

POLITECHNIKA ŚLĄSKA W GLIWICACH

WYDZIAŁ ELEKTRYCZNY

Instytut Elektrotechniki i Informatyki

**PROJEKT INŻYNIERSKI**

**Symulacja manewrów samochodowych wykonywanych na kursie prawa jazdy kategorii B.**

**Symulation of car maneuvers used on driving course category B.**

Student: **Grzegorz Kazimierz LISOWSKI**

Nr albumu: 210414

Studia: Stacjonarne I stopnia

Kierunek: Informatyka

Specjalność:

Prowadzący: dr inż. Maciej Sajkowski

Recenzent:

Spis treści

[1. Wstęp 5](#_Toc409359305)

[1.1 Przedmiot projektu 5](#_Toc409359306)

[1.2 Cel i zakres projektu 5](#_Toc409359307)

[1.3 Założenia projektu 5](#_Toc409359308)

[2. Opis technologii 6](#_Toc409359309)

[3. Projekt 10](#_Toc409359310)

[3.1 Menu główne 10](#_Toc409359311)

[3.2 Otoczenie 10](#_Toc409359312)

[3.3 Pojazd 11](#_Toc409359313)

[3.4 Scena pierwsza: łuk 11](#_Toc409359314)

[4. Implementacja 14](#_Toc409359315)

[5. Instrukcja użytkownika 15](#_Toc409359316)

[6. Podsumowanie i wnioski 16](#_Toc409359317)

[Bibliografia 17](#_Toc409359318)

# Wstęp

## Przedmiot projektu

Przedmiotem projektu jest opracowanie programu komputerowego mającego pomóc osobom przygotowującym się do zdawania egzaminu na prawo jazdy poprzez interaktywną naukę manewrów wymaganych na kursie prawa jazdy.

## Cel i zakres projektu

Celem projektu jest przygotowanie:

* menu głównego,
* otoczenia,
* zaprojektowanie i implementacja secnariusza manewru wykorzystywanego w nauce jazdy,
* implementacja narzędzi potrzebnych do weryfikacji wykonanych manewrów.

## Założenia projektu

Założenia dotyczące programu:

- jest interaktywną symulacją

- jest oparty o technologie 3-D

# Opis technologii

Jako silnik wybrana została technologia Unity3D w wersji 4.6 beta 20. Jest to rozbudowane narzędzie stosowane zarówno przez hobbystów jak i przez wielkie przedsiębiorstwa do tworzenia gier komputerowych oraz symulacji w środowisku 2D oraz 3D. Do najważniejszych cech Unity należą prostota oraz wieloplatformowość. Dzięki tym cechom można wykorzystać wszystkie zalety takich technologii jak OpenGL czy DirectX bez zagłębiania się w szczgóły ich działania. Wieloplatformowość pozwala tworzyć oprogramowanie na wiele systemów:

* Windows
* Mac
* Linux
* iOS
* Android
* Play Station 3
* Xbox 360
* Wii
* BlackBerry
* Windows Phone 8

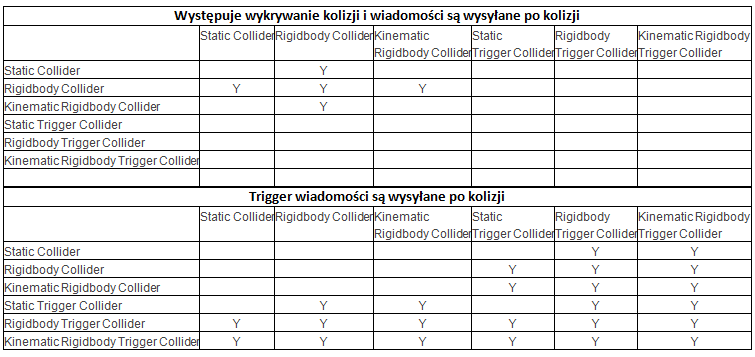
A także dzięki wykorzystaniu wtyczki webowej „Unity Web Player” na przeglądarki internetowe. Zapowiedziano też wsparcie dla technologii WebGL od wersji Unity 5.0.

