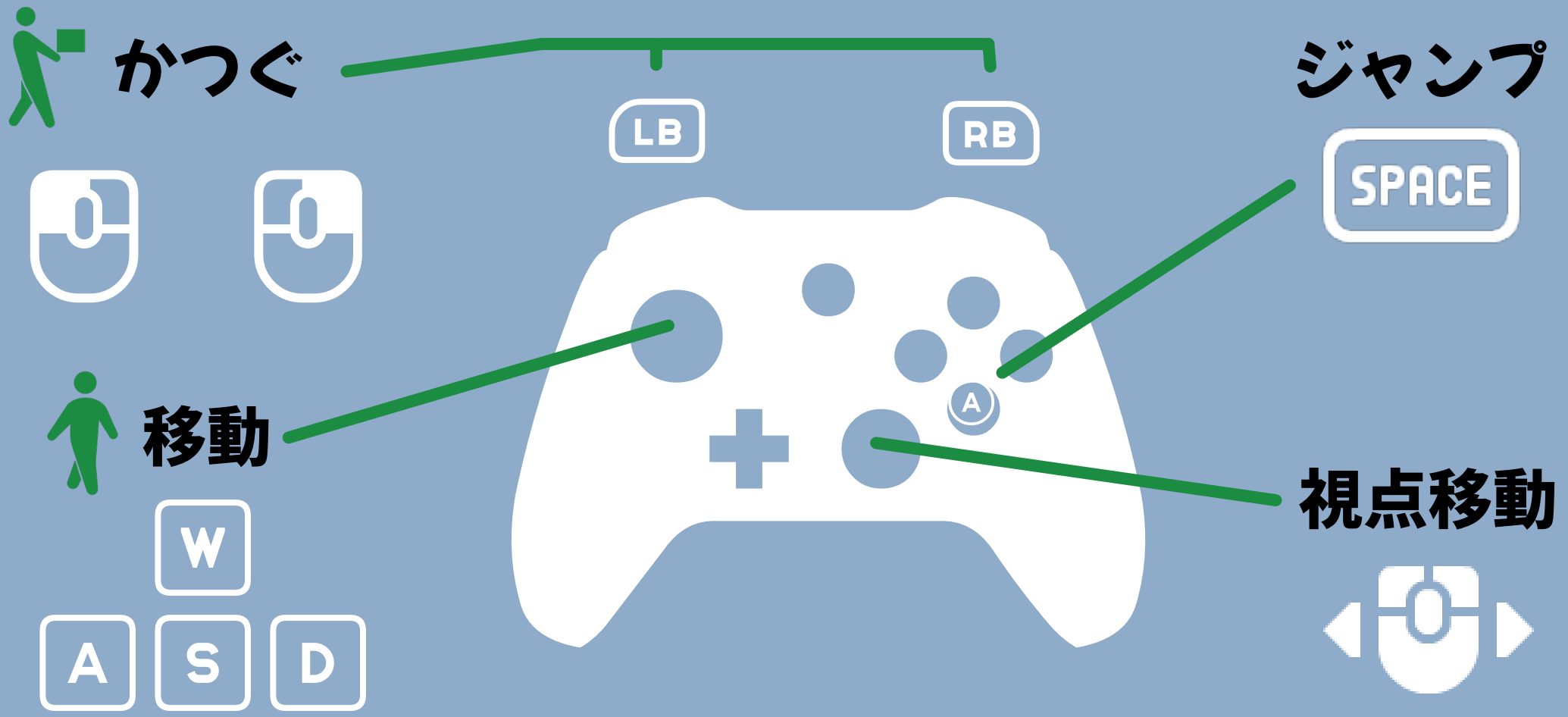
