

INSTRUKCJA WYKONANIA PROJEKTU GRY

UNITY 6000.0.0B16

# 3D Mini GOLF

„MINIGOLF”



Dane techniczne:

Silnik: Unity

Wersja silnika: 6000.0.0b16

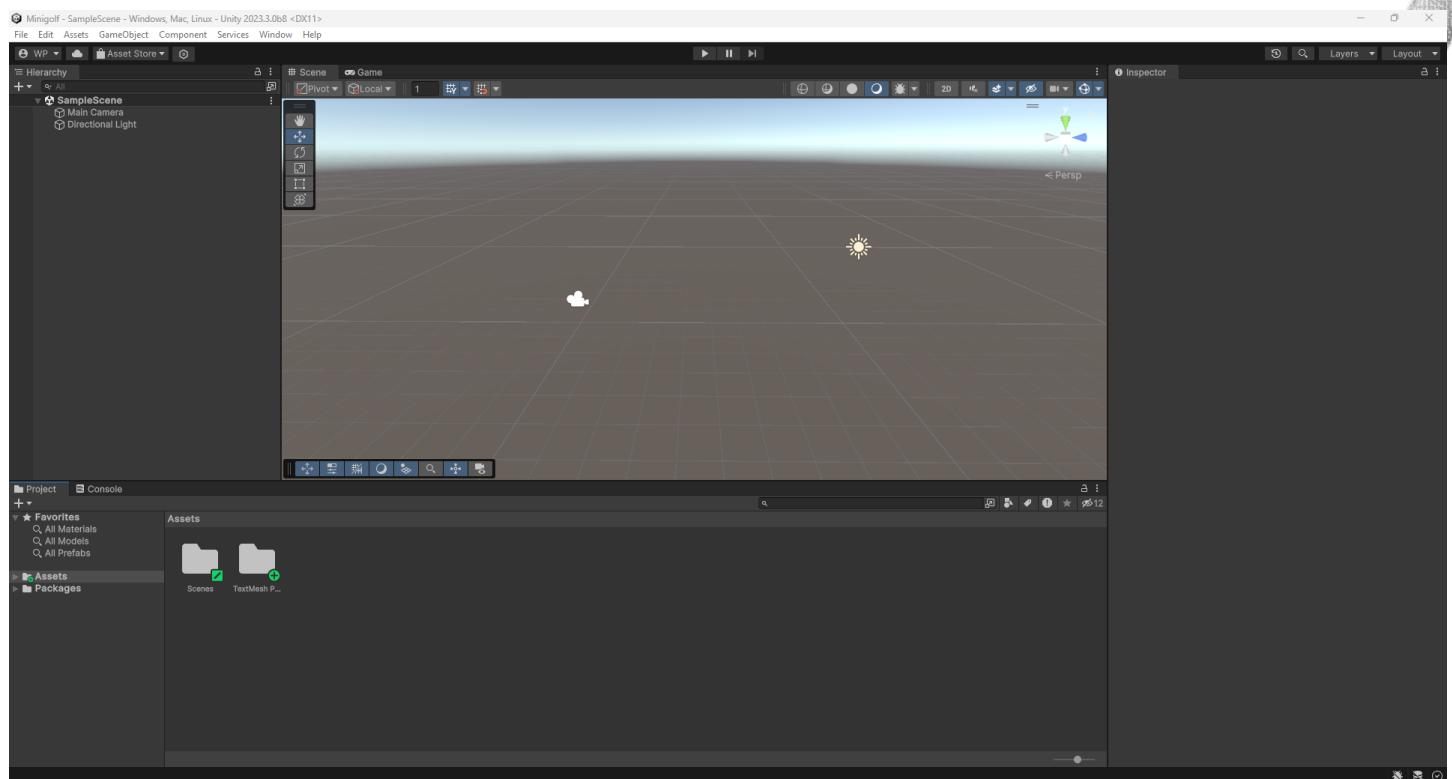
Platforma: PC, Android

Język programowania: C#

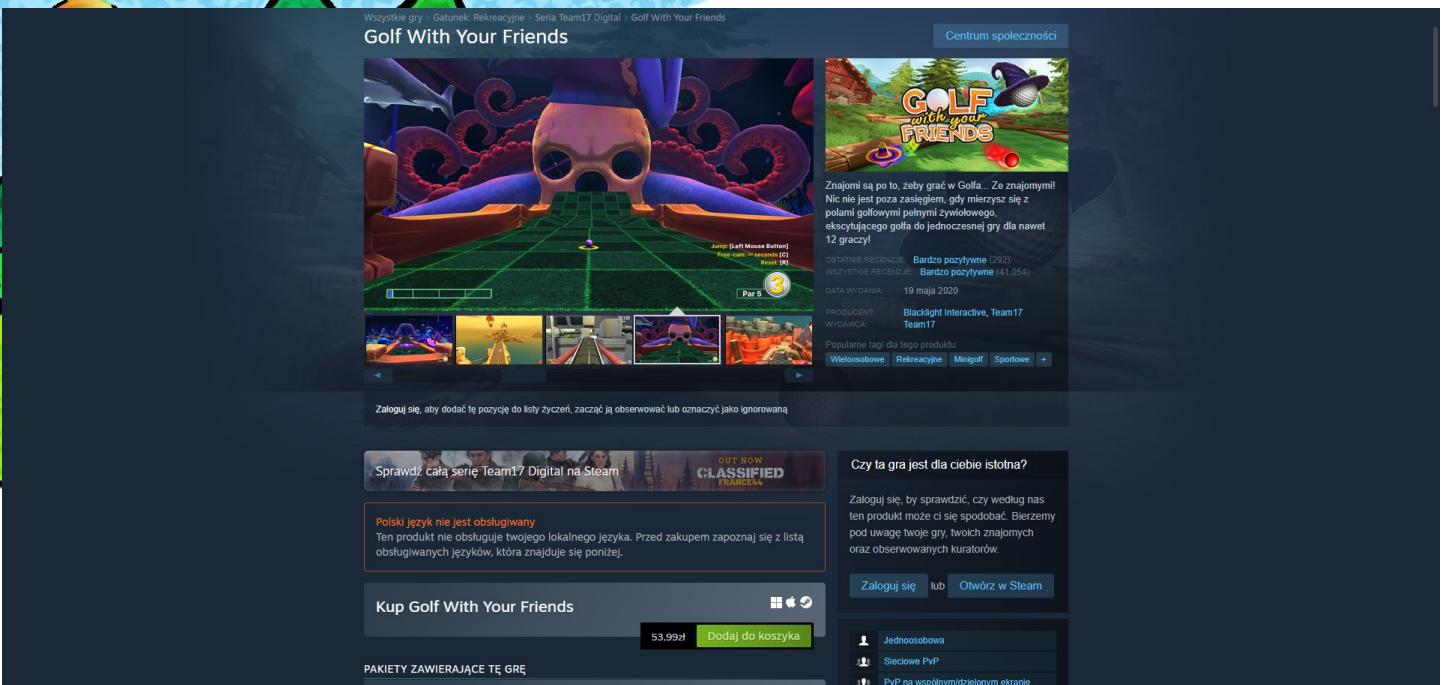
Tryb gry: Single/Multi Player

## Opis i Fabuła:

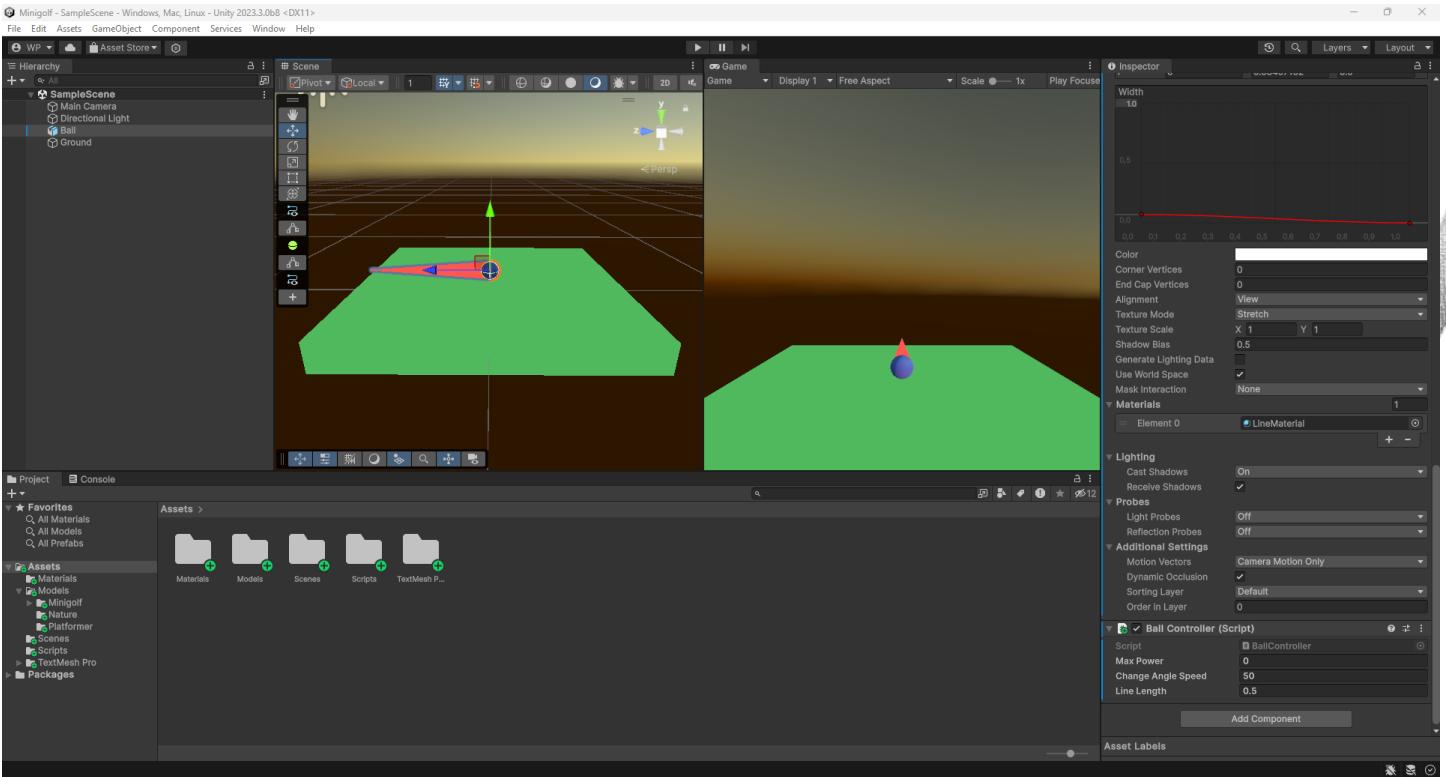
„Minigolf” to wciągająca gra zręcznościowa 3D, w której do pięciu graczy wciela się w rolę piłki do minigolfa. Celem gry jest wprowadzenie kulki do dołka na różnorodnych torach przy użyciu jak najmniejszej liczby ruchów. Gra oferuje trzy różnorodne poziomy, z których każdy stanowi wyzwanie dla graczy. Graj samodzielnie lub rywalizuj z przyjaciółmi, by zobaczyć, kto osiągnie najlepszy wynik na każdym z poziomów!



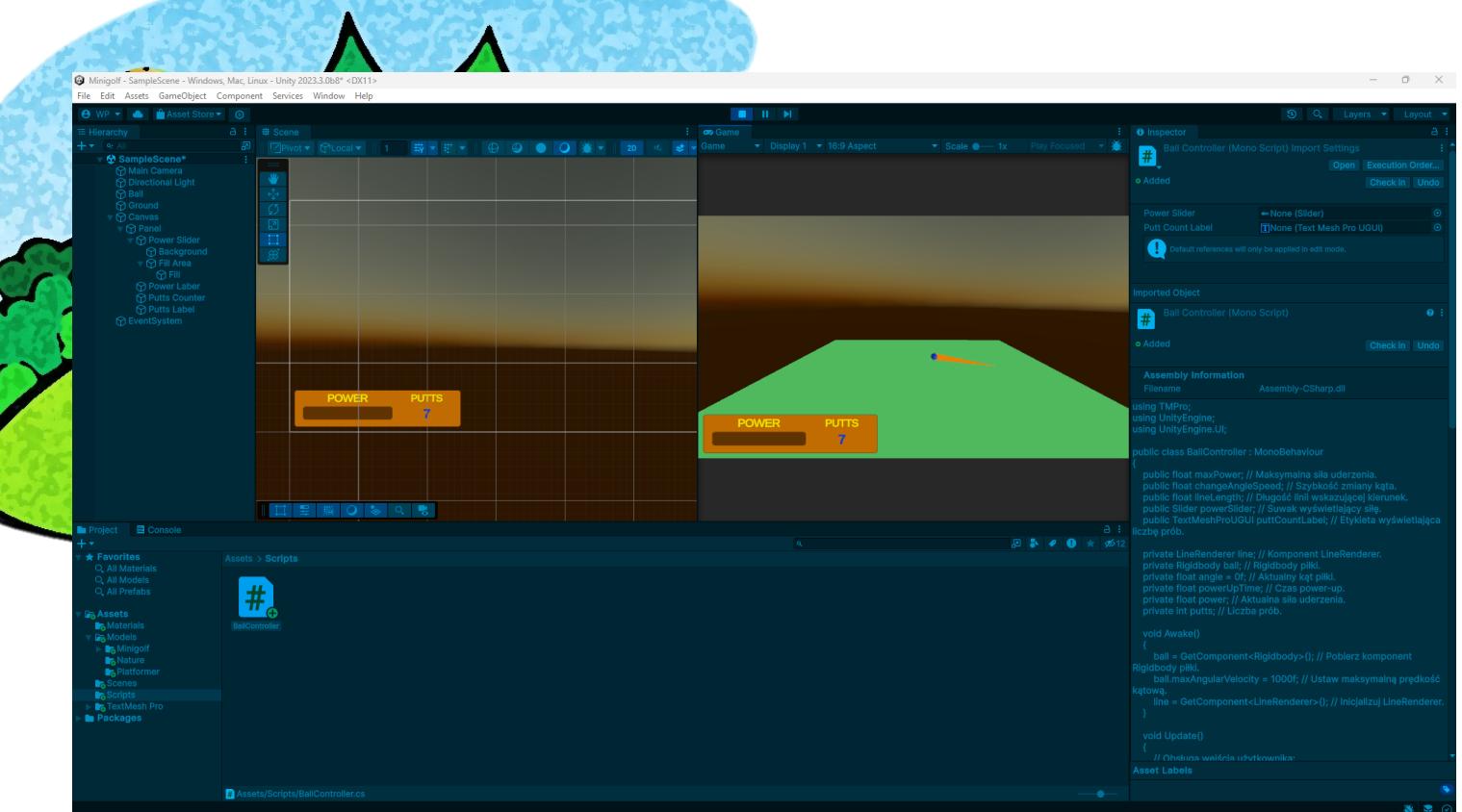
Zaczynamy od zera. Stworzyłem nowy projekt gry 3D w Unity 6000.0.0b16 – „Minigolf”.



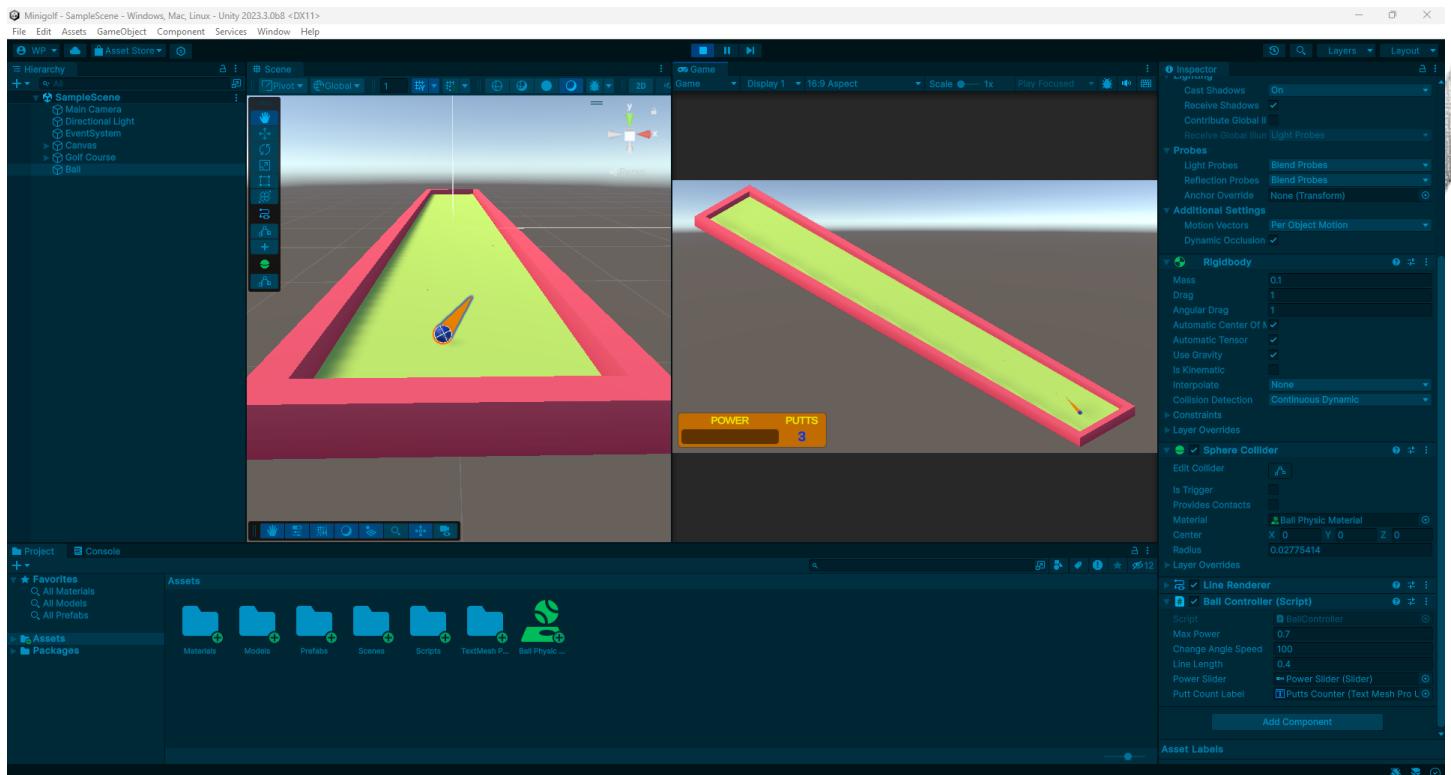
Częściowo inspirację będę czerpał z tej gry, w której grałem kiedyś ze znajomymi i jest bardzo fajna. Wiadomo, że będzie to tylko namiastka, ale będę się starał iść w tym kierunku. Link do gry: [https://store.steampowered.com/app/431240/Golf\\_With\\_Your\\_Friends/?l=polish](https://store.steampowered.com/app/431240/Golf_With_Your_Friends/?l=polish)



Zacząłem od dodania odpowiednich modeli i materiałów, które będą mi potrzebne w realizacji tej gry. Część pobrałem jako gotowce, a część wykonałem samodzielnie. Po tym zrobiłem proste podłożę i piłkę golfową. Pokolorowałem, dodałem Collidery. W piłce dodałem komponent Rigidbody do symulacji fizyki piłki (poruszania się) i Line Renderer do wyznaczenia trajektorii uderzenia piłki. Napisałem częściowo skrypt do obsługi piłki. Można już klawiszami obracać trajektorię uderzenia piłki. Wszystko wypożyczonowałem.



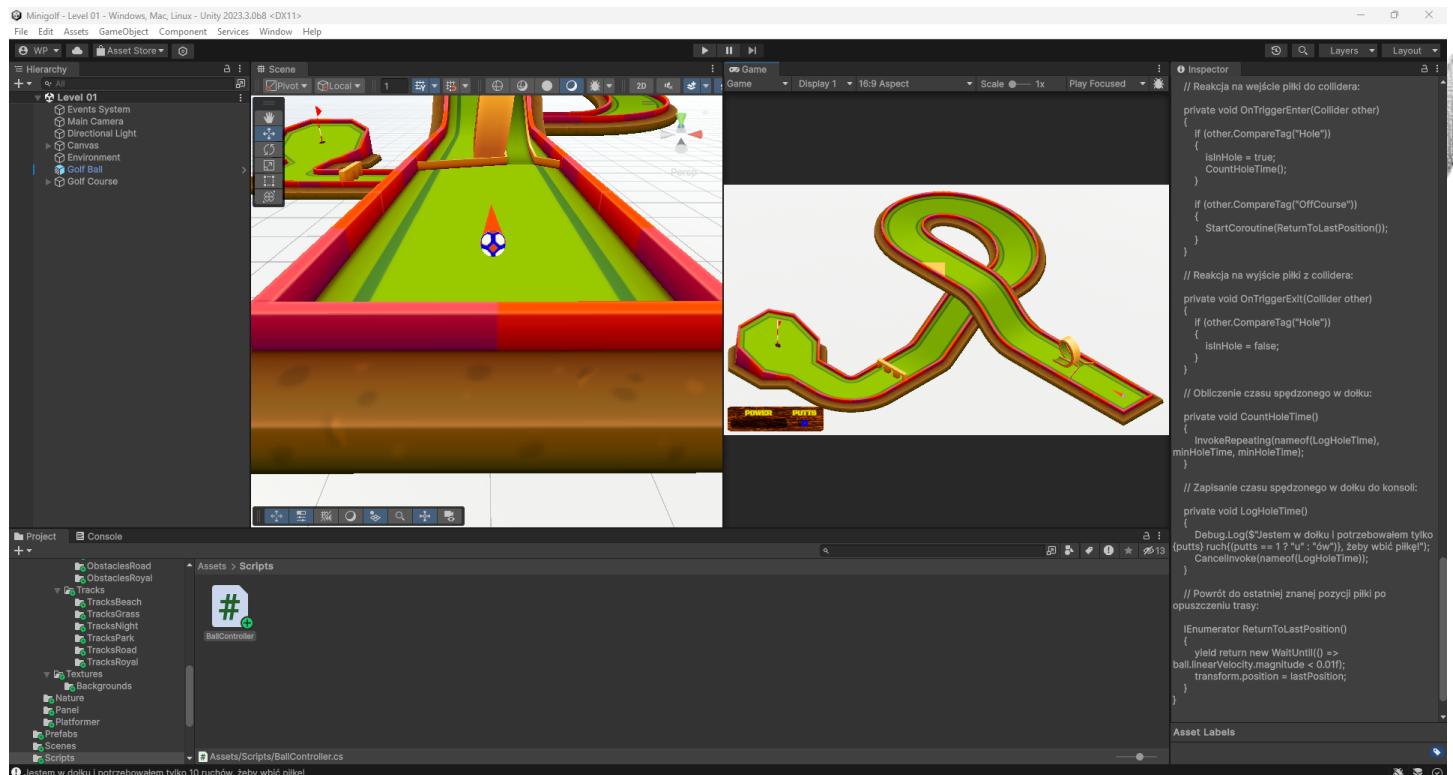
Kolejny etap polegał na zrobieniu panelu wyświetlającego siłę uderzenia i liczbę prób. Zmodyfikowałem odpowiednio skrypt żeby wszystko działało, a poza tym odpowiednio wypożyczonowałem panel, żeby pod wpływem skalowania zmieniał się odpowiednio jego rozmiar. Zmieniłem model piłki na ciekawszy oraz ukierunkowałem się w kolorystyce.



