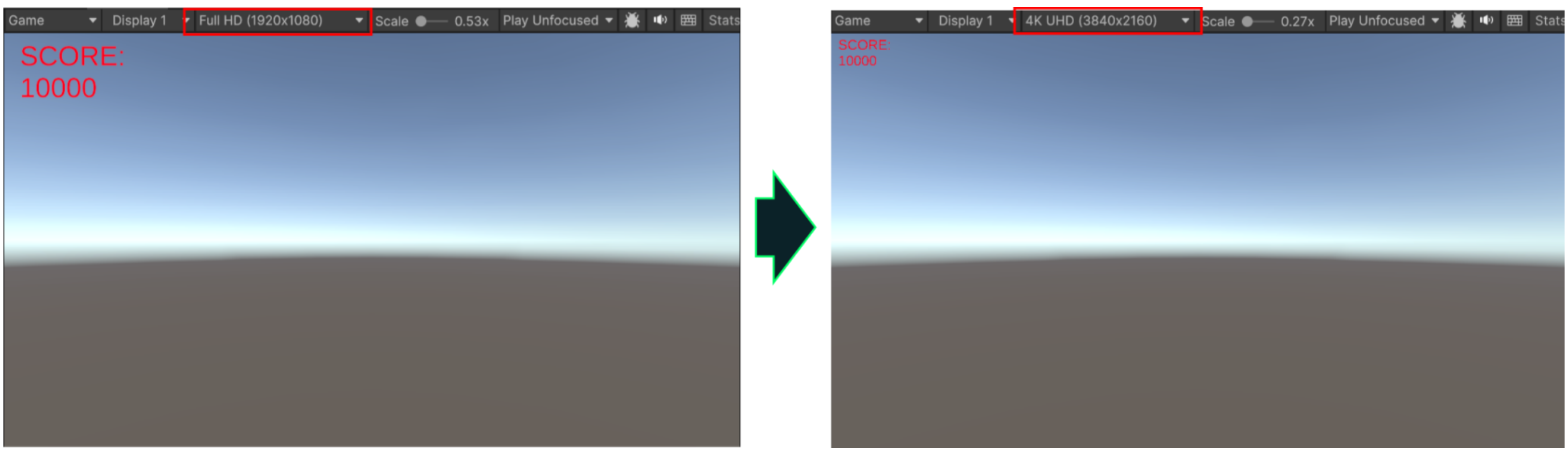
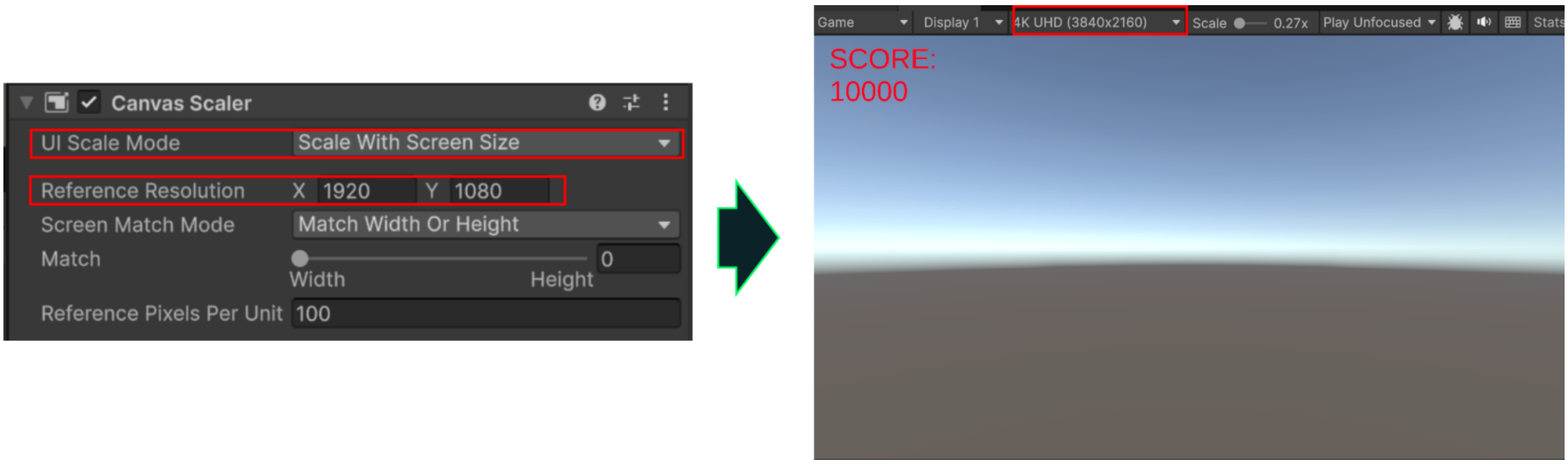




UI???

