2020年度
Unity講座(基礎編)

07回目

講師：幸田 将伍 (@MagurodonDev)

今回の講義の目的

- プログラムを自分で読めるようになる
- Unityを使って自分が実現したいことをできるようになる
- 自分一人でもゲームを作成できるレベルになる
- Unityの活用事例を学び、自分の進路に役立てる
- 実際のエンジニアがどういった仕事の進め方をしているかを知る
- ゲーム会社のクライアントエンジニアとして就職できるレベルになる

一緒にレベルアップして行きましょう！

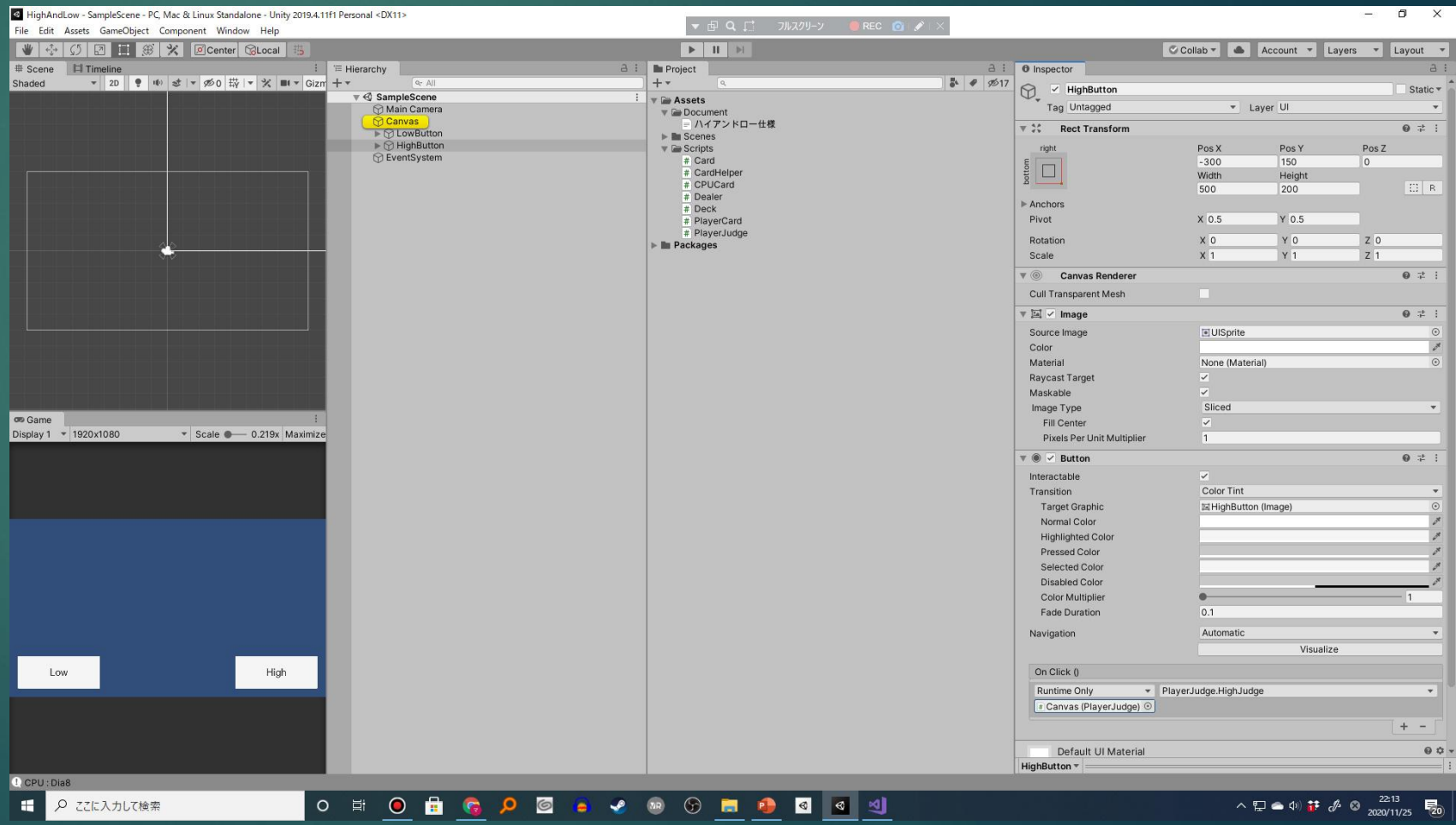
Unity

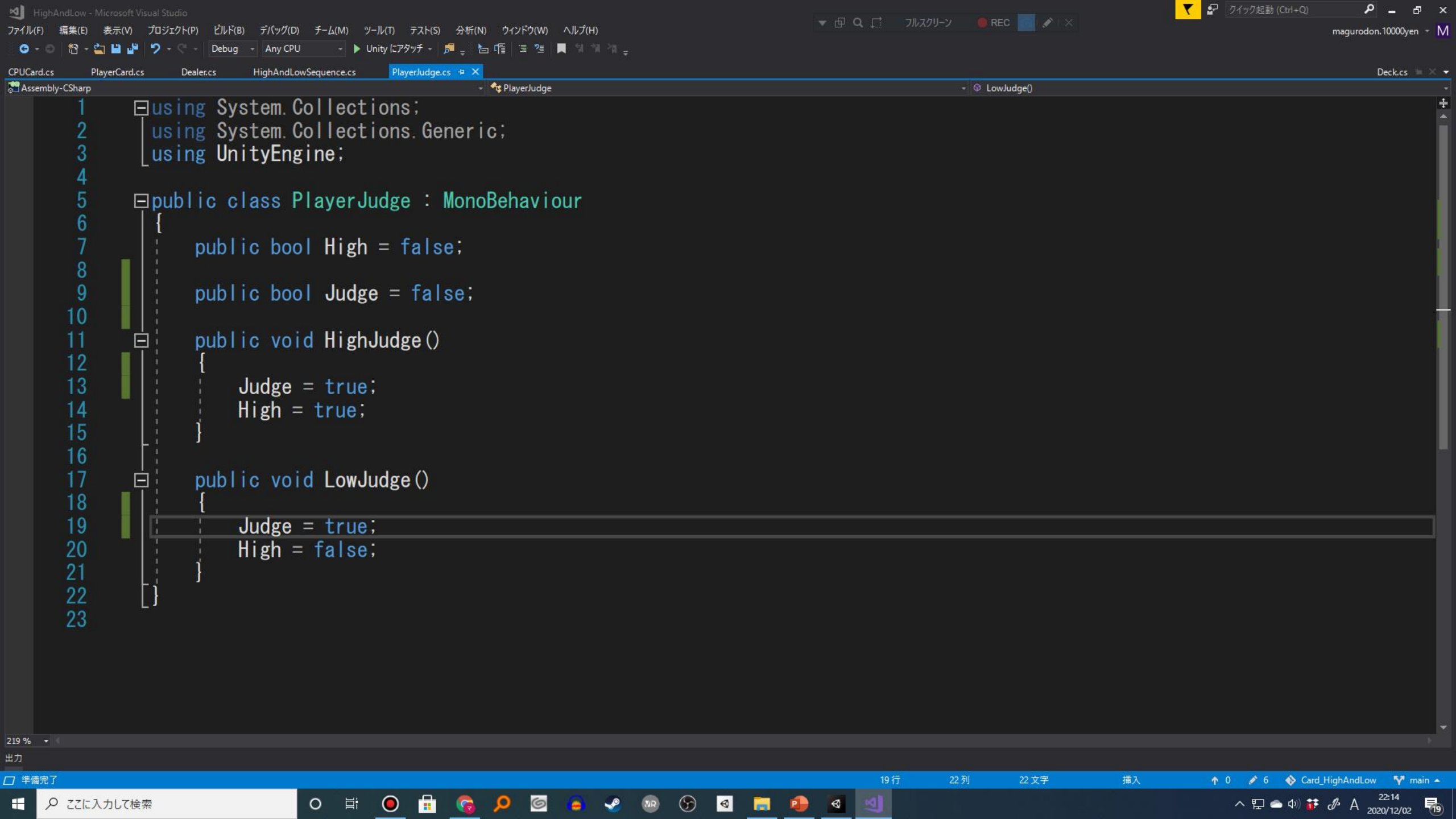
ハイアンドロー

- 前回はシーケンスのスクリプトを作成しました
- 今回はシーケンスを完成させたうえで、トランプの描画をしていきます

- PlayerJudge.cs
- PlayerCard.cs
- CPUCard.cs
- HighAndLowSequence.cs

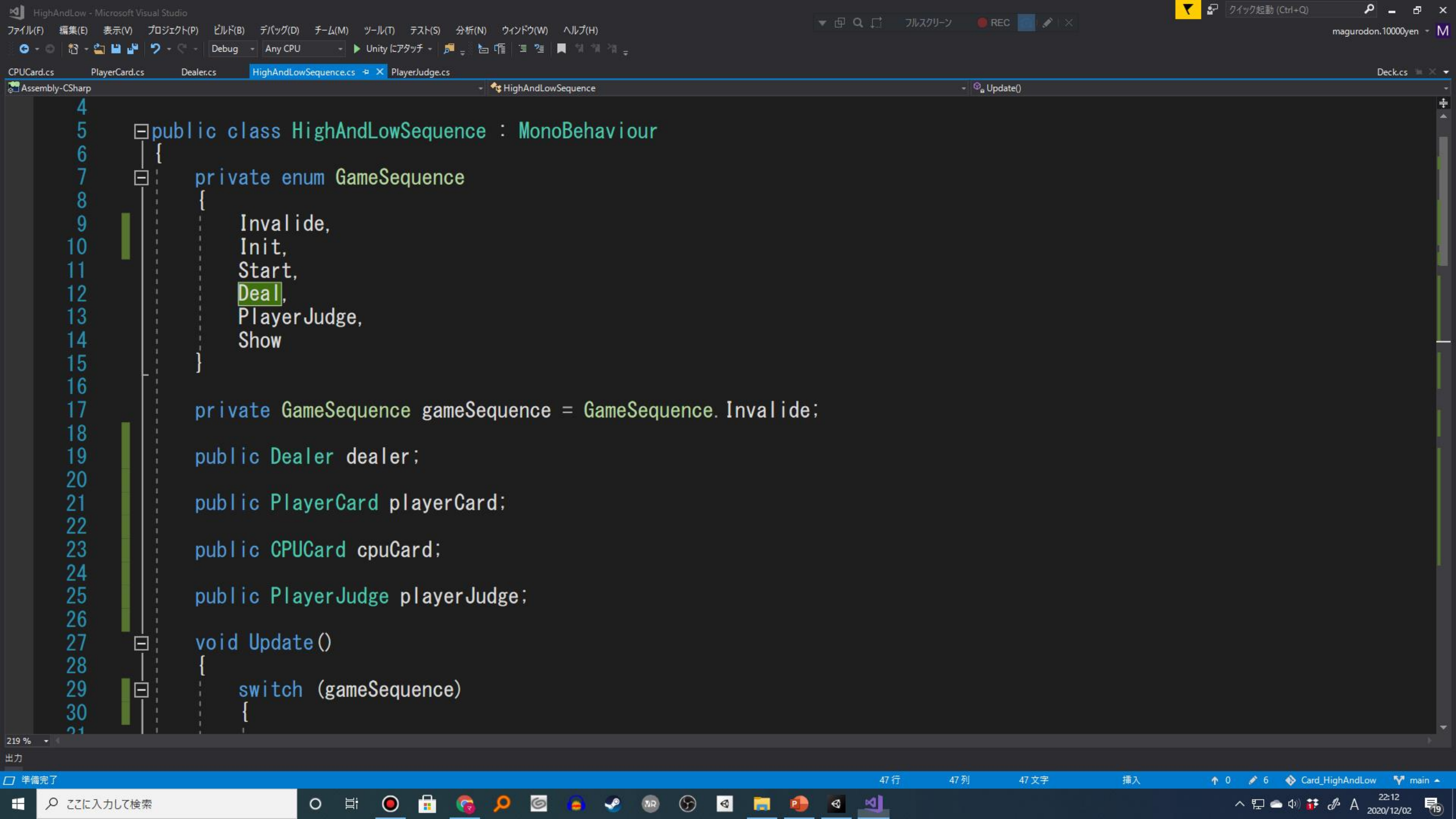
- ざっとこのスクリプトたちを修正していきましょう




HighAndLow - Microsoft Visual Studio
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
Debug
Any CPU
Unity にアタッチ
CPUCard.cs
PlayerCard.cs
Dealer.cs
HighAndLowSequence.cs
PlayerJudge.cs
Deck.cs
Assembly-CSharp
PlayerJudge
LowJudge()
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class PlayerJudge : MonoBehaviour
{
public bool High = false;
public bool Judge = false;
public void HighJudge()
{
Judge = true;
High = true;
}
public void LowJudge()