Następnie zrobiłem przykładowe pole do gry w golfa, po to aby zrobić fizykę piłki, np. odbijanie od ścianek, poruszanie piłki po torze, płynność, żeby nie zahaczała o krawędzie obiektów itd. Zmniejszyłem trochę panel informacyjny i pole kamery. Teraz piłka sunie już po polu jak trzeba.



Z każdym krokiem wygląda to coraz lepiej. Ulepszyłem panel informacyjny pod względem wizualnym. Zbudowałem tor z nowych obiektów. Zmieniłem teksturę piłki. Pobawiłem się fizyką piłki. Na ten moment można już swobodnie trafić piłką w dołek, póki co nie ma dodanego jeszcze skryptu. Następnym etapem będzie napisanie skryptu trafienia w dołek oraz wypadnięcia piłki poza tor.



Na tym etapie napisałem kolejną część skryptu odpowiadającą za obsługę trafienia w dołek. Wyświetla się log informujący o tym oraz po ilu próbach się to udało. Poza tym obsłużyłem wypadnięcie piłki poza tor, po zatrzymaniu wraca do ostatniej pozycji. Dodałem płaszczyznę.



Uwaga! Kolejne kroki opisuję po całkowitym ukończeniu gry.

Kolejnym etapem, który poczyniętem w tworzeniu gry było stworzenie kompletnego menu do gry. Zacząłem od stworzenia menu głównego. Są w nim zawarte opcje rozgrywki, opcji, informacji o grze oraz przycisk wyjścia z gry. W darmowych źródłach grafik znalazłem odpowiednie tło i obrazek golfisty z piłką, żeby nie było tak pusto.



W plikach z modelami dodałem odpowiednie obrazy i zmieniłem typ tekstury na Sprite/2D. Zmieniłem opcje kompresji i maksymalnego rozmiaru grafiki. Poza tym Poza tym zwiększyłem liczbę pikseli ze 100 na 200. Zrobiłem tak dla wszystkich plików graficznych w mojej grze, poza kurSORAMI, które muszą mieć ustawiony format na RGBA 32bit i włączony Read/Write.

```

using TMPro;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class MenuManager : MonoBehaviour
{
    public TMP_InputField inputPlayerName;
    public PlayerRecord playerRecord;
    public Button buttonStart;
    public Button buttonAddPlayer;
    public GameObject MainPanel;
    public GameObject PlayPanel;
    public GameObject OptionsPanel;
    public GameObject AboutPanel;
    public Color inputTextColor = Color.black;
    public Color placeholderColor = Color.gray;
    public int maxPlayers = 5;

    void Start()
    {
        inputPlayerName.GetComponent<TMP_Text>().color = inputTextColor;
        inputPlayerName.placeholder.GetComponent<TMP_Text>().color = placeholderColor;

        ValidatePlayerName(inputPlayerName.text);
        inputPlayerName.onValueChanged.AddListener(ValidatePlayerName);
        inputPlayerName.onSubmit.AddListener(delegate { AddPlayerOnEnter(); });
    }

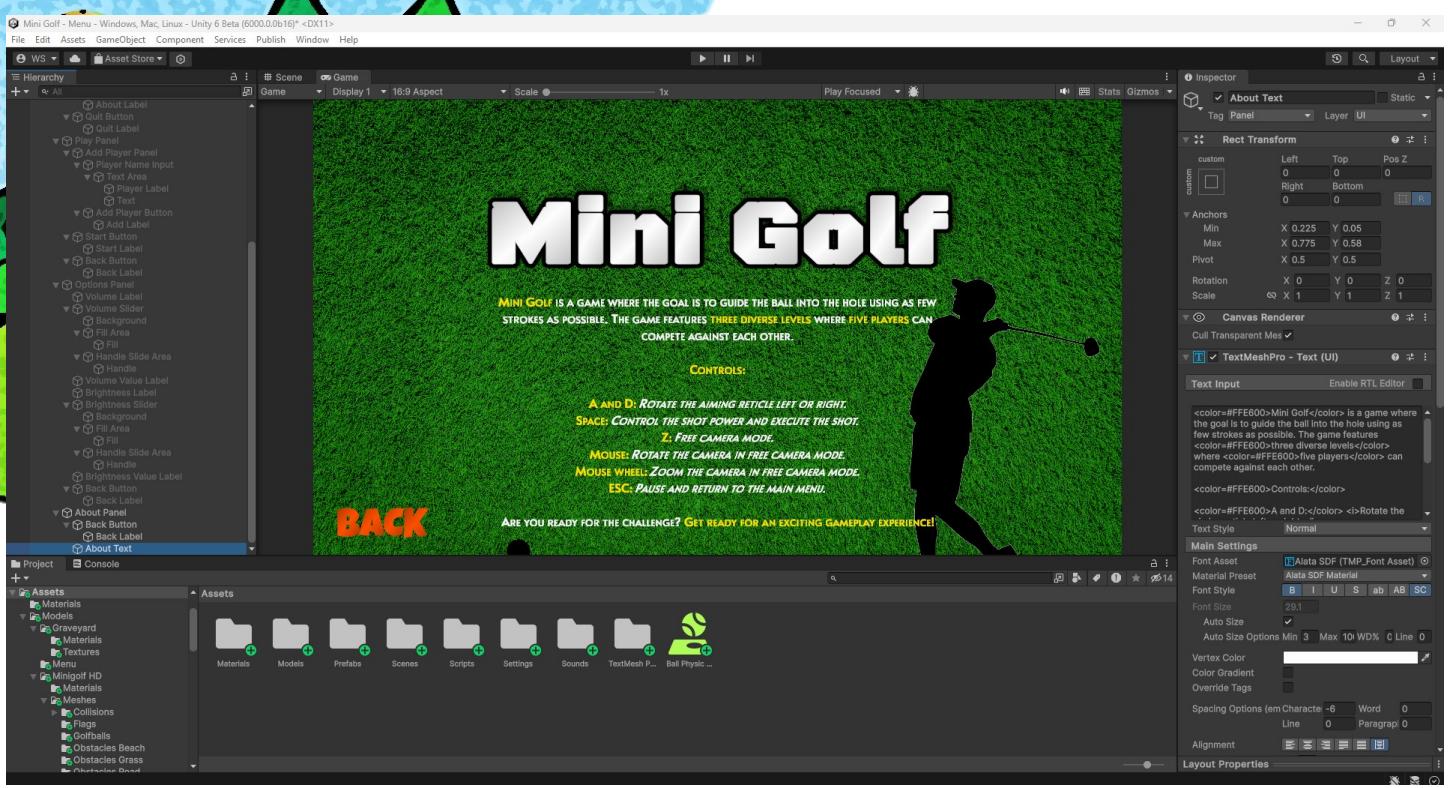
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
    }
}

```

To jest kod do obsługi menu głównego z odpowiednio opisanymi metodami. Przypisałem go do głównego Canvasu o nazwie Menu. Dodałem do zmiennych publicznych slidery, labele i butony żeby uzyskać poprawne działanie opcji gry, gdzie możemy zmienić głośność i jasność obrazu. Warto dodać, że poza obsługą myszki, w menu zastosowałem obsługę klawiszową, tzn. Enter zatwierdza w przypadku wprowadzania nazwy gracza, a ESC cofa do poprzedniej karty.



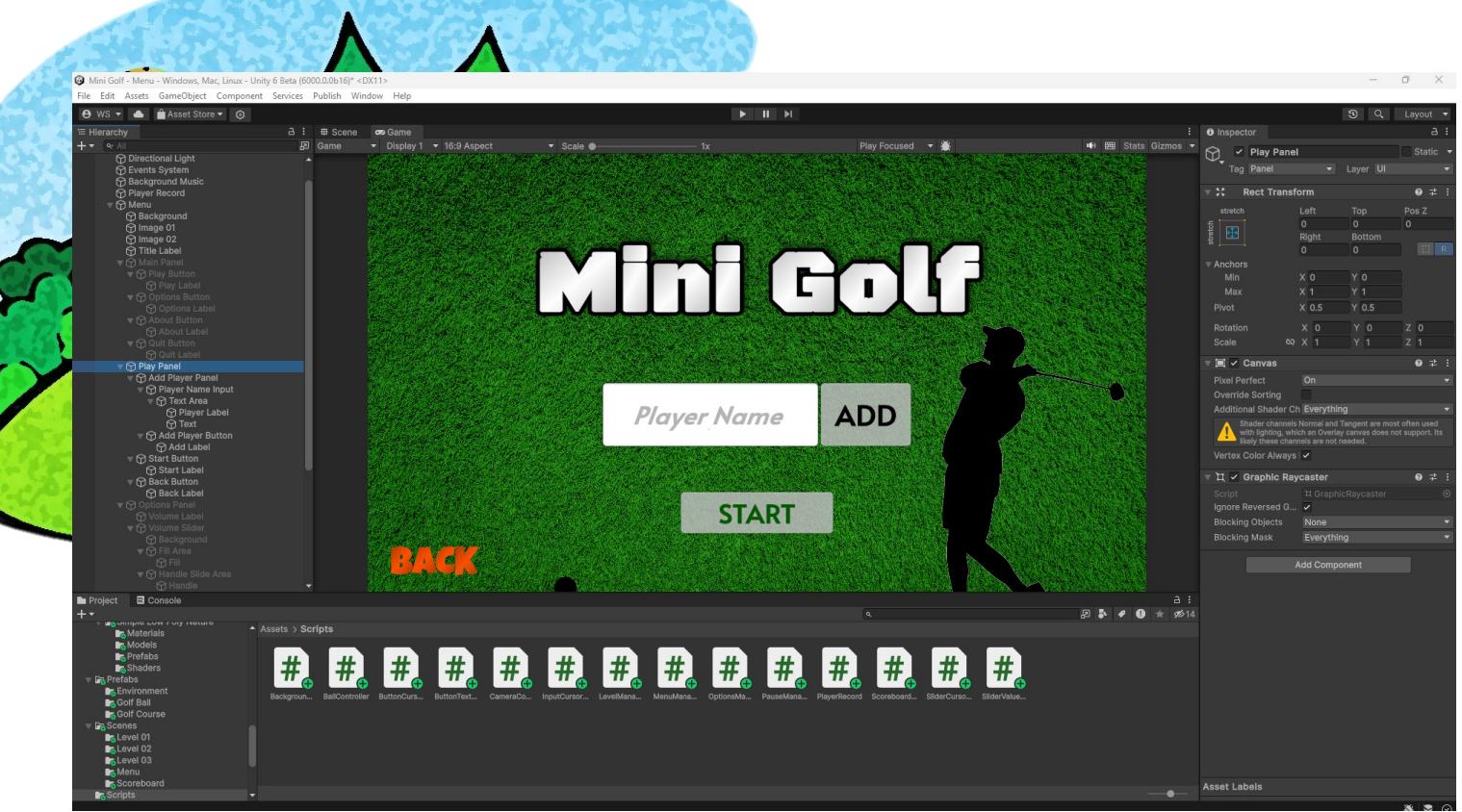
Wygląda to w ten sposób, mamy przycisk powrotu do głównego menu, labele, slidery z handle'ami oraz zrobiłem labele konwertujące poziom slidera na wartość z zakresu 0-100 (%). Po tym za obsługę opcji odpowiada osobny skrypt OptionsManager.cs z uwagi na odpowiednie ustawienie, jak ma działać jasność i dźwięk w grze. Dla jasności, poziom jasności gry dostosowuje się do oświetlenia na danym levelu i jest zmienny, żeby działało to jednakowo dla wszystkich scen, trzeba by zrobić Post Processing i ujednolicić poziom jasności, ale brakło na to czasu.



W podmenu informacji o grze dodałem zwykły tekst TMP, odpowiednio formatowany i skalowany. Inline dodałem zmiany kolorów pojedynczych słów, czy też pogrubienie. Wszystkie teksty w mojej grze są customowe, tzn. z Google Fonts dodałem czcionki do TextMeshPro i zrobiłem gotowe czcionki do użycia w grze, tak jak poniżej.



Musiałem się trochę pobawić, ze względu na to, że nieodpowiednie stworzenie czcionek powoduje, że pomiędzy literami jest siatka. Sprawę załatwia odpowiednia rozdzielcość czcionek i ustawienie paddingu. Poza tym czcionki mają ustawiony Auto Size (Żeby się skalowały zależnie od rozmiaru ekranu i rozdzielcość), spacing character, color, gradient itd.



Play Panel natomiast ma pole tekstowe do wpisywania nazw graczy (Do 5). Poza tym w skrypcie odpowiednio skonfigurowałem, że gra nie może zacząć się bez żadnego gracza, nie możemy dodać gracza bez nazwy, to wszystko przy pomocy skryptu wyłączającego interakcję przycisków. Zmieniłem również domyślne kursory, tzn. przy wpisaniu nazwy pojawia się karetka, a przy najeżdzaniu na przyciski handle, ale tylko pod warunkiem, że są interakcyjne. Odpowiadają za to osobne skrypty widoczne poniżej.

```

using UnityEngine;
using UnityEngine.EventSystems;
using TMPro;

public class InputCursorChanger : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler
{
    public Texture2D customCursorTexture;
    private Vector2 hotSpot = Vector2.zero;
    private TMP_InputField inputField;

    void Start()
    {
        inputField = GetComponent<TMP_InputField>();
    }

    void OnPointerEnter(PointerEventData eventData)
    {
        if (inputField != null && customCursorTexture != null)
        {
            Cursor.SetCursor(customCursorTexture, hotSpot, CursorMode.Auto);
        }
    }

    void OnPointerExit(PointerEventData eventData)
    {
        if (inputField != null)
        {
            Cursor.SetCursor(null, Vector2.zero, CursorMode.Auto);
        }
    }
}

public class ButtonCursorChanger : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler, IPointerUpHandler
{
    public Texture2D cursorTexture;
    private Vector2 hotSpot = Vector2.zero;

    void OnPointerEnter(PointerEventData eventData)
    {
        Button button = GetComponent<Button>();

        if (button != null && button.interactable && cursorTexture != null)
        {
            Cursor.SetCursor(cursorTexture, hotSpot, CursorMode.Auto);
        }
    }

    void OnPointerExit(PointerEventData eventData)
    {
        Button button = GetComponent<Button>();

        if (button != null && button.interactable)
        {
            Cursor.SetCursor(null, Vector2.zero, CursorMode.Auto);
        }
    }

    void OnPointerUp(PointerEventData eventData)
    {
        Cursor.SetCursor(null, Vector2.zero, CursorMode.Auto);
    }
}

```

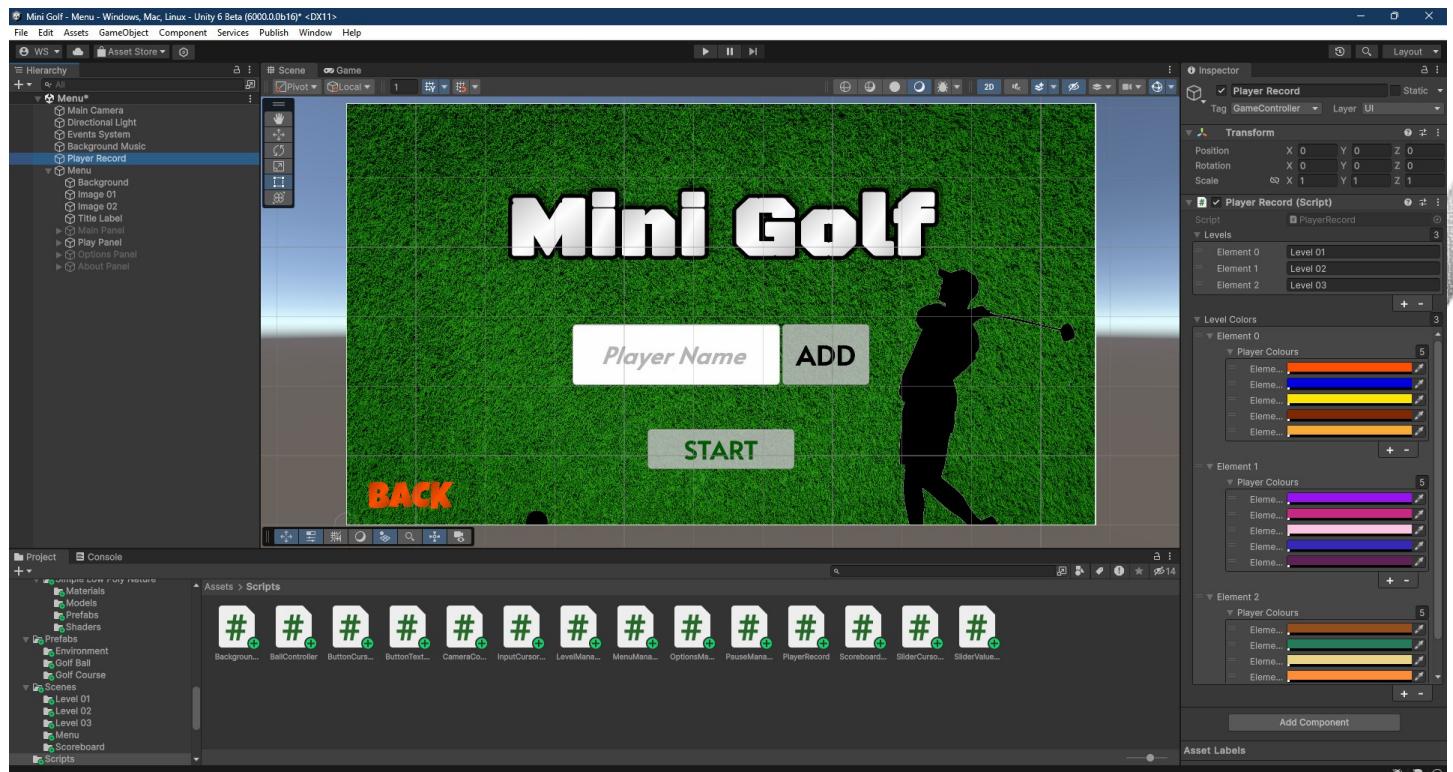
Jeden odpowiada za kurSOR Input Field, a drugi za buttony. Poza tym jest jeszcze zmiana kursora do sliderów w osobnym skrypcie. Działa to na tej samej zasadzie. Kursor zmienia się na handle, nawet jeśli będziemy trzymać LPM na sliderze i nim jeździć na boki po całym ekranie, dopiero jak puścimy włączy się domyślny pointer.

```

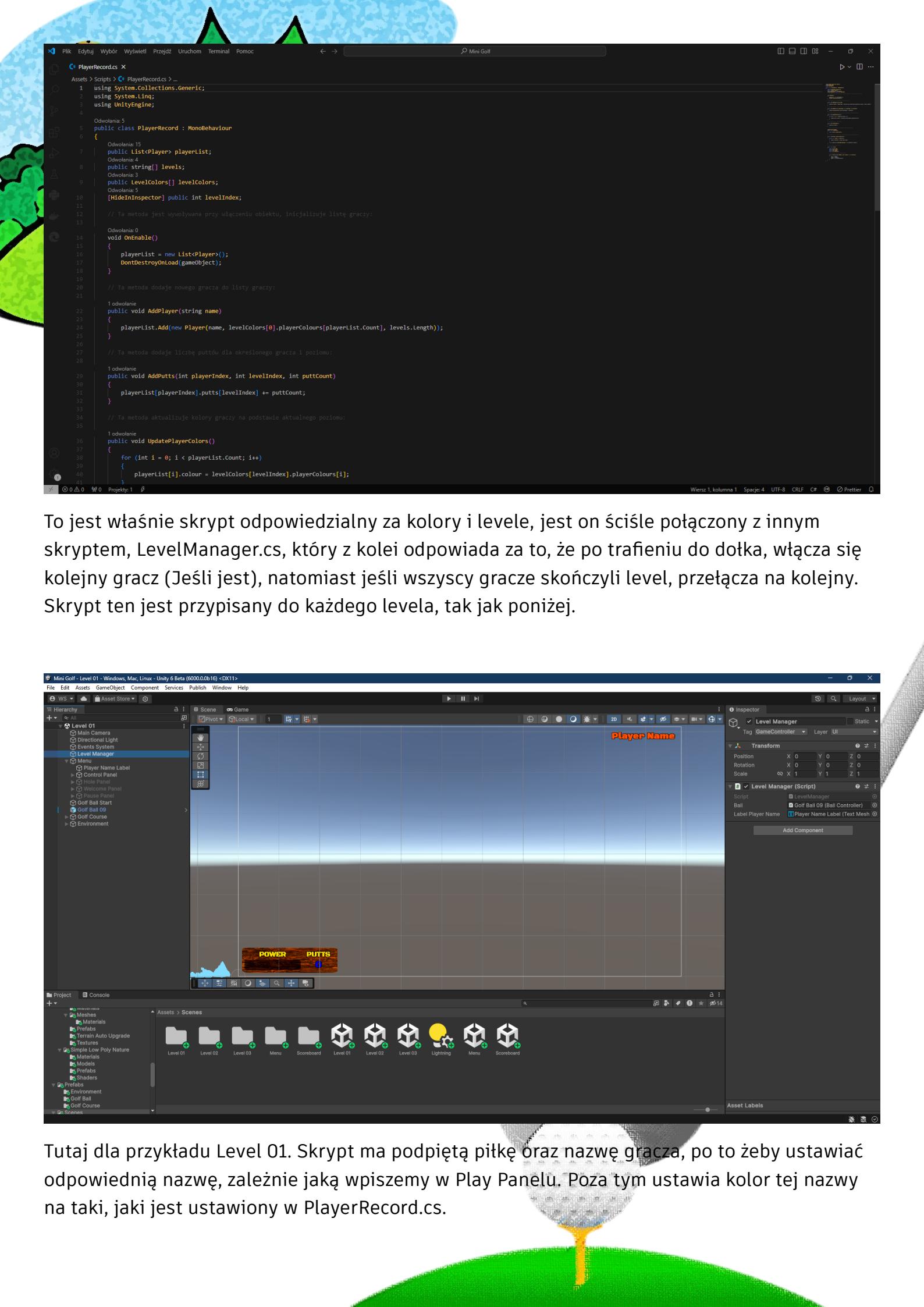
Assets > Scripts > C_SliderCursorChanger.cs ...
1 using UnityEngine;
2 using UnityEngine.EventSystems;
3 using UnityEngine.UI;
4
5 Odwołania: 0
6 public class SliderCursorChanger : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler, IPointerDownHandler
7 {
8     Odwołania: 4
9     public Texture2D cursorTexture;
10    Odwołania: 3
11    private Vector2 hotSpot = Vector2.zero;
12    Odwołania: 3
13    private bool isDragging = false;
14
15    // Ta metoda zmienia kursor na nietypowy, gdy wskaźnik wchodzi na slider:
16    Odwołania: 0
17    public void OnPointerEnter(PointerEventData eventData)
18    {
19        Slider slider = GetComponent<Slider>();
20        if (slider != null && slider.interactable && cursorTexture != null)
21        {
22            Cursor.SetCursor(cursorTexture, hotSpot, CursorMode.Auto);
23        }
24
25        // Ta metoda przywraca domyślny kursor, gdy wskaźnik opuszcza slider:
26        Odwołania: 0
27        public void OnPointerExit(PointerEventData eventData)
28        {
29            Slider slider = GetComponent<Slider>();
30            if (slider != null && slider.interactable && !isDragging)
31            {
32                Cursor.SetCursor(null, Vector2.zero, CursorMode.Auto);
33            }
34
35            // Ta metoda zmienia kursor na nietypowy podczas przeciągania slidera:
36            Odwołania: 0
37            public void OnPointerDown(PointerEventData eventData)
38            {
39                Slider slider = GetComponent<Slider>();
40                if (slider != null && slider.interactable)
41                {
42                    isDragging = true;
43                    Cursor.SetCursor(cursorTexture, hotSpot, CursorMode.Auto);
44                }
45            }
46        }
47    }
48
49    // Ta metoda zmienia cursor na nietypowy, gdy wskaźnik wchodzi na slider:
50    Odwołania: 0
51    public void OnPointerEnter(PointerEventData eventData)
52    {
53        Slider slider = GetComponent<Slider>();
54        if (slider != null & slider.interactable & cursorTexture != null)
55        {
56            Cursor.SetCursor(cursorTexture, hotSpot, CursorMode.Auto);
57        }
58
59        // Ta metoda aktualizuje tekst na podstawie wartości slidera:
60        Odwołania: 3
61        private void UpdateValueText(float value)
62        {
63            int displayValue = Mathf.RoundToInt(value * 100);
64            valueText.text = displayValue.ToString();
65
66            // Ta metoda jest wywoływana przy zniesieniu obiektu, usuwa listener ze slidera:
67            Odwołania: 0
68            private void OnDestroy()
69            {
70                if (slider != null)
71                {
72                    slider.onValueChanged.RemoveListener(UpdateValueText);
73                }
74            }
75        }
76    }
77}

```

To są właśnie skrypty do obsługi sliderów, tzn. wyświetlanie wartości procentowej względem poziomu slidera oraz zmiana kursora.



