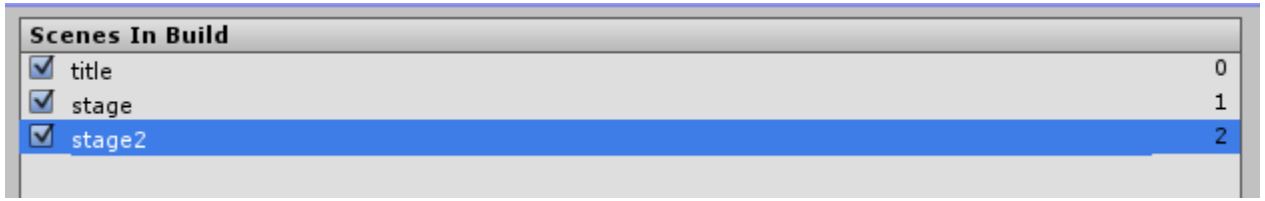


- ゲーム実行
- 真ん中にあるブロックが破壊されたらクリアになる



6-2. ステージ2の準備

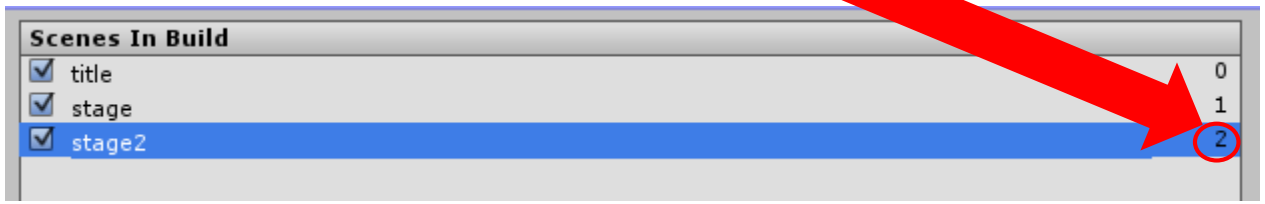
- 1からステージ2を作るのはめんどくさい
- 今のステージをコピペしよう！
- 「File」→「Save Scene as…」
- 名前は「stage2」
- シーン遷移のためにビルドセッティングをする
- 「File」→「Build&Setting」を開き、Scenes In Buildに「stage2」を追加
- 「Build&Setting」を閉じる



- 遷移用のプログラムを書く
- 「S02_Stage」を開く
- 赤枠を追加

- gameClearがtrueのとき
  マウスが左クリックされたら
  2番目のシーンに移動する
  この2番目とは

```
using UnityEngine;
using System.Collections;
using UnityEngine.SceneManagement;

public class S02_Stage : MonoBehaviour {
    public GameObject[] prefab;
    private bool gameClear = false;
    public GUIStyle gui_gameClear;

    void Start()
    {
        blockSetting();

    }

    void Update()
    {
        if (gameClear)
        {
            if (Input.GetMouseButtonDown(0))
            {
                SceneManager.LoadScene(2);
            }
        }
    }
```



6-3. スコア引継ぎ

- シーンごとに変数を管理するためステージ1と2では別のスコアとなる
- 1のスコアに2のスコアを足していくにはどうするか？

  →ゲーム起動時に作成され終了まで保持され続ける変数を使う
- 「S02_Score」を開く
- 赤枠追加

```
using UnityEngine;
using System.Collections;

public class S02_Score : MonoBehaviour {

    private static int score = 0;
    public GUIStyle gui_score;
```

- staticという1単語追加するだけ
- ※staticは多用しない

6-4. ステージ2の作成

- 土台はステージ1のコピペにより完成している
- あとはプログラムをちょっと書き直すだけ
- 「S02_Stage」を開く
- 赤枠変更

- 何ステージ目かを保持する変数stage_noを用意
- stage_noによってブロックの配置を変更（後で設定）
- stage_noが用意したステージの数を超える場合
  遷移しないようにする

```
public GameObject[] prefab;
private bool gameClear = false;
public GUIStyle gui_gameClear;
static private int stage_no = 1;

void Start()
{
    switch (stage_no)
    {
        case 1:
            blockSetting();
            break;
        case 2:
            blockSetting();
            break;
    }
}

void Update()
{
    if (gameClear)
    {
        if (Input.GetMouseButtonDown(0))
        {
            stage_no++;
            if (stage_no < 3)
            {
                SceneManager.LoadScene(stage_no);
            }
        }
    }
}
```