{
Judge = true;
High = false;
}
}
219 %
出力
準備完了
19 行
22 列
22 文字
挿入
Card_HighAndLow
main
ここに入力して検索
22:14
2020/12/02

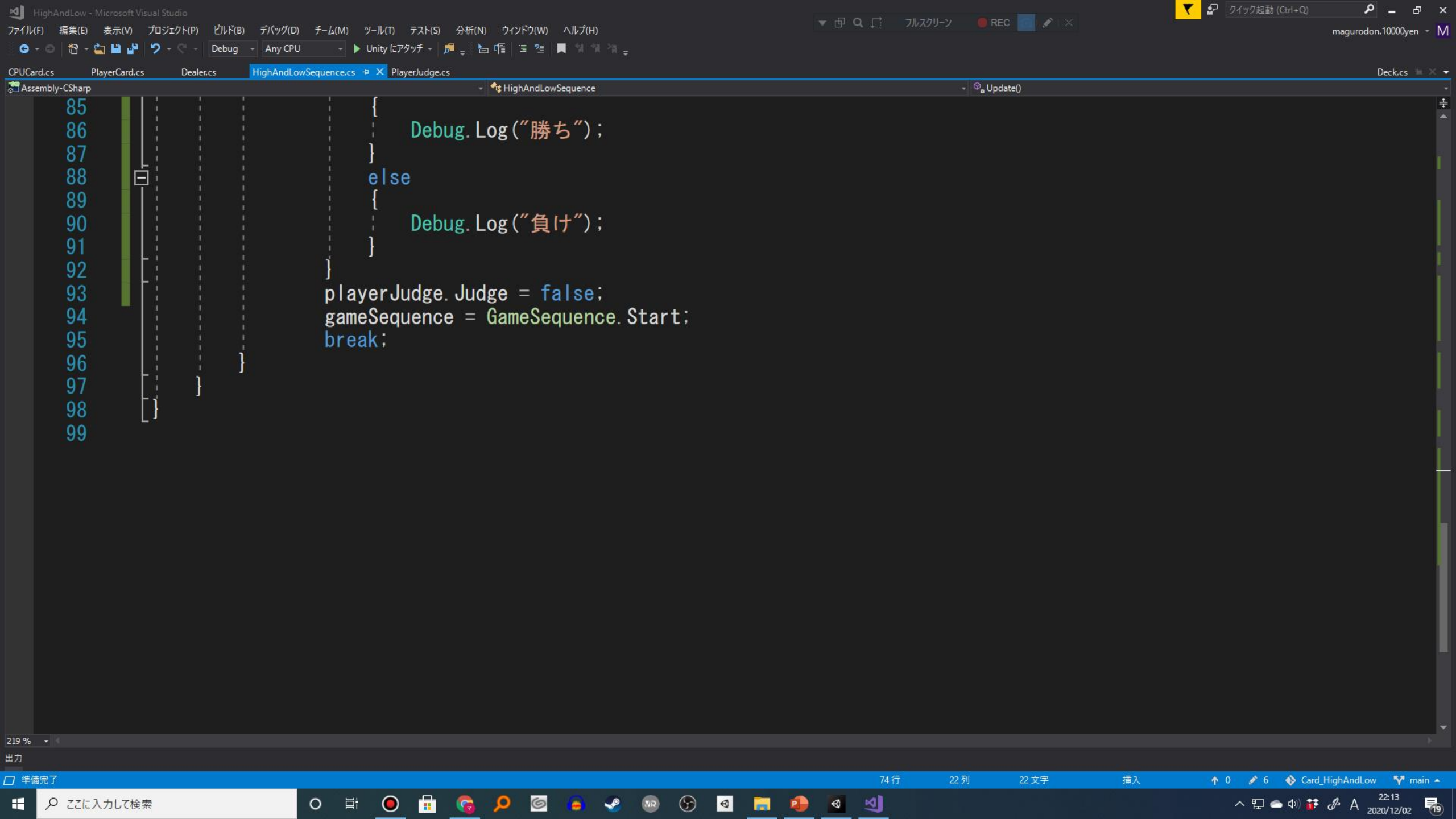
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerCard : MonoBehaviour
{
    public Dealer Dealer;

    private List<Card> playerDeck = new List<Card>();

    public Card playerCard;

    public void SetPlayerDeck()
    {
        playerDeck = Dealer.GetPlayerDeck();
    }

    public void SetPlayerCard()
    {
        playerCard = Deck.GetCard(playerDeck);
        Debug.Log(playerCard.Number);
    }

}
```

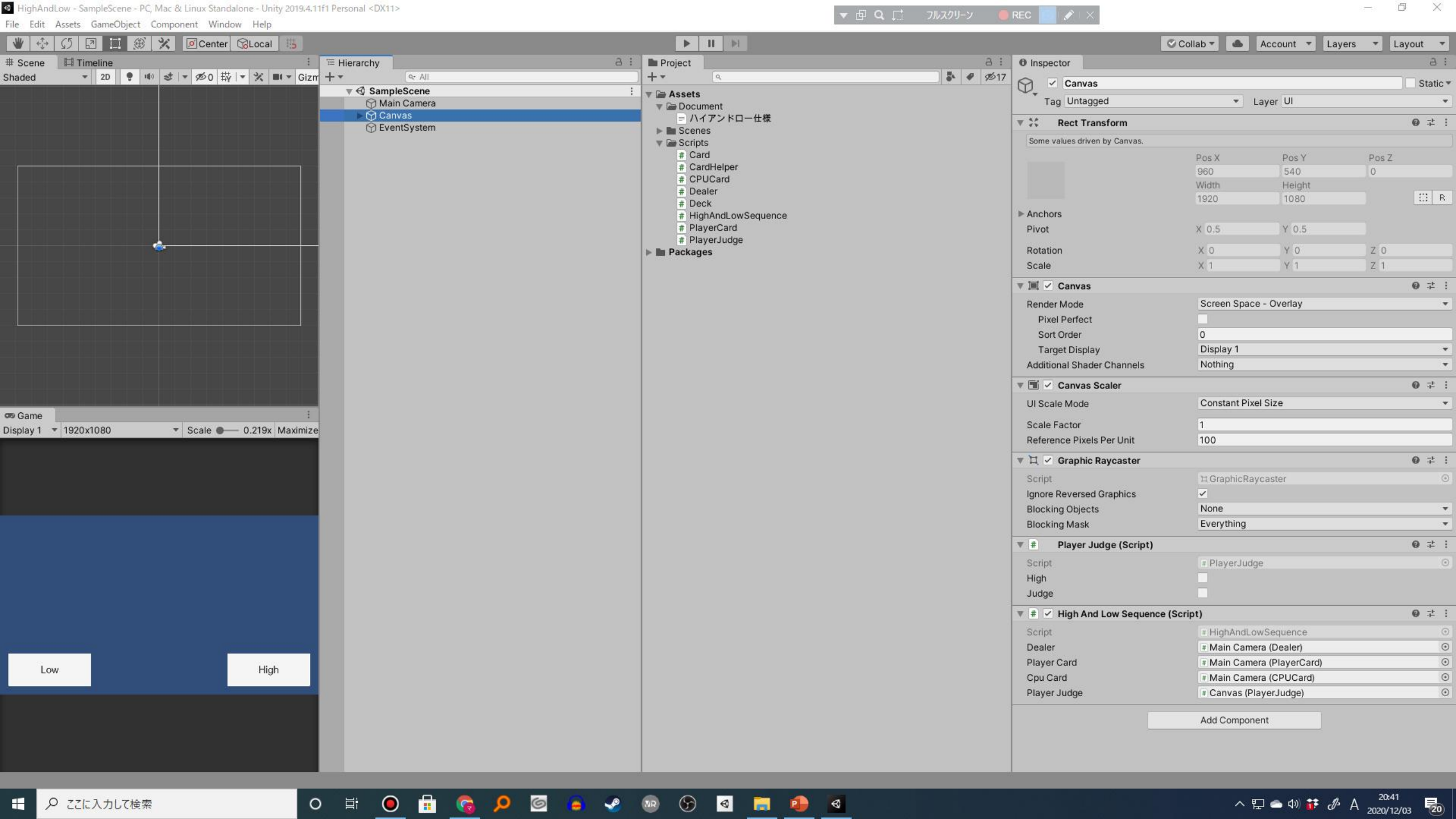
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CPUCard : MonoBehaviour
{
    public Dealer Dealer;

    private List<Card> cpuDeck = new List<Card>();

    public Card cpuCard;

    public void SetCPUDeck()
    {
        cpuDeck = Dealer.GetCPUDeck();
    }

    public void SetCPUCard()
    {
        cpuCard = Deck.GetCard(cpuDeck);
        Debug.Log(cpuCard.Number);
    }
}
```