Play Panel wymagał ode mnie stworzenia skryptu PlayerRecord.cs, który jest odpowiedzialny za obsługę poziomów gry oraz liczby graczy. Gra z założenia jest wieloosobowa, ale pod względem, że może grać w nią do 5 graczy kolejno po sobie, nie jednocześnie. Kod jest napisany na tyle elastycznie, że gdybym chciał, mogę dodać dowolną liczbę graczy i leveli. Kolory są odpowiedzialne za rozróżnianie graczy poza nazwą. Ustawiają one odpowiednio nazwę gracza i LineRendererer na kolor ustawiony w zmiennej publicznej, zależny od levela gry.



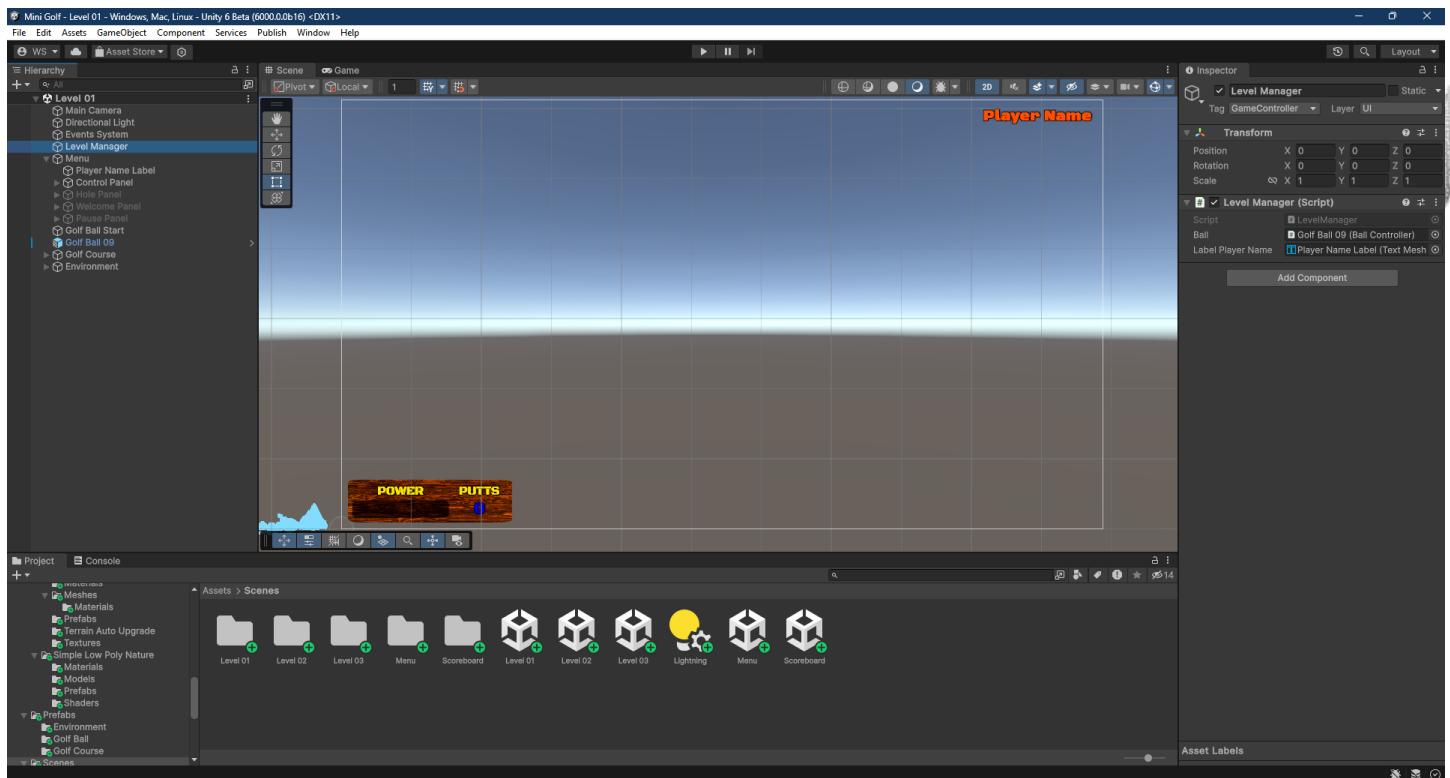
```

PlayerRecord.cs X
Assets > Scripts > C# PlayerRecord.cs > ...
1  using System.Collections.Generic;
2  using System.Linq;
3  using UnityEngine;
4
5  Odwołania: 5
6  public class PlayerRecord : MonoBehaviour
7  {
8      Odwołania: 15
9      public List<Player> playerList;
10     Odwołania: 4
11     public string[] levels;
12     Odwołania: 3
13     public LevelColors[] levelColors;
14     Odwołania: 5
15     [HideInInspector] public int levelIndex;
16
17     // Ta metoda jest wywoływana przy włączeniu obiektu, inicjalizuje listę graczy:
18
19     Odwołania: 0
20     void OnEnable()
21     {
22         playerList = new List<Player>();
23         DontDestroyOnLoad(gameObject);
24     }
25
26     // Ta metoda dodaje nowego gracza do listy graczy:
27
28     Odwołanie
29     public void AddPlayer(string name)
30     {
31         playerList.Add(new Player(name, levelColors[0].playerColours[playerList.Count], levels.Length));
32     }
33
34     // Ta metoda dodaje liczbę puttów dla określonego gracza i poziomu:
35
36     Odwołanie
37     public void AddPutts(int playerIndex, int levelIndex, int puttCount)
38     {
39         playerList[playerIndex].putts[levelIndex] += puttCount;
40     }
41
42     // Ta metoda aktualizuje kolory graczy na podstawie aktualnego poziomu:
43
44     Odwołanie
45     public void UpdatePlayerColors()
46     {
47         for (int i = 0; i < playerList.Count; i++)
48         {
49             playerList[i].colour = levelColors[levelIndex].playerColours[i];
50         }
51     }

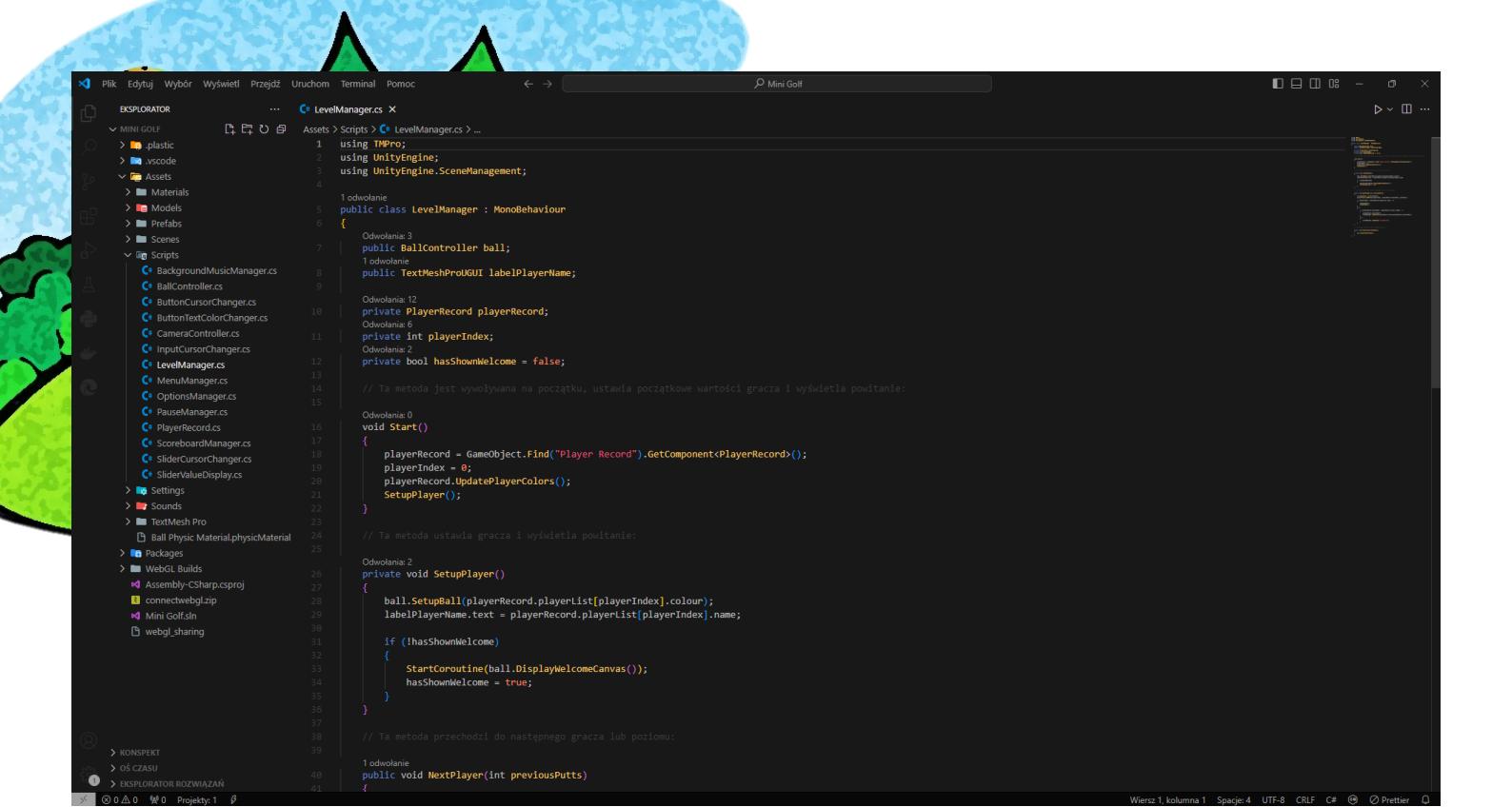
```

Wiersz 1, kolumna 1 Spacje: 4 UTF-8 CRLF C# ⌂ Prettier

To jest właśnie skrypt odpowiedzialny za kolory i levele, jest on ściśle połączony z innym skryptem, `LevelManager.cs`, który z kolei odpowiada za to, że po trafieniu do dołka, włącza się kolejny gracz (Jeśli jest), natomiast jeśli wszyscy gracze skończyli level, przełącza na kolejny. Skrypt ten jest przypisany do każdego levela, tak jak poniżej.



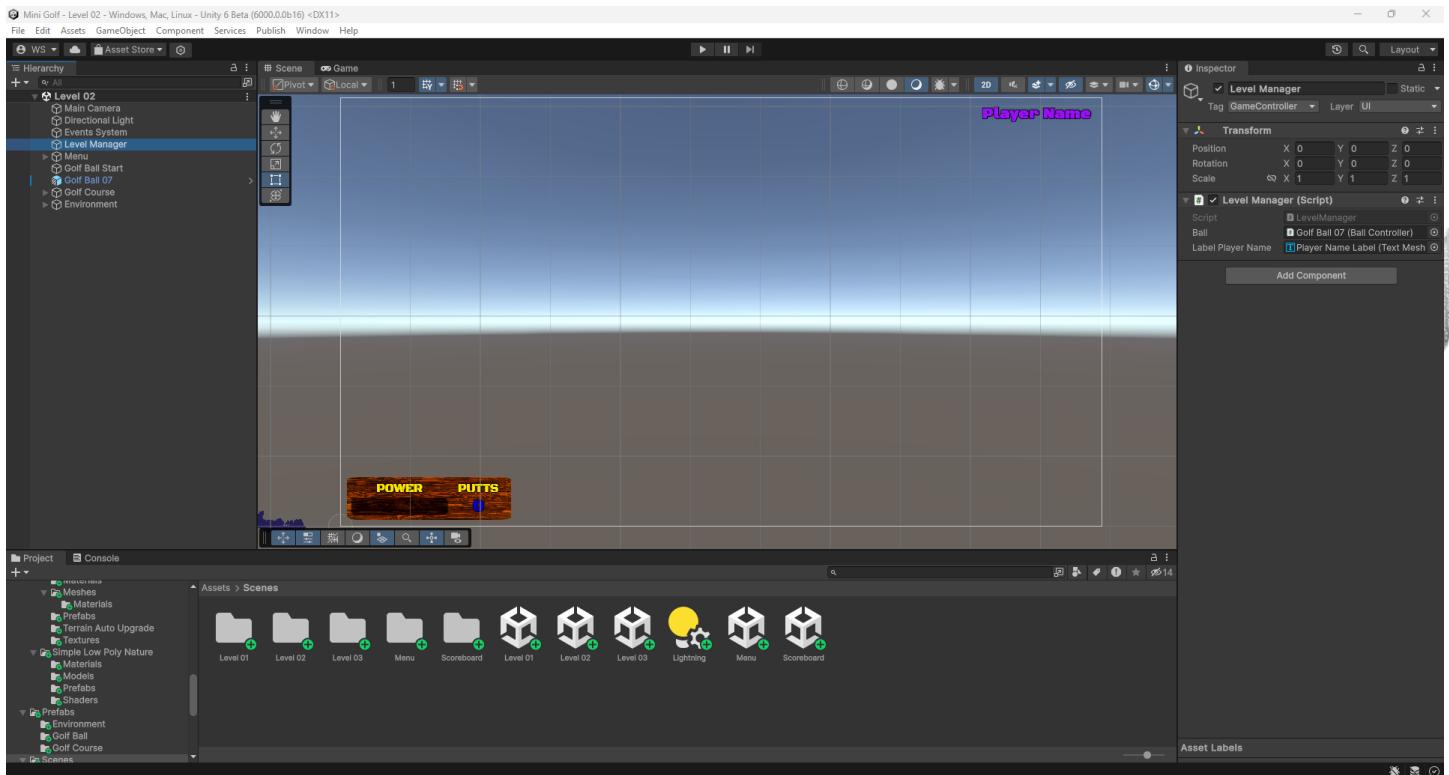
Tutaj dla przykładu Level 01. Skrypt ma podpiętą piłkę oraz nazwę gracza, po to żeby ustawać odpowiednią nazwę, zależnie jaką wpiszemy w Play Panelu. Poza tym ustawia kolor tej nazwy na taki, jaki jest ustwiony w `PlayerRecord.cs`.



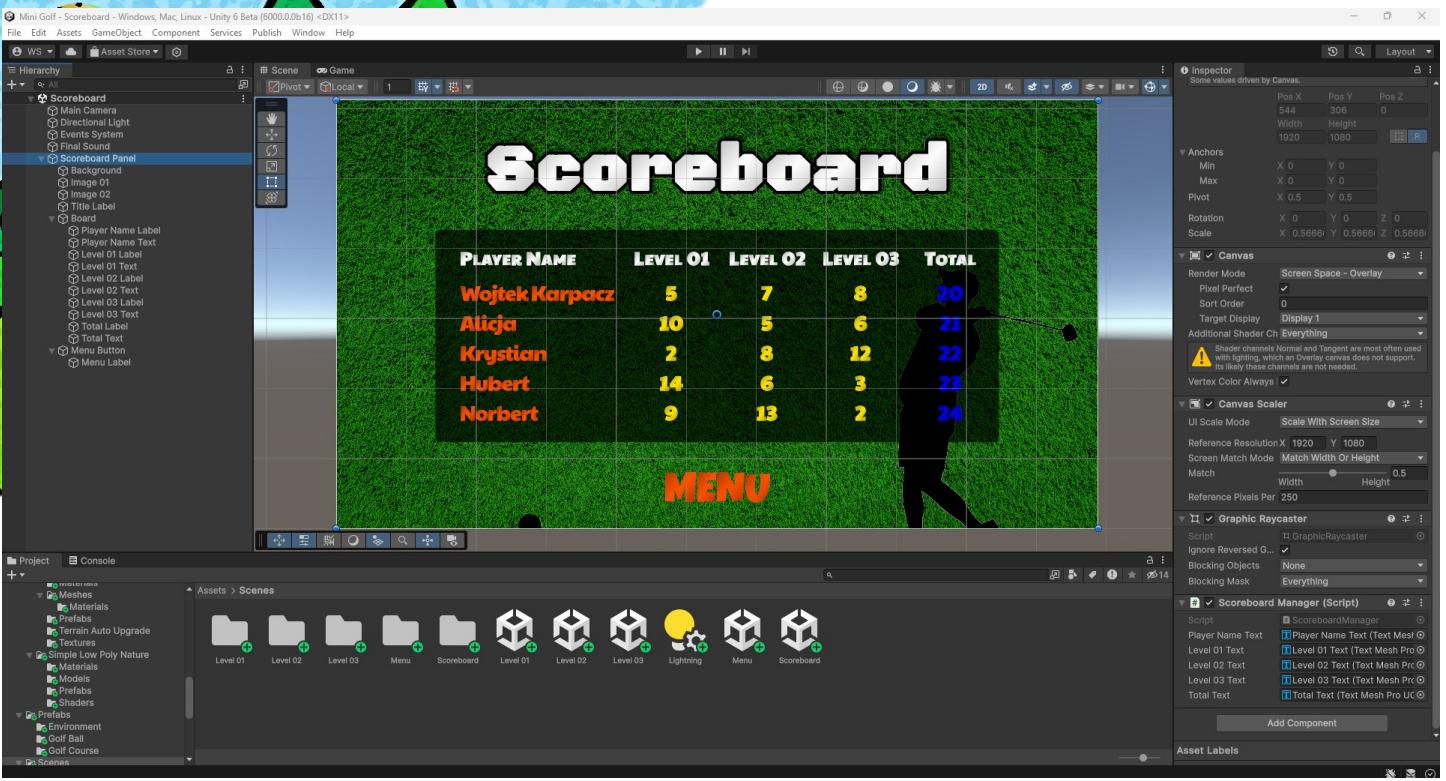
```
1 using TMPro;
2 using UnityEngine;
3 using UnityEngine.SceneManagement;
4
5 // odwołanie
6 public class LevelManager : MonoBehaviour
7 {
8     // odwołania
9     public BallController ball;
10    public TextMeshProUGUI labelPlayerName;
11
12    // odwołania
13    private PlayerRecord playerRecord;
14    private int playerIndex;
15    private bool hasShownWelcome = false;
16
17    // Ta metoda jest wywoływana na początku, ustawia początkowe wartości gracza i wyświetla powitanie:
18
19    void Start()
20    {
21        playerRecord = GameObject.Find("Player Record").GetComponent<PlayerRecord>();
22        playerIndex = 0;
23        playerRecord.UpdatePlayerColors();
24        SetupPlayer();
25    }
26
27    // Ta metoda ustala gracza i wyświetla powitanie:
28
29    private void SetupPlayer()
30    {
31        ball.SetupBall(playerRecord.playerList[playerIndex].colour);
32        labelPlayerName.text = playerRecord.playerList[playerIndex].name;
33
34        if (!hasShownWelcome)
35        {
36            StartCoroutine(ball.DisplayWelcomeCanvas());
37            hasShownWelcome = true;
38        }
39    }
40
41    // Ta metoda przechodzi do następnego gracza lub poziomu:
42
43    void NextPlayer(int previousPutts)
44    {
45    }
46}
```

Wiersz 1, kolumna 1 Spacja: 4 UTF-8 CRLF C# Prettier

Tak wygląda ten skrypt.



Tak samo to wygląda dla Levelu 02 i tym samym kolejnych leveli, po prostu odpowiednio przypisujemy obiekt piłki i Player Name Label. Kolory nazw graczy i linii celowania zmieniają się pomiędzy levelami z uwagi na to, że różne levele mają różną oprawę graficzną (Kolorystykę), więc nie zawsze, np. fioletowy będzie odpowiedni dla wszystkich leveli.



Kolejna kwestia z menu, to wyniki, czyli Scoreboard. Zrobiony jest na bazie menu głównego dla spójności. Zrobiłem prosty czarny Panel z delikatną przezroczystością. Na nim dodałem liczne TMP. Odpowiednio je wypożyczyłem. Ustawiłem No Wrap, żeby jeśli nazwa gracza będzie dłuża, to rozmiar pola się nie zwiększy, a tekst nie przejdzie do kolejnej linii, tylko będzie zmieniał rozmiar na mniejszy. Jest ustawiony Auto Size dla nazwy gracza, natomiast w skrypcie, który oczywiście jest i obsługuje liczbę uderzeń dla danego gracza na poszczególnych levelach (Również sumuje i po tym sortuje na liście), ustawiłem, że rozmiar tekstu liczby uderzeń ustawia się do nazwy gracza, bo nigdy liczba uderzeń nie będzie dłuższa niż nazwa gracza. Jakby wszystko było na auto, to nazwa gracza byłaby mniejsza od liczby uderzeń.