Narzędzie to pozwala na szybkie tworzenie prototypów przy pomocy edytora, a Unity Store pozwala znaleźć wiele gotowych grafik i rozszerzeń do edytora. Dokumentacja technologii Unity jest rozbudowana i na bieżąco aktualizowana przez twórców.

Do symulacji fizyki w środowisku unity używana jest technologia NVIDIA PhysX.

Wśród elementów PhysX w Unity można wyróżnić:

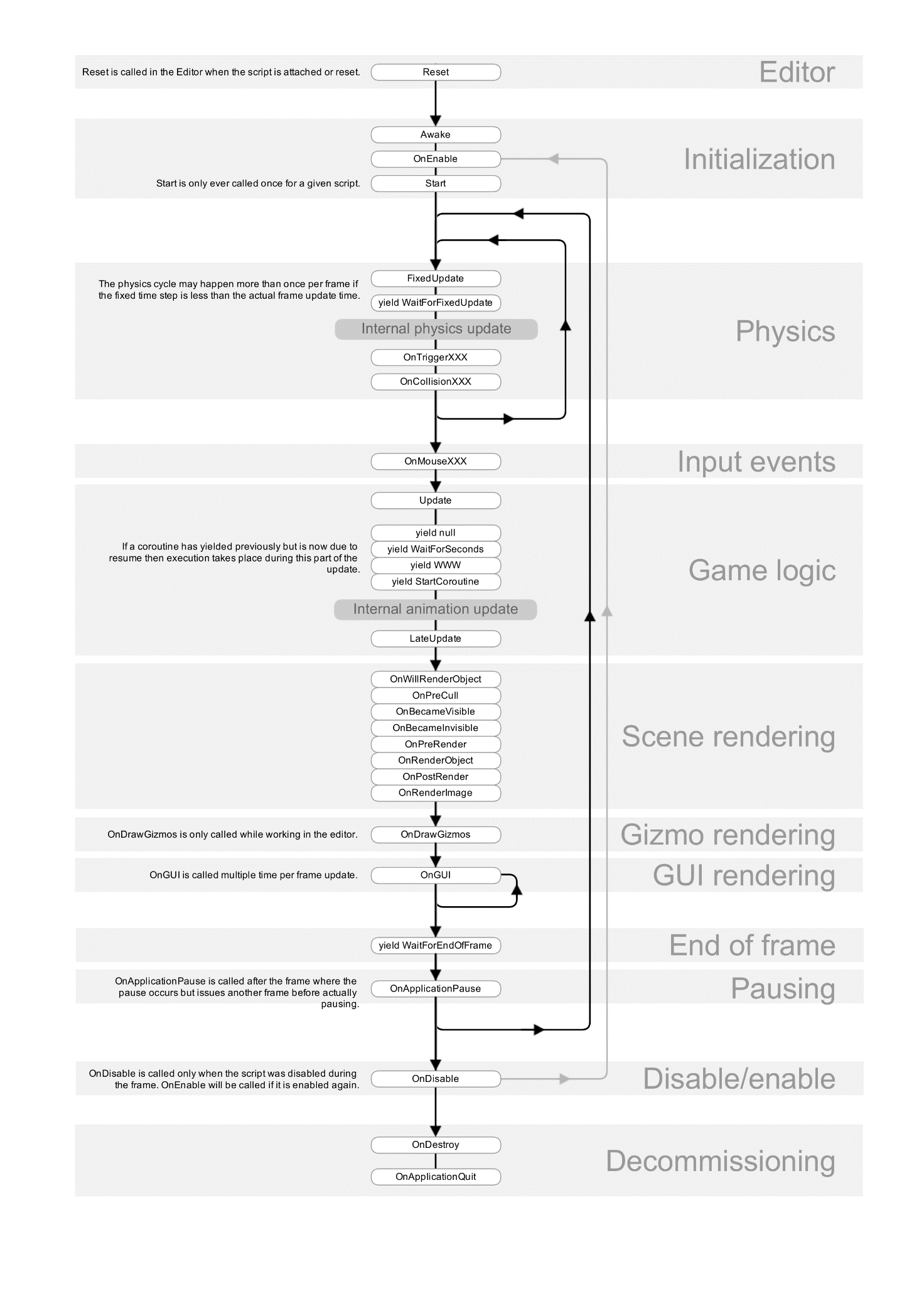
* Rigidbody – symulacja fizyki bryły sztywnej.
* Colliders – wykrywanie kolizji obiektów.
* Materiały Fizyczne – ustalanie oddziaływań na tarcie i odbicie.
* WheelColiders – fizyka kół pojazdu.