HighAndLow - Microsoft Visual Studio
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
Debug
Any CPU
Unity にアタッチ
CPUCard.cs
PlayerCard.cs
Dealer.cs
HighAndLowSequence.cs
PlayerJudge.cs
Deck.cs
Assembly-CSharp
HighAndLowSequence
Update()
public class HighAndLowSequence : MonoBehaviour
{
private enum GameSequence
{
Invalide,
Init,
Start,
Deal,
PlayerJudge,
Show
}
private GameSequence gameSequence = GameSequence.Invalide;
public Dealer dealer;
public PlayerCard playerCard;
public CPUCard cpuCard;
public PlayerJudge playerJudge;
void Update()
{
switch (gameSequence)
{
219 %
出力
準備完了
47 行
47 列
47 文字
挿入
Card_HighAndLow
main
ここに入力して検索
22:12
2020/12/02

```
        case GameSequence.Invalide:

            gameSequence = GameSequence.Init;
            break;

        case GameSequence.Init:

            // PlayerとCPUにデッキをディールする
            playerCard.SetPlayerDeck();
            cpuCard.SetCPUDeck();
            gameSequence = GameSequence.Deal;
            break;

        case GameSequence.Start:

            gameSequence = GameSequence.Deal;
            break;
        case GameSequence.Deal:

            // PlayerとCPUにカードをディールする
            playerCard.SetPlayerCard();
            cpuCard.SetCPUCard();

            Debug.Log(playerCard.playerCard.Number);
            Debug.Log(cpuCard.cpuCard.Number);
            gameSequence = GameSequence.PlayerJudge;
```

```
            break;
        case GameSequence.PlayerJudge:

            // 数を予想してボタンを押したらShowに進む
            if (playerJudge.Judge)
            {
                gameSequence = GameSequence.Show;
            }
            break;
        case GameSequence.Show:

            // プレイヤーが確認したらStartに戻って次のゲーム

            if (playerJudge.High)
            {
                if (playerCard.playerCard.Number > cpuCard.cpuCard.Number)
                {
                    Debug.Log("勝ち");
                }
                else
                {
                    Debug.Log("負け");
                }
            }
            else
            {
                if (playerCard.playerCard.Number < cpuCard.cpuCard.Number)
```

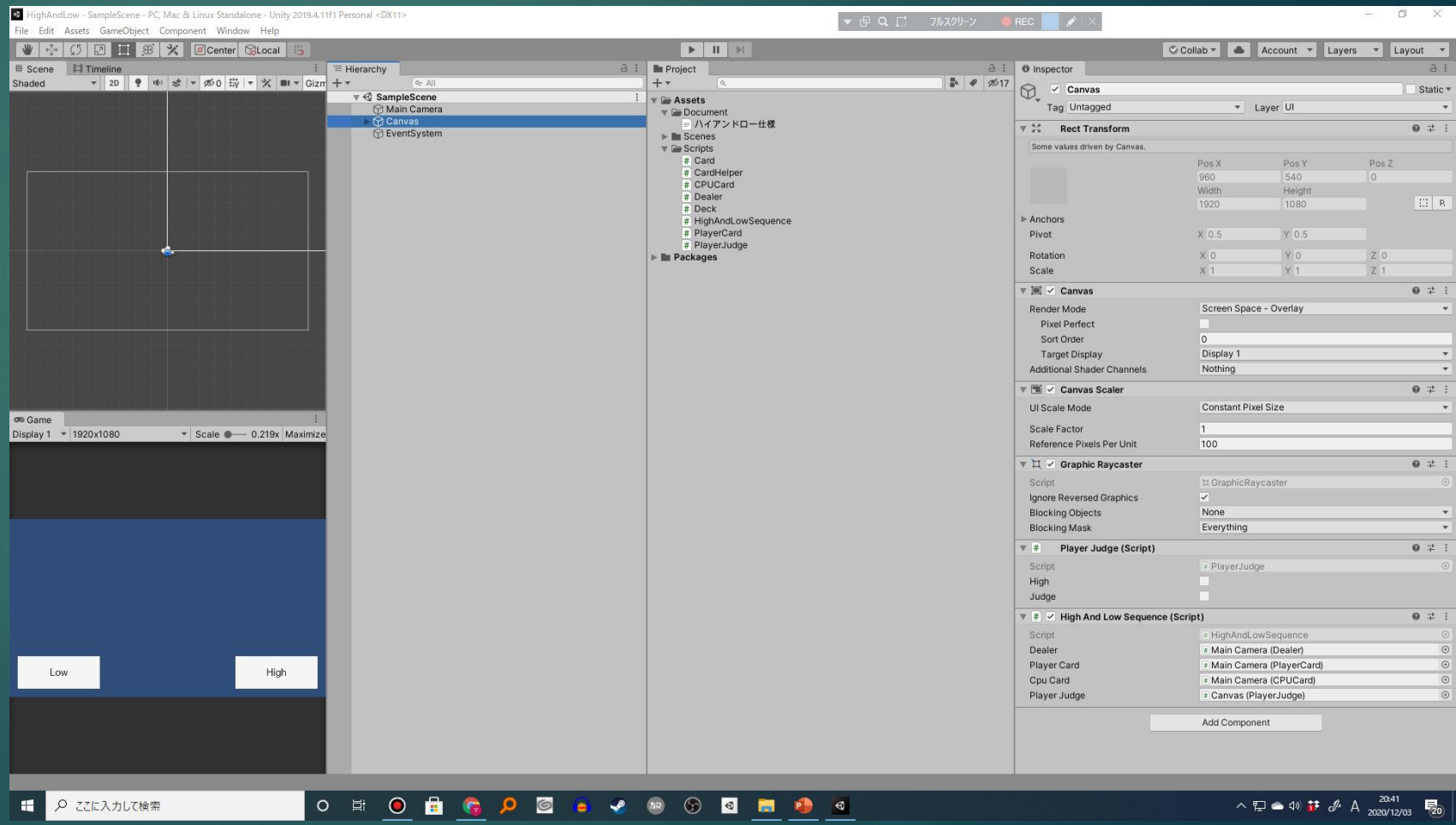

HighAndLow - Microsoft Visual Studio
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
Debug
Any CPU
Unity にアタッチ
CPUCard.cs
PlayerCard.cs
Dealer.cs
HighAndLowSequence.cs
PlayerJudge.cs
Deck.cs
Assembly-CSharp
HighAndLowSequence
Update()
85 {
86 Debug.Log("勝ち");
87 }
88 else
89 {
90 Debug.Log("負け");
91 }
92 }
93 playerJudge.Judge = false;
94 gameSequence = GameSequence.Start;
95 break;
96 }
97 }
98 }
99
219 %
出力
準備完了
74 行
22 列
22 文字
挿入
Card_HighAndLow
main
ここに入力して検索
22:13
2020/12/02