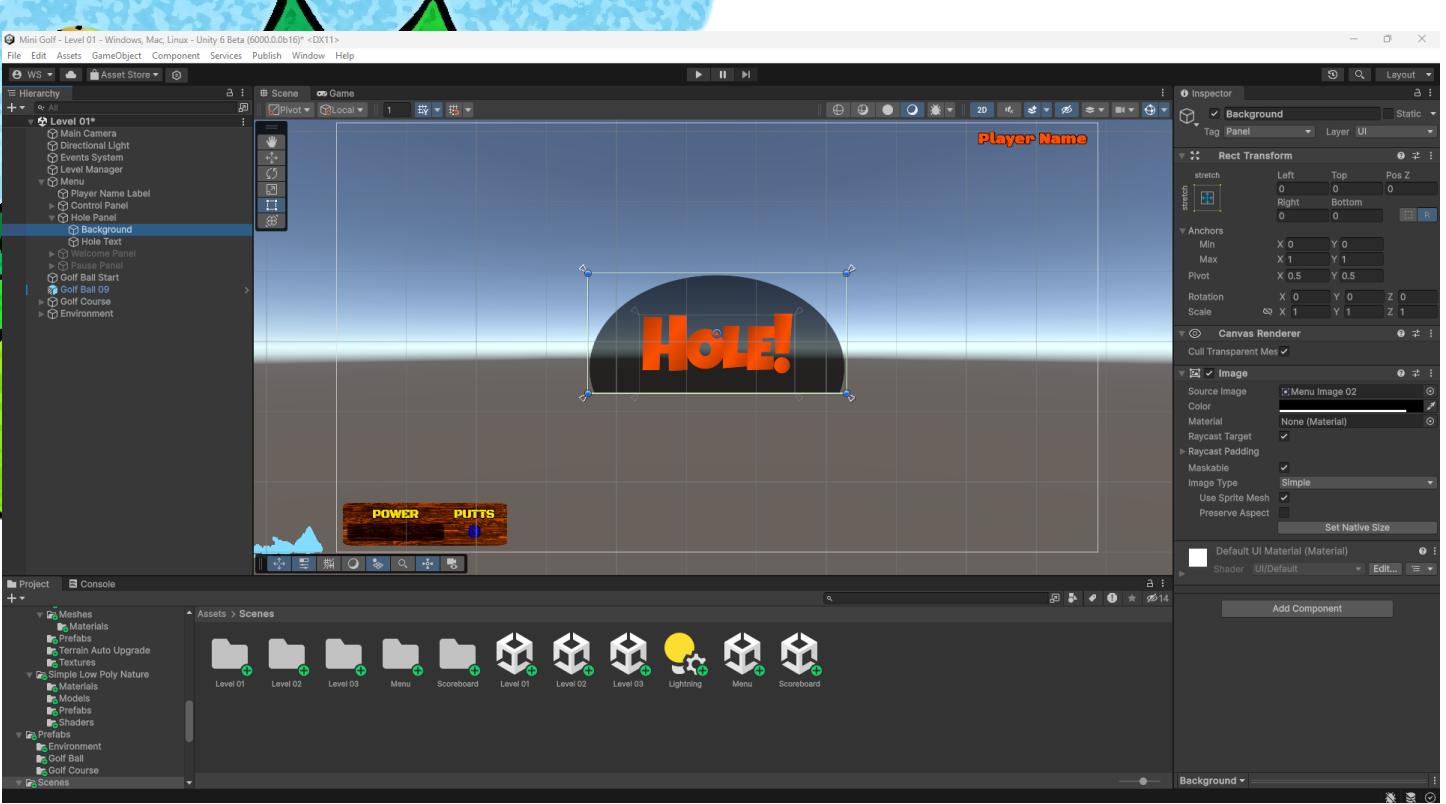
```

    EKSPLORATOR
    > MINI GOLF
    > plastic
    > vscode
    > Assets
    > Materials
    > Models
    > Prefabs
    > Scenes
    > Scripts
        ScoreboardManager.cs
        BackgroundMusicManager.cs
        BallController.cs
        ButtonColorChanger.cs
        CameraController.cs
        InputCursorChanger.cs
        LevelManager.cs
        MenuManager.cs
        OptionsManager.cs
        PauseManager.cs
        PlayerRecords.cs
        ScoreboardManager.cs
        SliderColorChanger.cs
        SliderValueDisplay.cs
    > Settings
    > Sounds
    > TextMesh Pro
        Ball_Physics_Material_physicMaterial
    > Packages
    > WebGL Builds
        Assembly-CSharp.csproj
        connectwedge.zip
        MiniGolf.jsn
        webgl_sharing
    > KONSPEKT
    > OŚ CZASU
    > EKSPLORATOR ROZWIĄZANIA

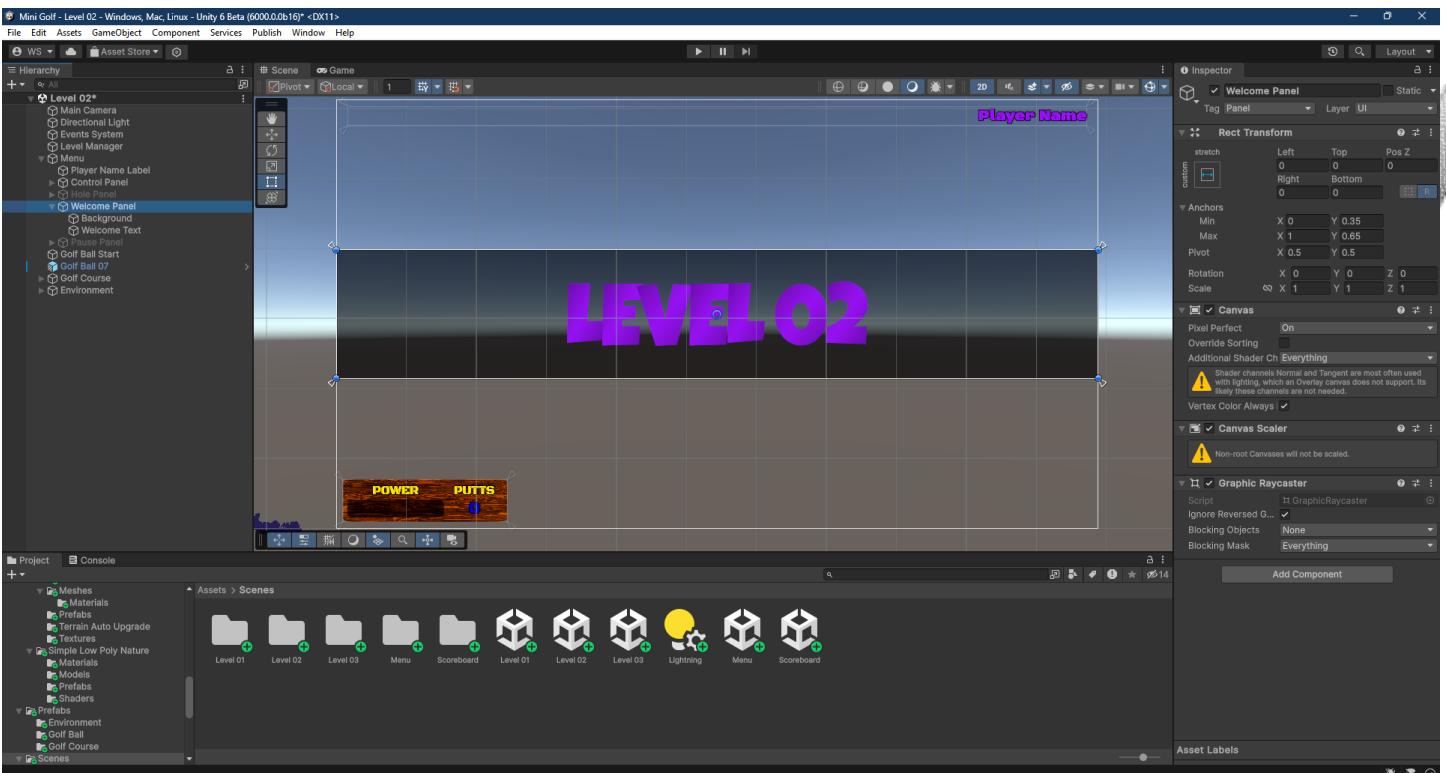
    ScoreboardManager.cs
    Assets > Scripts > ScoreboardManager.cs ...
    1  using TMPro;
    2  using UnityEngine;
    3  using UnityEngine.SceneManagement;
    4
    5  public class ScoreboardManager : MonoBehaviour
    6  {
    7      Odwołania: 3
    8      public TextMeshProUGUI playerNameText;
    9      Odwołania: 3
    10     public TextMeshProUGUI level01Text;
    11     Odwołania: 3
    12     public TextMeshProUGUI level02Text;
    13     Odwołania: 3
    14     public TextMeshProUGUI level03Text;
    15     Odwołania: 3
    16     private PlayerRecord playerRecord;
    17
    18     // Ta metoda jest wywoływana na początku, inicjalizuje tablicę wyników:
    19
    20     void Start()
    21     {
    22         playerRecord = GameObject.Find("Player Record").GetComponent<PlayerRecord>();
    23
    24         // Czyszczenie tekstów tablicy wyników:
    25         playerNameText.text = "";
    26         level01Text.text = "";
    27         level02Text.text = "";
    28         level03Text.text = "";
    29         totalText.text = "";
    30
    31         // Wypełnianie tablicy wyników danymi graczy:
    32         foreach (var player in playerRecord.GetScoreboardList())
    33         {
    34             playerNameText.text += player.name + "\n";
    35             level01Text.text += player.putts[0].ToString() + "\n";
    36             level02Text.text += player.putts[1].ToString() + "\n";
    37             level03Text.text += player.putts[2].ToString() + "\n";
    38             totalText.text += player.totalPutts.ToString() + "\n";
    39         }
    40
    41         // Ta metoda jest wywoływana co klatkę, synchronizuje rozmiar czcionki wszystkich tekstów:
    42     }

    Wiersz 1, kolumna 1  Spacja: 4  UTRF-8  CRLF  C#  Prettier  Q
  
```

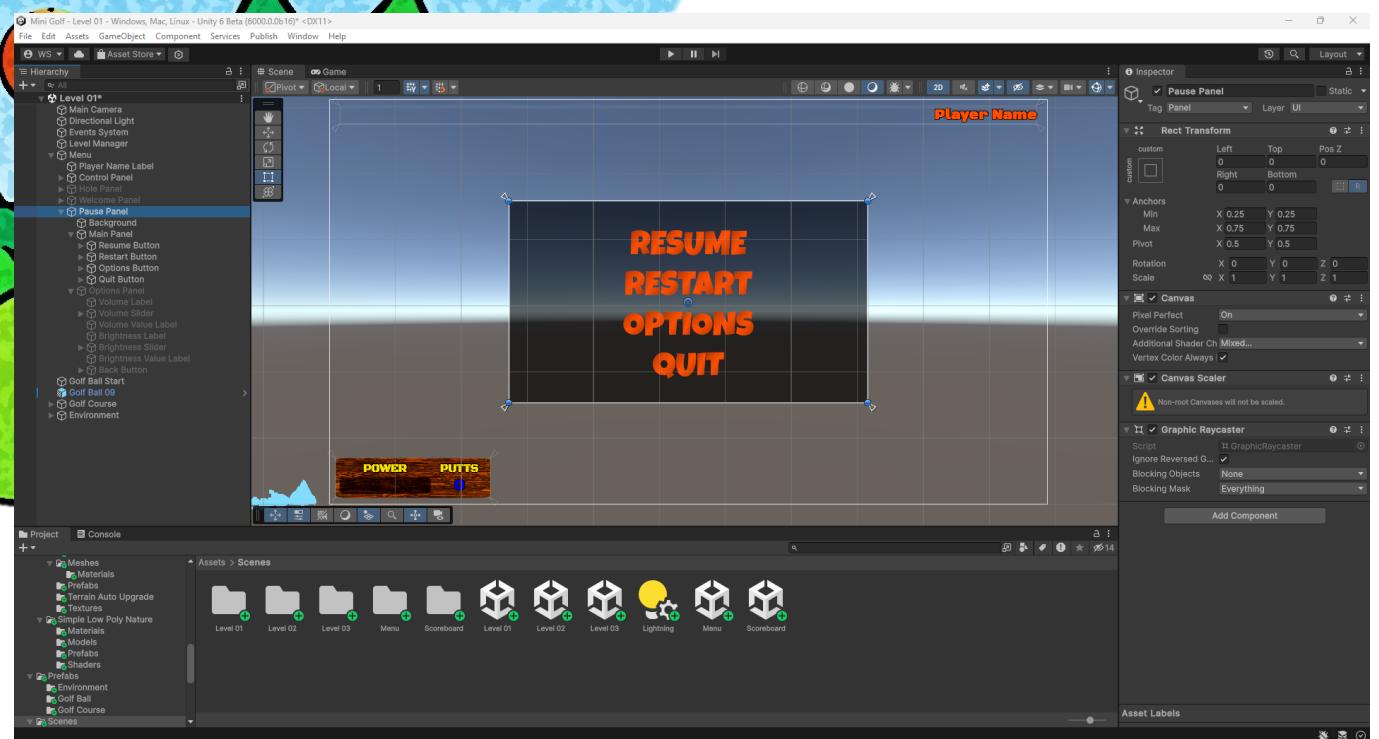
To jest właśnie skrypt ScoreboardMaganer.cs do obsługi wyników.



Poza tym z kwestii Canvas, dodałem powiadomienie o trafieniu w dołek dla wszystkich. Kolor tekstu ustawia się zależnie od koloru przypisanego danemu graczu na konkretnym levelu. Skrypt do obsługi tego obiektu jest wpisany w skrypt do obsługi piłki BallController.cs i tam są przypisane do zmiennych publicznych obiekty.



Idąc za ciosem zrobiłem też obiekt wprowadzający w level. Pojawia się raz dla pierwszego gracza na levelu. Jego kolor jest po prostu ustawiony odpowiednio kolorystycznie do 2D dla danego levela, nie jest zależny od koloru gracza. Skrypt obsługujący jest również w pliku BallController.cs.



Z kwestii 2D muszę również poruszyć temat Pause Menu, które jest na każdym levelu. Podczas rozgrywki można klawiszem ESC włączyć takie menu, wtedy gra pauzuje się (Time.timeScale = 0). Mamy do wyboru wznowić grę, którą możemy również wznowić ponownym naciśnięciem klawisza ESC. Restart służy do resetowania rozgrywki na danym levelu dla danego gracza, jest to o tyle ważne, że nie resetuje levela dla wszystkich graczy i nie resetuje całej gry, co myślę, że byłoby nieodpowiednie. Jest przypisana metoda wywołana ze skryptu BallController.cs, która zeruje wszystkie wartości, uderzenia etc. i nie wlicza ich do wyników ogólnych. Poza tym mamy panel opcji, taki sam jak w głównym menu, możemy w nim zmienić to samo. Oczywiście wartości są zapamiętane, ponieważ obydwa podmenu obsługuje ten sam skrypt OptionsManager.cs. Obsługa klawisza ESC do cofania i wznowiania gry jest również zastosowana. Przycisk Quit opuszcza rozgrywkę do głównego menu. Za menu odpowiada skrypt PauseManager.cs widoczny poniżej.

The screenshot shows the Unity Editor interface with the following details:

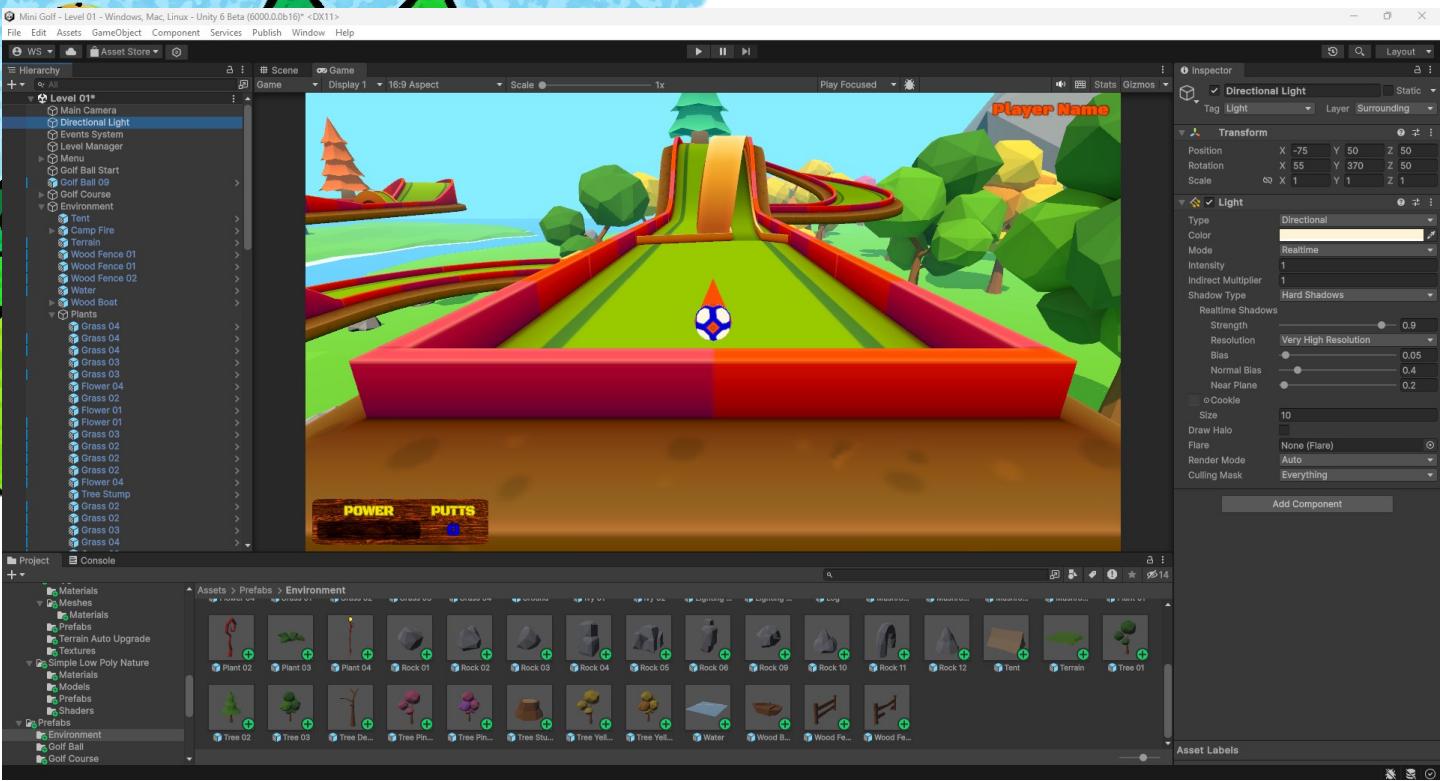
- Project Tab:** Shows the project structure for "MINI GOLF".
- Scripts Tab:** Displays the "PauseManager.cs" script under the "Assets > Scripts" path.
- Code Content:** The script defines a `PauseManager` class that manages game pause states and UI panels. It includes fields for pausePanel, mainPanel, optionsPanel, resumeButton, restartButton, optionsButton, quitButton, backButton, and originalTextColor. It also includes methods for `Start`, `Update`, and `OnEnable`. The `Start` method initializes the panels and sets up click listeners for the UI buttons. The `Update` method checks if the game is paused and updates the UI accordingly. The `OnEnable` method starts the background music.



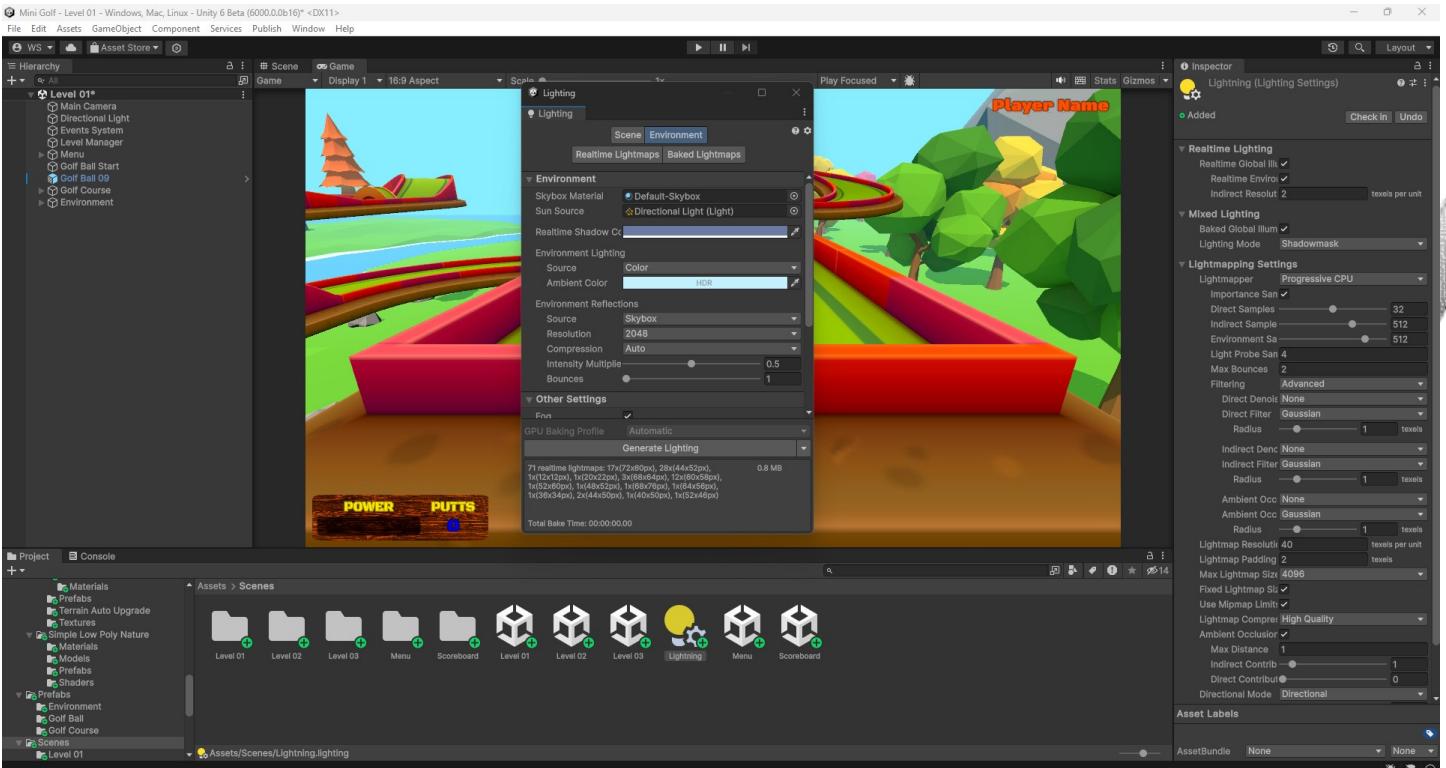
I to będzie na tyle z 2D. Przejdźmy do 3D. Zaczynając od Levela 01. Zrobiłem świat z assetów. Zmodyfikowałem, dodałem mesh collidery, ustawiłem tagi OffCourse, bo piłka po wypadnięciu poza tor musi mieć trigger, że wypadła i ma wrócić do ostatniej pozycji. Pobawiłem się z ustawieniem i efekt widzimy na obrazku. I teraz z kwestii oświetlenia, renderowania, kamery itd.



Ustawiłem Main Camera, zrobiłem Solid Color zamiast Skyboxa i osiągnąłem fajne niebo z chmurkami, zmieniłem też ustawienia graficzne na wyższe.