rys. 1 Detekcja kolizji. (źródło: http://wiki.unity3d.pl/).

rys. 1 przedstawia gdzie jest wysyłana informacja o kolizji przy jej wykryciu jest to istotna informacja przy tworzeniu aplikacji w tym środowisku gdyż decyduje o tym po stronie którego obiektu powinna znaleźć się logika odpowiedzialna za reakcję na daną kolizję.

Input Manager jest narzędziem unity które wprowadza warstwę abstrakcji pomiędzy urządzenia wejścia-wyjścia a program dzięki czemu można tworzyć oprogramowanie kontrolowane różnego rodzaju kontrolerami bez wprowadzania zmian w skryptach.



rys. 2 cykl wywołań funkcji skryptu (źródło: http://docs.unity3d.com/Manual).

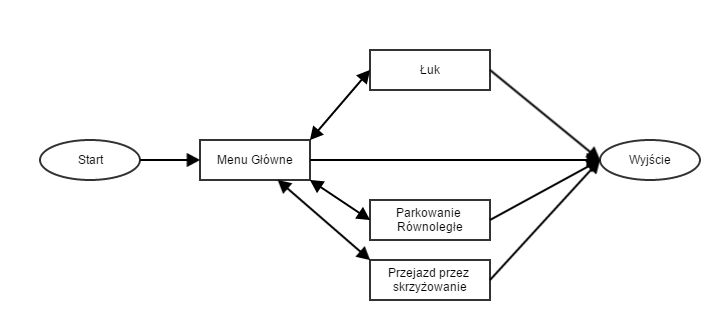
Technologia unity narzuca specyficzny sposób tworzenia aplikacji polegający na tworzeniu skryptów dokładanych do obiektów, w których metody wywoływane są w określonej kolejności. kolejność tą ukazuje (rys. 2) cykl wywołań funkcji skryptu. Pozwala on na wywołanie kodu przy uruchomieniu aplikacji dzięki funkcjom Awake, OnEnable oraz Start, wywoływanie regularnie kodu dzięki funkcjom Update, LateUpdate oraz FixedUpdate lub też reakcje na wydarzenia dzięki metodom OnTriggerEnter, OnTriggerExit, OnCollisionEnter, OnCollisionExit.

Ten sposób pracy oparty na skryptach oraz powyżej wymienione narzędzia czynią z Unity3D bardzo uniwersalne środowisko do tworzenia symulacji 3D oraz gier komputerowych różnego rodzaju.

# Projekt

## Menu główne

Jako menu główne zastosowano prosty interfejs użytkownika zawierający tytuł aplikacji oraz przyciski. Jako tło zastosowano trójwymiarowe otoczenie opisane w punkcie 3.2.



rys. 3 Schemat przejść między scenami

Na schemacie przejść między scenami (rys. 3) ukazano pożądany sposób działania menu głównego widać na nim, że w menu głównym należy umieścić przyciski odpowiedzialne za przejście na poszczególne sceny oraz przycisk opuszczenia aplikacji.