インベントリの中にあるアイテム（コイン、鍵）を画面上で表示し、主人公の体力も可視化しましょう。

Canvas??

ヒエラルキーに新しい　　Canvas を追加してください。

Canvas Scaler ???

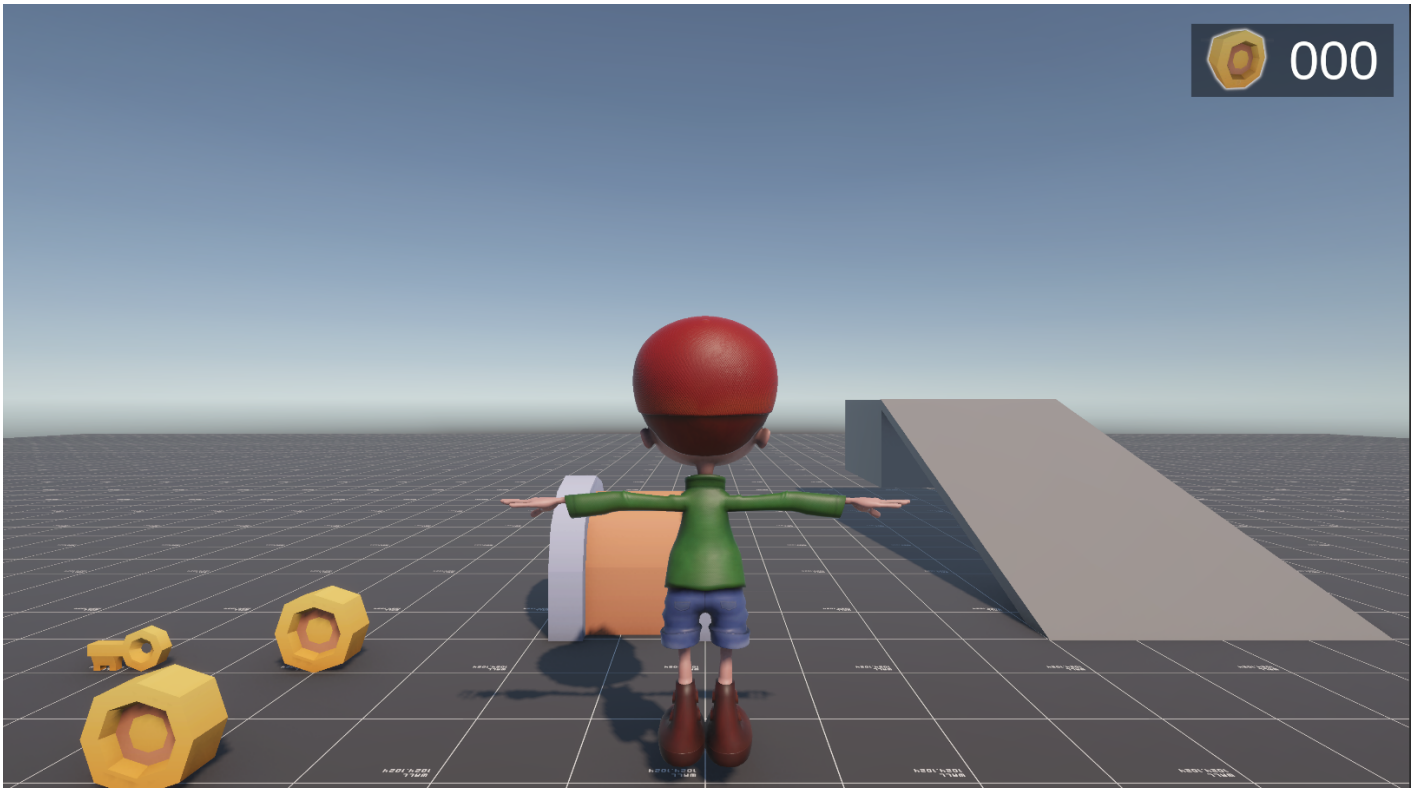
Canvas を作成するときに、同時に「　　Canvas Scaler
」のコンポーネントを作成される。このコンポーネントは想定する画面のサイズに合わせて、自動的に
Canvasを拡大縮小する。

