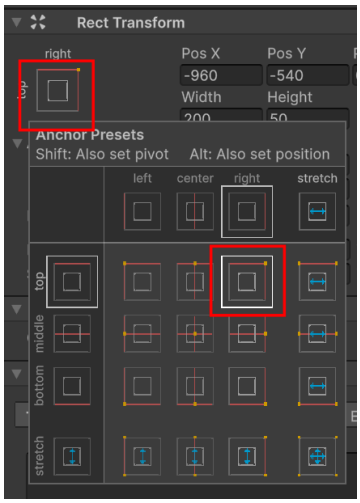




UI???

ゲームが動くが、画面になにも情報がなく、　　Canvas を使い、ゲームの状況を可視化しましょう。

Canvas???

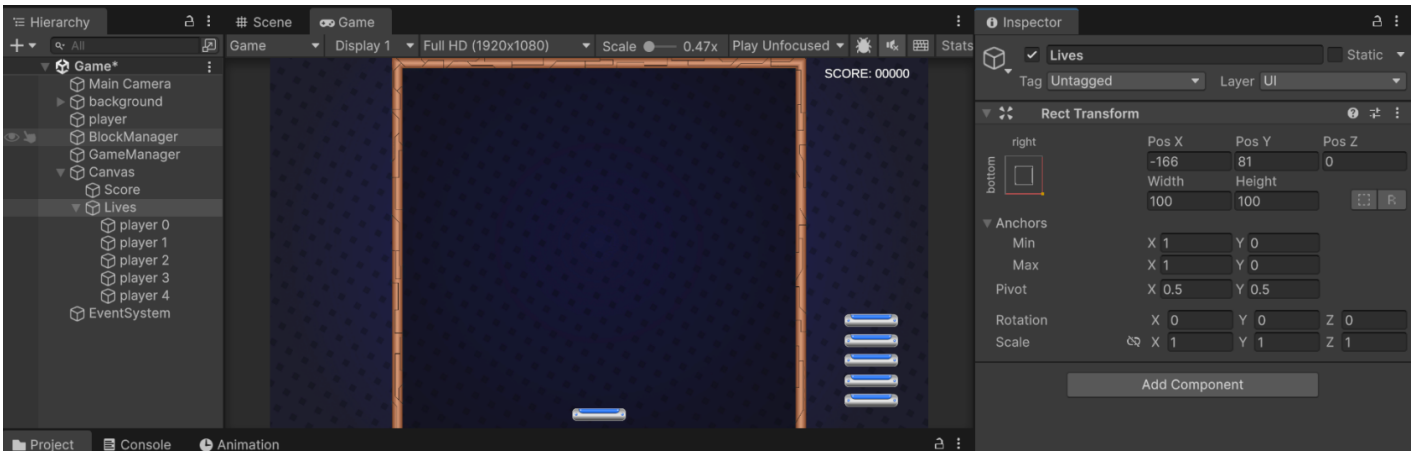
まず、　Canvas を作成し、残機とスコアの表示場所を作りましょう。

???

スコアは画面の右上に表示しよう。テキストをを表示するので「　　TextMeshPro」で良いでしょう。必ず右上に出したいので、アンカーが画面の右上を追跡するようにしてください。位置を文字のサイズを調整し、とりあえず、「　　SCORE: 00000」を表示するようにしてください:

??

残機は、スプライトの表示・非表示で作りましょう。とりあえず、最大残機５個までにし、画面の右下に配置しましょう:



ここでヒエラルキーの順番を注目してください

- Canvasは一番上にあるもので、
- その中に残機のグループ「　　Lives」を準備した
- さらにその中では5つの「　　Image」を使うゲームオブジェクトを準備した

?????

1つのスクリプト＝1つの責任の考えを従って、　　UIの管理するのは「　　UIManager」にお任せしましょう。ゲーム監督　　`GameManager`　が　`UIManager`　を操作し、ゲームの情報を画面上で表示してもらう。

???