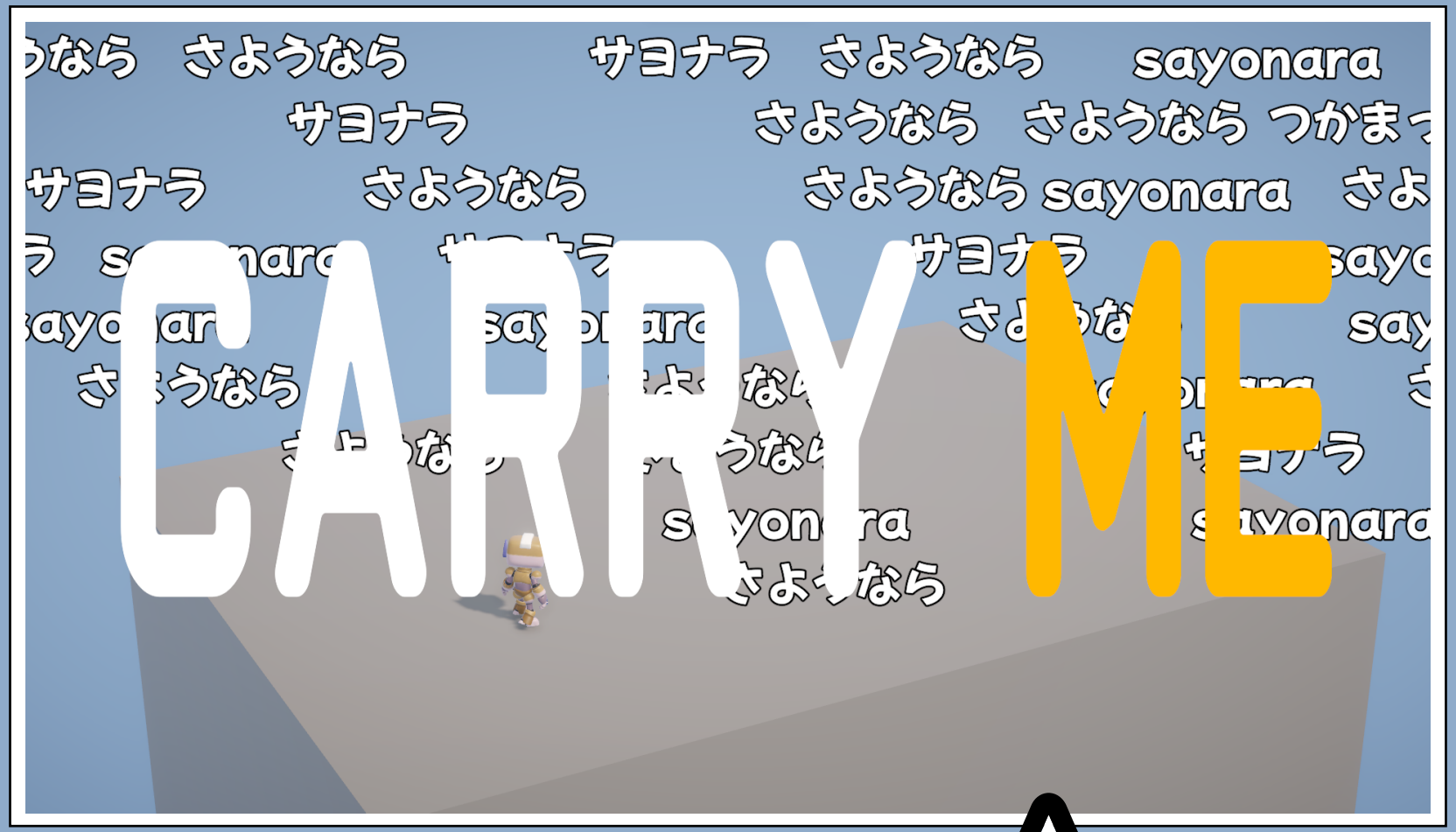
操作説明