例えば、　　UIを開発したときに、想定していた画面サイズは　　FullHD (1920 x 1080)
だった。ただ、プレーヤーは　　4K（ 3840 x 2160）で遊ぶとしたら、画面がすごく小さくなってしまう。

この場合は　　Canvas Scaler
を使用すれば、画面が大きくなっても、小さくなっても、いつも正しく表示される

Canvas Scaler 無し:　　FullHD から　　4K に切り替えると、　　UIが小さくなってしまう

Canvas Scaler で想定サイズを入力すると、　　4K に切り替えても、　　UIがのサイズを保持する

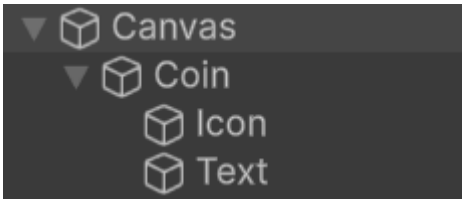


今回、上記の設定で進めましょう。

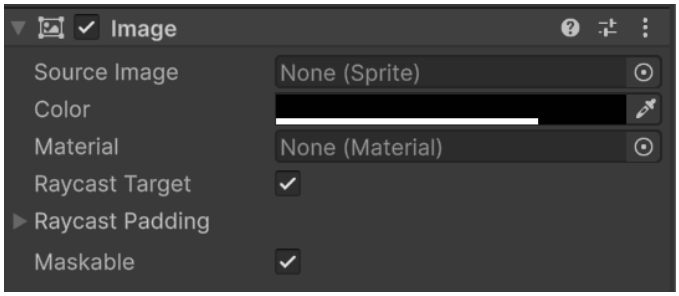
?????

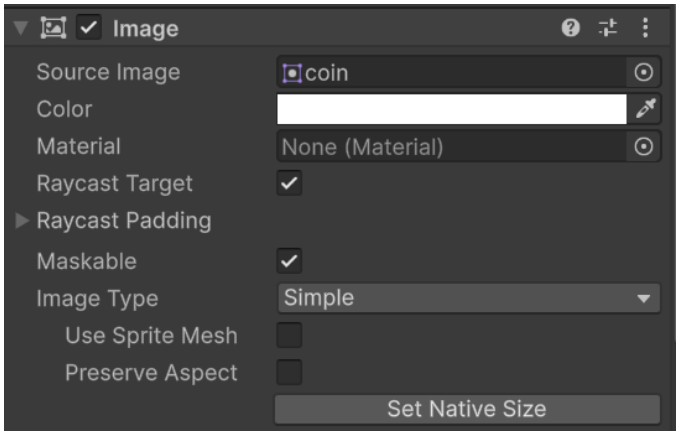
画面の右上にコインの数を表示しましょう。

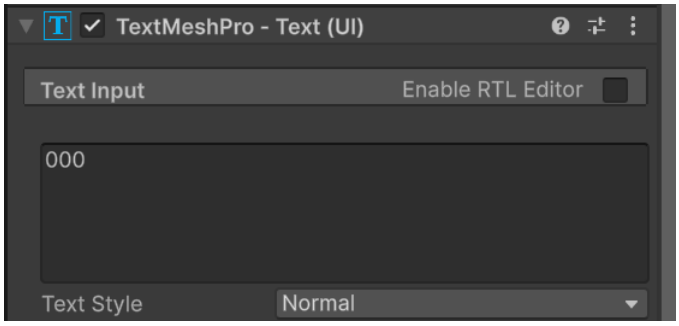
Canvas の中で、以下のようにヒエラルキーを作ってください:











こうなるように:

- Coin: Image
  （画像）にし、背景は半透明の黒にする。この「部品」が画面の右上に表示されるので、アンカーポイントが右上にしてください。

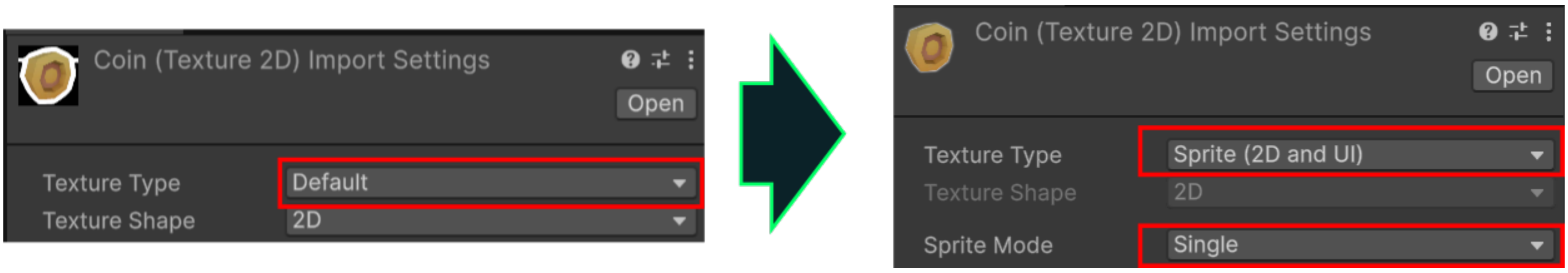
- Icon: Image
  （画像）にし、コインのアイコンにする

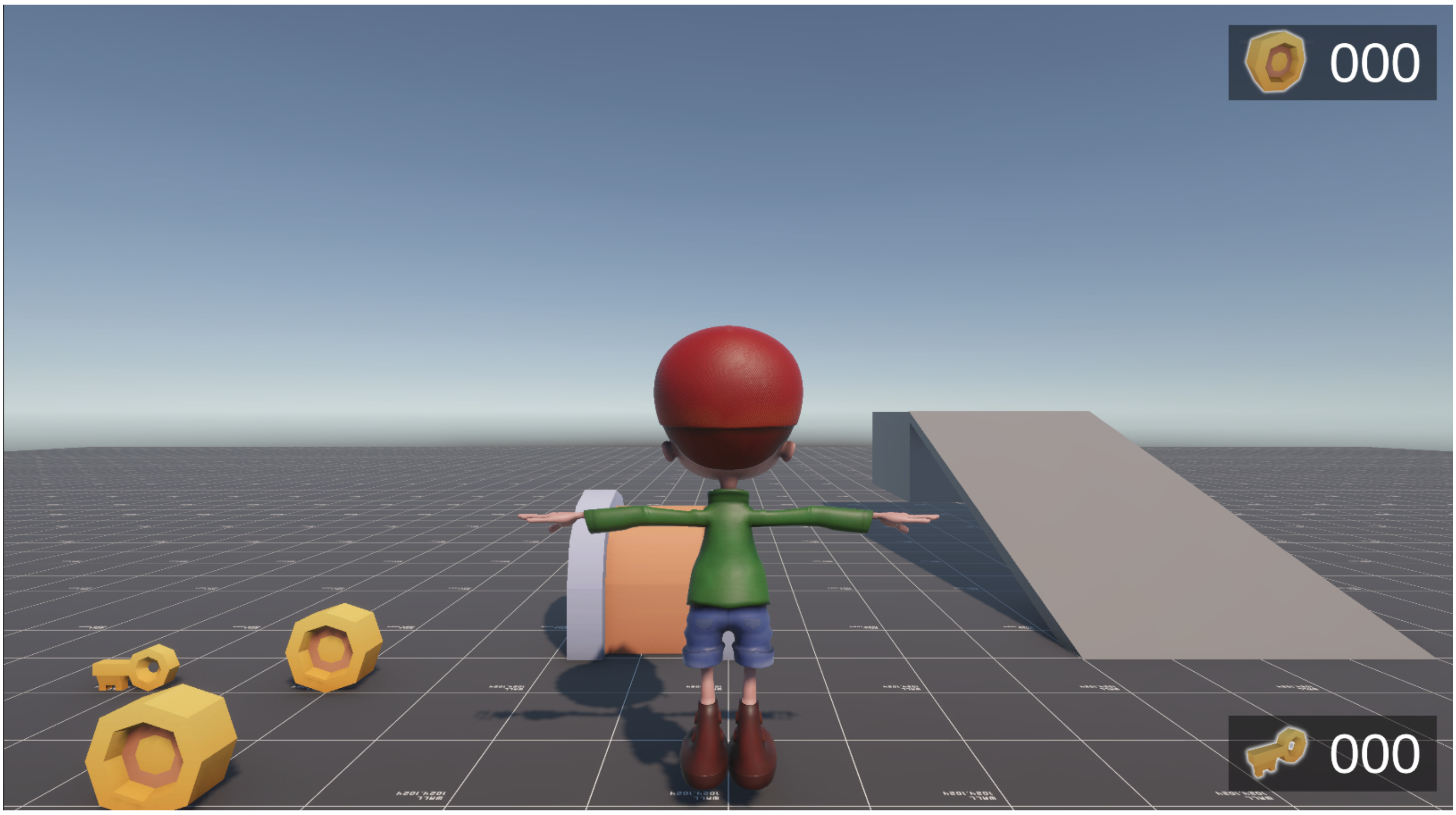
- Text: TextMeshPro にし、 000
  ので始まるようにする

Coin?????????