## Otoczenie

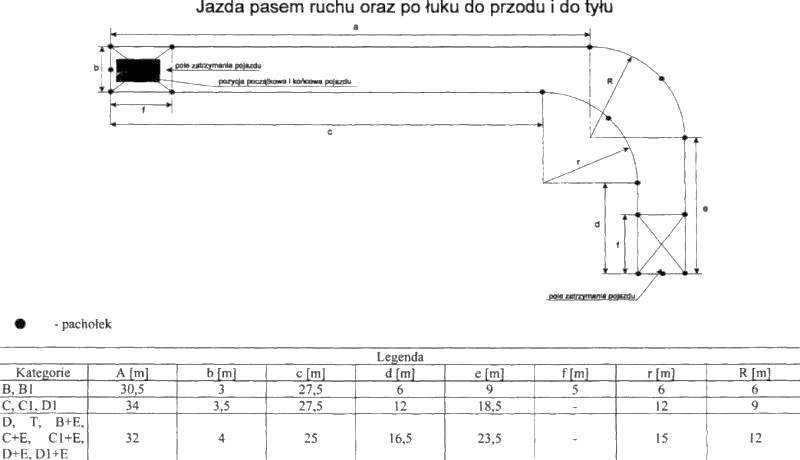
Otoczenie w którym odbywają się sceny składa się z 3 elementów: podłoża, budynku oraz muru ogradzającego cały teren.

## Pojazd

Jako pojazd stworzony został na podstawie skyCar z paczki new sample assets beta będącej częścią unity 5.0, w paczce tej znajdują się wszystkie komponenty potrzebne do stworzenia pojazdów w unity wraz z przykładowym modelem.

## Łuk

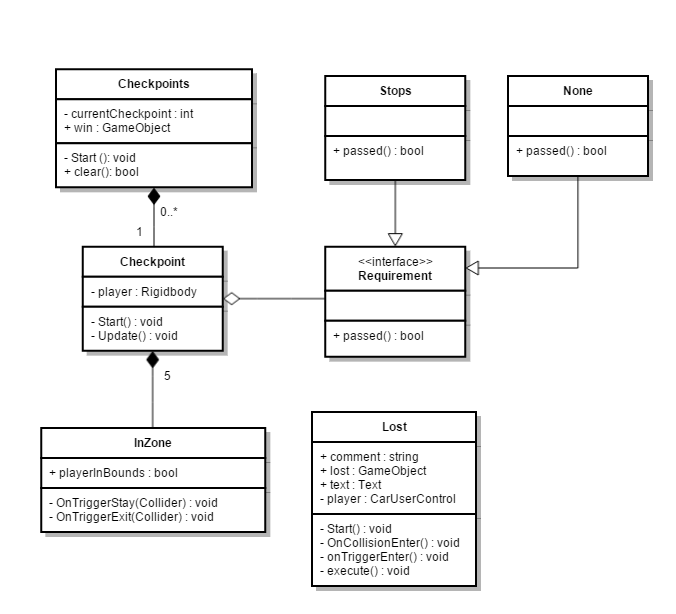
Manewrem rozpatrywanym w programie będzie jazda po łuku. Chociaż sam program przewiduje istnienie większej ilości manewrów w tej pracy opiszę tworzenie jazdy po łuku gdyż pozostałe tworzone są w sposób analogiczny z wykorzystaniem dokładnie tych samych narzędzi. W tym celu należy stworzyć tor o wymiarach podanych w rys. 4 ukazującym długości poszczególnych linii oraz rozmieszczenie słupków.



rys. 4 rozmieszczenie linii i pachołków na łuku (źródło: prawko.com)

W celu weryfikacji poprawnego wykonania manewru konieczne jest wykrywanie kolizji ze słupkami oraz z liniami, należy też wykryć czy pojazd znalazł się i zatrzymał się w poprawnej pozycji.