- ゲーム実行
- 1度目のゲームクリア時に左クリック　→　ステージ2に進む
- 2度目のゲームクリア時に左クリック　→　何も起きない

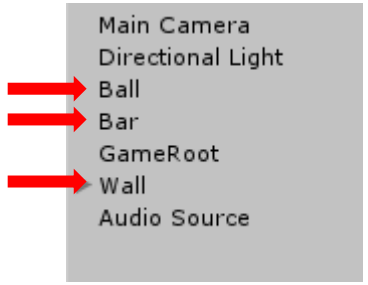
- 以上が確認できれば大丈夫

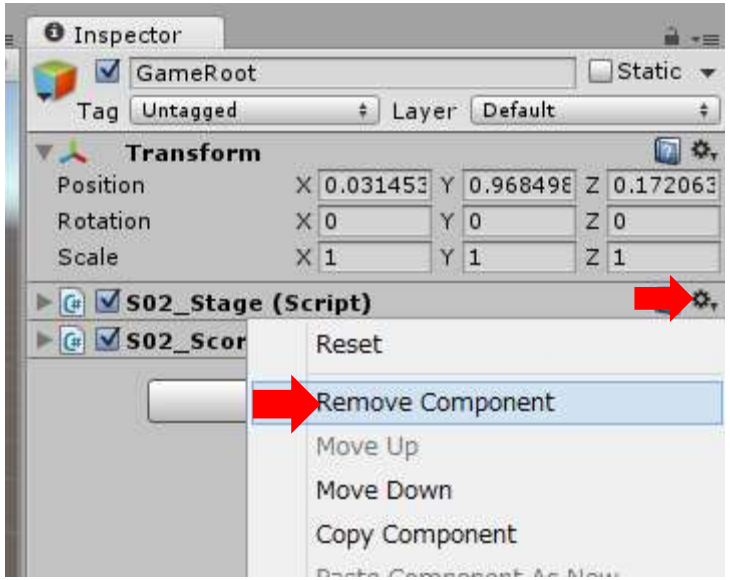
- ここまで出来たらステージ2のブロック生成は置いといて
- 先にエンディングを作ります

- 「File」→「Save Scene as...」
- 名前は「ending」

- 「ending」シーンを開く

- 開いたらHierarchyから「Ball」「Bar」「Wall」を削除
- 「GameRoot」のInspectorを開き「S02_Stage」を削除





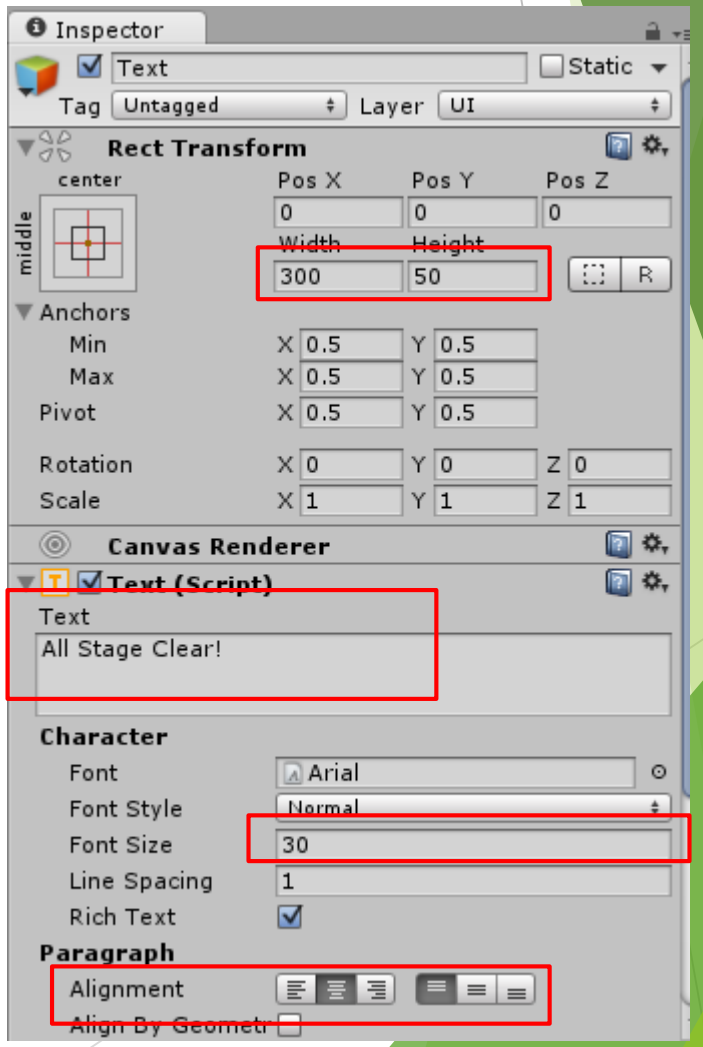
- エンディング用のプログラムを書く
- 「Assets」→「Create」→「C# Script」
- 名前は「S03_ending」
- できたらHierarchyの「GameRoot」にドラッグ＆ドロップ