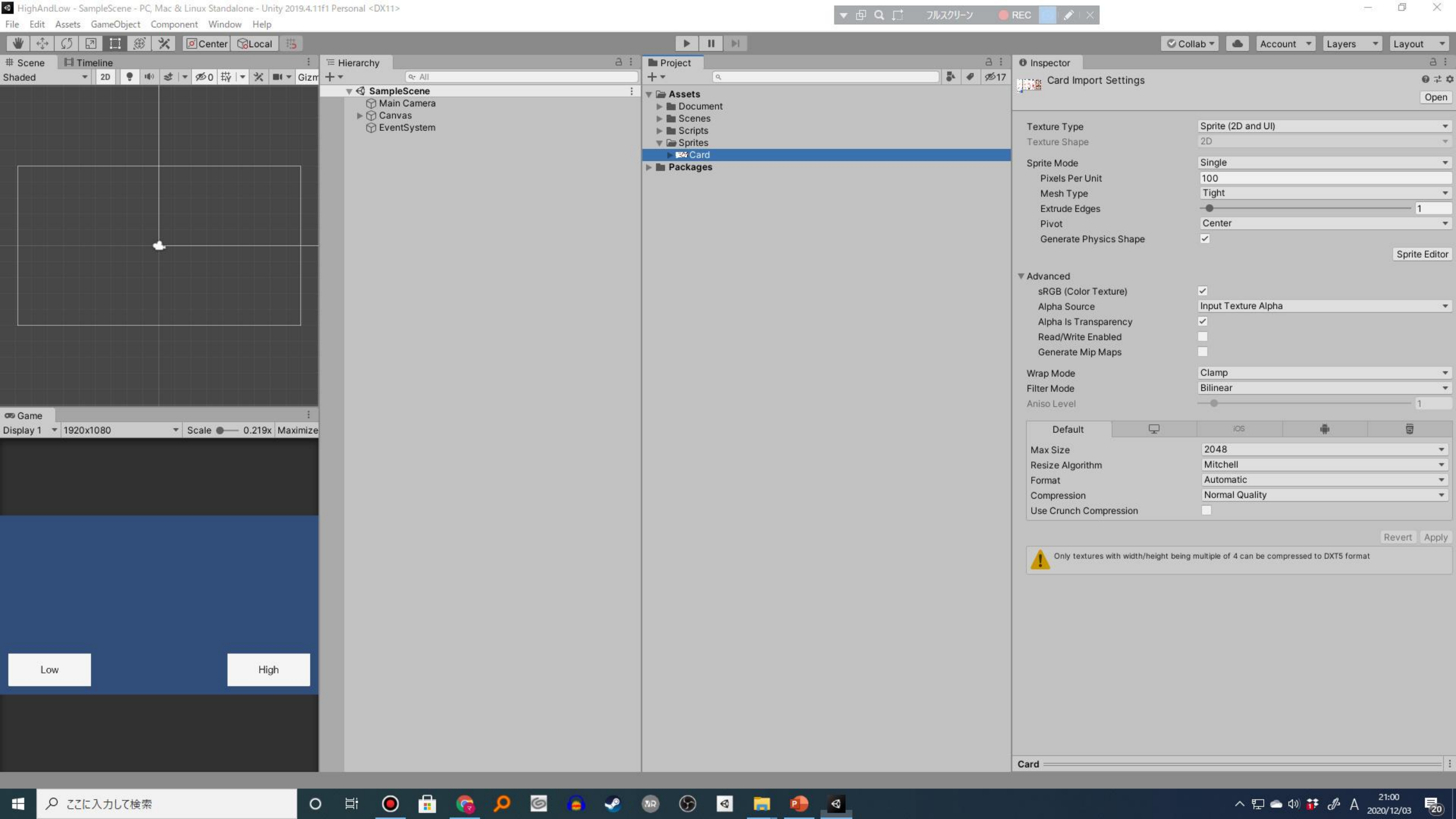

HighAndLow - SampleScene - PC, Mac & Linux Standalone - Unity 2019.4.11f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
フルスクリーン REC
Center Local
Collab Account Layers Layout
Scene Timeline
Shaded 2D
Hierarchy
All
SampleScene
Main Camera
Canvas
EventSystem
Project
Assets
Document
ハイアンドロー仕様
Scenes
Scripts
Card
CardHelper
CPUCard
Dealer
Deck
HighAndLowSequence
PlayerCard
PlayerJudge
Packages
Inspector
Canvas
Static
Tag Untagged
Layer UI
Rect Transform
Some values driven by Canvas.
Pos X 960
Pos Y 540
Pos Z 0
Width 1920
Height 1080
Anchors
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Canvas
Render Mode Screen Space - Overlay
Pixel Perfect
Sort Order 0
Target Display Display 1
Additional Shader Channels Nothing
Canvas Scaler
UI Scale Mode Constant Pixel Size
Scale Factor 1
Reference Pixels Per Unit 100
Graphic Raycaster
Script GraphicRaycaster
Ignore Reversed Graphics
Blocking Objects None
Blocking Mask Everything
Player Judge (Script)
Script PlayerJudge
High
Judge
High And Low Sequence (Script)
Script HighAndLowSequence
Dealer Main Camera (Dealer)
Player Card Main Camera (PlayerCard)
Cpu Card Main Camera (CPUCard)
Player Judge Canvas (PlayerJudge)
Add Component
Game
Display 1 1920x1080 Scale 0.219x Maximize
Low
High
ここに入力して検索
20:41
2020/12/03

Unity

ハイアンドロー

- Unityでの設定を終わらせてプレイしてみてください
- Consoleを見ながら、勝ち負けが正しく反映されているのを確認します
- さて、では実際にカードを表示していきましょう
- Assetsフォルダ直下に新しく”Sprites”というフォルダを作成してください
- 本日お配りしたCard.pngをその中にいれます



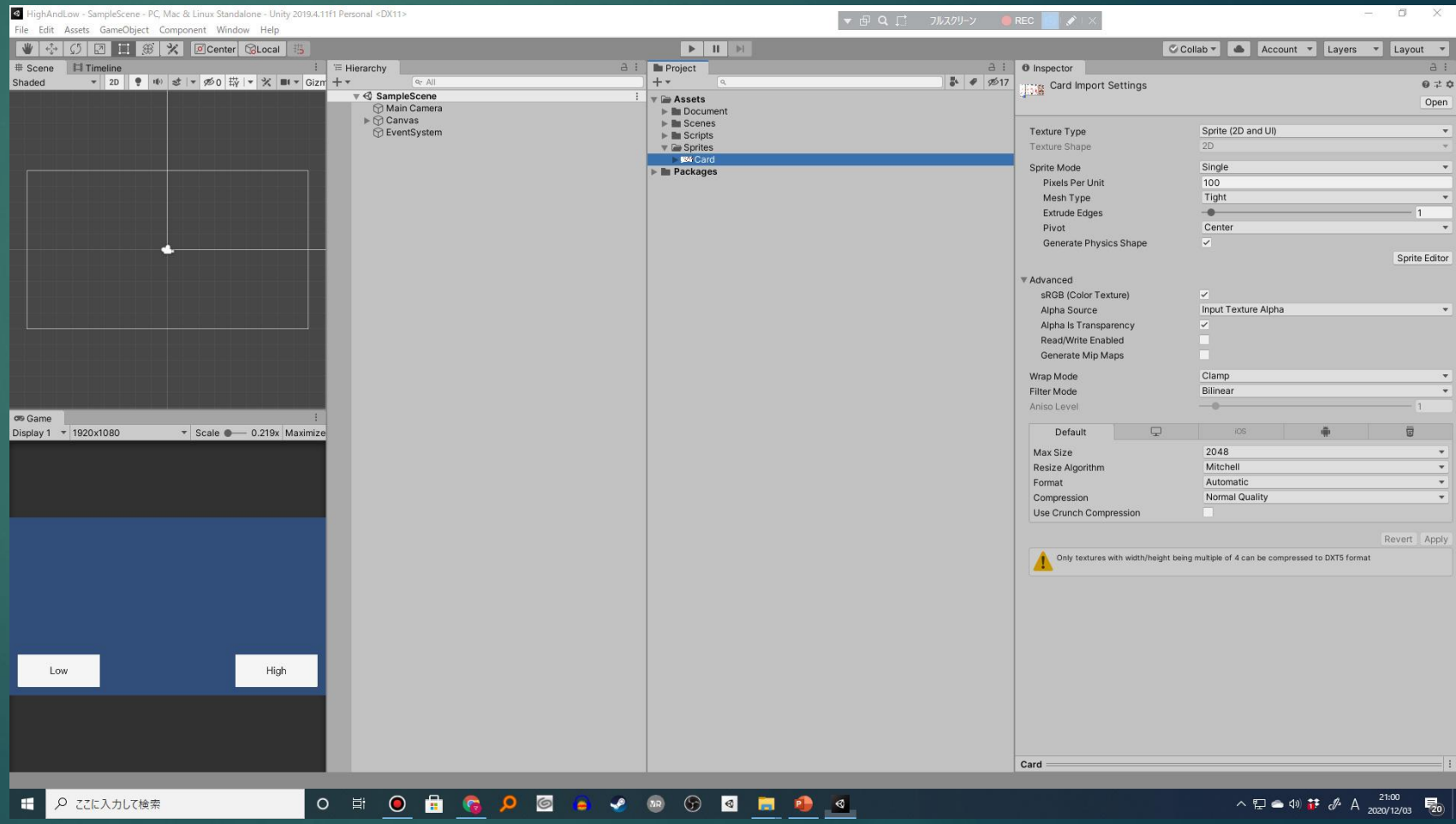

HighAndLow - SampleScene - PC, Mac & Linux Standalone - Unity 2019.4.11f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
フルスクリーン REC
Center Local
Collab Account Layers Layout
Scene Timeline
Shaded 2D
Hierarchy
All
SampleScene
Main Camera
Canvas
EventSystem
Project
Assets
Document
Scenes
Scripts
Sprites
Card
Packages
Inspector
Card Import Settings
Open
Texture Type Sprite (2D and UI)
Texture Shape 2D
Sprite Mode Single
Pixels Per Unit 100
Mesh Type Tight
Extrude Edges 1
Pivot Center
Generate Physics Shape
Sprite Editor
Advanced
sRGB (Color Texture)
Alpha Source Input Texture Alpha
Alpha Is Transparency
Read/Write Enabled
Generate Mip Maps
Wrap Mode Clamp
Filter Mode Bilinear
Aniso Level 1
Default iOS
Max Size 2048
Resize Algorithm Mitchell
Format Automatic
Compression Normal Quality
Use Crunch Compression
Revert Apply
Only textures with width/height being multiple of 4 can be compressed to DXT5 format
Card
Game
Display 1 1920x1080 Scale 0.219x Maximize
Low
High
ここに入力して検索
21:00
2020/12/03

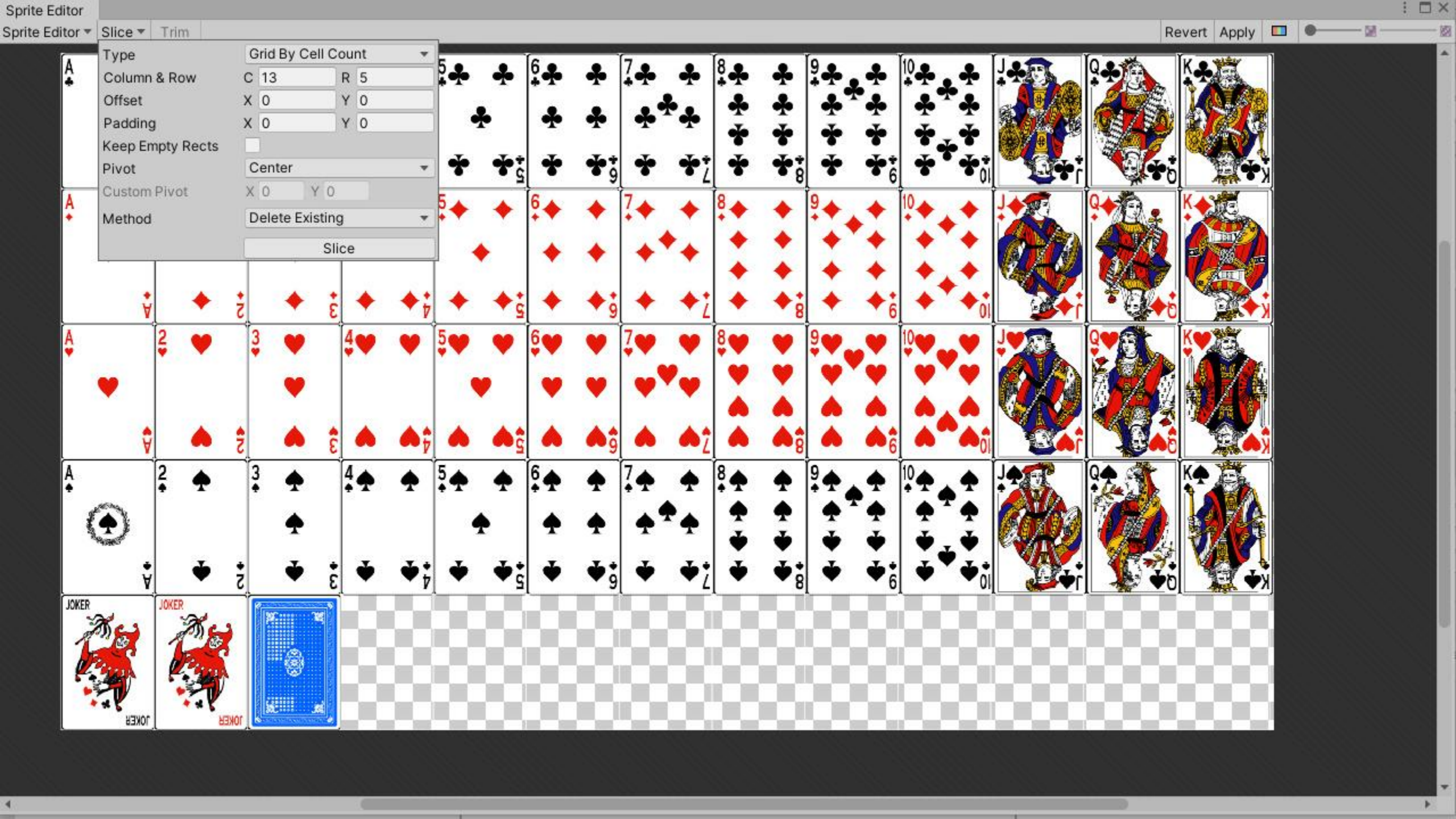
Unity

ハイアンドロー

- 今からですが、一枚の画像から複数枚の画像を切り出す作業を行います
- 意外とクライアントエンジニアの仕事になったりもするので覚えておきましょう

- 画像のInspector欄でSpriteModeを”multiple”に変更して”Apply”ボタンを押してください
- できましたらSprite Editorボタンをクリックしてください
- SpriteEditorで画像を切り出していきます