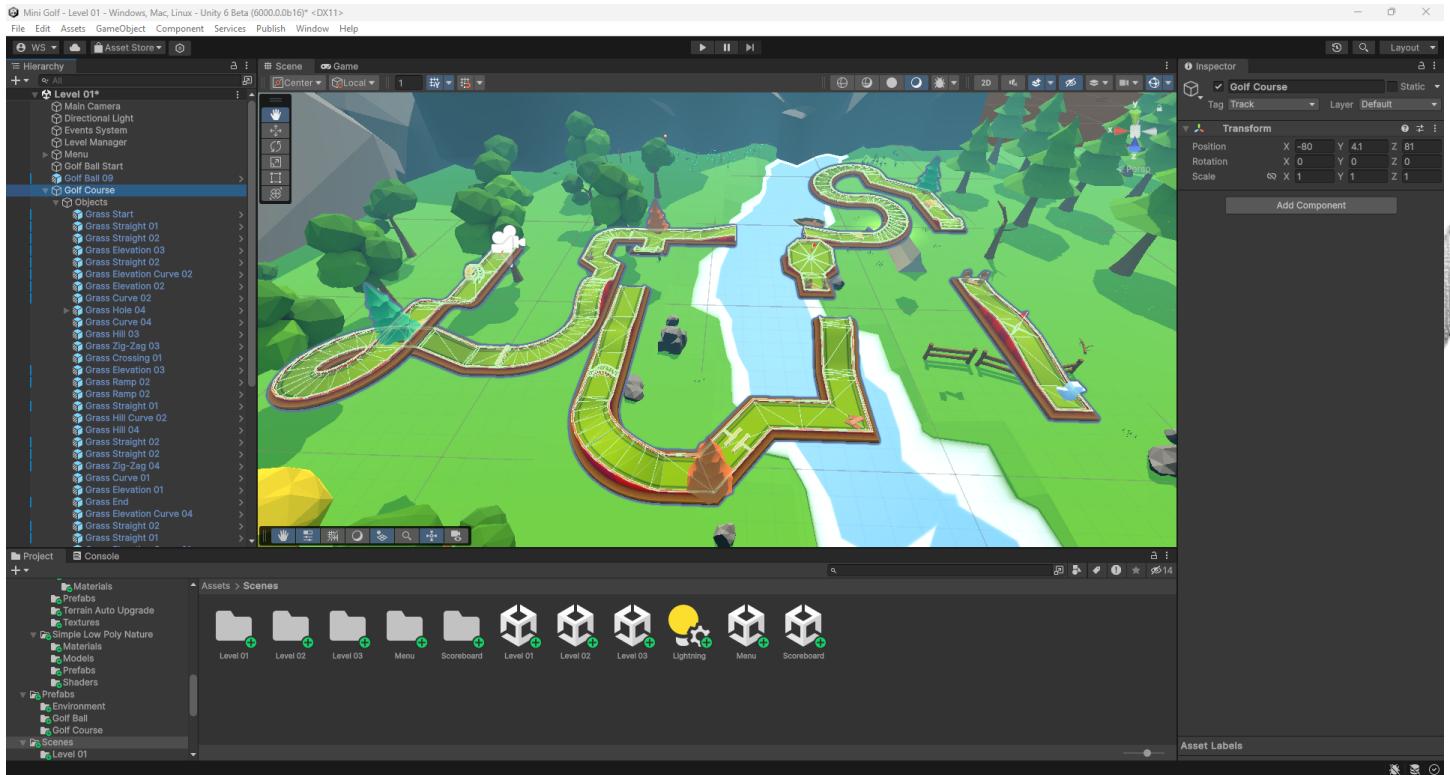
I teraz słoneczko, czyli Directional Light. Tutaj ustawiłem kolor oświetlenia, rodzaj, cienie, ustawienia graficzne itd. Warto dodać że ustawienia kamery i światła będą różniły się zależnie od poziomu. Tutaj kolor jest inny niż na Levelu 02, bo tam jest ciemno.



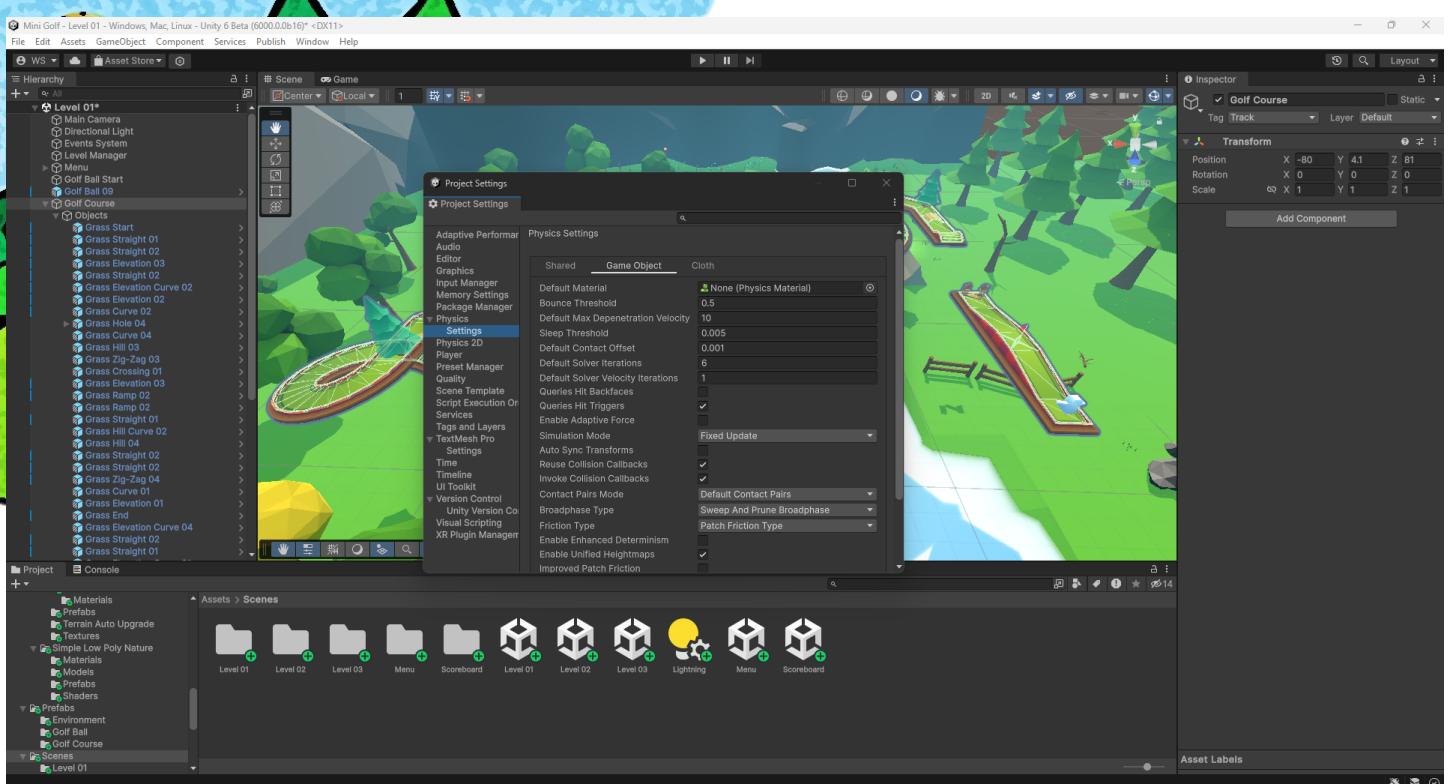
Zrenderowałem oświetlenie dla wszystkich scen z osobna, mam zrobiony plik z globalnymi ustawieniami światła, ale poza tym dla każdego poziomu są inne ustawienia środowiskowe, tutaj pozwoliłem sobie na delikatną mgiełkę, ustawiłem stały kolor promieni świetlnych padających na tekstury, cienie itp. Zrobiłem to, ponieważ moje prefaby mają lightmaps, więc pozwoliło mi to uzyskać bardzo fajne żywe tekstury z odpowiednimi cieniami. Woda świetnie wygląda za sprawą shadera, jest również animacja ogniska.



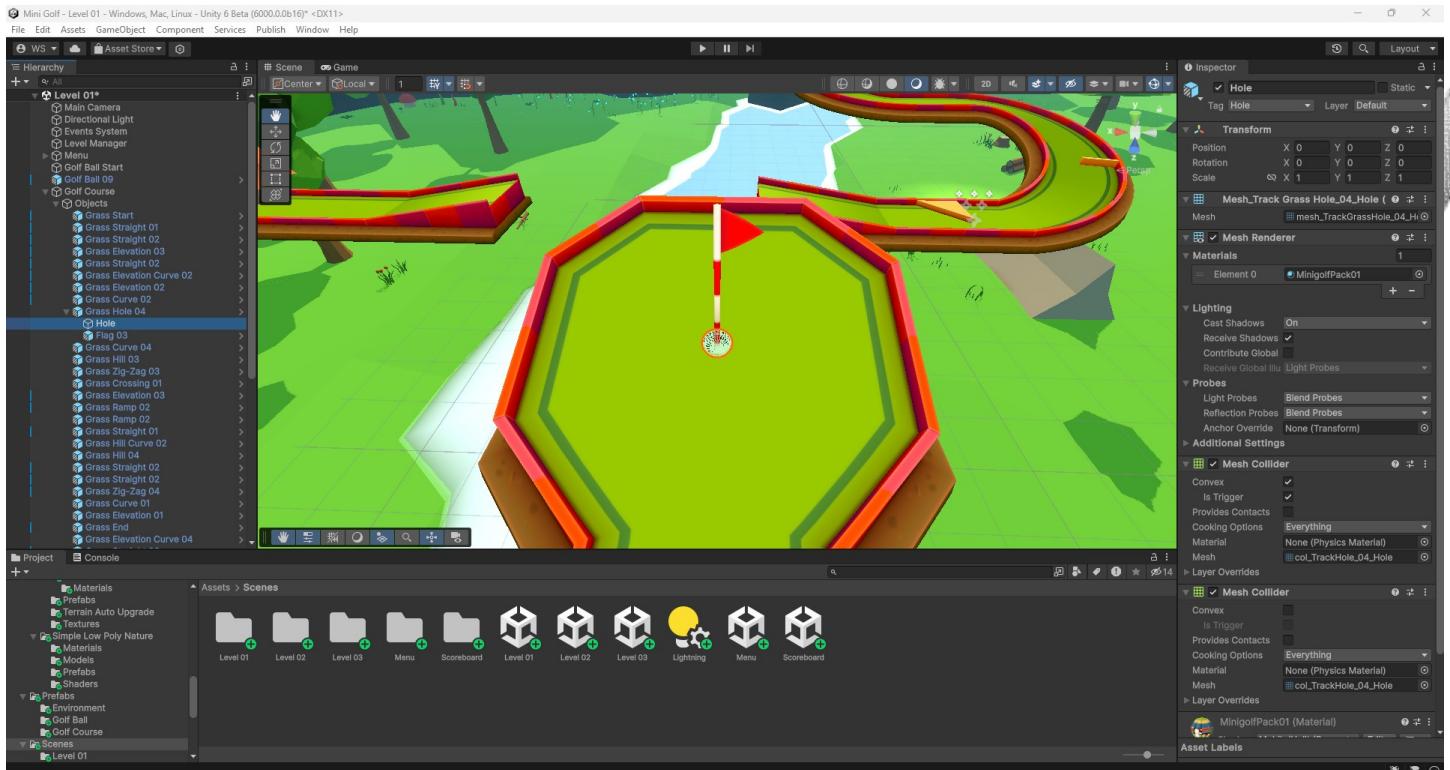
A tak to wygląda podczas rozgrywki z trybu swobodnej kamery (Klawisz Z).



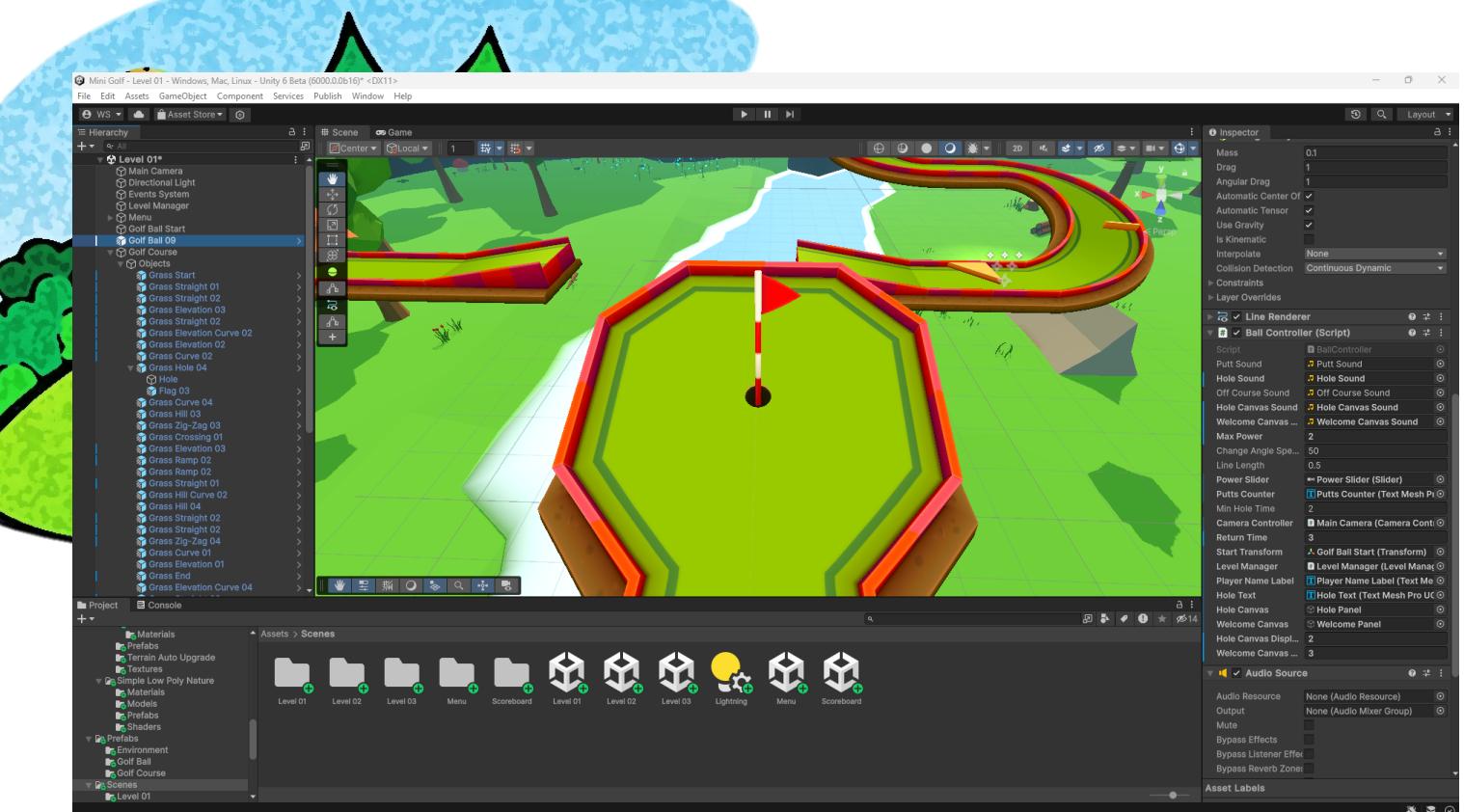
Na poziomie 01 zmodyfikowałem również tor golfowy na bardziej skomplikowany, zrobiłem jakieś hopki, kręciołki i tego typu sprawy. Dodałem też z Nature Packa jakieś drzewa, deski itp. jako utrudnienia z colliderami. Wszystko poustawiałem. Czasem zdarzy się, że piłka zahaczy o krawędź obiektu, szczególnie podczas uderzenia pod górkę. Jest to spowodowane fizyką projektu, na jakim etapie mamy styczność z colliderem. Myślę, że i tak ustawiłem to bardzo dobrze, ale czasem zdarzy się przy mocnym uderzeniu, że piłka zahaczy o krawędź, mimo, że jej nie widać.



A ustawia się to w ustawieniach fizyki (Project Settings). To dokładnie opcja Default Contact Offset, im bliżej zera, tym teoretycznie ten kontakt z krawędziami powinien być minimalny, ale jest różnie. 0.001 to najbardziej optymalna wartość dla mnie. Warto też dodać, że zmieniłem Bounce Treshold na 0.5 bo jest to powiązane również dokładnie z tym samym efektem.



Na torze mamy dołki, które mają odpowiedni tag Hole, który wywołuje trigger do Hole Canvas i przechodzi po oczekaniu 2s (Bo przecież piłka może zahaczyć, ale nie wpadnie) do następnego gracza lub levela. Wartości sekundowe czy takie parametry ustawiam w publicznych zmiennych dla skryptu piłki, tak jak widać dalej.



Dodane wartości i obiekty do skryptu piłki dla Levela 01.



Screenshot of the Unity Editor showing the BallController.cs script in the code editor. The script is a MonoBehavior with various public and private variables and methods. It includes imports for System.Collections, TMPro, UnityEngine, and UnityEngine.UI. The code handles ball movement, sound effects, and interacts with other game components like the Main Camera and Level Manager.

```

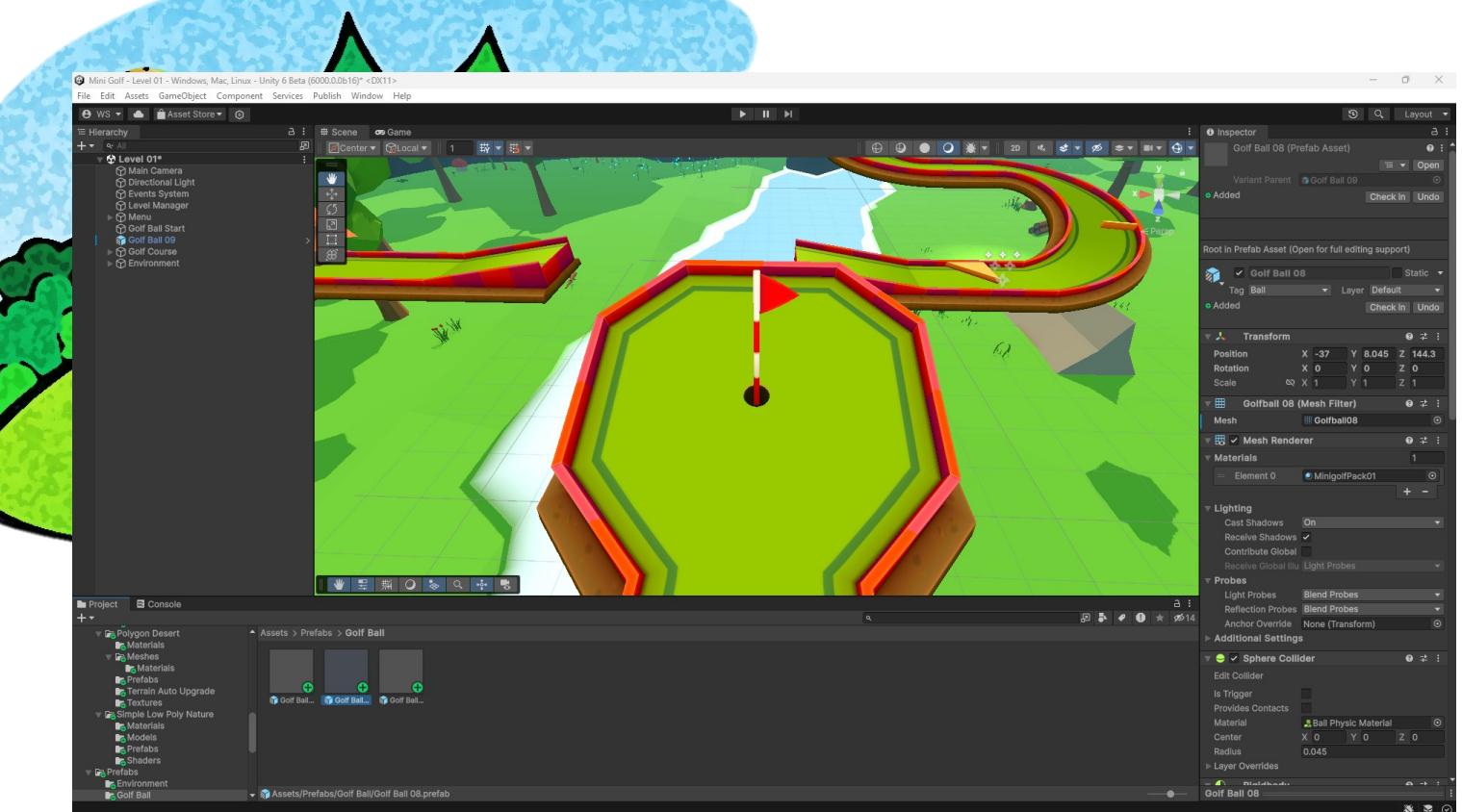
using System.Collections;
using TMPro;
using UnityEngine;
using UnityEngine.UI;

public class BallController : MonoBehaviour
{
    // Deklaracja zmiennych publicznych, które będą używane w inspektorze Unity:
    public AudioClip puttSound, holeSound, offCourseSound, holeCanvasSound, welcomeCanvasSound;
    public float maxPower; // Maksymalna moc uderzenia piłki.
    public float changeAngleSpeed; // Szybkość zmiany kąta.
    public float lineLength; // Długość linii pokazującej kierunek uderzenia.
    public Slider powerSlider; // Slider mocy.
    public TextMeshProUGUI putsCounter; // Licznik uderzeń.
    public float minHoleTime; // Minimalny czas spędzony w dołku.
    public CameraController cameraController; // Kontroler kamery.
    public float returnTime = 3f; // Czas powrotu piłki na poprzednią pozycję.
    public Transform startTransform; // Początkowa pozycja piłki.
    public LevelManager levelManager; // Menadżer poziomu.
    public TextMeshProUGUI playerNameLabel; // Etykieta nazwy gracza.
    public TextMeshProUGUI holeText; // Etykieta dołka.
    public GameObject holeCanvas; // Canvas dołka.
    public GameObject welcomeCanvas; // Canvas powitalny.
    public float holeCanvasDisplayTime = 2f; // Czas wyświetlania canvasa dołka.
    public float welcomeCanvasDisplayTime = 3f; // Czas wyświetlania canvasa powitalnego.

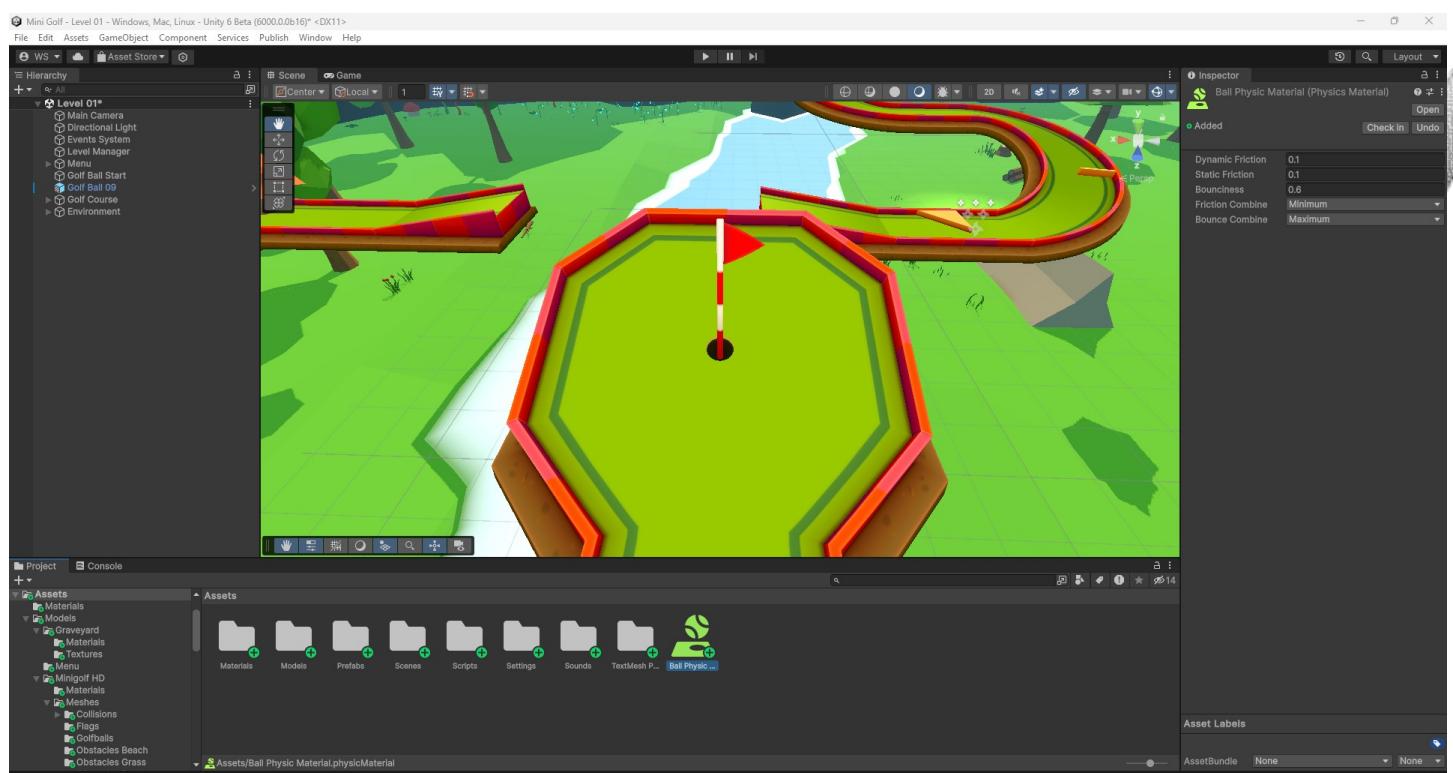
    // Deklaracja zmiennych prywatnych:
    private AudioSource audioSource;
    private LineRenderer line;
}