- プログラムを書く

```
using UnityEngine;
using System.Collections;
using UnityEngine.SceneManagement;

public class S03_ending : MonoBehaviour {
    private float step_timer = 0.0f;

    // Update is called once per frame
    void Update () {
        step_timer += Time.deltaTime;
        if (step_timer > 5.0f)
        {
            SceneManager.LoadScene(0);
        }
    }
}
```

- 秒数をカウントする変数を用意
- 5秒経ったら0番目のシーン（title）に遷移

- 出来たら保存
- Unityの画面に戻り、エンディング用のテキストを書く
- 「GameObject」→「UI」→「Text」
- Inspectorの赤枠変更

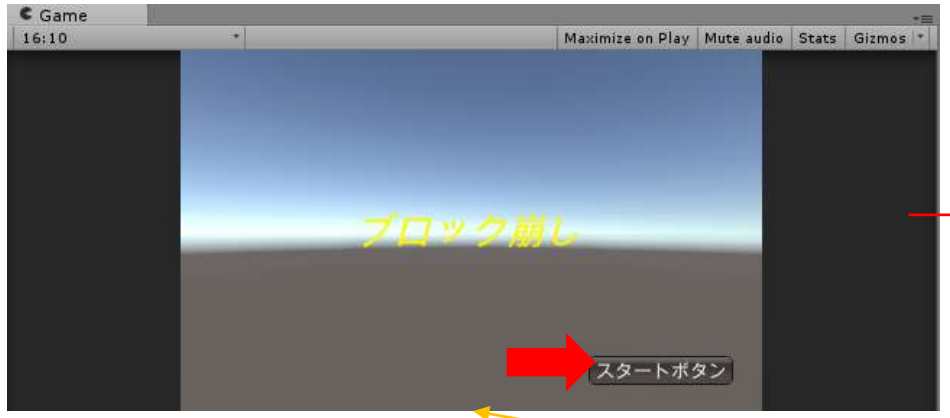


- 「File」→「Build&Setting」でScenes In Buildに「ending」を追加
- endingへの遷移をプログラムに追加
- 「S02_Stage」を開く

```
void Update()
{
    if (gameClear)
    {
        if (Input.GetMouseButtonDown(0))
        {
            stage_no++;
            if (stage_no < 3)
            {
                SceneManager.LoadScene(stage_no);
            }
            else
            {
                SceneManager.LoadScene("ending");
            }
        }
    }
}
```