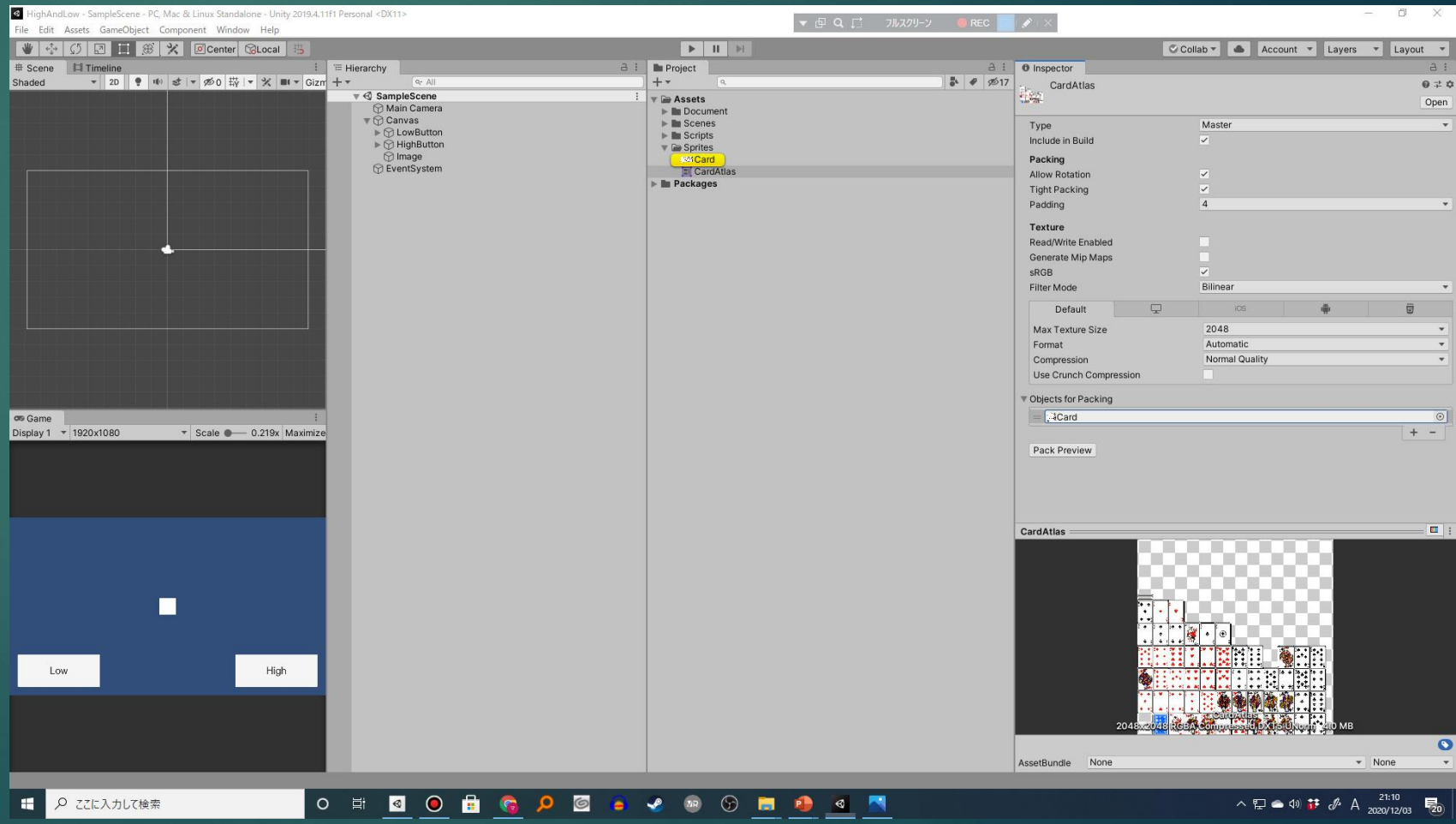



Sprite Editor
Sprite Editor
Slice
Trim
Revert
Apply
Type
Grid By Cell Count
Column & Row
C 13
R 5
Offset
X 0
Y 0
Padding
X 0
Y 0
Keep Empty Rects
Pivot
Center
Custom Pivot
X 0
Y 0
Method
Delete Existing
Slice

Unity
ハイアンドロー

- 先の設定ができたらSliceボタンを押してください
- そしてApplyボタンを押すと、Cardの中身が65枚になります

- 次に同じフォルダを右クリックしてCreateからSpriteAtlasを選択します
- 名前を”CardAtlas”として、ObjectsForPackingにCard.pngを指定してPackPreviewをクリックしてください
- ※SpriteAtlasとは
- 画像をパッキングしてくれる機能です。4*4の画素数じゃなくても、自動で4*4のAtlas(画像)を作成し、メモリに優しい画像を作成できます

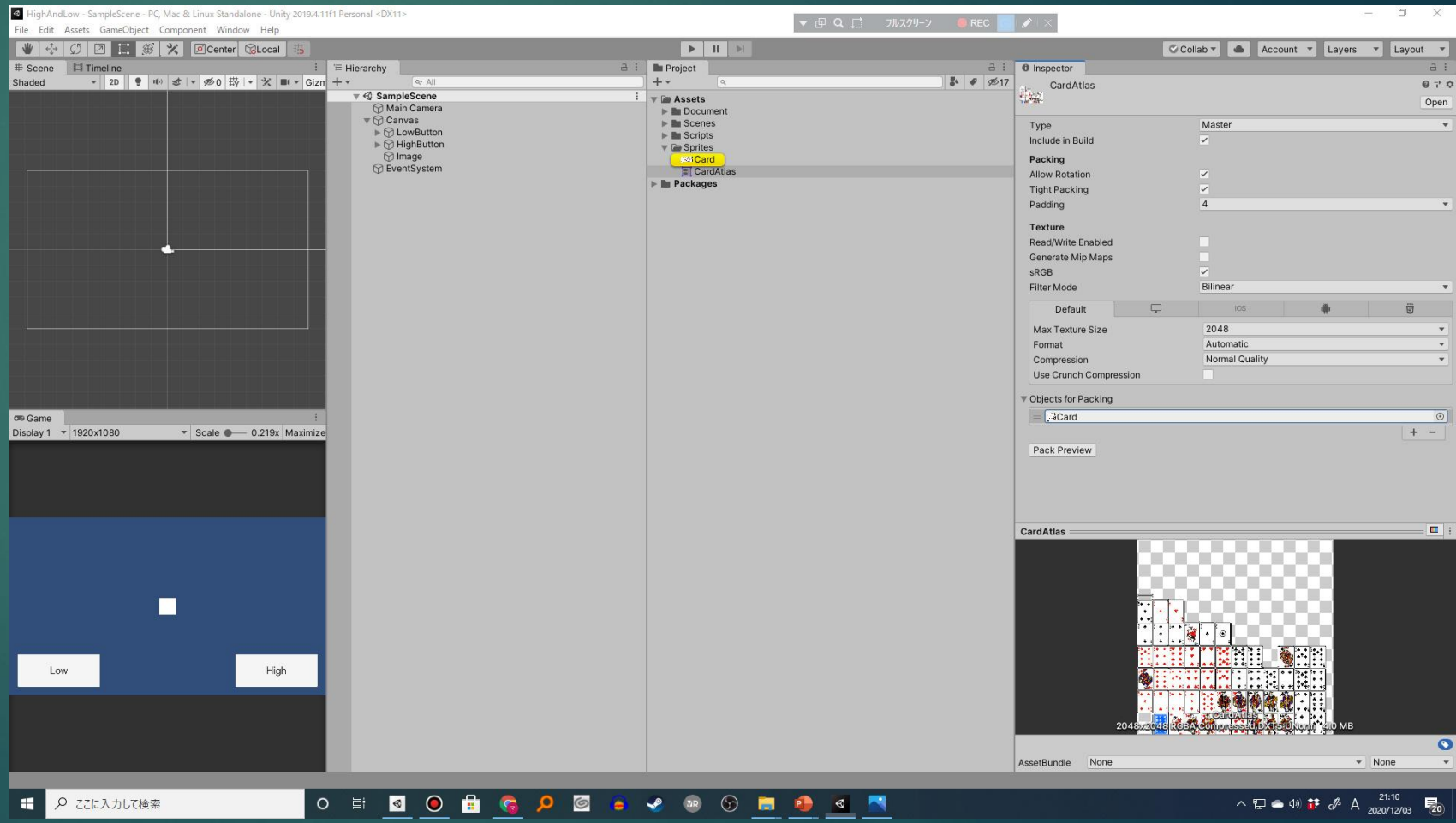


Unity
ハイアンドロー

- この画像の塊(SpriteAtlas)から実際に使用するカードの画像を取得します

- Canvas直下にImageを2つ作成し、
各々”PlayerCard”、”CPUCard”と名前を付けてください
- 画像の大きさは
- Width:167
- Height:243

- では、新しくCardImageHelper.csを作成し、またPlayerHand.csとCPUHand.csを編集していきます



HighAndLow - Microsoft Visual Studio

ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)

Debug | Any CPU | Unity にアタッチ

CardHelper.cs | CardImageHelper.cs | CPUCard.cs | PlayerCard.cs | Dealer.cs | HighAndLowSequence.cs | PlayerJudge.cs | Card.cs