Rys. 5 przedstawia diagram klas. Klasa Checkpoints przechowuje referencje na wszystkie należące do niej skrypty klasy Checkpoint dba o to żeby aktywowane były one po kolei oraz o to aby wyświetlić menu wygranej po oczyszczeniu ostatniego checkpointa. Klasa Checkpoint sprawdza czy pojazd znalazł się we właściwej pozycji otrzymując informacje od pięciu obiektów klasy InZone oraz czy jest spełniony warunek podany przez interfejs Requirement jeśli obie te rzeczy są spełnione Checkpoint inforuje klasę Checkpoints ze należy przejść do kolejnego punktu, po skończeniu wszystkich punktów należy wyświetlić wiadomość o wygranej. Klasa InZone ustawia playerInBounds jeśli wykryje, że pojazd znalazł się w danej strefie, natomiast przy jej opuszczeniu playerInBounds przyjmuje wartość false. Interfejs Requirement realizuje wzorzec projektowy strategii, gdzie wymiennymi strategiami są klasy Stops oraz None. None pozwala na skończenie checkpointu poprzez najechanie w odpowiednie miejsce. Stops pozwala zakończyć checkpoint dopiero po zatrzymaniu się w wyznaczonej strefie. Ostatnią klasą jest klasa Lost dokładana jest ona do każdego obiektu i każdej strefy której naruszenie powoduje przegranie rozgrywki, po przegranej gracz straci kontrole nad samochodem oraz wyświetlane jest menu z powodem przegranej.

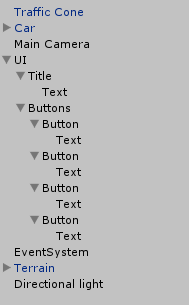


rys. 5 Diagram klas.

# Implementacja

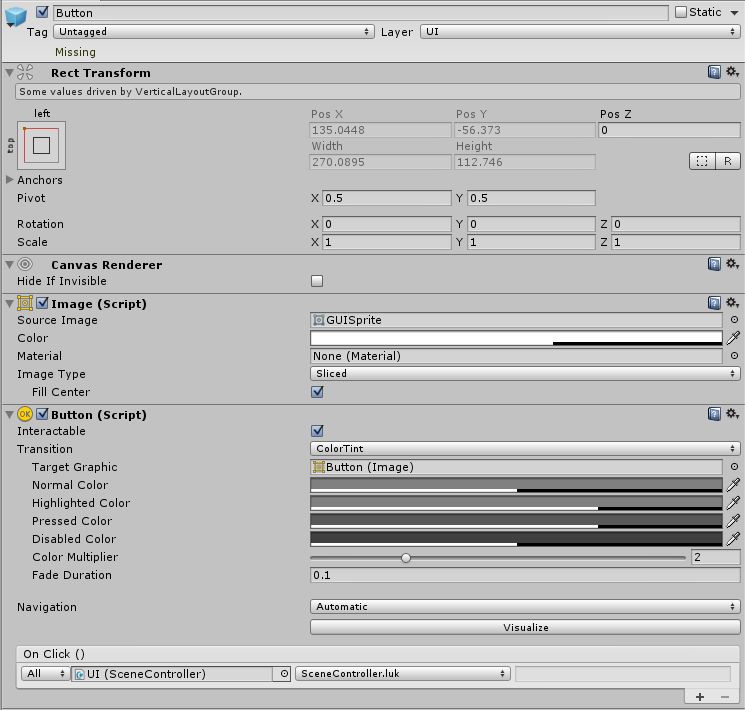
## Menu główne

Menu główne przygotowane jest w nowym systemie UI wydanym przez Unity.



