Programowanie urządzeń mobilnych

Projekt:
Gra mobilna - warcaby

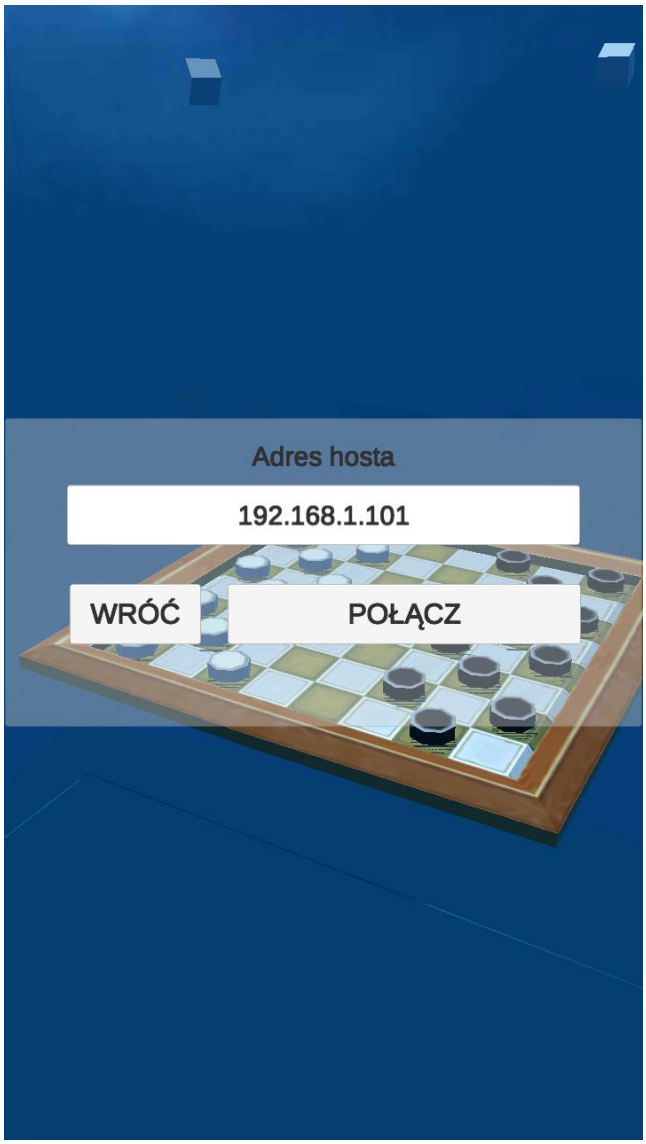
Wykonał:
Bartosz Jednacz

Opis Projektu

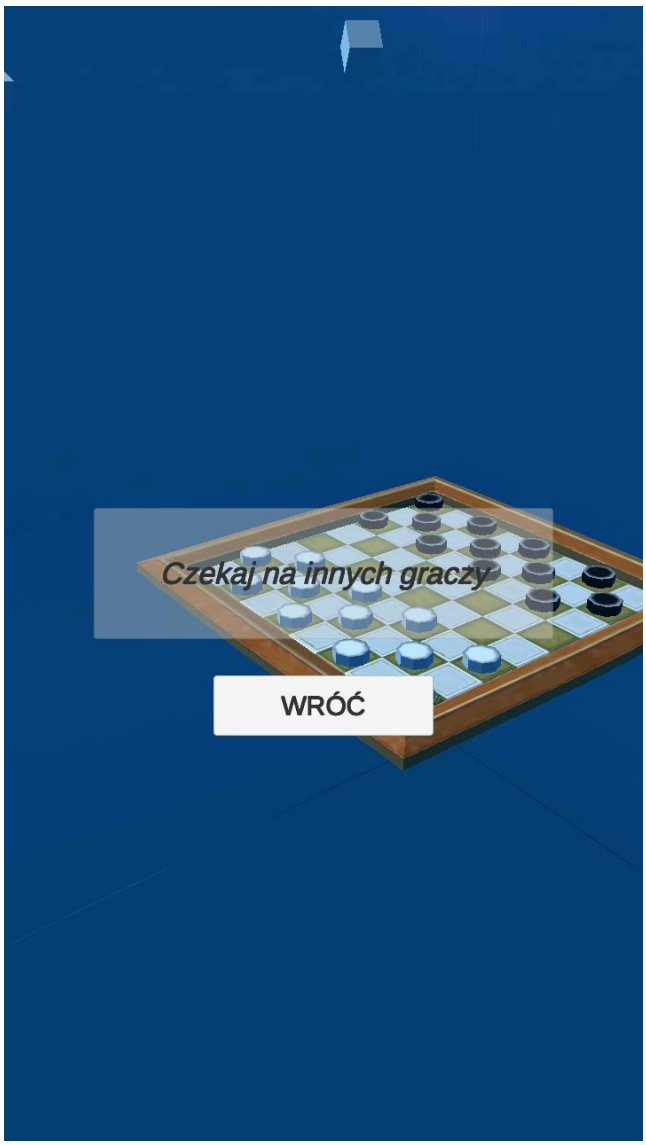
Jest to aplikacja gry mobilnej - warcaby, każdemu dobrze znana gra planszowa w wersji na smartfon. Są to dokładniej - Czekersy (są to warcaby diagonalne) - pionki nie biją do tyłu, królowa porusza się o jedno pole.
Mamy kilka opcji grania w warcaby: możemy grać na jednym urządzeniu lub lokalnie z innym użytkownikiem, identyfikatorem w tym przypadku będzie nazwa użytkownika jaką podaliśmy.