Assembly-CSharp | CardImageHelper | GetCardSprite(Card card)

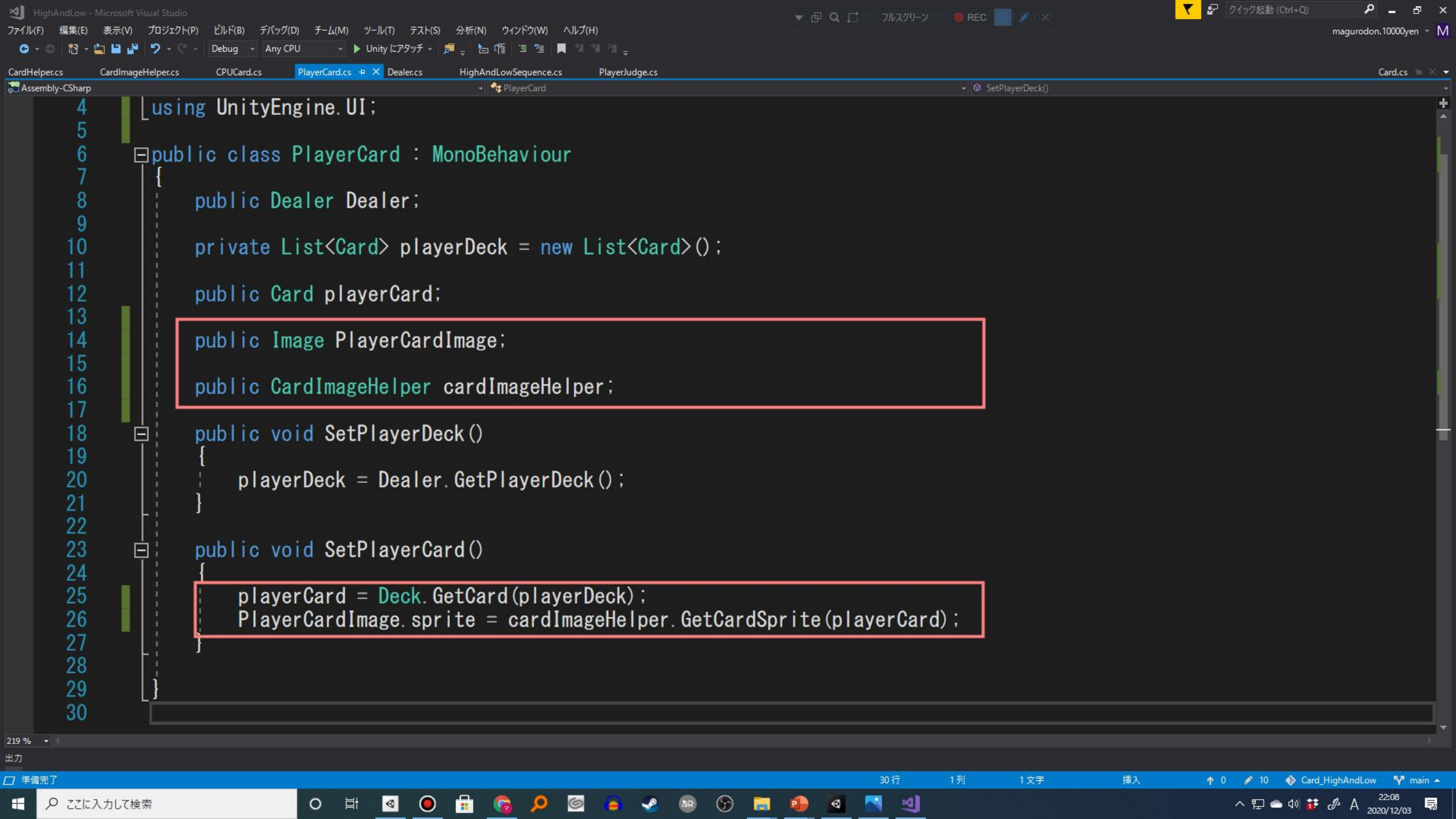
```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.U2D;
5
6   public class CardImageHelper : MonoBehaviour
7   {
8       public SpriteAtlas CardAtlas;
9
10      public Sprite GetCardSprite(Card card)
11      {
12          var cardNum = card.Number + ((int)card.CardSuit * 13) - 1;
13          var cardSprite = CardAtlas.GetSprite($"Card_{cardNum}");
14          return cardSprite;
15      }
16  }
17
```