- できたら保存

- ここまでできたらtitleシーンを開いてからゲーム実行
- ※stageのInspector「S02_Stgae」にチェックが入っているか確認



5秒後

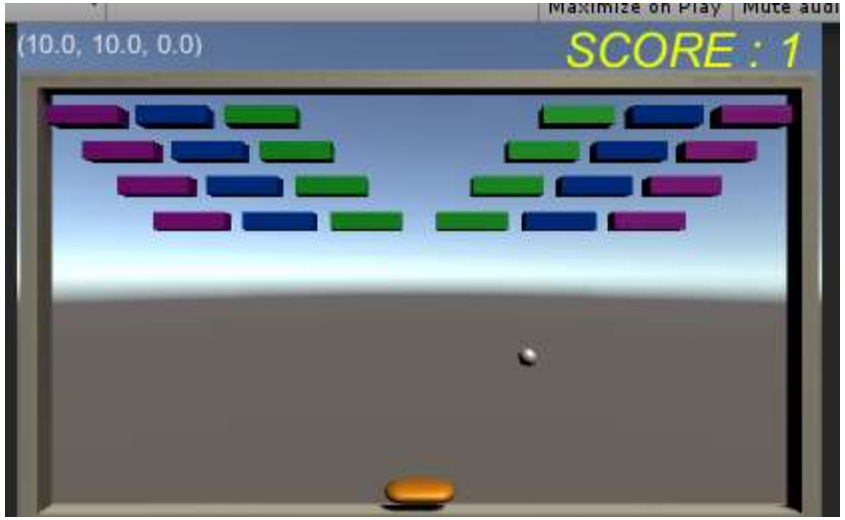
- 前ページの動作がちゃんとできた人はステージ2のブロック生成をやる
- 「S02_Stage」を開く

```
void Start()
{
    switch (stage_no)
    {
        case 1:
            blockSetting1();
            break;
        case 2:
            blockSetting2();
            break;
    }
}
```

```
private void blockSetting1()
{

    GameObject block_right1 = GameObject.Instantiate(prefab[0]) as GameObject;

    block_right1.transform.position = new Vector3(18.0f , 19.0f, 0.0f);
    /*for (int i = 0; i < 4; i++)
    {
        GameObject block_right1 = GameObject.Instantiate(prefab[0]) as GameObject;
        GameObject block_right2 = GameObject.Instantiate(prefab[1]) as GameObject;
        GameObject block_right3 = GameObject.Instantiate(prefab[2]) as GameObject;
        GameObject block_left1 = GameObject.Instantiate(prefab[0]) as GameObject;
        GameObject block_left2 = GameObject.Instantiate(prefab[1]) as GameObject;
        GameObject block_left3 = GameObject.Instantiate(prefab[2]) as GameObject;

        block_right1.transform.position = new Vector3(3.0f + (5 * i), 19.0f, 0.0f);
        block_right2.transform.position = new Vector3(3.0f + (5 * i), 21.0f, 0.0f);
        block_right3.transform.position = new Vector3(3.0f + (5 * i), 23.0f, 0.0f);
        block_left1.transform.position = new Vector3(-3.0f - (5 * i), 19.0f, 0.0f);
        block_left2.transform.position = new Vector3(-3.0f - (5 * i), 21.0f, 0.0f);
        block_left3.transform.position = new Vector3(-3.0f - (5 * i), 23.0f, 0.0f);
    }*/
}
```

```
private  void blockSetting2()
{
    for (int i = 0; i < 3; i++){
        for(int j = 0; j < 4; j++)
        {
            GameObject block_r = GameObject.Instantiate(prefab[i]) as GameObject;
            GameObject block_l = GameObject.Instantiate(prefab[i]) as GameObject;

            block_r.transform.position = new Vector3(3.0f + (5 * i) + (2 * j), 17.0f + (2 * j), 0.0f);
            block_l.transform.position = new Vector3(-3.0f - (5 * i) - (2 * j), 17.0f + (2 * j), 0.0f);
        }
    }
}
```

- blockSetting1のコメントアウトはそのままでtitleシーンから実行
- ステージ2に進んだ時にブロックの配置がこんなんならおっけー



- ここまで出来た人はblockSetting1のコメントアウトを戻して
  上の2行を削除

- ゲーム実行して遊んでみる

おまけ？

- 遊んでる人は気づいたと思いますがボールが左右にしか動かなくなることがある
  →初速と物理法則のみで動くため変な計算されると終わる

- 衝突ごとに速度を設定してあげればいい

- 「S02_Ball」に赤枠を追加
- これで衝突時に速度を再設定するので
  常に10か-10の速度を持つ

```
void OnCollisionEnter()
{
    audioSource.Play();

    Vector3 v = transform.GetComponent<Rigidbody>().velocity;
    if (v.x < 0.0f)
    {
        v.x = -10.0f;
    }
    else
    {
        v.x = 10.0f;
    }
    if (v.y < 0.0f)
    {
        v.y = -10.0f;
    }
    else
    {
        v.y = 10.0f;
    }

    transform.GetComponent<Rigidbody>().velocity = v;

    if (s02_Stage.checkAllBlocks())
    {
        transform.GetComponent<Rigidbody>().velocity = Vector3.zero;
```

- 今回はここまで

- 次回は
  - クリア時の初期化
  - 挙動改善
  - アイテム追加？
  - etc...

- よりゲームっぽいことを付け足したいと思います

- 質問がある人は連絡ください

お疲れ様でした