Tak wygląda menu główne aplikacji

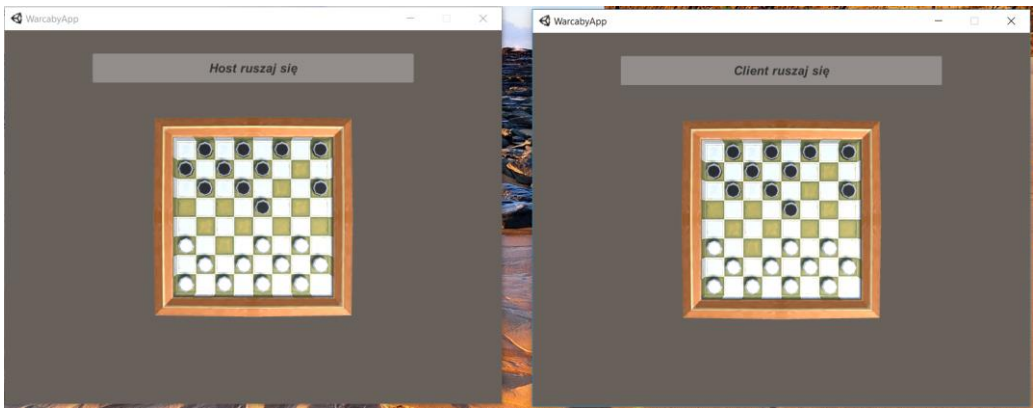
Po kliknięciu w przycisk połącz, wyświetli nam takie okno „konfiguracji” sieci. Tutaj podajemy adres IP urządzenia, na którym jest uruchomiony Host - Serwer. Nie wpisując żadnych danych połączy nas w Hostem lokalnym. Przycisk wróć - cofnie nas do menu głównego.



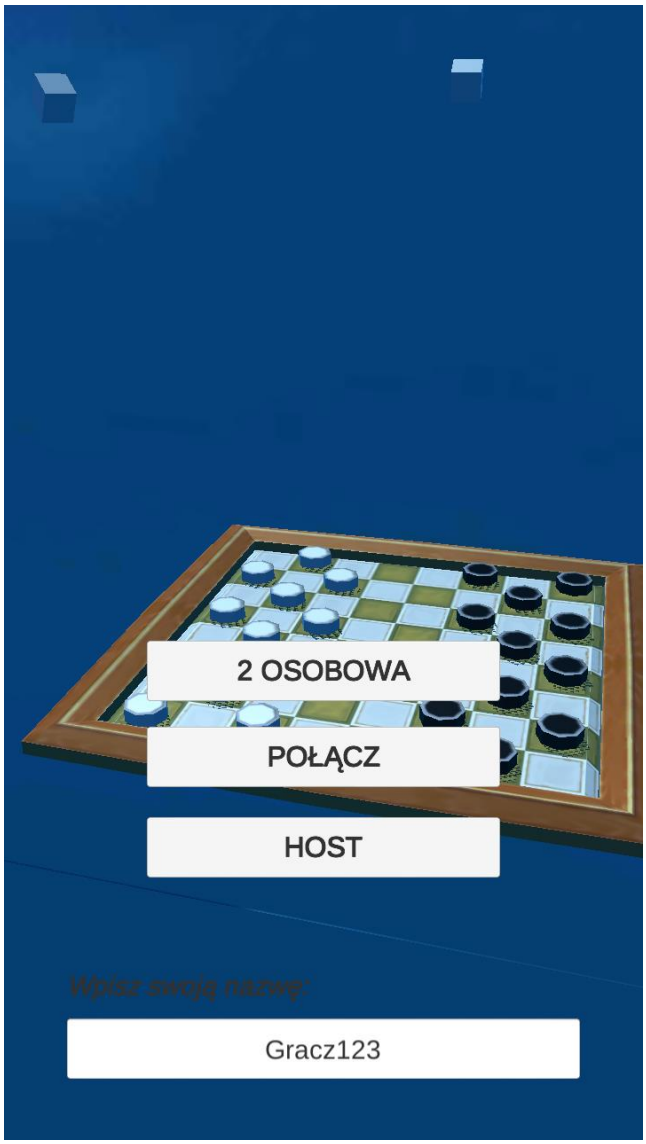
Gdy jednak wybierzemy z głównego menu - HOST. Wyświetli się to o to takie okno. Informuje nas o tym, że my stajemy się „serwerem” dla innego gracza, który ten jak we wcześniejszym obrazie wybierze przycisk - POŁĄCZ, a następnie ustawi adres tego serwera.