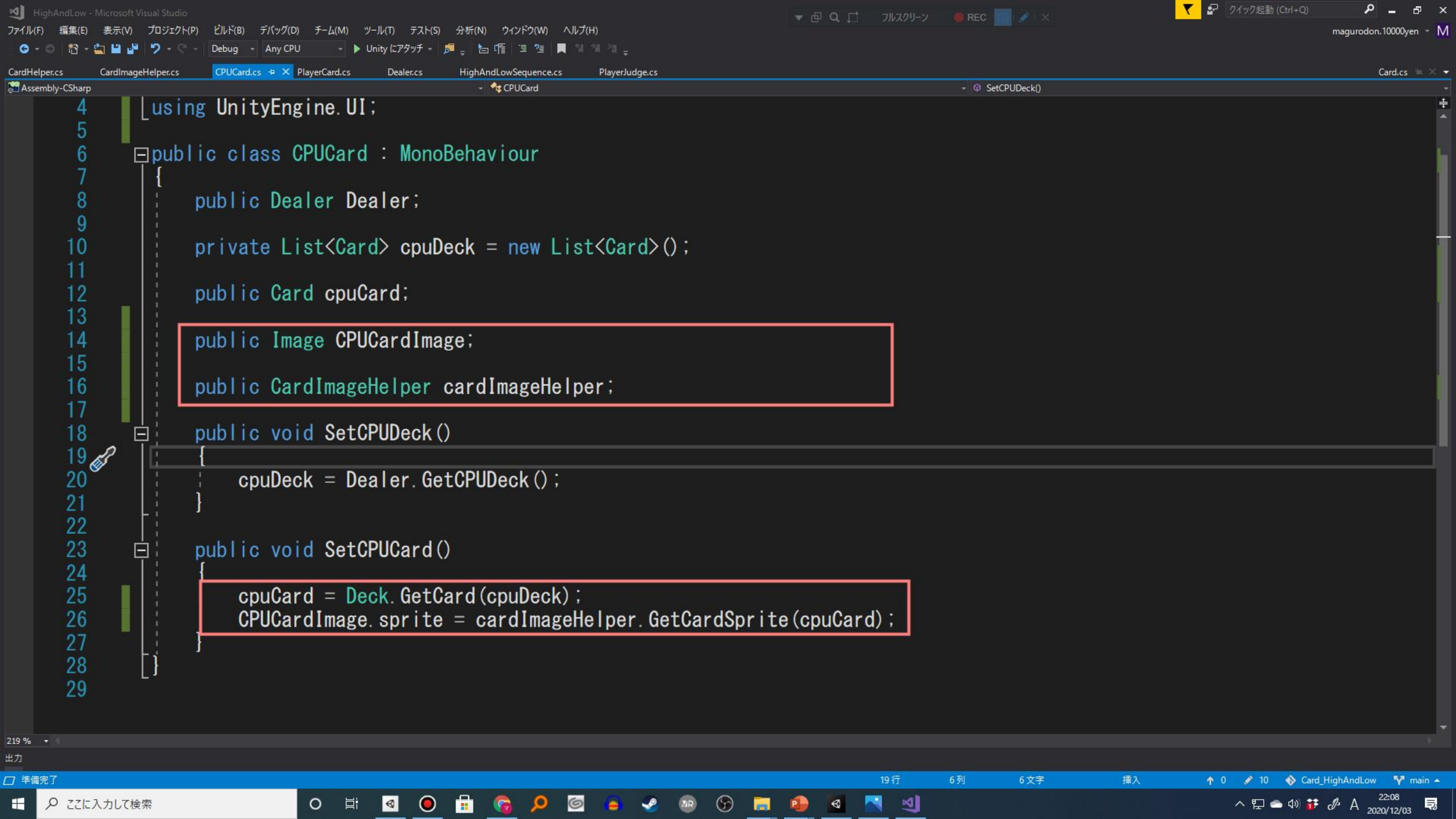
219 %

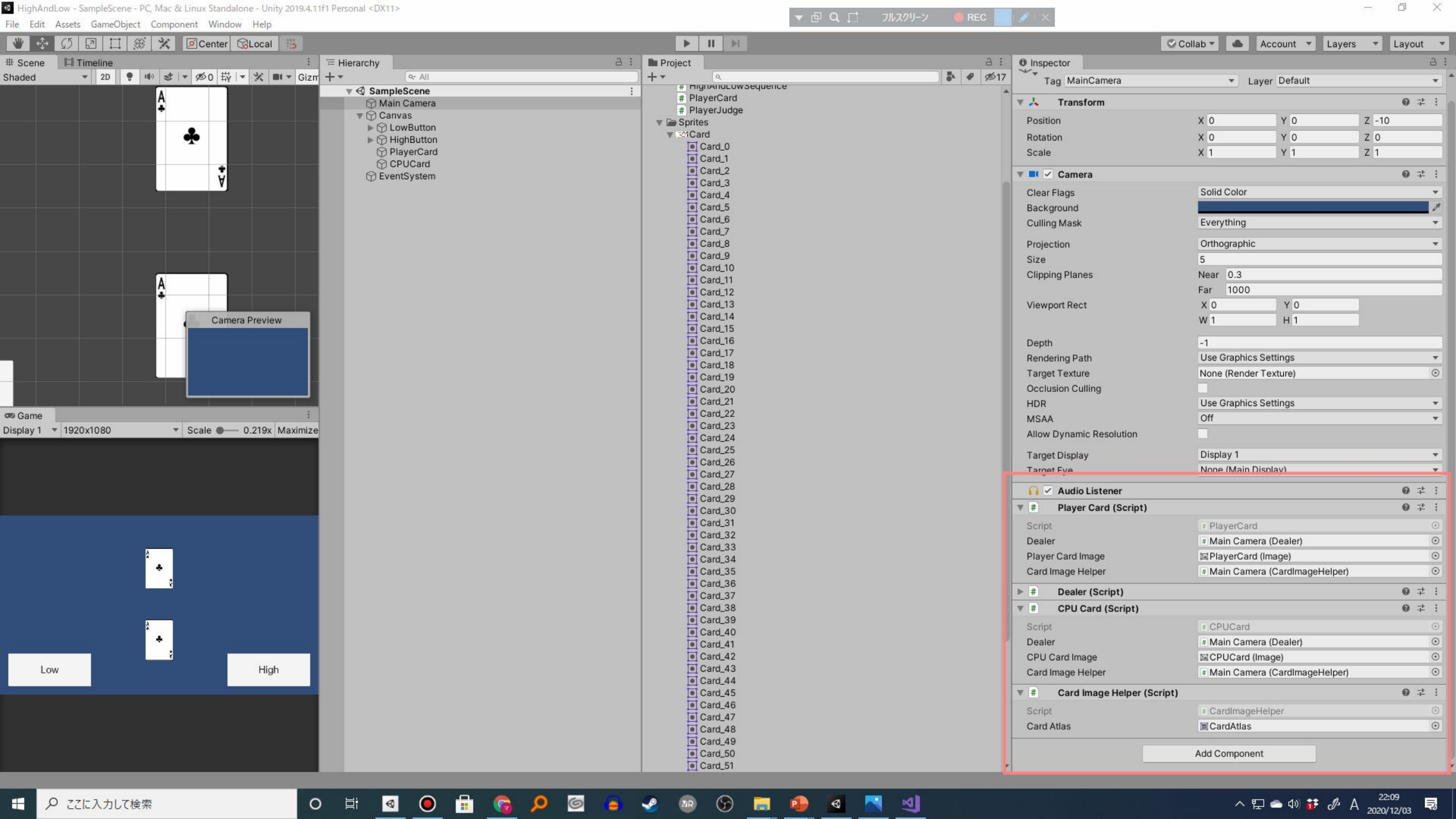
出力

アイテムが保存されました 17 行 1 列 1 文字 挿入 0 10 Card_HighAndLow main

ここに入力して検索 22:06 2020/12/03

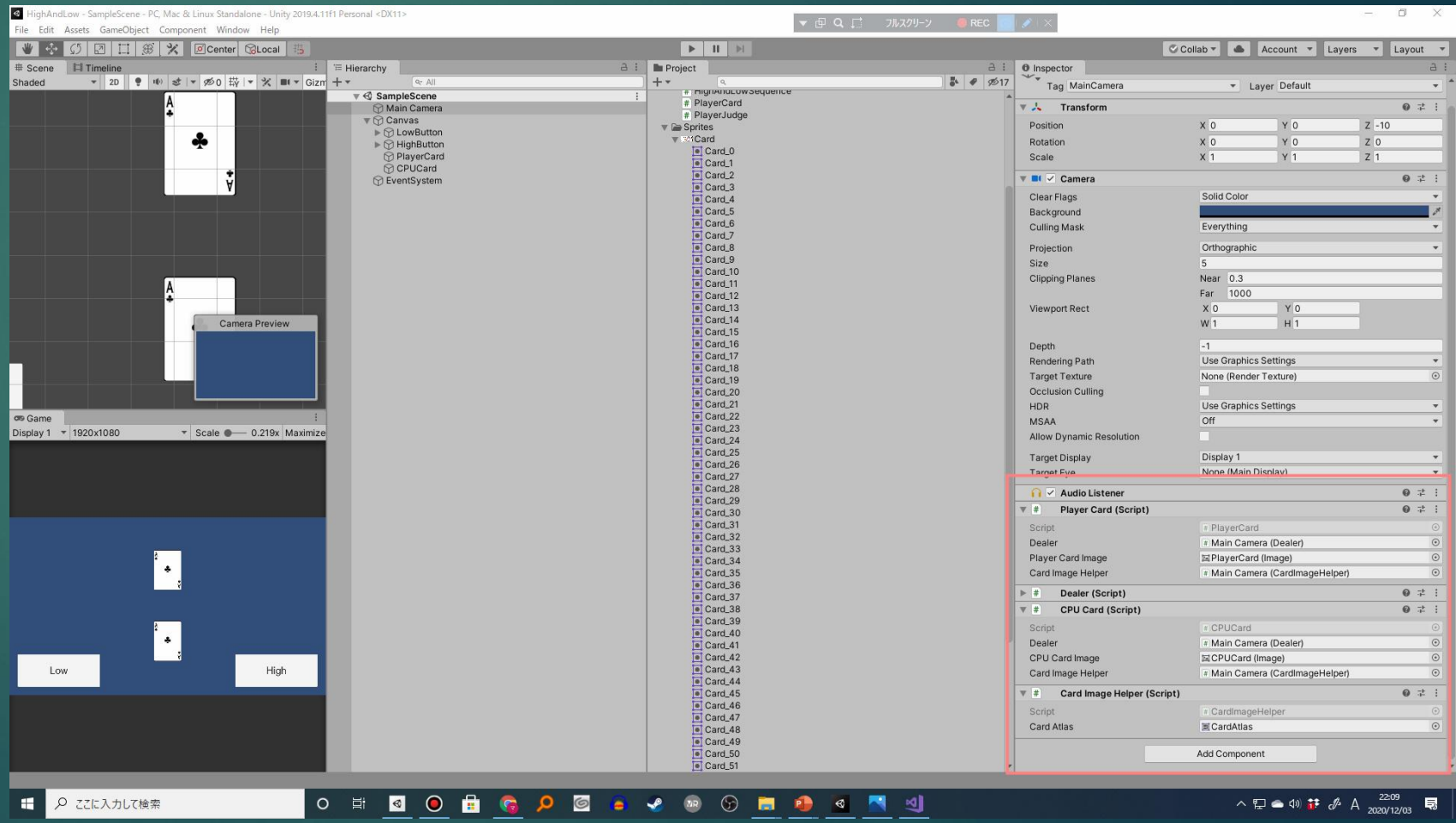

HighAndLow - Microsoft Visual Studio
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
Debug
Any CPU
Unity にアタッチ
CardHelper.cs
CardImageHelper.cs
CPUCard.cs
PlayerCard.cs
Dealer.cs
HighAndLowSequence.cs
PlayerJudge.cs
Card.cs
Assembly-CSharp
PlayerCard
SetPlayerDeck()
using UnityEngine.UI;
public class PlayerCard : MonoBehaviour
{
public Dealer Dealer;
private List<Card> playerDeck = new List<Card>();
public Card playerCard;
public Image PlayerCardImage;
public CardImageHelper cardImageHelper;
public void SetPlayerDeck()
{
playerDeck = Dealer.GetPlayerDeck();
}
public void SetPlayerCard()
{
playerCard = Deck.GetCard(playerDeck);
PlayerCardImage.sprite = cardImageHelper.GetCardSprite(playerCard);
}
}
219 %
出力
準備完了
30 行
1 列
1 文字
挿入
Card_HighAndLow
main
ここに入力して検索
2020/12/03


HighAndLow - Microsoft Visual Studio
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
Debug
Any CPU
Unity にアタッチ
CardHelper.cs
CardImageHelper.cs
CPUCard.cs
PlayerCard.cs
Dealer.cs
HighAndLowSequence.cs
PlayerJudge.cs
Card.cs
Assembly-CSharp
CPUCard
SetCPUDeck()
using UnityEngine.UI;
public class CPUCard : MonoBehaviour
{
public Dealer Dealer;
private List<Card> cpuDeck = new List<Card>();
public Card cpuCard;
public Image CPUCardImage;
public CardImageHelper cardImageHelper;
public void SetCPUDeck()
{
cpuDeck = Dealer.GetCPUDeck();
}
public void SetCPUCard()
{
cpuCard = Deck.GetCard(cpuDeck);
CPUCardImage.sprite = cardImageHelper.GetCardSprite(cpuCard);
}
}
219 %
出力
準備完了
19 行
6 列
6 文字
挿入
Card_HighAndLow
main
ここに入力して検索
22:08
2020/12/03