3Dゲームを作るときに、すべての画像が 3Dモデルで使う「 Texture (テクスチャ)
」と思われる。テクスチャが UI で使えないので、 スプライト になるように設定しなければならない。

UIで使う画像を選択し、 Inspector で設定を変えてください:





これで、「 Sprite」（ 2Dゲームと UI）にし、スプライトモードは「 Single」（単体）にする。

※：同じ画像にたくさんのスプライトがある場合はここは「 Multiple」（多数）になる

???

同じ手順を従って、画面の右下に鍵の数が表示できる UIを作成してください

?????

UI管理するスクリプト「 UIManager」を作成し、コインの数と鍵の数を更新するメソッドを実装しましょう。まず、必要なのは、テキストの参照なので：

```
// UIを管理する
public class UIManager : MonoBehaviour
{
```

```
    // コインのテキスト
    [SerializeField]
    private TextMeshProUGUI coinText;

    // 鍵のテキスト
    [SerializeField]
    private TextMeshProUGUI keyText;
}
```

そして、アイテムのタグで、数を設定できるメソッド:

```
// アイテム（コイン、鍵）の数を設定
public void SetCount(string item, int count)
{
    // タグで確認
    switch (item)
    {
        // コイン
        case "Coin":
            coinText.text = $"{count:000}";
            break;
        // 鍵
        case "Key":
            keyText.text = $"{count:000}";
            break;
    }
}
```

Inventory???

そして、 Inventory スクリプトを更新し、鍵、またはコインを拾ったら、 UIを更新しましょう。まず、 Unity で紐づけできるように:

```
// アイテムを管理するインベントリ
public class Inventory : MonoBehaviour
{
    // UIを更新するため
    [SerializeField]
    private UIManager uiManager;

    //（省略）
```

```
}
```

そして、拾うときに:

```
// トリガーが発動した
private void OnTriggerEnter(Collider other)
{
    // 相手のタグを求める
    string itemTag = other.tag;

    // そのタグあるかないかを確認
    // "ContainsKey" は、ディクショナリーの中に
    // 定めた「キー」があると、「true」を返す
    bool isItem = itemCount.ContainsKey(itemTag);

    // 有効なアイテじゃなければ、何もしない
    if (!isItem)
        return; // 終了

    // コライダー「の」トランスフォーム「の」親「の」ゲームオブジェクト
    Destroy(other.transform.parent.gameObject);
    itemCount[itemTag]++;

    // UIを更新
    uiManager.SetCount(itemTag, itemCount[itemTag]);
}
```

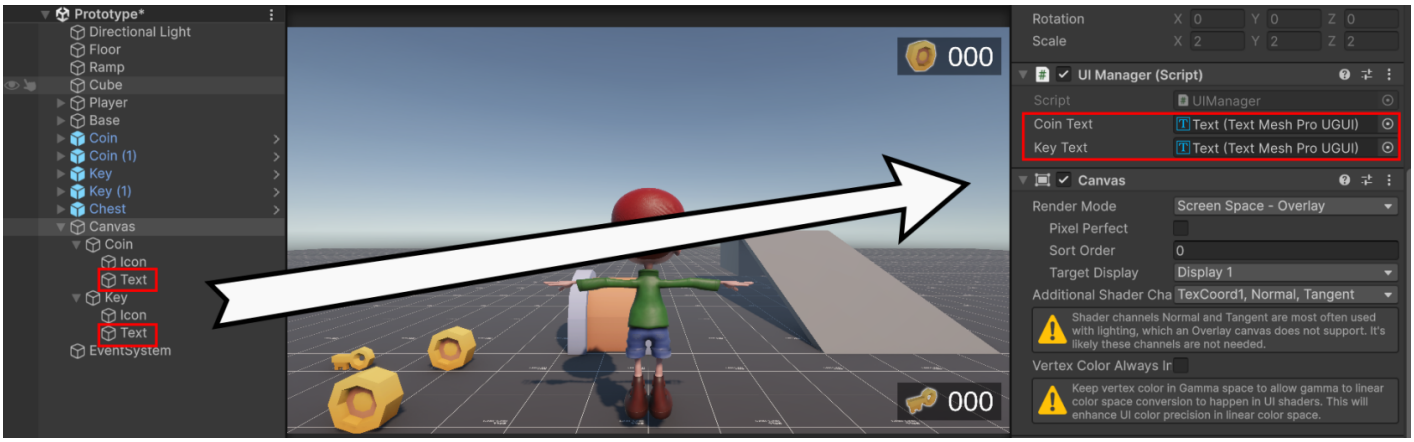
もちろん、消費のときも　　...

```
// アイテムを消費する
public void UseItem(string item, int count)
{
    // 現在の数を確認
    int nowCount = GetCount(item);

    // 足りたら、消費する
    if (nowCount >= count)
        itemCount[item] -= count;

    // UIを更新
```