```

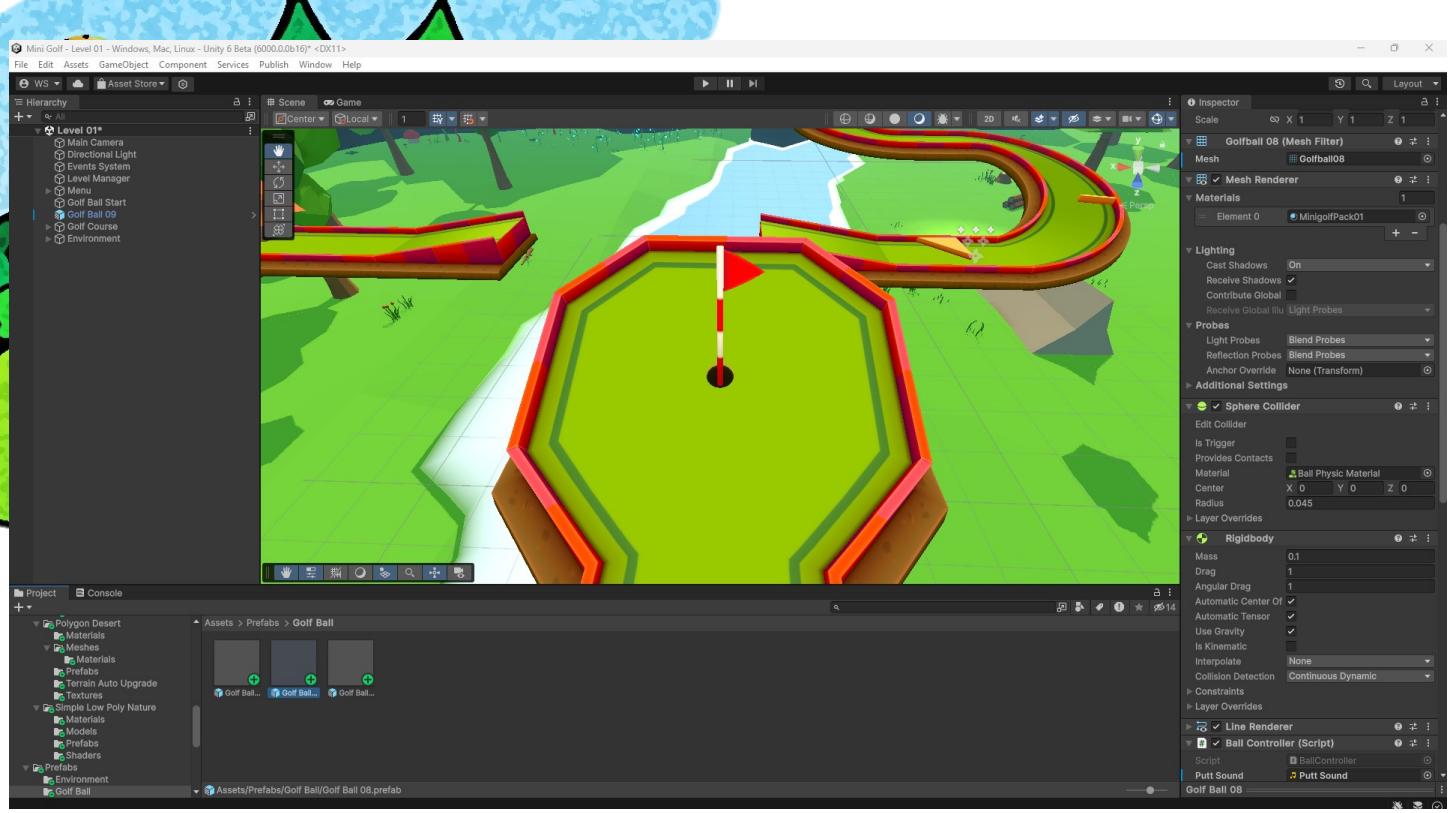
A tutaj widzimy skrypt BallController.cs, czyli najważniejszy skrypt w tej grze. Odpowiada za masę funkcji, masa linijek kodu, wszystko odpowiednio opisane i skomentowane w kodzie. Przypisany do wszystkich piłek w tej grze i trzęsie całą ekipą obiektów i spina je w kupę.



Wszystkie piłki mam dodane do prefabów. Piłki właściwie zrobiłem sam. Zrobiłem tekstury (Wzorki i kolorki), po to żeby pasowały do poszczególnych leveli, bo na każdym jest inną teksturą piłki. Wszystkie łącza ta sama fizyka i skrypty, jedynie pozycje mają inne.



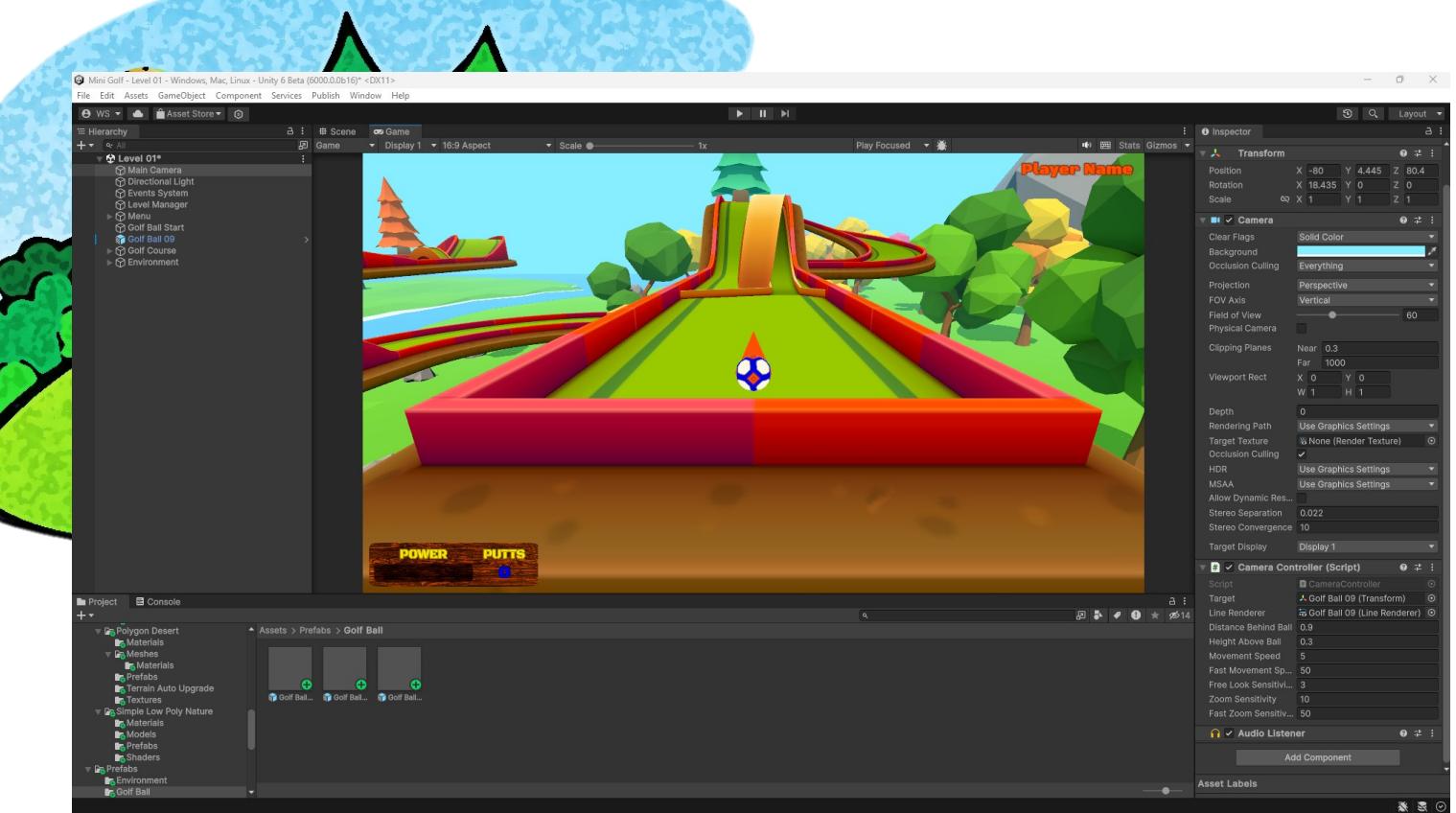
Tutaj mam jeden materiał z fizyką dla skoków piłki.



No i w samej piłce też mam ustawiony Rigidbody i Collider, żeby nie była za lekka, nie lewitowała, latała dobrze w powietrzu i wszystko działało poprawnie.



Zrobiłem jeszcze drugi obiekt, który jest dokładnie tą samą piłką, ale bez niczego. Ma właściwie tylko Mesh Renderer, ale i tak wyłączony, żeby nie było widać kompletnie piłki. Golf Ball Start ma tylko za zadanie trzymać mi pozycję startową piłki i jak przesunę gdzieś właściwą piłkę w ramach testów, to po uruchomieniu gry i tak będę zaczynał od początku toru, bo pobiera sobie pozycję z tej piłki startowej.



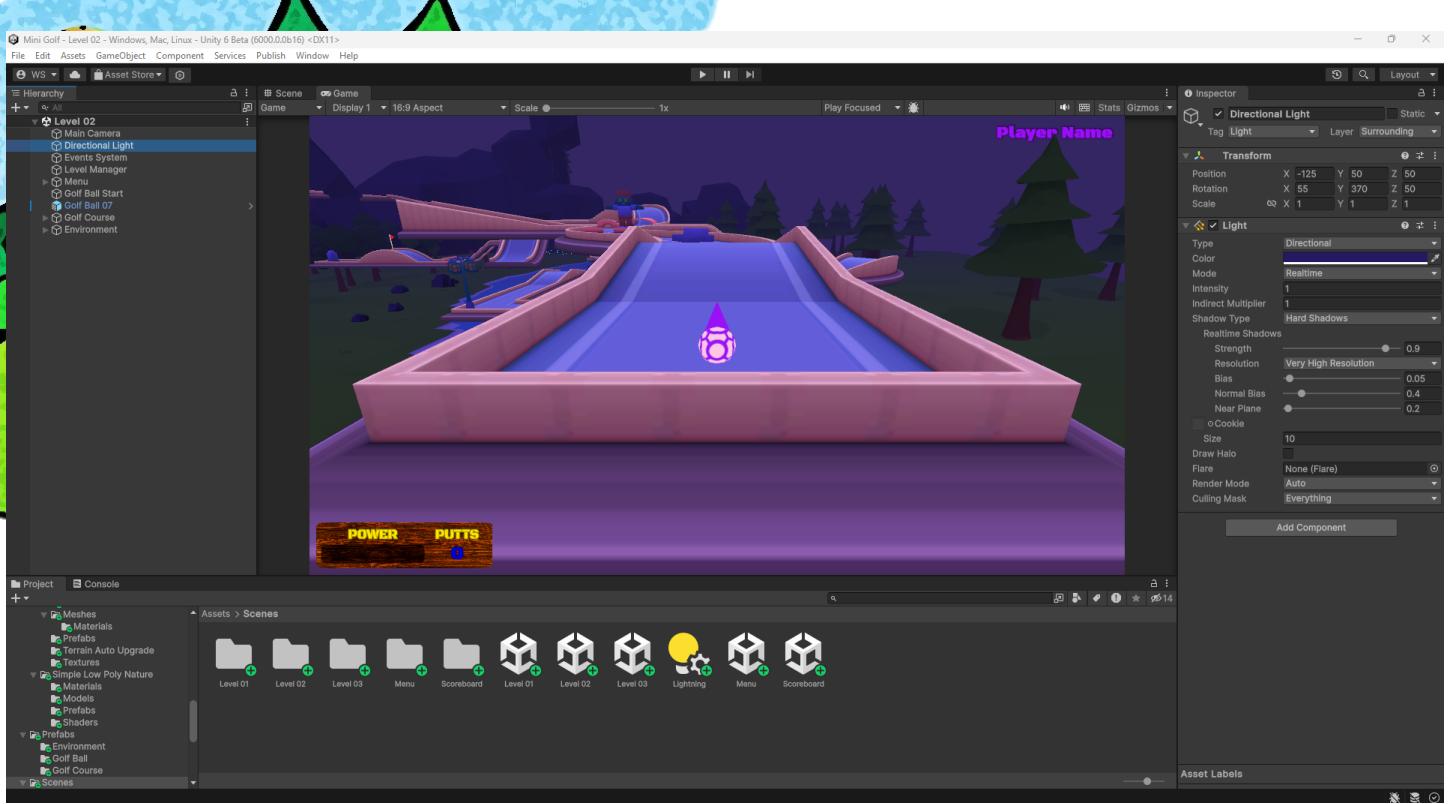
Przejdźmy teraz do kamery. Nie jest to taka zwykła kamera, bo to tzw. Third-Person Camera. Śledzi mi ona piłkę i lata za nią, gdzie by nie poleciała. Mogę ustawać sobie wysokość i odległość podążania za piłką, prędkość. Muszę tylko przypisać do niej obiekt, który ma śledzić. Kamera też ma tryb swobodny. Aktywuje się klawiszem Z. Możemy sobie latać po całej mapie, sterujemy myszką, a W i S do przodu albo do tyłu. Scroll daje nam zooma. W parametrach skryptu kamery możemy regulować sobie szybkość poruszania się po mapie, czułość myszki itd.

The screenshot shows the Unity Editor interface with the following details:

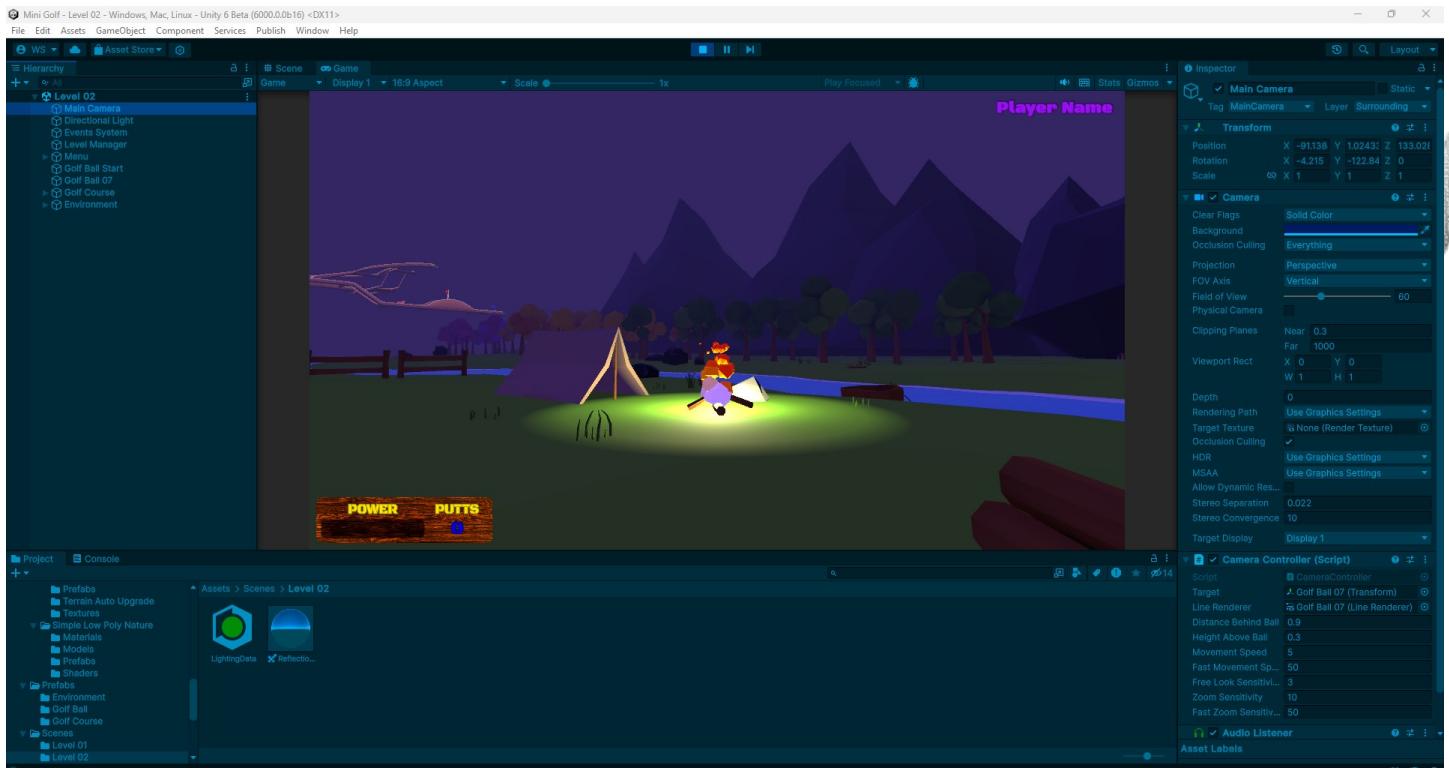
- Menu Bar:** Plik, Edytuj, Wybierz, Wyświetl, Przejdz, Uruchom, Terminal, Pomoc.
- Toolbar:** Includes icons for file operations like Open, Save, and Build.
- Sidebar:** EKSPLORATOR (Assets) tree view:
  - MINI GOLF
    - .plastic
    - .vscode
  - Assets
    - Materials
    - Models
    - Prefabs
    - Scenes
    - Scripts
      - BackgroundMusicManager.cs
      - BallController.cs
      - ButtonCursorChanger.cs
      - ButtonTextColorChanger.cs
      - CameraController.cs
      - InputCursorChanger.cs
      - LevelManager.cs
      - MenuManager.cs
      - OptionsMenu.cs
      - PauseManager.cs
      - PlayerRecord.cs
      - ScoreboardManager.cs
      - SliderCursorChanger.cs
      - SliderValueDisplay.cs
    - Settings
    - Sounds
    - TextMesh Pro
      - Ball\_Physic Material.physicMaterial
  - Packages
  - WebGL Builds
  - Assembly-CSharp.csproj
  - MiniGolf.zip
  - webgl\_sharing
- Editor Window:** Mini Golf (Assets > Scripts > CameraController.cs)

```
1 using UnityEngine;
2
3 public class CameraController : MonoBehaviour
4 {
5     public Transform target;
6     public LineRenderer lineRenderer;
7     public float distanceBehindBall = 1f;
8     public float heightAboveBall = 0.5f;
9     public float movementSpeed = 5f;
10    public float fastMovementSpeed = 50f;
11    public float freeLookSensitivity = 5f;
12    public float zoomSensitivity = 10f;
13    public float fastZoomSensitivity = 50f;
14    [HideInInspector] public bool looking = false;
15
16    // Ta metoda jest wywoływana co klatkę, aktualizuje pozycję kamery na podstawie położenia piłki.
17
18    void LateUpdate()
19    {
20        if (Time.timeScale == 0)
21        {
22            return;
23        }
24
25        if (target != null && lineRenderer != null && lineRenderer.positionCount >= 2)
26        {
27            if (Input.GetKeyDown(KeyCode.Z))
28            {
29                looking = !looking;
30            }
31
32            if (!looking)
33            {
34                Vector3 direction = lineRenderer.GetPosition(1) - lineRenderer.GetPosition(0);
35                direction.Normalize();
36
37                Vector3 targetPosition = target.position - direction * distanceBehindBall;
38                targetPosition.y += heightAboveBall;
```
- Bottom Status Bar:** Wiersz 1, kolumna 1 Space: 4 UTF-8 CRLF C# ⓘ ⓘ Prettier

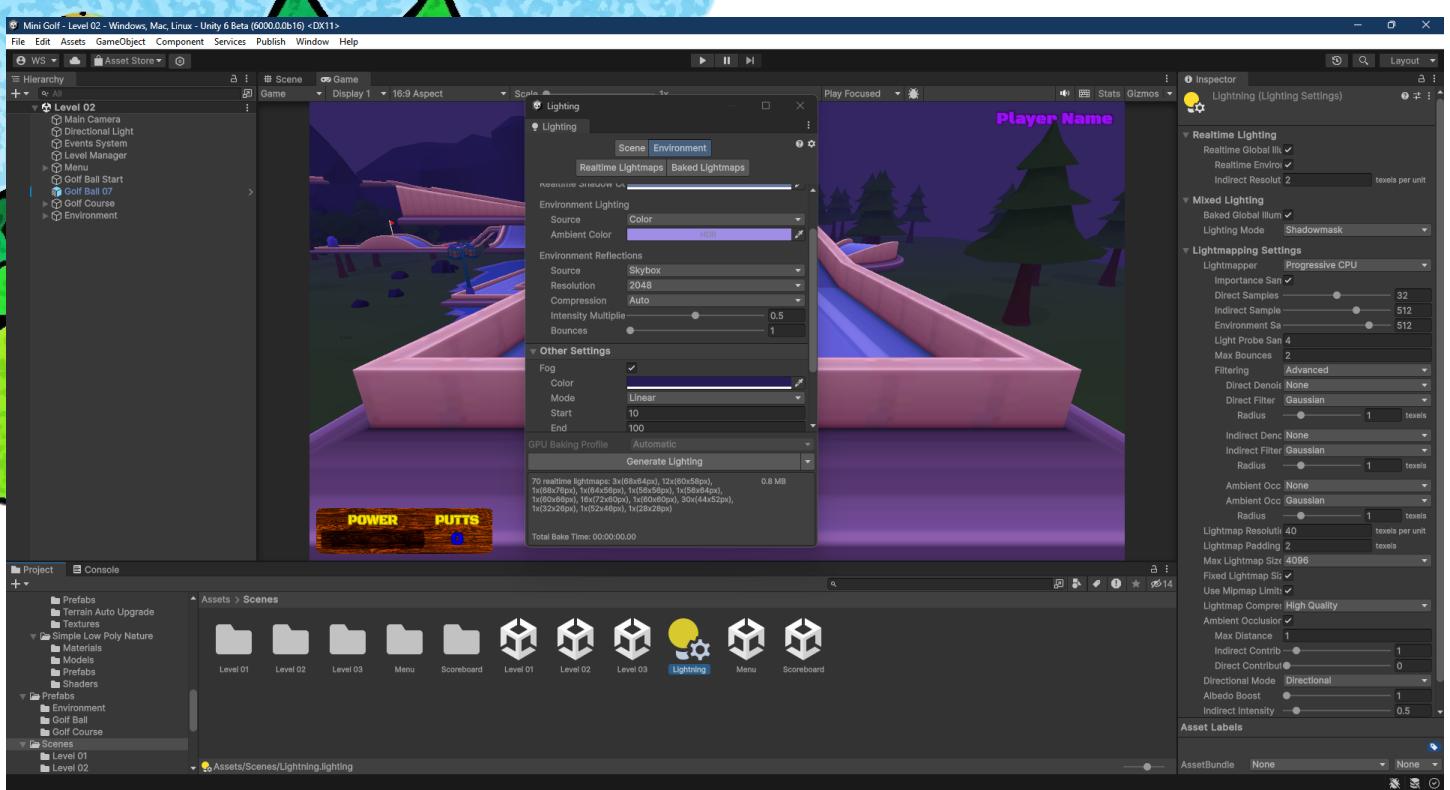
Odpowiedzialny jest za nią taki osobliwy skrypt CameraController.cs.