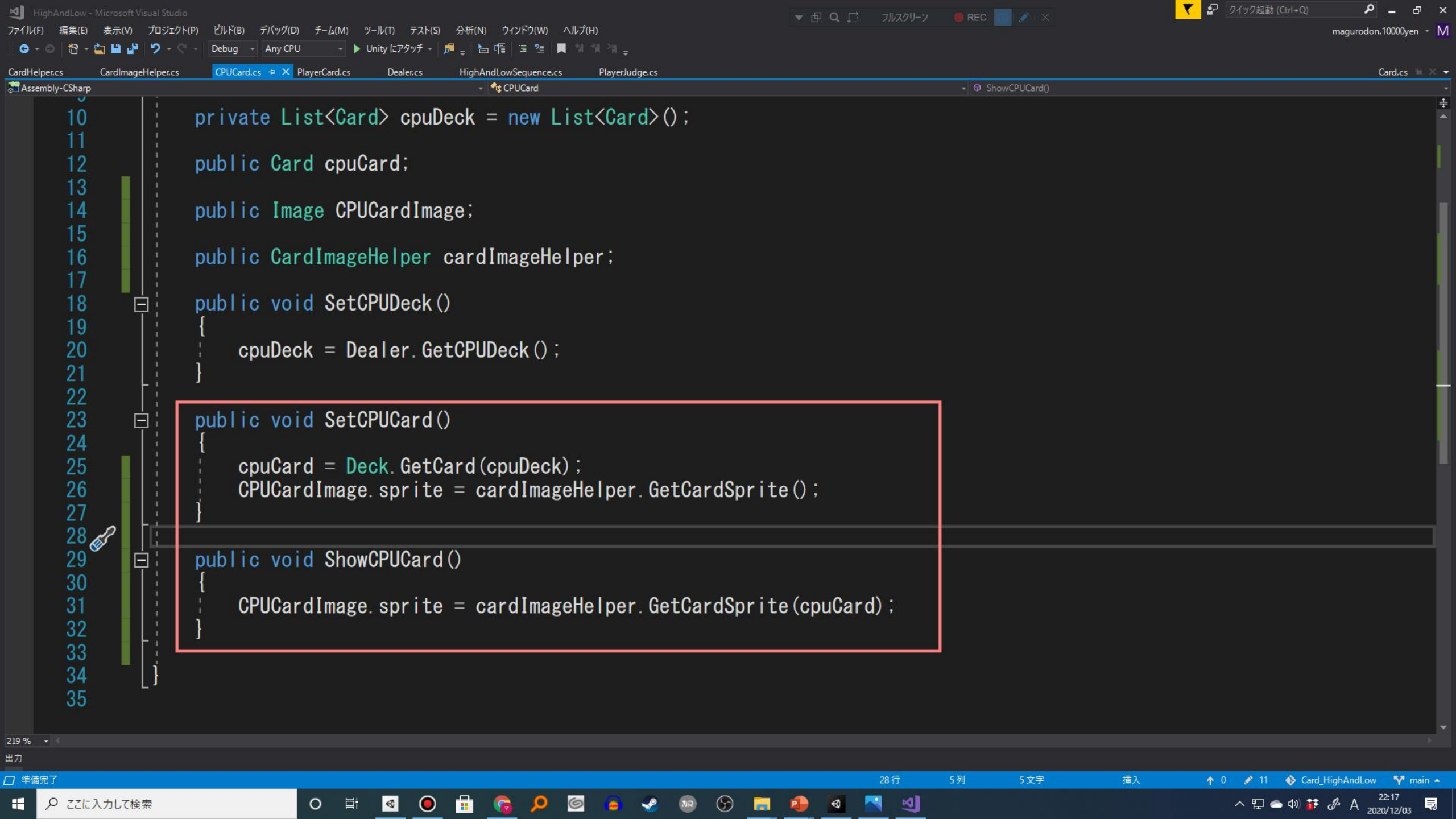

HighAndLow - SampleScene - PC, Mac & Linux Standalone - Unity 2019.4.11f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
フルスクリーン REC
Center Local
Collab Account Layers Layout
Scene Timeline
Shaded 2D
Camera Preview
Game
Display 1 1920x1080 Scale 0.219x Maximize
Low
High
Hierarchy
SampleScene
Main Camera
Canvas
LowButton
HighButton
PlayerCard
CPUCard
EventSystem
Project
PlayerCard
PlayerJudge
Sprites
Card
Card_0
Card_1
Card_2
Card_3
Card_4
Card_5
Card_6
Card_7
Card_8
Card_9
Card_10
Card_11
Card_12
Card_13
Card_14
Card_15
Card_16
Card_17
Card_18
Card_19
Card_20
Card_21
Card_22
Card_23
Card_24
Card_25
Card_26
Card_27
Card_28
Card_29
Card_30
Card_31
Card_32
Card_33
Card_34
Card_35
Card_36
Card_37
Card_38
Card_39
Card_40
Card_41
Card_42
Card_43
Card_44
Card_45
Card_46
Card_47
Card_48
Card_49
Card_50
Card_51
Inspector
Tag MainCamera Layer Default
Transform
Position X 0 Y 0 Z -10
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Camera
Clear Flags Solid Color
Background
Culling Mask Everything
Projection Orthographic
Size 5
Clipping Planes Near 0.3 Far 1000
Viewport Rect X 0 Y 0 W 1 H 1
Depth -1
Rendering Path Use Graphics Settings
Target Texture None (Render Texture)
Occlusion Culling
HDR Use Graphics Settings
MSAA Off
Allow Dynamic Resolution
Target Display Display 1
Audio Listener
Player Card (Script)
Script PlayerCard
Dealer Main Camera (Dealer)
Player Card Image PlayerCard (Image)
Card Image Helper Main Camera (CardImageHelper)
Dealer (Script)
CPU Card (Script)
Script CPUCard
Dealer Main Camera (Dealer)
CPU Card Image CPUCard (Image)
Card Image Helper Main Camera (CardImageHelper)
Card Image Helper (Script)
Script CardImageHelper
Card Atlas CardAtlas
Add Component
ここに入力して検索
22:09
2020/12/03

Unity
ハイアンドロー

- 先の設定が終わり次第、プレイしてみてください
- 二枚のカードが表示されていると思います
- 今の現状ではCPUのカードも分かってしまうので、これは良くありません
- 隠すためにCPUCard.csとCardImageHelper.csをもう一度編集します
- また、HighAndLowSequence.csも編集しましょう



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.U2D;

public class CardImageHelper : MonoBehaviour
{
    public SpriteAtlas CardAtlas;

    public Sprite GetCardSprite(Card card = null)
    {
        var cardNum = 0;

        if (card == null)
        {
            cardNum = 54;
        }
        else
        {
            cardNum = card.Number + ((int)card.CardSuit * 13) - 1;
        }

        var cardSprite = CardAtlas.GetSprite($"Card_{cardNum}");
        return cardSprite;
    }
}
```


HighAndLow - Microsoft Visual Studio
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
Debug
Any CPU
Unity にアタッチ
CardHelper.cs
CardImageHelper.cs
CPUCard.cs
PlayerCard.cs
Dealer.cs
HighAndLowSequence.cs
PlayerJudge.cs
Card.cs
Assembly-CSharp
CPUCard
ShowCPUCard()
10 private List<Card> cpuDeck = new List<Card>();
12 public Card cpuCard;
14 public Image CPUCardImage;
16 public CardImageHelper cardImageHelper;
18 public void SetCPUDeck()
19 {
20 cpuDeck = Dealer.GetCPUDeck();
21 }
23 public void SetCPUCard()
24 {
25 cpuCard = Deck.GetCard(cpuDeck);
26 CPUCardImage.sprite = cardImageHelper.GetCardSprite();
27 }
29 public void ShowCPUCard()
30 {
31 CPUCardImage.sprite = cardImageHelper.GetCardSprite(cpuCard);
32 }
34 }
219 %
出力
準備完了
28 行
5 列
5 文字
挿入
Card_HighAndLow
main
ここに入力して検索
2020/12/03

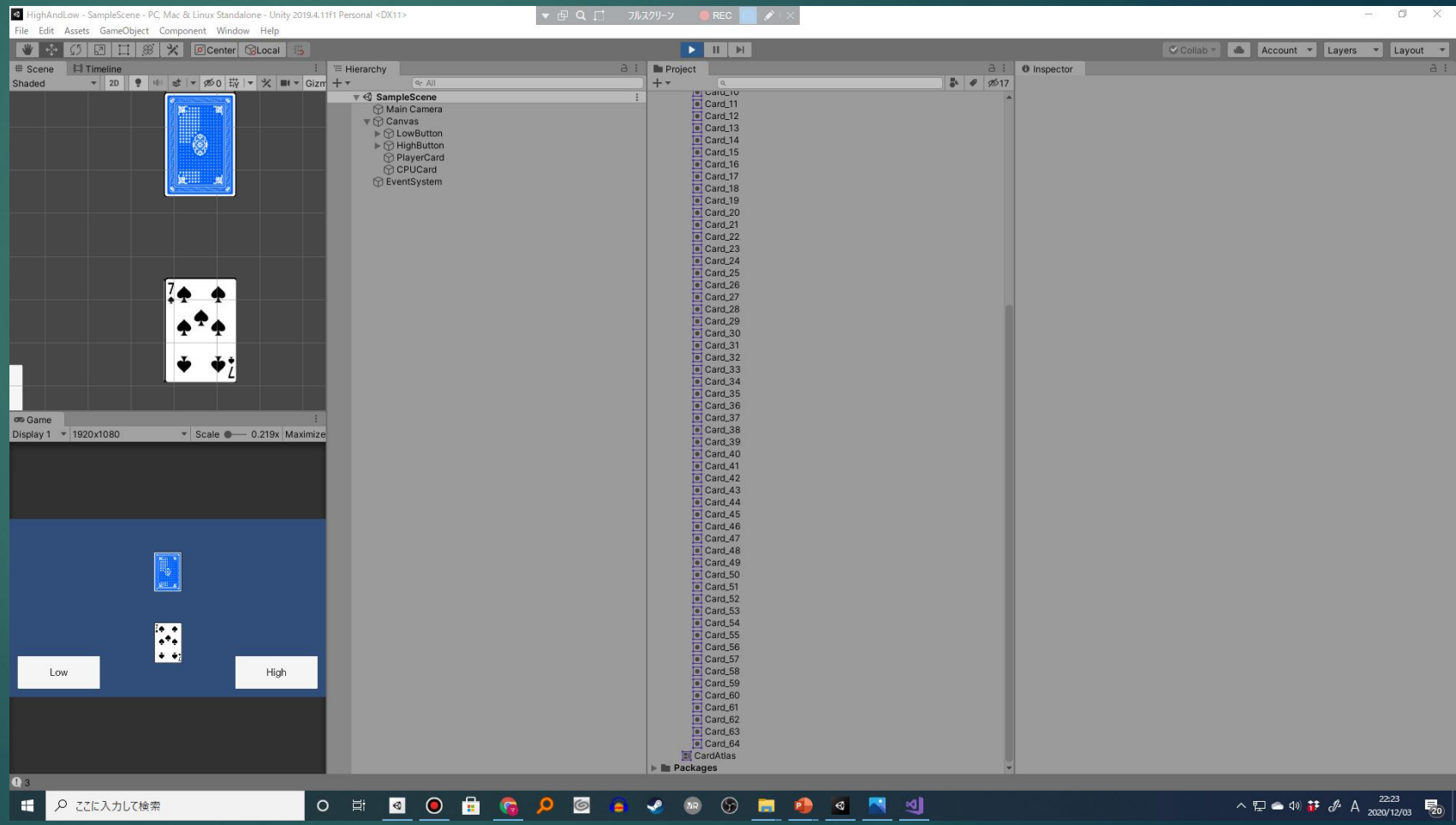
```
            Debug.Log(playerCard.playerCard.Number);
            Debug.Log(cpuCard.cpuCard.Number);
            gameSequence = GameSequence.PlayerJudge;
            break;
        case GameSequence.PlayerJudge:

            // 数を予想してボタンを押したらShowに進む
            if (playerJudge.Judge)
            {
                gameSequence = GameSequence.Show;
            }
            break;
        case GameSequence.Show:

            // プレイヤーが確認したらStartに戻って次のゲーム
            cpuCard.ShowCPUCard();

            if (playerJudge.High)
            {
                if (playerCard.playerCard.Number > cpuCard.cpuCard.Number)
                {
                    Debug.Log("勝ち");
                }
                else
                {
                    Debug.Log("負け");
                }
            }
```

Unity
ハイアンドロー

- これでゲームの進行があらかた完了したと思います
- Consoleを見れば、勝ち負けが分かるようになっております。

- では実習です。
- ここでUIを使って、勝ったときは”Win”、負けたときは”Lose”と表示してみてください

- Scriptは新しくGameJudge.csを作成します
- その中で結果を表示するメソッドを作成し、HighAndLowSequence.cs内で呼び出してください



Tips：Listや配列の利用方法

- リストや配列は本当によく使われます
- ひと昔前は
- [1,0,0,0,0]
- [1,1,1,1,0]
- [0,0,0,1,0]
- [0,0,0,1,2]
- これでMap等を表現していました(0が進行不能、2がゴール)
- ちなみに上は多次元配列と呼ばれるものです