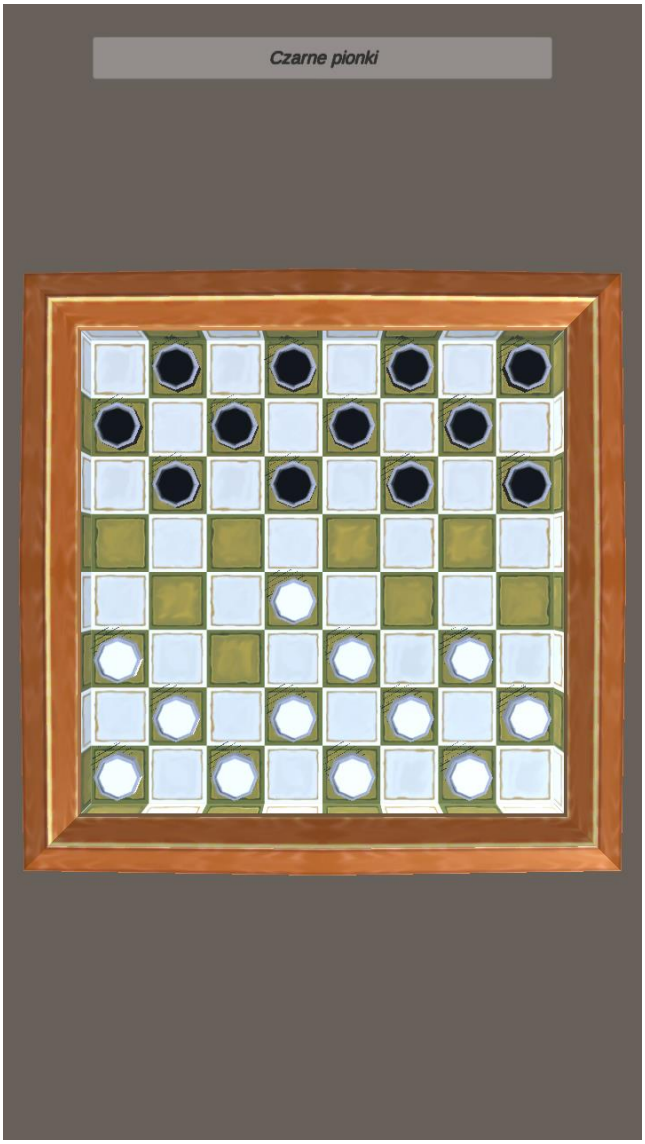
Tak wygląda gra lokalnie



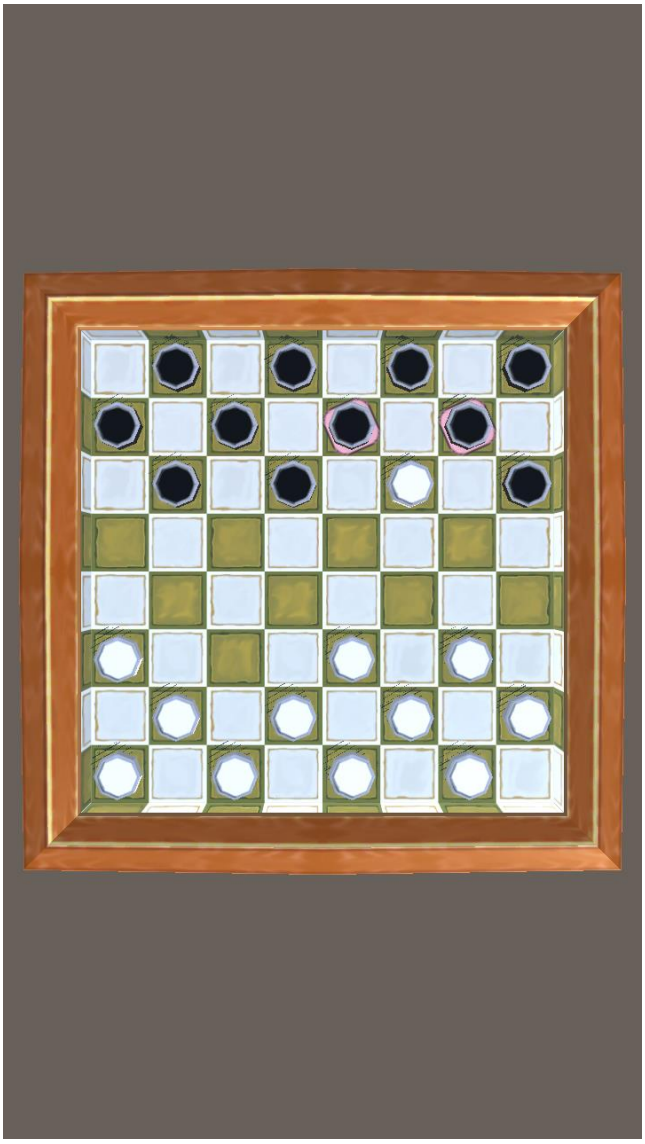
Możemy oczywiście wpisać nazwę naszego gracza.

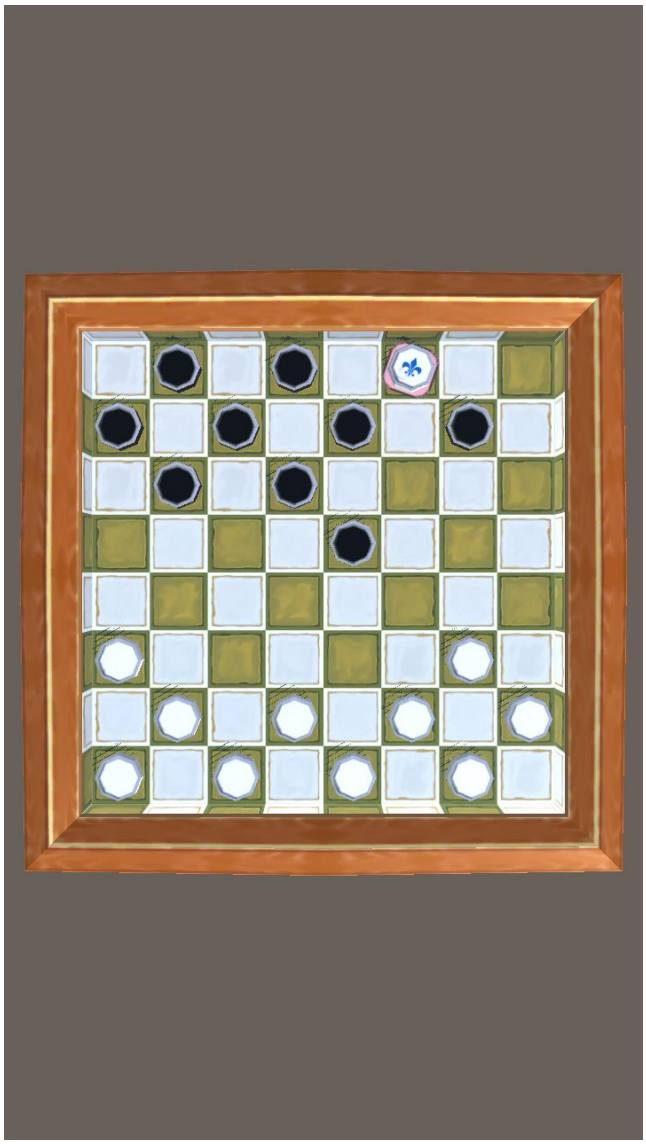


Jednak gdy chcemy pograć w trybie HOTSEAT - 2 osobowy tryb gry na jednym urządzeniu wybieramy 1 opcję z menu głównego.
Wyświetli się nam od razu plansza do gry i zostanie wyświetlony odpowiedni komunikat - w tym przypadku, informują nas o tym, czyj jest teraz ruch.



Jeśli konfiguracja pionków wymaga „zbicia” przeciwnika - zostanie podświetlony pionek, który musi ma „bicie”. Jest także królowa.