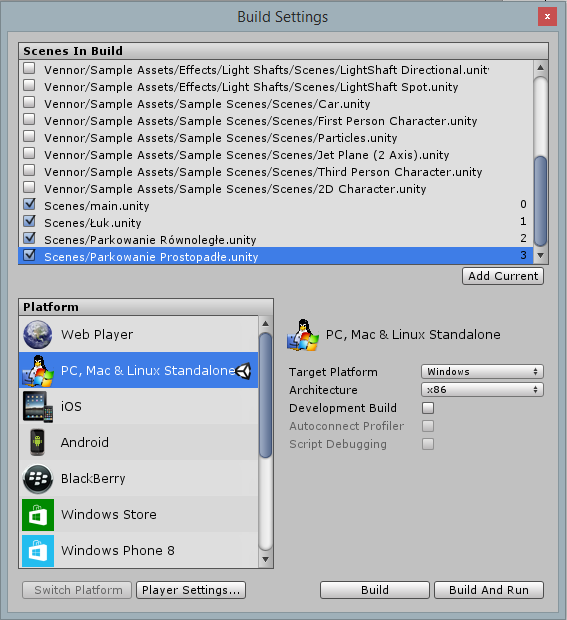
rys. 6 Struktura obiektów menu głównego

rys. 6 przedstawia strukturę obiektów na scenie Menu głównego. Element UI jest głównym canvasem, jest on nakładany na całą powierzchnię ekranu a komponent Graphic Raycaster zapewnia możliwość klikania w elementy menu. Kolejnym elementem jest Title jest to zwykły lekko szary obrazek obrazek tworzący tło dla elementu Text będącego jego dzieckiem. Dla zachowania skalowalności kotwice ustawione są w taki sposób aby tytuł zajmował 100% szerokości oraz 20% wysokości ekranu. Element Buttons przechowuje wszystkie przyciski jego najważniejszym komponentem jest Vertical Layout Group zapewni on, że wszystkie podległe elementy są rozstawione jeden pod drugim. Komponenty elementów Button przedstawione są na rys. 7 najważniejszym z nich jest komponent Button który odpowiada za sposób animowania akcji przycisku takich jak najechanie czy kliknięcie w tym przypadku jest to prosta zmiana kolorów w zależności od stanu przycisku, dalsza część tego komponentu pozwala ustalić akcje przycisków.



rys. 7 komponenty dołączone do elementu Button

Do oprogramowania wszystkich przycisków w aplikacji wykorzystany jest skrypt Scene Controller który ma przygotowane metody pozwalające mu na przejście na dowolną inną scenę a także na zamknięcie aplikacji. Jednak aby ten skrypt mógł działać konieczne jest wcześniejsze ustawienie aktywnych scen, dokonuje się tego na panelu Build Settings (rys. 8). Jako otoczenie wykorzystane zostało otoczenie wykorzystywane na wszystkich scenach, jako dodatki dołożony został później przygotowany samochód oraz pachołek, gotowy efekt można zobaczyć na rys. 9



rys. 8 Build Settings



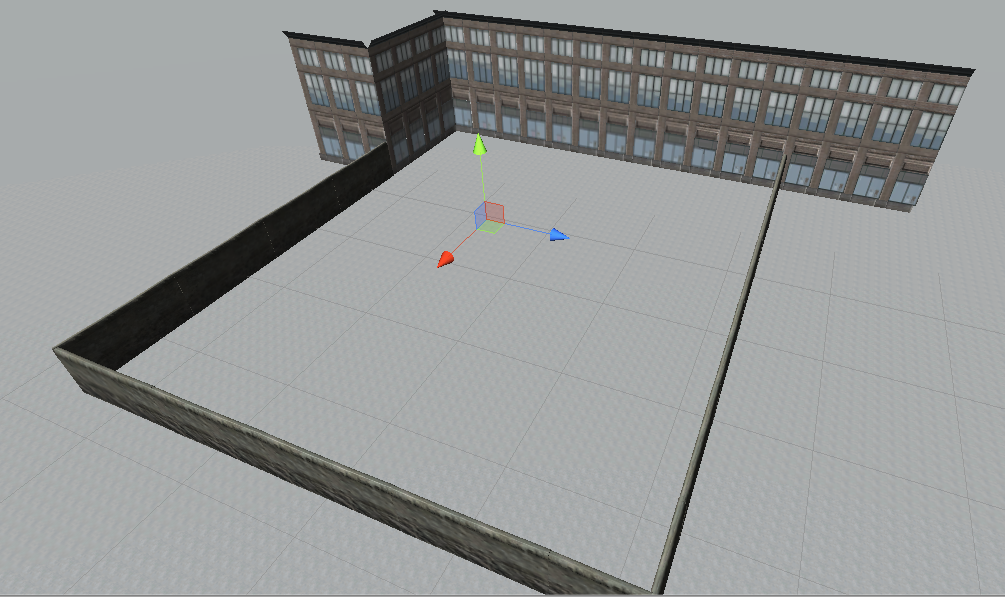
rys. 9 Menu główne

## Otoczenie

Podłoże jest wykonane jako terrain component na który nałożono teksturę betonu.