新しい　`UIManager`　スクリプトを作成し、スコアの「　　`TextMeshPro`」を　Inspector　で設定できるようにし、中身を変えられるメソッドを実装しましょう:

```
using TMPro;
using UnityEngine;

// UIを管理するスクリプト
public class UIManager : MonoBehaviour
{
    // スコア用のテキスト
    [SerializeField]
    private TextMeshProUGUI scoreText;

    // GameManagerからスコアを設定できるようにする
    public void SetScore(int score)
    {
```

```
        scoreText.text = $"SCORE: {score}";
    }
}
```

そして、 GameManager を更新し、スコアを数え、表示しましょう。一応、ブロック1個消えたら、 100 点を与えることにしましょう。すでにブロックが廃止の時に呼ばれるメソッド「 BlockDestroyed 」があるので、そこで点数を増やせば良いでしょう。

まず、 GameManager のメンバー変数で:

```
// ゲーム監督
public class GameManager : MonoBehaviour
{
    // ボールを投げる処理
    [SerializeField]
    private BallShooter ballShooter;

    // 残機の数
    private int _lives;

    // ブロックの数
    private int _blockCount;

    // スコア
    private int _score;

    // 後は同じ
}
```

そして、 `BlockDestroyed` を編集し、スコアを更新:

```
// ブロック廃止のとき呼び出される
public void BlockDestroyed()
{
    // 100点を与える
    _score += 100;

    _blockCount--;
    if (_blockCount == 0)
    {
        // コンソールで表示し、ボール投げれなくなる
```

```
            Debug.Log("ゲームクリア");
            ballShooter.enabled = false;
        }
    }
```

UIManager????

スコアを更新しているので、次に `UIManager` に表示するように命令する。参照が必要であるため、`GameManager` で `UIManager` の紐づけできるようにする:

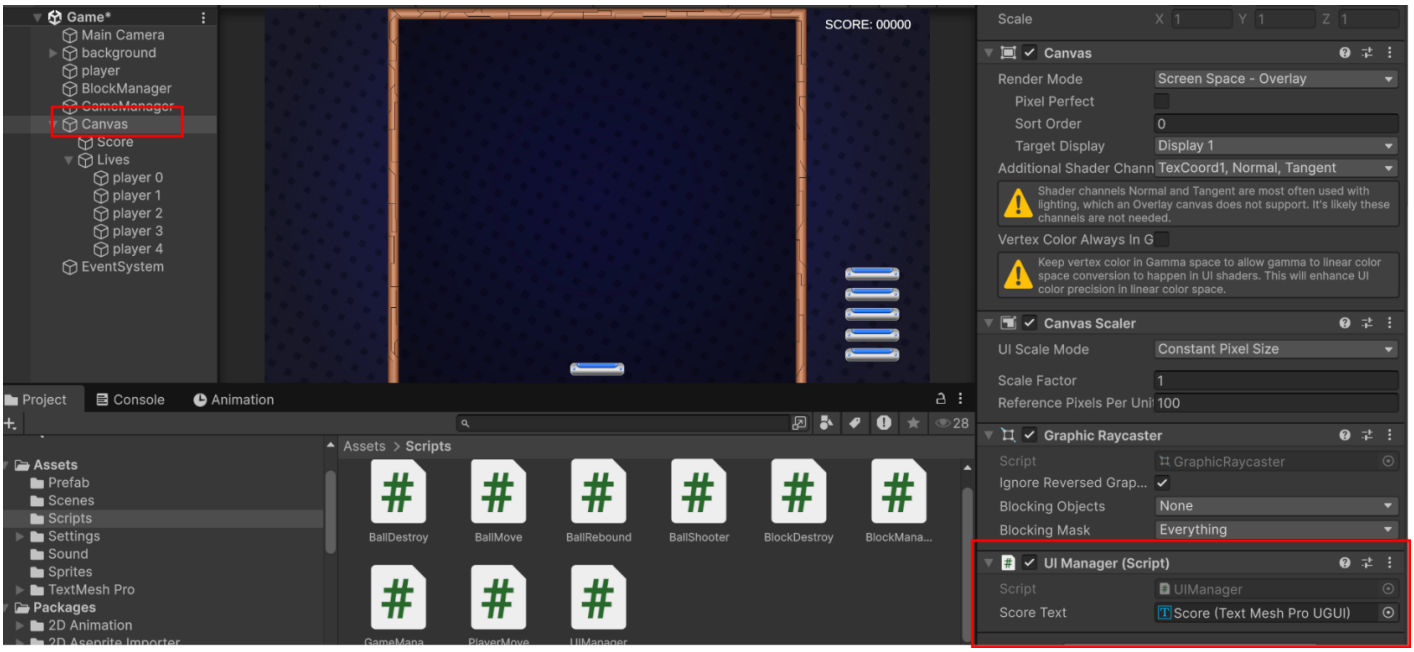
```
// ゲーム監督
public class GameManager : MonoBehaviour
{
    // ボールを投げる処理
    [SerializeField]
    private BallShooter ballShooter;

    // ゲームの情報を表示できるように
    [SerializeField]
    private UIManager uiManager;

    // 後は同じ
}
```