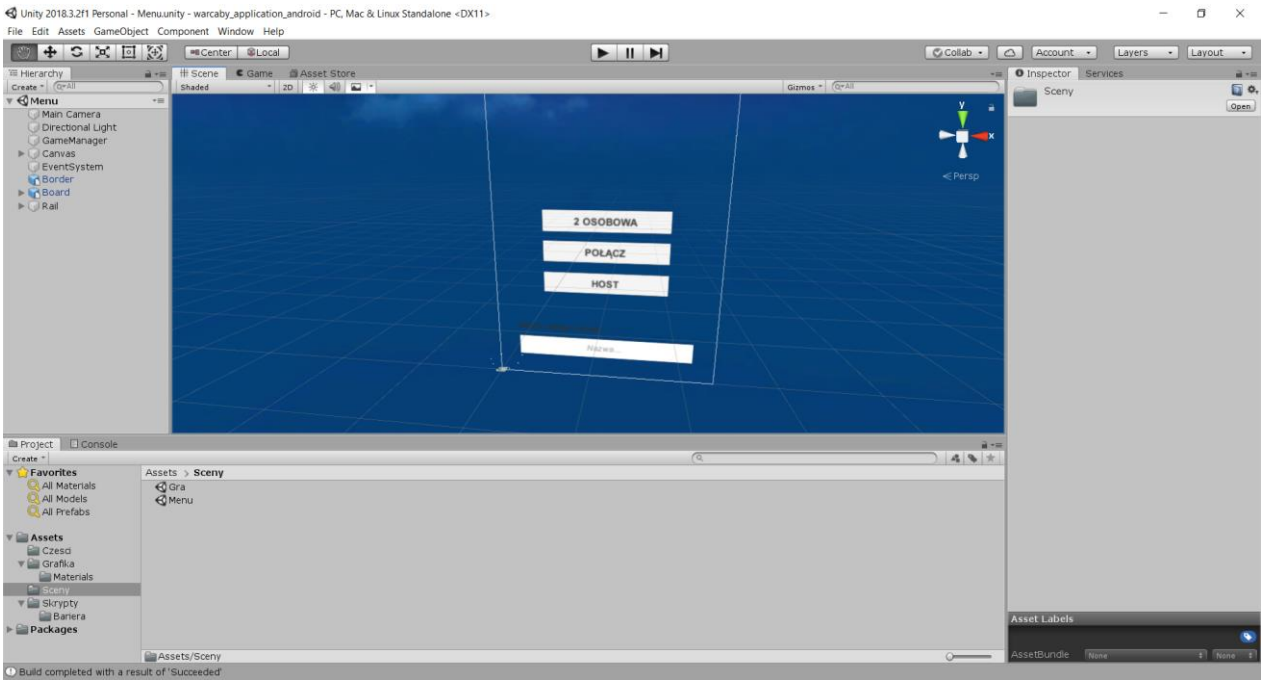


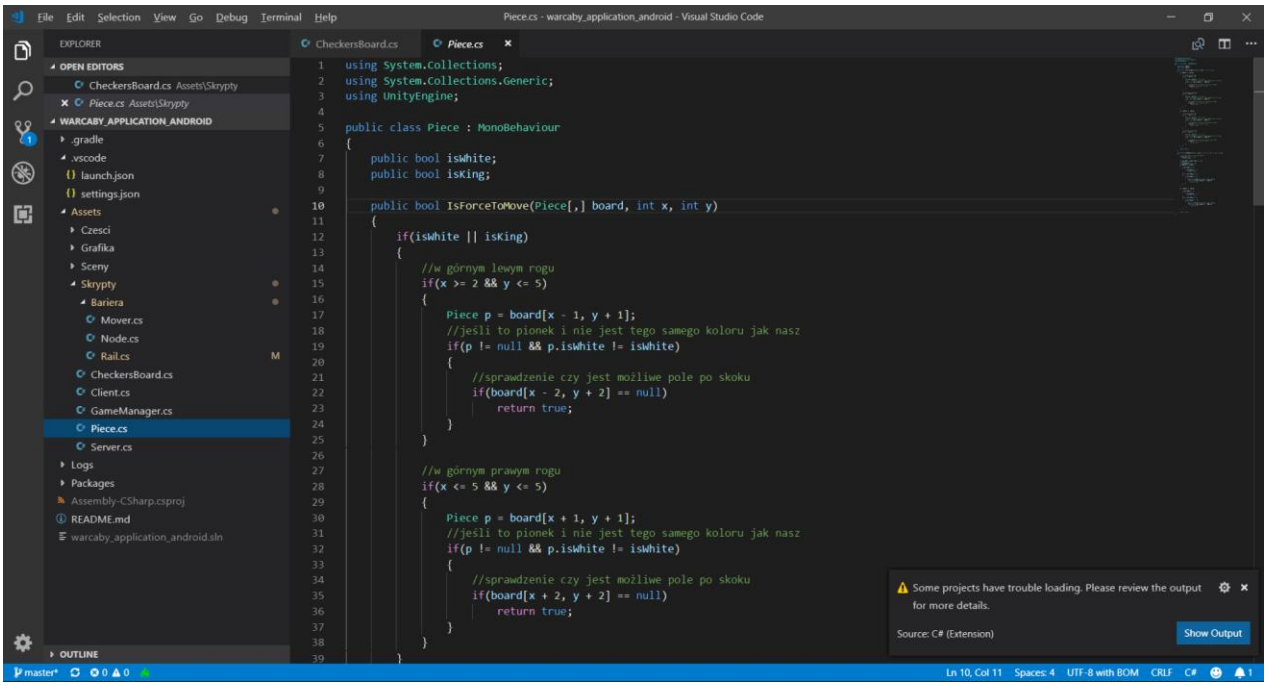


Opis kodu źródłowego

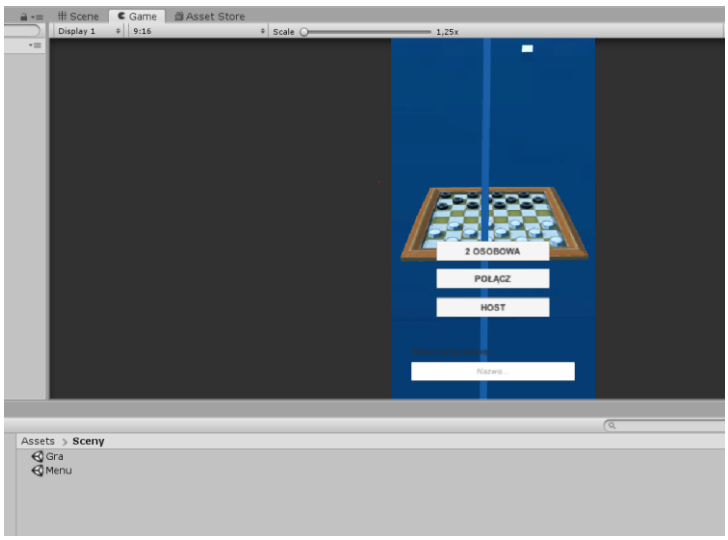
Użyte technologie:

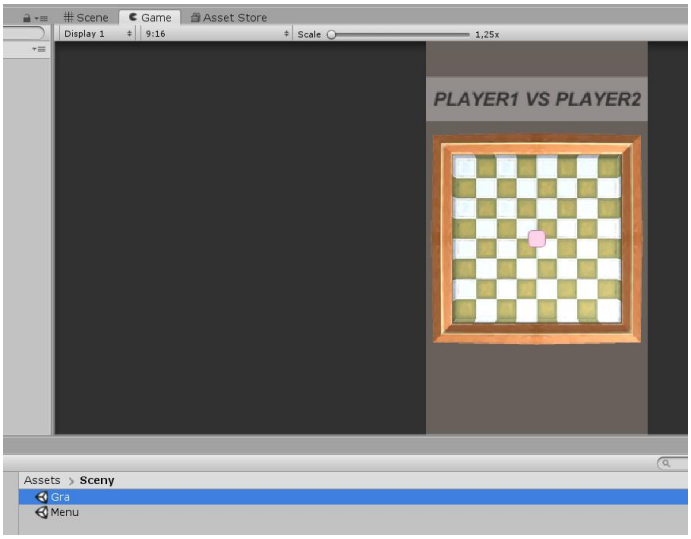
Projekt został stworzony w środowisku do tworzenia 3D oraz 2D gier komputerowych lub innych materiałów interaktywnych. Został tu stworzony model planszy, pionków, działanie obiektów, działanie „kamer”, działanie skryptów oraz działanie całej gry. Do pisania skryptów w C# korzystałem ze środowiska programistycznego MIcrosoft Visual Studio Code.



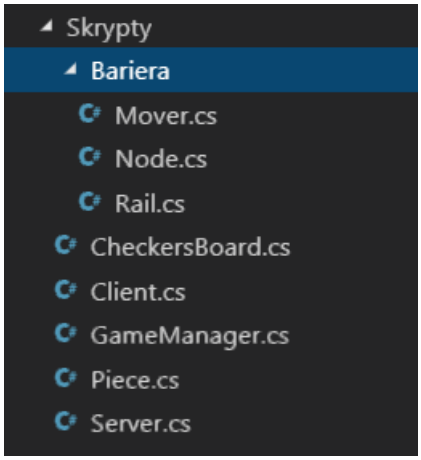


Projekt opiera się na 2 scenach: scena menu oraz scena gry (plansza).