Level 02 to właściwie z zasadą działania to samo. Mamy różnice w oświetleniu i torze. Zostawiłem to samo otoczenie, tylko je ściemniłem, teraz ognisko wygląda jeszcze bardziej urokliwie.



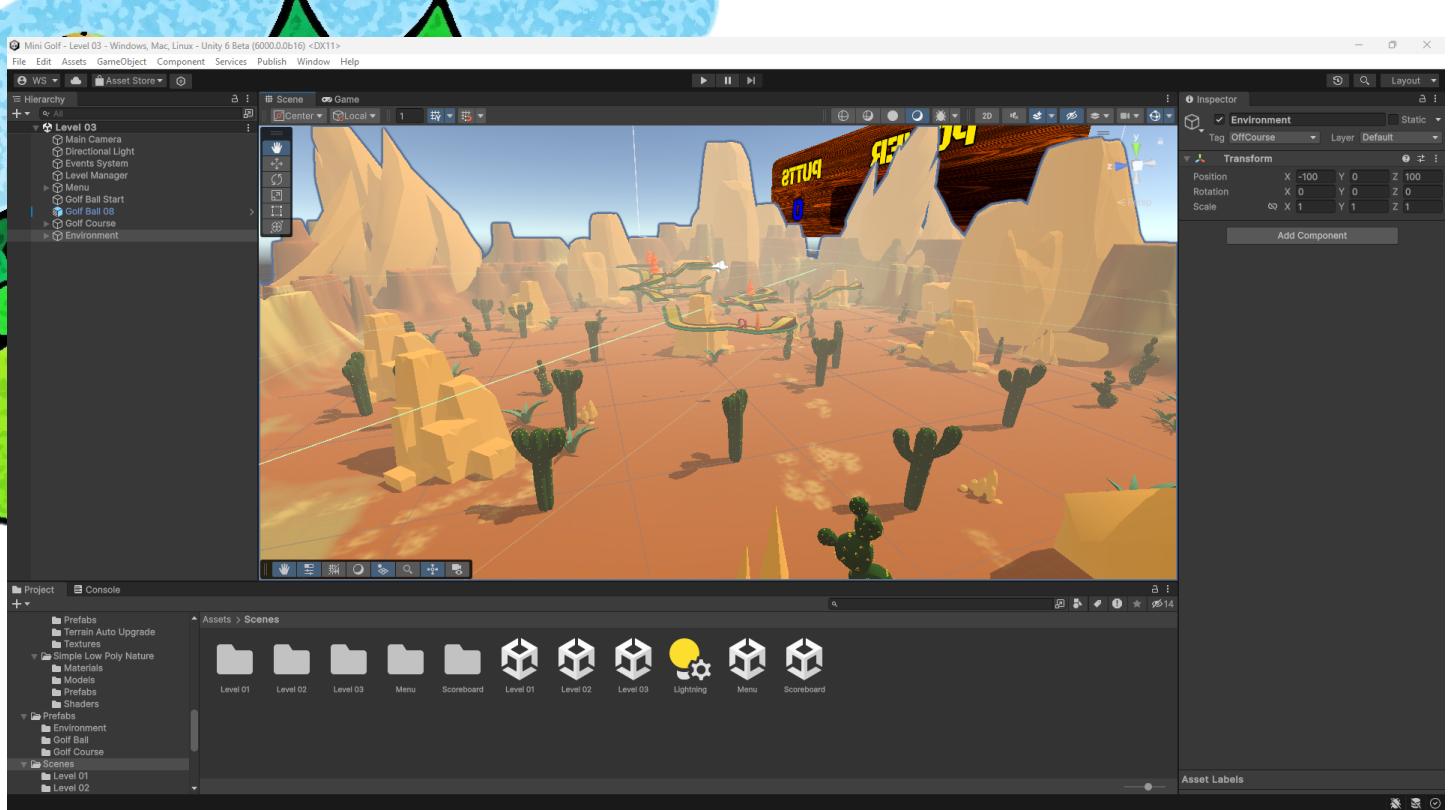
Dokładnie tak wygląda to podczas rozgrywki. Pokazałem też, że jest ReflectionProbe i LightningData od renderowania oświetlenia.



Tak wyglądają natomiast ustawienia oświetlenia w renderingu. Też dodałem mgłę, ale delikatniejszą, jest ta sama zasada co na poziomie 01, tylko kolory inne.



Level 03 odnośnie oświetlenia to te same zmiany co na Levelu 02, czyli zmienione kolory w Main Camera, Directional Light oraz w Lightning Renderer. Tym razem postawiłem na pustynię. Obiekty wziąłem z assetów i ustawiłem je w fajny sposób na mapie. Wszystkie obiekty mają dodane collidery rzecz jasna i są dodane do prefabów.



Tak wygląda mapa. Oczywiście zrobiłem też inny tor, tak jak to miało miejsce na Levelu 01 i 02. Zmieniłem też kolorki wszystkich napisów.



Na koniec postawiłem na dodanie dźwięków do gry. Ze strony z darmowymi plikami dźwiękowymi [freesound.org](http://freesound.org) pobrałem wszystkie potrzebne dźwięki. Zrobiłem nastrojową muzykę w tle, która się zapętla i wybudza się po włączeniu gry automatycznie. Jest ona obecna w całej grze, niezależnie od pauzy itd. Obsługuje ją skrypt BackgroundMusicManager.cs, po to, żeby nie zniknęła po włączeniu kolejnych scen.



Pliki dźwiękowe wcześniej musiałem podocinać i odpowiednio obrobić w programie [WavePad](#), trochę wyciszyć, żeby wszystko było bardziej stonowane. Poza tym ustawiłem je odpowiednio w Unity, żeby nie straciły na jakości.

```

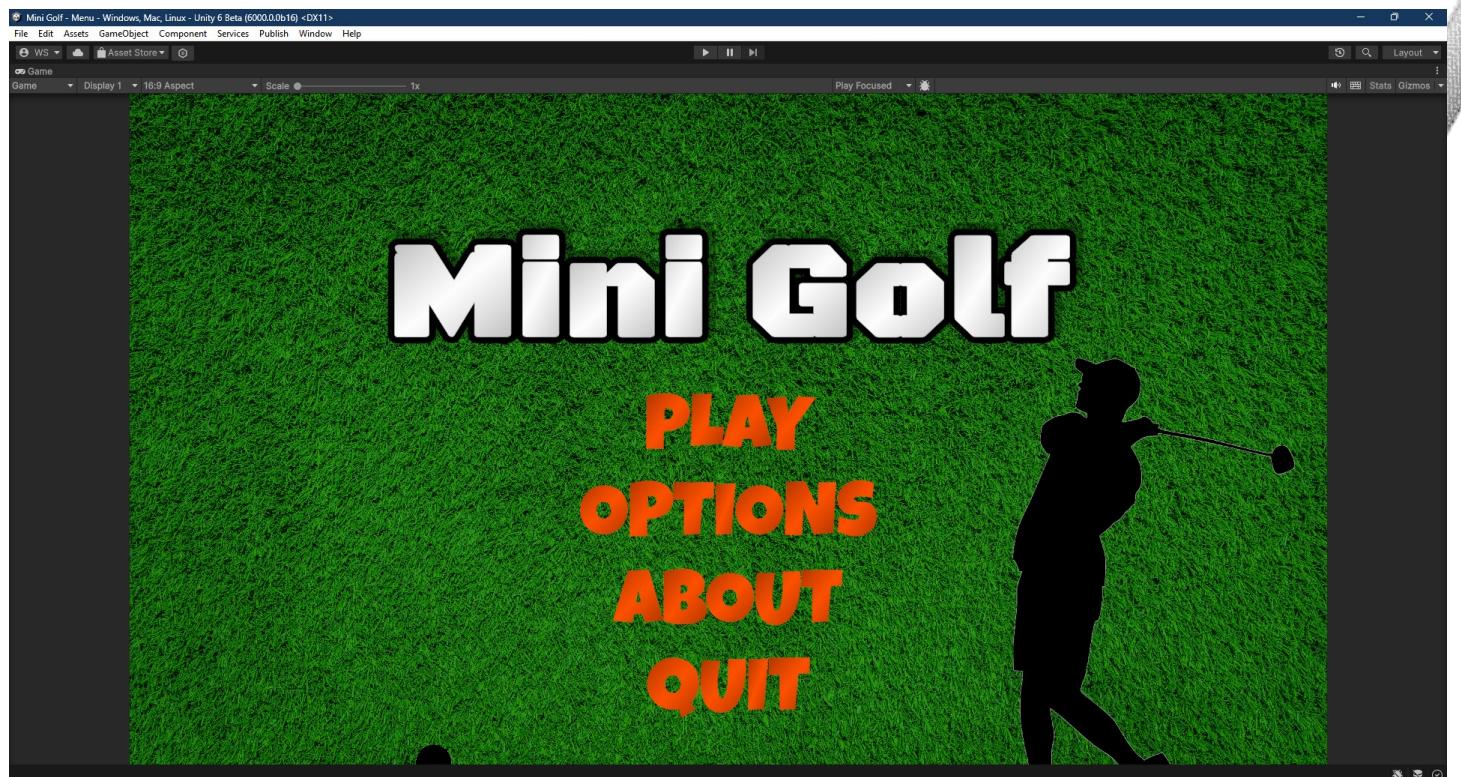
    EKSPLORATOR
    MINI GOLF
    > plastic
    > vscode
    > Assets
        > Materials
        > Models
        > Prefabs
        > Scenes
            Scripts
                BackgroundMusicManager.cs
                BallController.cs
                ButtonCursorChanger.cs
                ButtonTextColorChanger.cs
                CameraController.cs
                InputCursorChanger.cs
                LevelManager.cs
                MenuManager.cs
                OptionsManager.cs
                PauseManager.cs
                PlayerRecord.cs
                ScoreboardManager.cs
                SliderCursorChanger.cs
                SliderValueDisplay.cs
            > Settings
            > Sounds
            > TextMeshPro Pro
                Ball_Physic Material.physMaterial
            > Packages
            > WebGL Builds
                Assembly-CSharp.csproj
                connectwebgl.zip
                Mini.Golf.sln
                webgl_sharing
    ... BackgroundMusicManager.cs ...
    Assets > Scripts > BackgroundMusicManager.cs ...
    1  using UnityEngine;
    2
    3  public class BackgroundMusicManager : MonoBehaviour
    4  {
    5      // Odwołanie do jednego egzemplarza klasy
    6      private static BackgroundMusicManager instance;
    7
    8      // Metoda wywoływana przy tworzeniu obiektu:
    9
    10     Odwołanie: 0
    11     private void Awake()
    12     {
    13         if (instance != null)
    14         {
    15             Destroy(gameObject);
    16             return;
    17         }
    18
    19         instance = this;
    20         DontDestroyOnLoad(gameObject);
    21     }
    22
    23     // Metoda wywoływana przy uruchomieniu gry:
    24
    25     Odwołanie: 0
    26     private void Start()
    27     {
    28         if (!GetComponent< AudioSource >().isPlaying)
    29         {
    30             GetComponent< AudioSource >().Play();
    31         }
    32     }
    33
    34     // Metoda wywoływana przyniszczeniu obiektu:
    35
    36     Odwołanie: 0
    37     private void OnDestroy()
    38     {
    39         if (instance == this)
    40         {
    41             instance = null;
    42         }
    43     }
    44 }

```

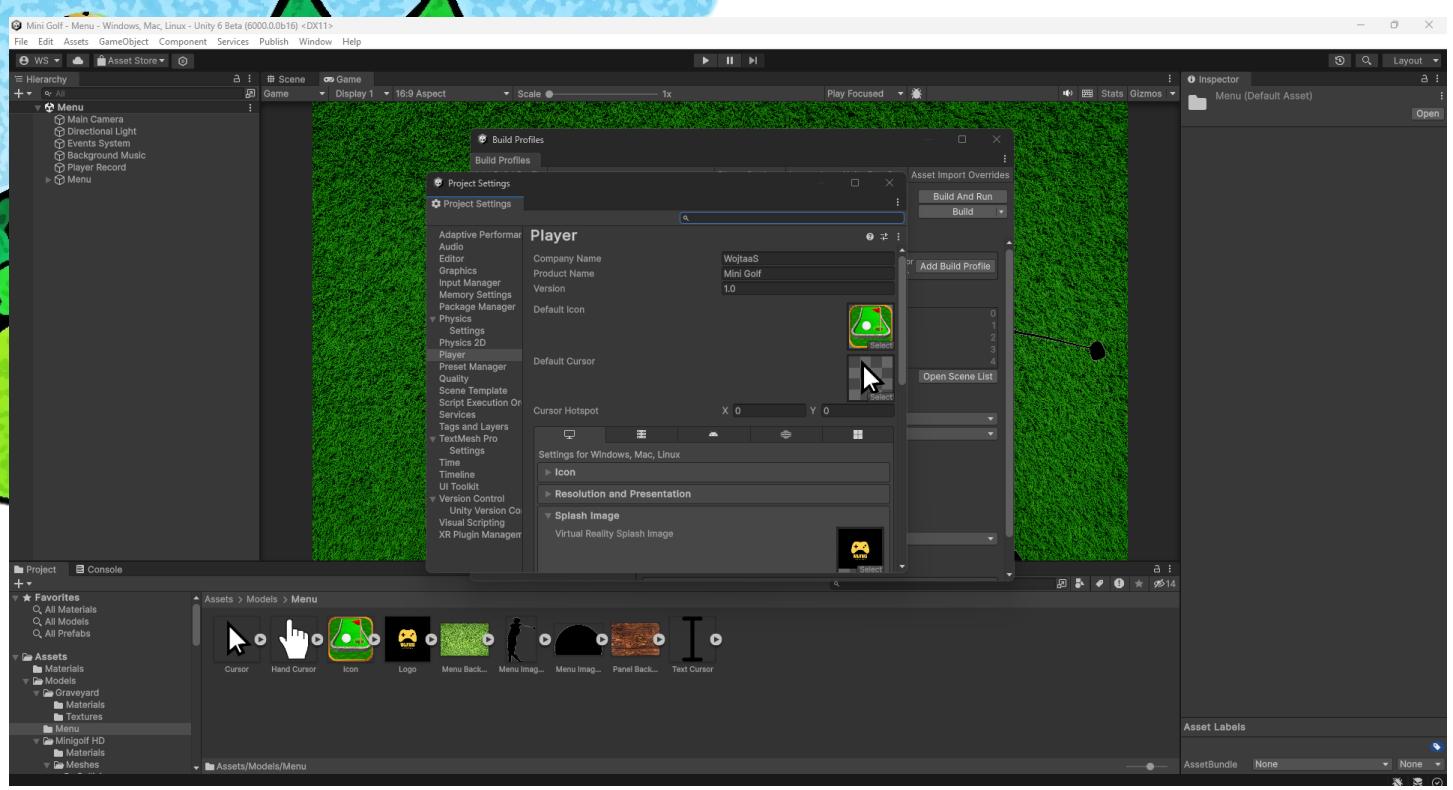
A to jest skrypt odpowiedzialny za utrzymywanie i poprawne działanie muzyki w tle.



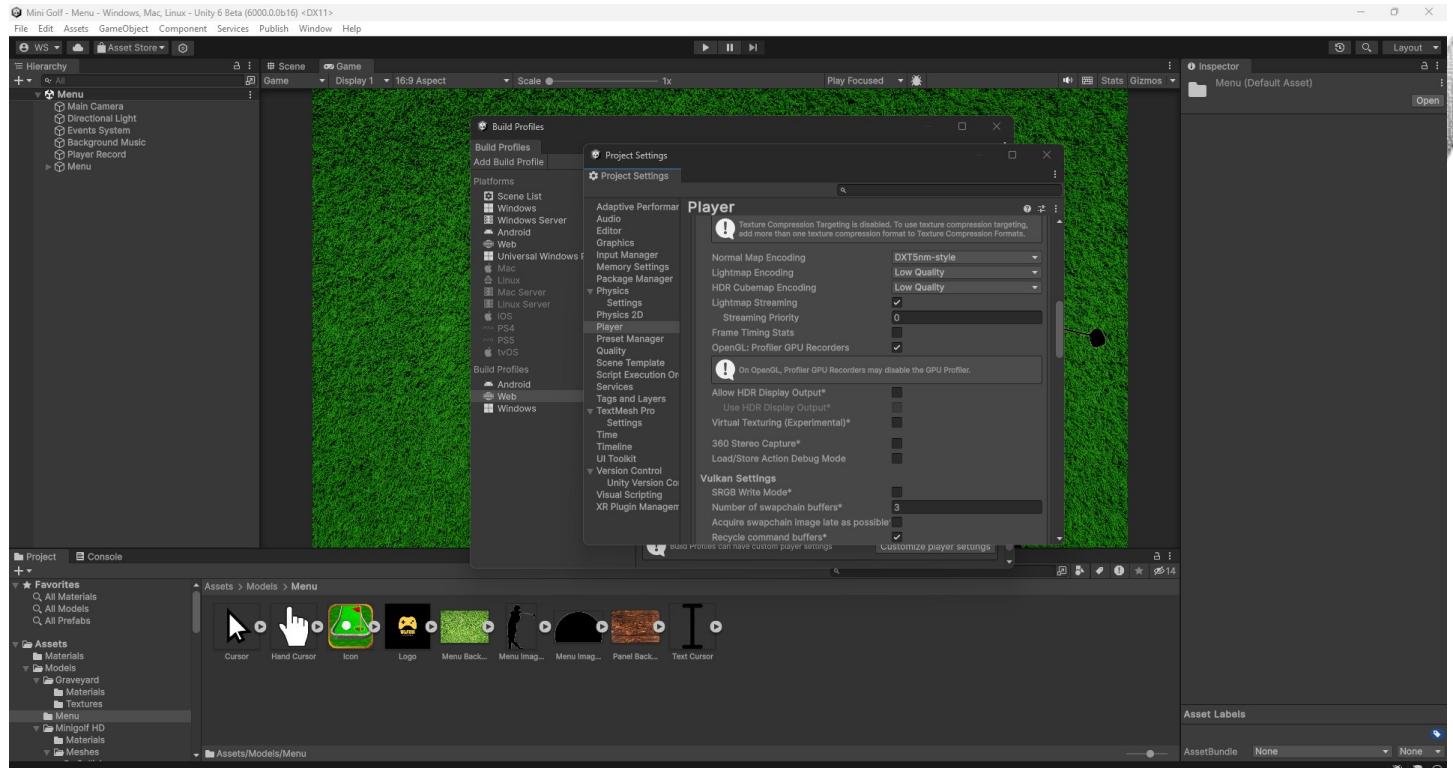
Oczywiście dźwięki musiały znaleźć się również w piłce, ponieważ dodałem tam dźwięk, kiedy uderzamy piłkę (Puszczamy spację), ale także kiedy piłka wpada do dołka. Poza tym pozostałe dźwięki, które są dodane to: kiedy piłka wypadnie poza tor, kiedy mamy wstęp Level Canvas oraz po zaliczeniu dołka (Hole Canvas). Te dźwięki są zaimplementowane w skrypcie BallController.cs, żeby wywoływały się w odpowiednich momentach rozgrywki. Oczywiście jest to zaimplementowane dla wszystkich piłek w mojej grze przy dodaniu komponentu Audio Source, bez którego nie działałoby to.