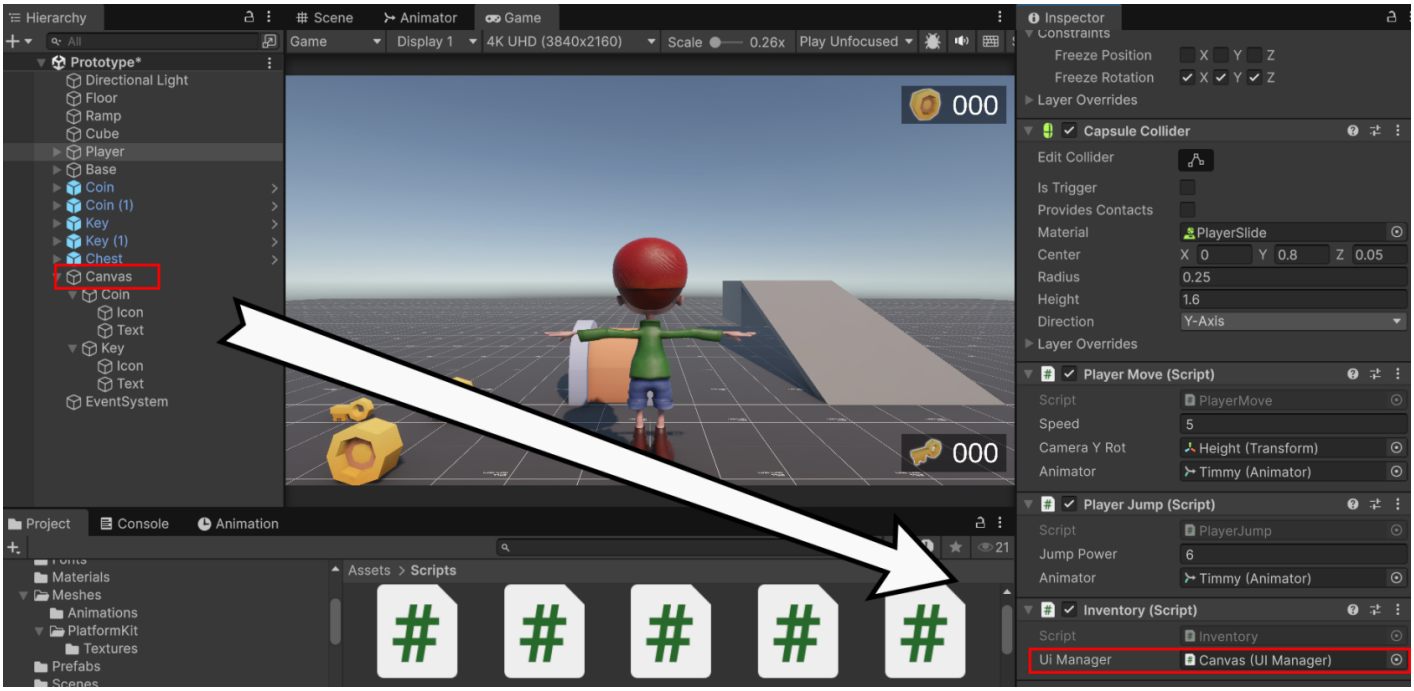
```
    uiManager.SetCount(item, itemCount[item]);
}
```

????????

□□ Canvas □ UIManager □□□□□□□□□□□□□□□□□□



□□□□ Inventory □ UIManager □□□□□



□□□□□□□□□□□□□□□

Revision #3
Created 2026-05-04 08:14:06 UTC by Menendez Francisco
Updated 2026-06-02 06:29:41 UTC by Menendez Francisco