そして、スコアを更新したら、UIの更新を UIManager に頼む

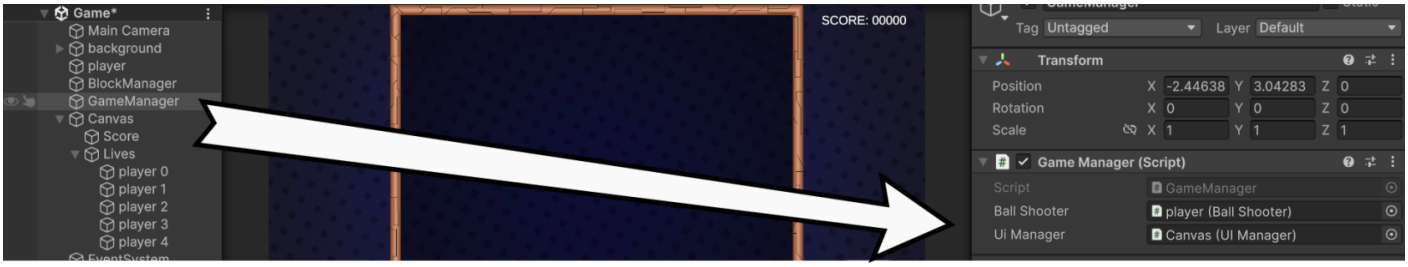
```
// ブロック廃止のとき呼び出される
public void BlockDestroyed()
{
    // 100点を与える
    _score += 100;

    // UIを更新
    uiManager.SetScore(_score);

    // 後は同じ
}
```

Unity????

スクリプトができたので、Unity のシーンを更新しましょう。`UIManager` は UI を管理するので、Canvas にアタッチするのは適切と思われる。スコアの TextMeshPro を設定してね～



そして、 GameManager に UIManager の参照を設定～



これで確認ができる！

??

UIManager で
ゲームオブジェクトの表示・非表示することによって、プレーヤーに残機の数を教えましょう。

まず、 player 0 ～ player 4 の５個のゲームオブジェクトがどこにあるのかを UIManager に教えましょう：

// UIを管理するスクリプト
public class UIManager : MonoBehaviour
{
// スコア用のテキスト
[SerializeField]
private TextMeshProUGUI scoreText;

// 残機表示用
[SerializeField]
private GameObject [] playerImages;

```
    // 後は同じ
}
```

そして、`SetScore`と同じように、`GameManager`からアクセスができる`SetLives`メソッドを作ろう:

```
// GameManagerから残機を設定できるようにする
public void SetLives(int lives)
{
    // ゲームオブジェクトを1個ずつに繰り返す
    for (int i = 0; i < playerImages.Length; i++)
    {
        // i番目の残機画像を表示するか?
        bool isAlive = i < lives;
        playerImages[i].SetActive(isAlive);
    }
}
```

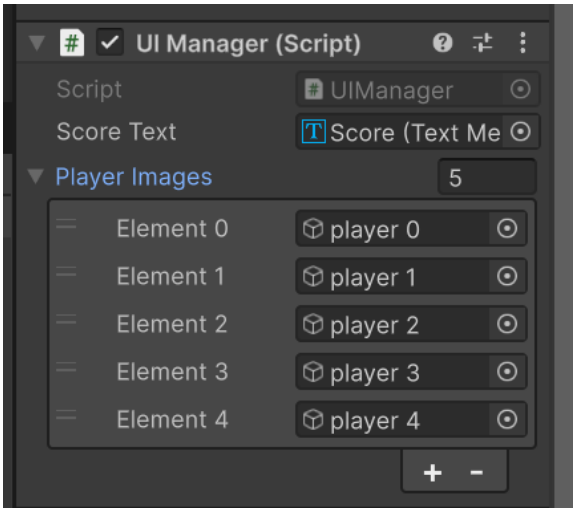
最後に、ゲームが始まるときと、残機が減るときに`SetLives`で残機の表示を更新しよう:

```
private void Start()
{
    _lives = 3; // 3個で始まる
    uiManager.SetLives(_lives);
}

// ボール廃止されたときに呼び出される
public void BallDestroyed()
{
    // 残機が減る
    _lives--;
    uiManager.SetLives(_lives);
    if (_lives == 0)
    {
        // とりあえず、コンソールで表示
        Debug.Log("ゲームオーバー");

        // スクリプトを無効化する
        ballShooter.enabled = false;
    }
```

```
}
```

Unity????

UIManager □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□



Revision #1
Created 2026-04-28 06:17:58 UTC by Menendez Francisco
Updated 2026-05-04 04:41:27 UTC by Menendez Francisco