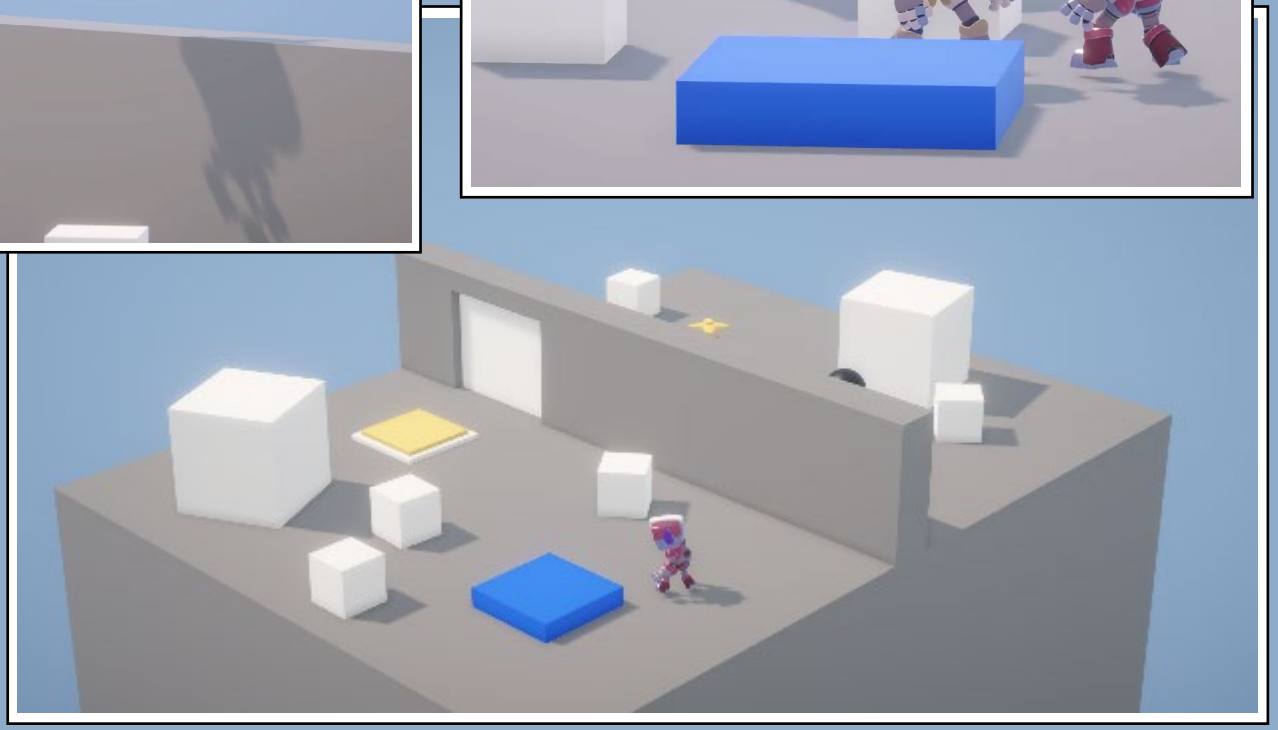


「担ぐ」をテーマに２週間でゲームを作る
というクラスの課題で制作したゲームです。

ゲーム概要

パートナーや
アイテムを担いで
ステージをクリアする
謎解きパズルゲーム

青いボタンを踏めば
次のステージへ！



入力の複製

- InputSystemを介したデバイスの入力を
  複数のキャラクターに反映するために
  UniRxというライブラリを使用しています。

```
6
7       // PlayerInputの入力を監視対象の変数として保持する
8
        2 個の参照
9       public class PlayerModel
10      {
11          ReactiveProperty<Vector2> m_inputMove = new ReactiveProperty<Vector2>(Vector2.zero);
12          ReactiveProperty<bool> m_inputJump = new ReactiveProperty<bool>(false);
13          ReactiveProperty<bool> m_inputCarry1 = new ReactiveProperty<bool>(false);
14          ReactiveProperty<bool> m_inputCarry2 = new ReactiveProperty<bool>(false);
15
            1 個の参照
16          public IReadOnlyReactiveProperty<Vector2> InputMove { get { return m_inputMove; } }
            1 個の参照
17          public IReadOnlyReactiveProperty<bool> InputJump { get { return m_inputJump; } }
            1 個の参照
18          public IReadOnlyReactiveProperty<bool> InputCarry1 { get { return m_inputCarry1; } }
            1 個の参照
19          public IReadOnlyReactiveProperty<bool> InputCarry2 { get { return m_inputCarry2; } }
20
21          PlayerInput m_playerInput;
22
            1 個の参照
23          public void SetPlayerInput(PlayerInput playerInput)
```

Assets/Player/PlayerModel

```
32
33      // modelの値の変更を監視して、複数のプレイヤーに通知
        1 個の参照
34      void Bind()
35      {
36          foreach(Player player in m_playerList)
37          {
38              m_playerModel.InputMove
39                  .Subscribe(player.OnMove)
40                  .AddTo(player.gameObject);
41
42              m_playerModel.InputJump
43                  .Subscribe(_ => player.OnJump())
44                  .AddTo(player.gameObject);
45
46              if(player.MainCharactor)
47              {
48                  m_playerModel.InputCarry1
49                      .Subscribe(_ => player.OnCarryAndDrop())
50                      .AddTo(player.gameObject);
51              }
52              else
53              {
54                  m_playerModel.InputCarry2
55                      .Subscribe(_ => player.OnCarryAndDrop())
56                      .AddTo(player.gameObject);
57              }
58          }
59      }
```