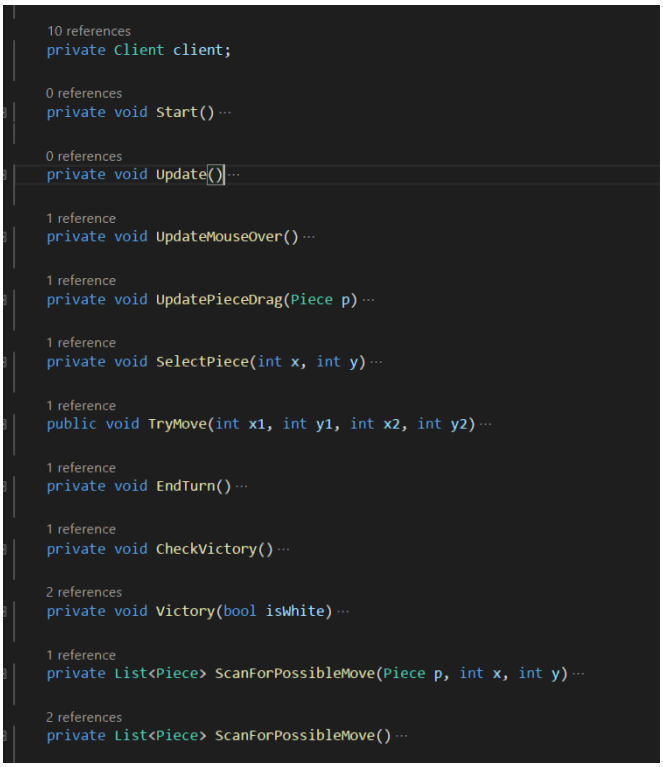




Działanie aplikacji funkcjonuje, dzięki zaimplementowanych skryptów w języku programowania C# .W podfolderze Bariera - tu są skrypty od działania kamer i ich ruchu (w menu głównym).

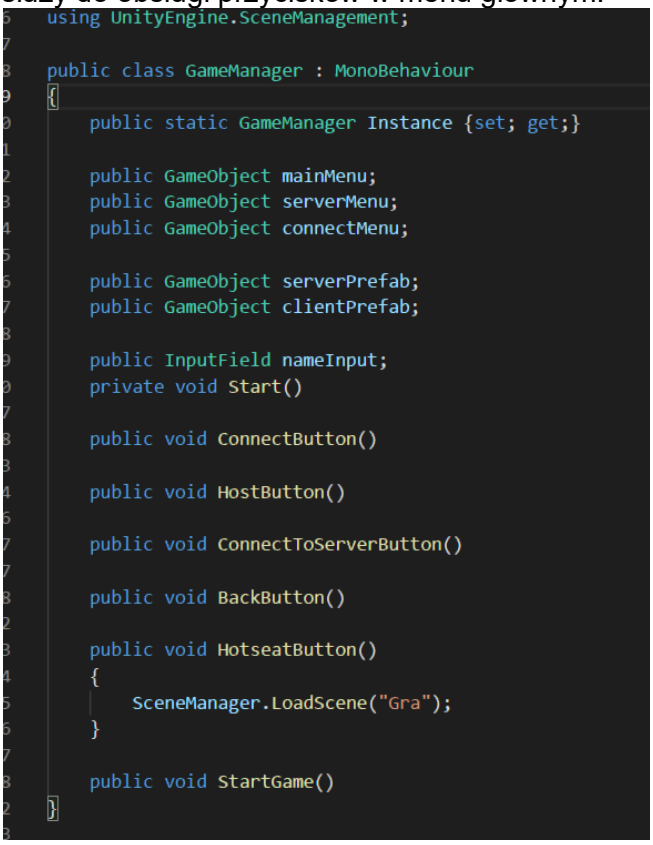


Klasy posiadają wiele metod, każda jest jest odpowiedzialna za jakąś część działania klasy. Część kodu poniżej to kod ze skryptu CheckersBoard - ta klasa odpowiada za funkcjonowanie rozgrywki do ostatniego pionka.