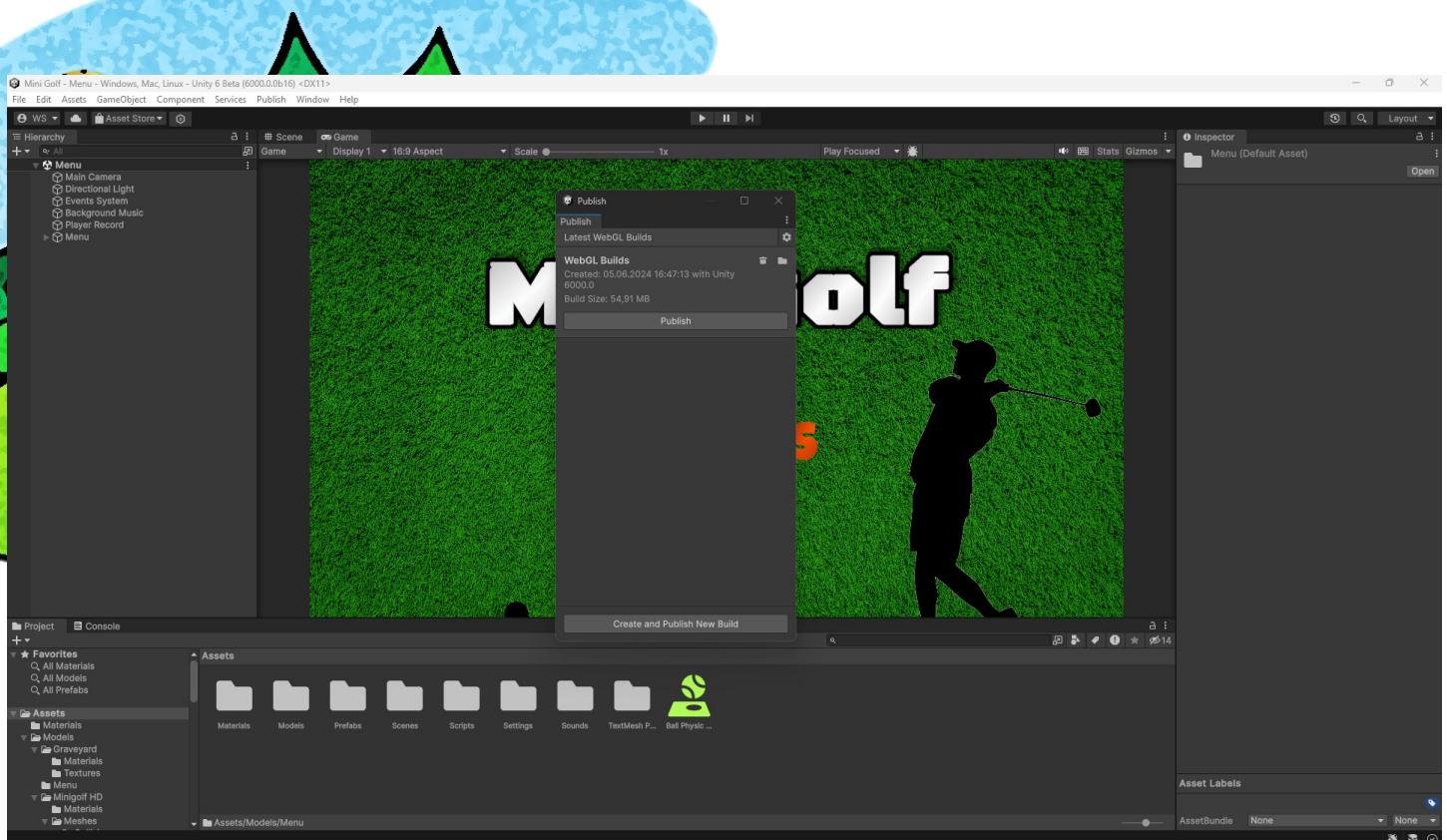
I na koniec po wszystkich testach wygląda to następująco. Oczywiście było też wiele drobnych poprawek, np. jeśli pauzujemy grę to nie możemy używać swobodnej kamery itp.



Na koniec po wszystkich działaniach przystąpiłem do tworzenia swojego buildu. Zacząłem konfigurować wszystkie parametry już na finalnym etapie komplikacji gry. Dodałem swój domyślny kursor pointer. Dodałem również customową ikonę gry i moje własne logo, jako producent gry. Ustawiłem sobie wszystko, łącznie z ustawieniami graficznymi, kompresowaniem itd. Zrobiłem to dla Windowsa, Androida i WebGL.



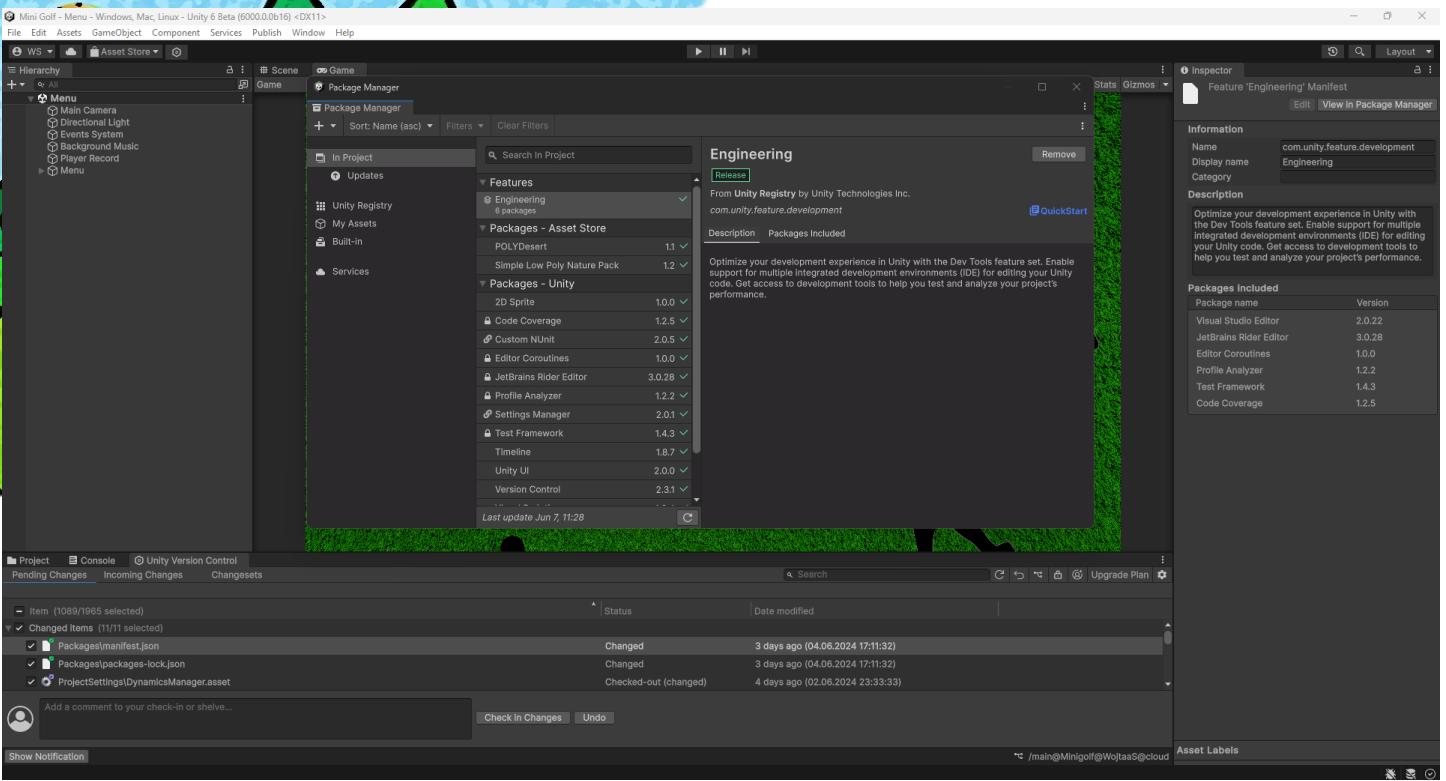
Dodałem swoje własne profile do buildów gry. Poza tym grę udostępniłem na stronie play.unity.com poprzez WebGL do rozgrywek online, bez konieczności pobierania gry na własny PC.



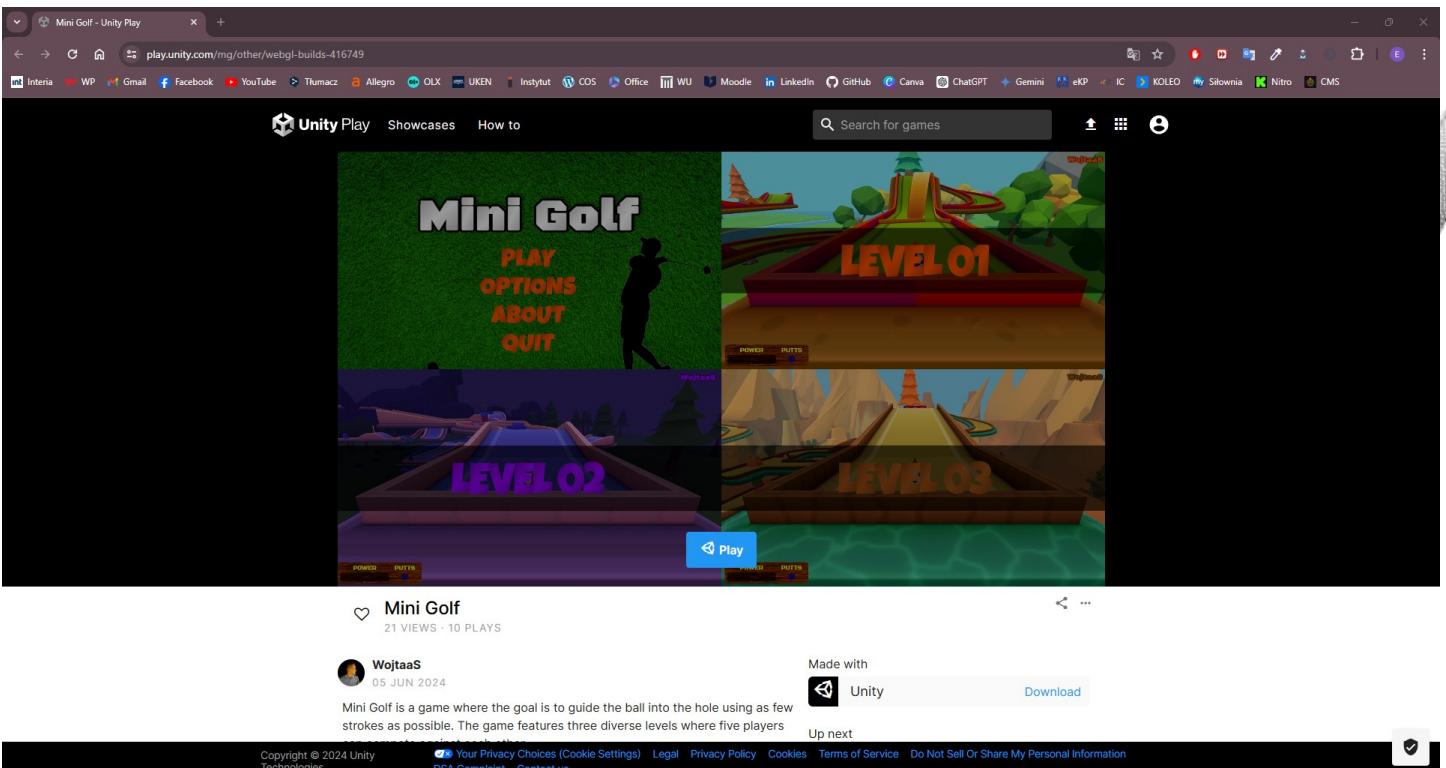
Publikacja WebGL. Maksymalny rozmiar do udostępnienia to 200MB. Początkowo przekraczałem tą wartość, ale zmniejszyłem rozmiar tekstur assetów z 16384 na 2048 i gra finalnie waży 55MB.

A screenshot of the Unity Asset Store website at assetstore.unity.com/account/assets. The page title is 'My Assets'. It shows two items: 'POLYDesert' by RUNEMARK STUDIO (1.4 MB, purchase date May 28, 2024) and 'Simple Low Poly Nature ...' by NEUTRONCAT (1.2 MB, purchase date May 7, 2024). Both have 'Open in Unity' buttons. To the right are filters for 'Status' (Unlabeled, Hidden Assets, Deprecated), 'Labels(0)', 'Categories' (3D), and 'Publishers'. The top of the page features a navigation bar with links like Home, My Assets, and a search bar.

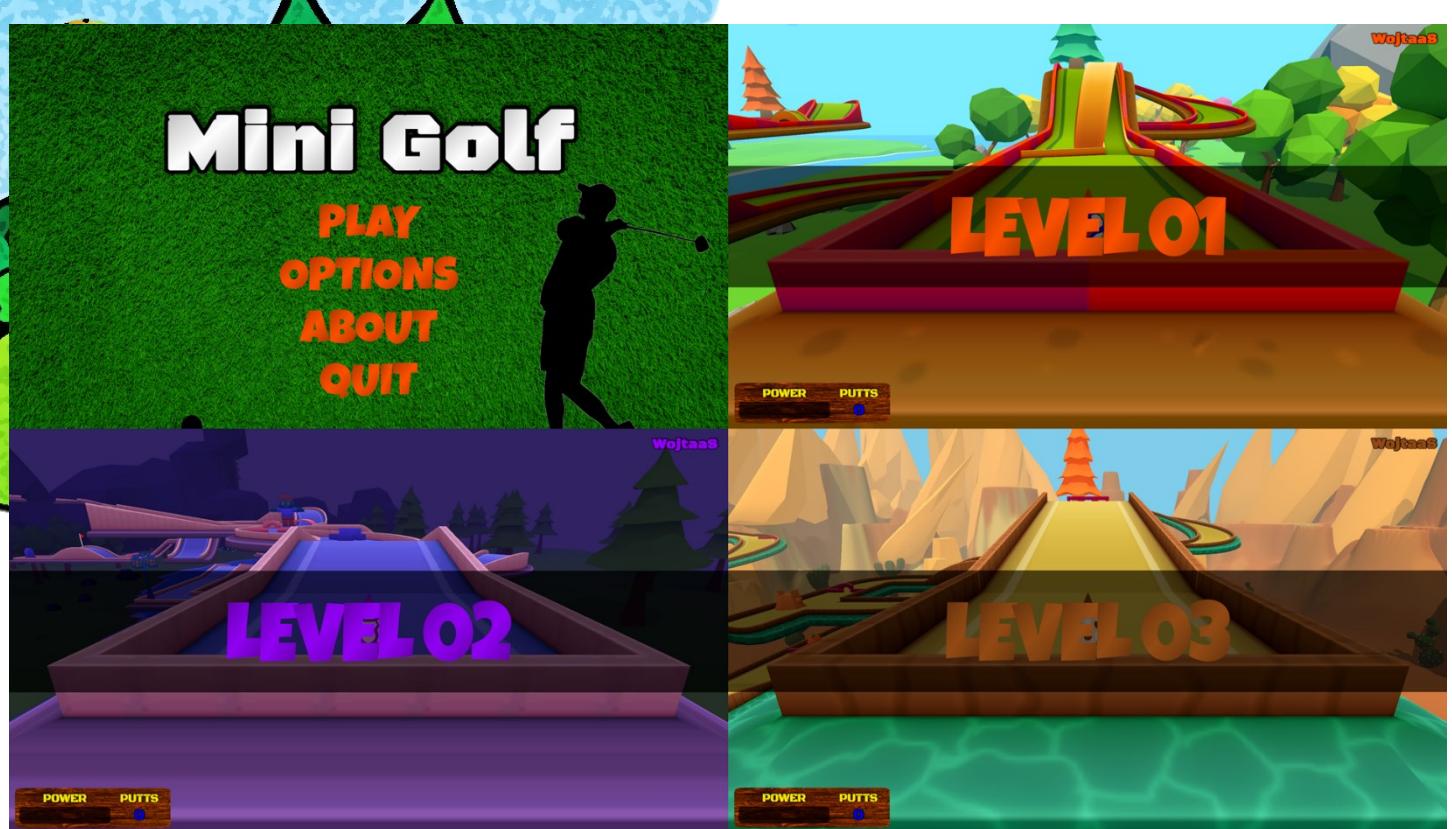
Moje assety wykorzystane do tworzenia map.



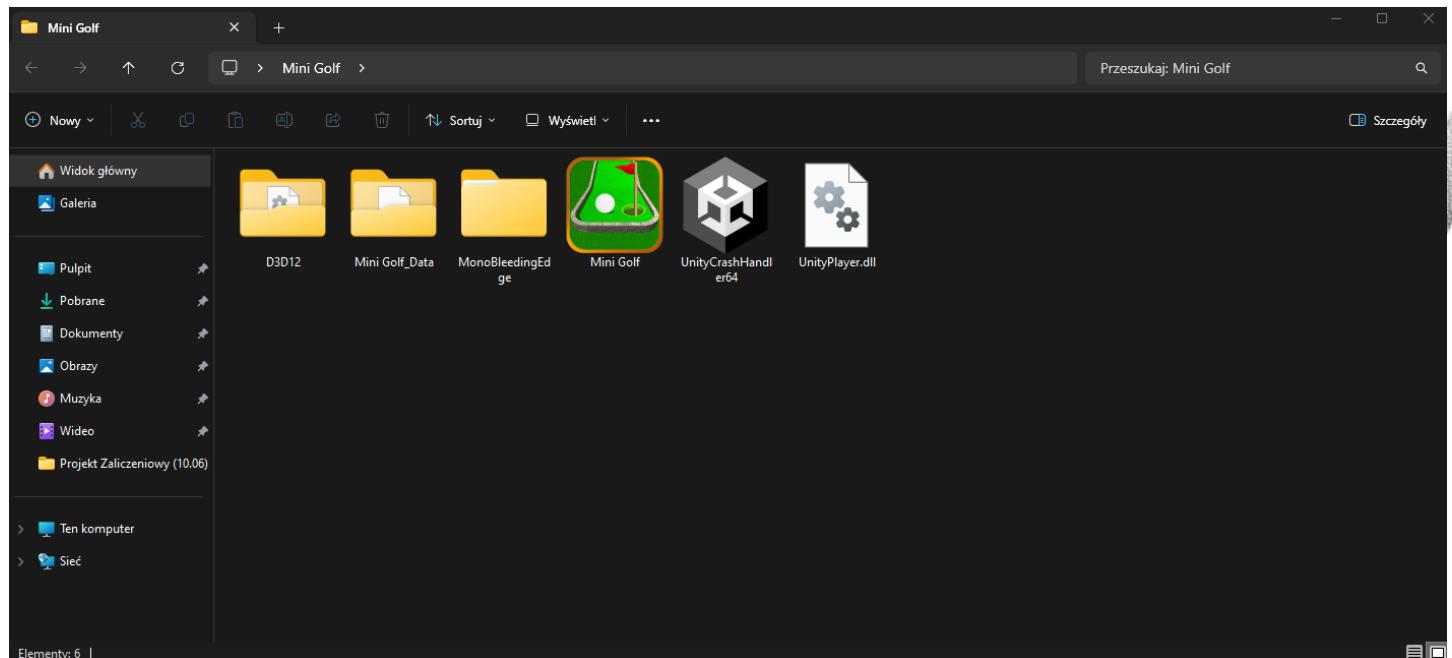
A tutaj wszystkie moje paczki zaimportowane w tworzeniu tego projektu, łącznie z assetami. Jak widać gra przechodzi również kontrolę błędów bez żadnych errorów.



A tutaj gra na stronie Unity Play udostępniona poprzez WebGL z możliwością rozgrywki online.



Miniatura do gry.



Folder z grą.



# Mini Golf

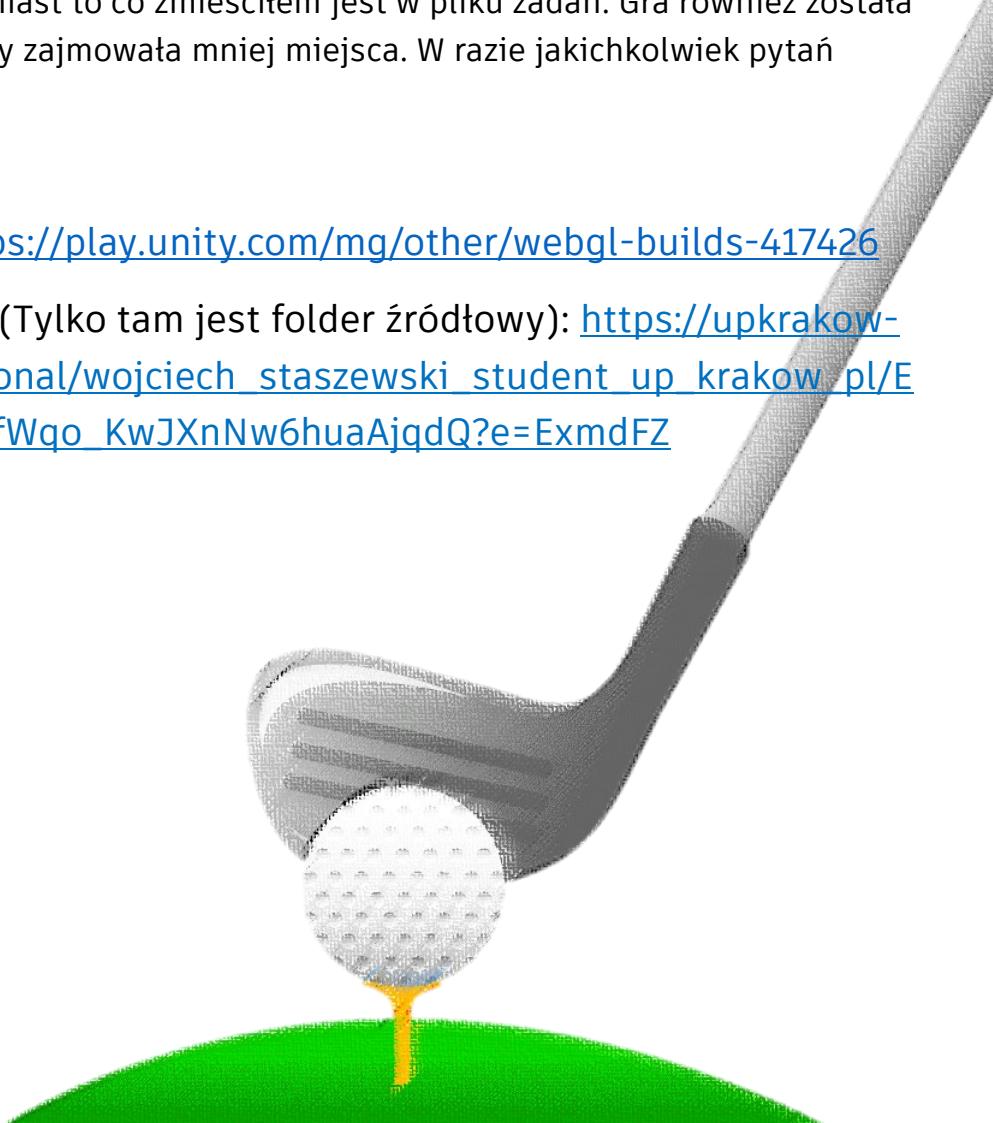
**PLAY  
OPTIONS  
ABOUT  
QUIT**



I gotowa włączona gra. Finalnie pełna wersja gry waży ponad 750MB z uwagi na tekstury. Nagrałem film prezentujący. Gra jest skompilowana w wersji na Androida i Windowsa. Poza tym spakowałem folder źródłowy do archiwum, ale nie podzieliłem go na części bo waży ponad 10GB, nie zmieściłby się do zadania tak czy tak (Ograniczenie do 500MB i 10 plików). Wrzuciłem wszystko na Dysk OneDrive, natomiast to co zmieściłem jest w pliku zadań. Gra również została skompresowana do archiwum, żeby zajmowała mniej miejsca. W razie jakichkolwiek pytań zapraszam do kontaktu.

Link do wersji online gry: <https://play.unity.com/mg/other/webgl-builds-417426>

Link do dysku OneDrive z grą (Tylko tam jest folder źródłowy): [https://upkrakow-my.sharepoint.com/:f/g/personal/wojciech\\_staszewski\\_student\\_up\\_krakow\\_pl/EggHKdDrdjtAoMaXeusi\\_dkBhfWqo\\_KwJXnNw6huaAjqdQ?e=ExmdFZ](https://upkrakow-my.sharepoint.com/:f/g/personal/wojciech_staszewski_student_up_krakow_pl/EggHKdDrdjtAoMaXeusi_dkBhfWqo_KwJXnNw6huaAjqdQ?e=ExmdFZ)



„KONIEC”

3D Mini

GOLF

INSPIRACJA:

[HTTPS://STORE.STEAMPOWERED.COM/APP/43124](https://store.steampowered.com/app/43124)

O/GOLF\_WITH\_YOUR\_FRIENDS/?L=POLISH