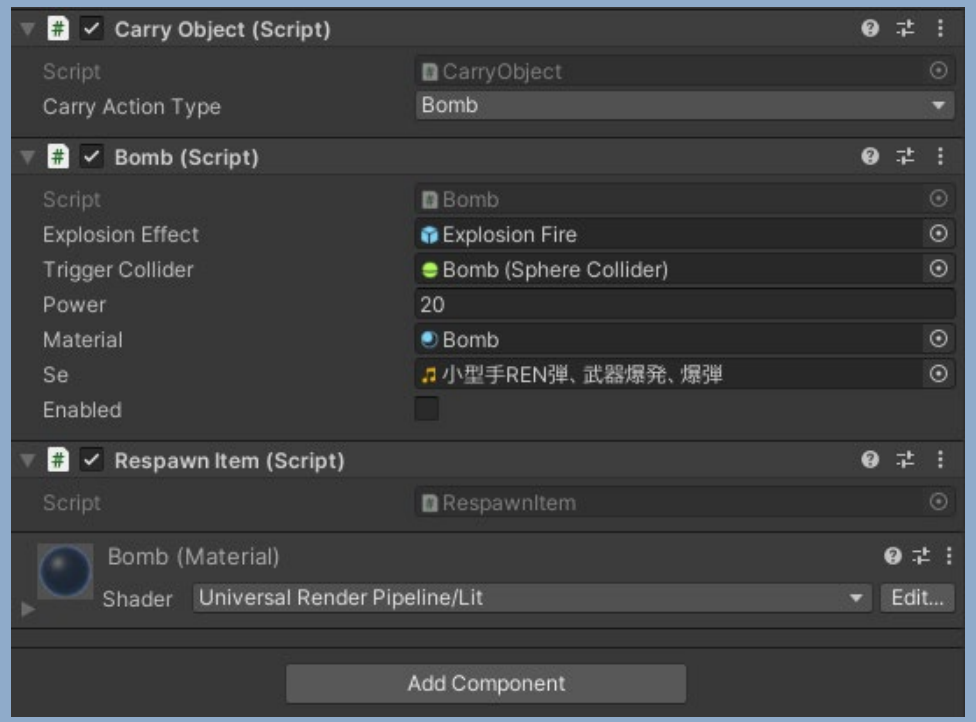
Assets/Player/PlayerPresenter

ポリモーフィズム

- オブジェクトを担いだ時の処理を
  簡単に追加、編集できるように
  ポリモーフィズムを用いて実装しています。

```
4
5   // このスクリプトがついていれば、プレイヤーが担ぐことが出来る
6
    Unity スクリプト (28 件のアセット参照)|15 個の参照
7   public class CarryObject : MonoBehaviour
8   {
9       [SerializeField] CarryActionType m_carryActionType; // 担がれたときの処理を指定
10      CarryAction m_carryAction;                          // 担がれたときのアクションの基底クラス
11
38
        2 個の参照
39      public void OnCarry(GameObject parent)
40      {
41          m_carryAction.OnStart(gameObject, parent);
42          m_isCarrying = true;
43      }
44
        Unity メッセージ|0 個の参照
45      private void Update()
46      {
47          if(m_isCarrying)
48          {
49              m_carryAction.OnUpdate();
50          }
51      }
52
        2 個の参照
53      public void OnDrop()
54      {
55          m_carryAction.OnEnd();
56          m_isCarrying = false;
57      }
58  }
```

Assets/Carry/CarryObject

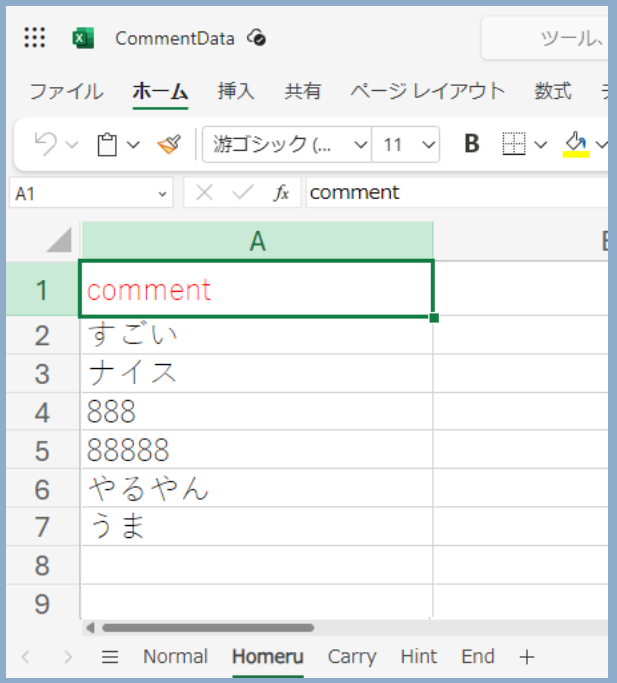


Assets/Stage
/Bomb(PrefabAsset)

コメント欄

- 画面のにぎやかしのために
  コメントが流れる機能を
  実装しています。

- コメントはエクセルで
  管理しているため、
  プログラマでなくとも
  編集しやすくなっています。



Assets/Comment/
CommentData.xlsx

課題

・使用するアセットを絞ったため見た目がやや寂しい
→Blenderで簡単なオブジェクトは
自作できるように勉強していきます。

・一部のスクリプトは期限に間に合わせるために
かなり雑な実装になってしまった
→速度との両立をできるように、
より多くのゲームを制作していきます。