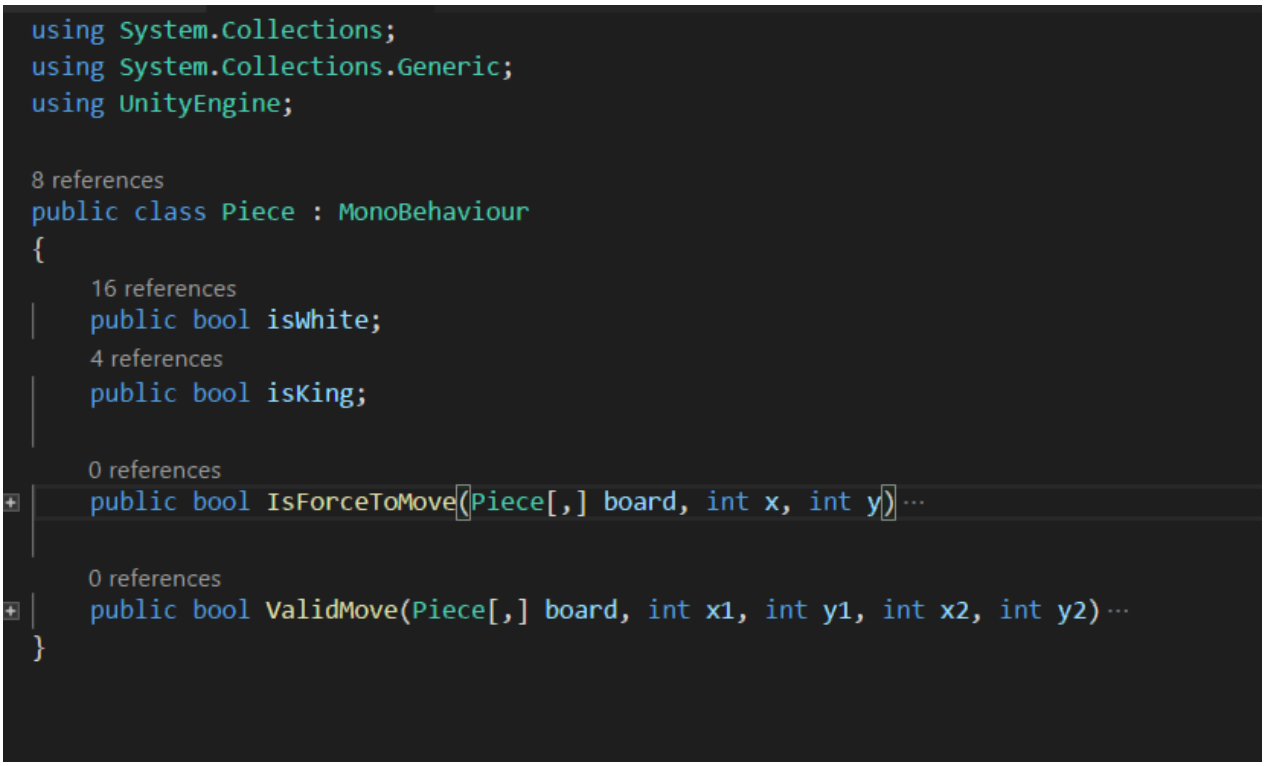
```
10 references
private Client client;

0 references
private void Start() ···

0 references
private void Update() ···

1 reference
private void UpdateMouseOver() ···

1 reference
private void UpdatePieceDrag(Piece p) ···

1 reference
private void SelectPiece(int x, int y) ···

1 reference
public void TryMove(int x1, int y1, int x2, int y2) ···

1 reference
private void EndTurn() ···

1 reference
private void CheckVictory() ···

2 references
private void Victory(bool isWhite) ···

1 reference
private List<Piece> ScanForPossibleMove(Piece p, int x, int y) ···

2 references
private List<Piece> ScanForPossibleMove() ···
```

Klasa GameManager - służy do obsługi przycisków w menu głównym.



```
using UnityEngine.SceneManagement;

public class GameManager : MonoBehaviour
{
    public static GameManager Instance {set; get;}

    public GameObject mainMenu;
    public GameObject serverMenu;
    public GameObject connectMenu;

    public GameObject serverPrefab;
    public GameObject clientPrefab;

    public InputField nameInput;
    private void Start()

    public void ConnectButton()

    public void HostButton()

    public void ConnectToServerButton()

    public void BackButton()

    public void HotseatButton()
    {
        SceneManager.LoadScene("Gra");
    }

    public void StartGame()
}
```

Klasy Client i Server: napisane funkcjonowanie klienta i serwera gry. W klasie Piece odpowiada za działanie (ruch) i generowanie pionków białych i czarnych.



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

8 references
public class Piece : MonoBehaviour
{
    16 references
    public bool isWhite;
    4 references
    public bool isKing;

    0 references
    public bool IsForceToMove(Piece[,] board, int x, int y) ···

    0 references
    public bool ValidMove(Piece[,] board, int x1, int y1, int x2, int y2) ···
}
```