Budynek jest stworzony z modularnych płaskich elementów z darmowej paczki udostępnionej na Unity asset store o nazwie TheLoom\_Building\_1\_Free.

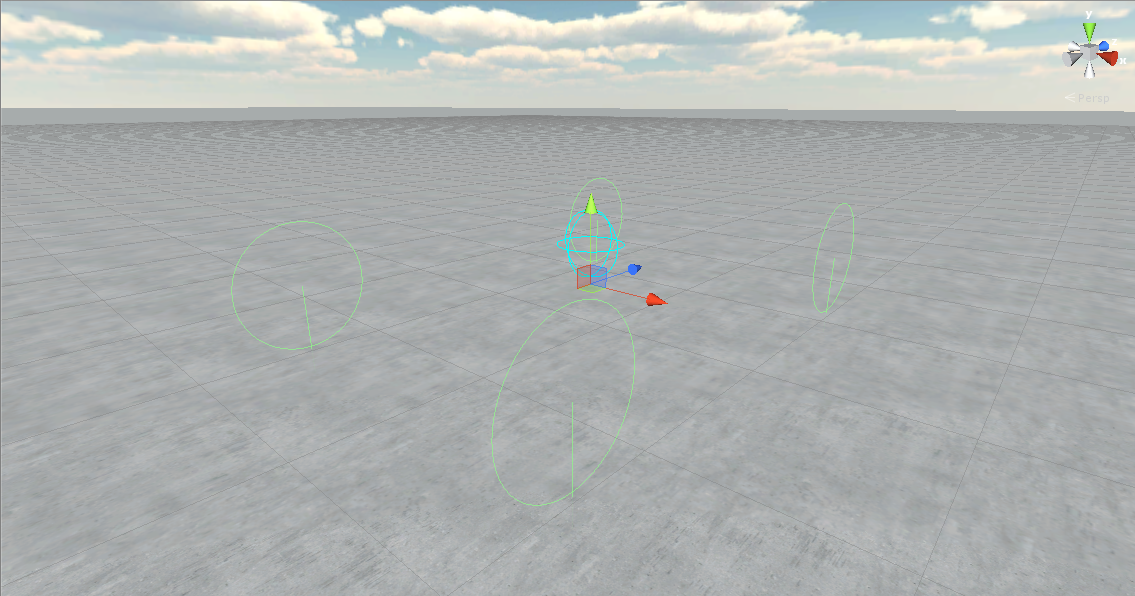
Mur jest obiektem 3D z paczki wallConcreteRough. Dla nadania światła na scenach do otoczenia zostało dodane także directional light.



rys. 10 Ostateczny wygląd otoczenia

## Pojazd

1. Tworzenie pojazdu rozpoczyna ustawienie elementu Empty Car Prefab rys. 10



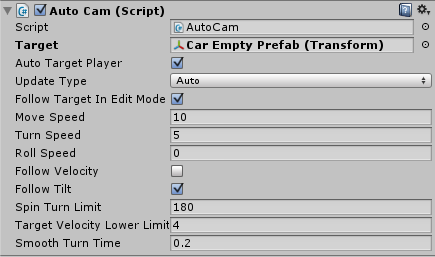
rys. 11 Prefabrykant pustego samochodu

prefabrykant ten posiada skrypt Car Controller na nadrzędnym elemencie który pozwala na ustawianie takich opcji jak kąt sterowania, szybkość odpowiedzi kierownicy, maksymalną prędkość oraz przyspieszenie, wśród bardziej zaawansowanych opcji można wyróżnić dostosowanie środka ciężkości, liczbę biegów, efekty poślizgu, a także siłę dociskającą samochód do podłoża, element ten posiada także skrypt odpowiadający za możliwość sterowania samochodem przez gracza. Element ten posiada 4 dzieci będące kołami posiadają one komponent Wheel oraz WheelCollider, symulują one prawdziwe zachowanie kół wraz z poślizgiem.



rys. 12 Dodanie modelu oraz przypisanie kół

Kolejnym etapem tworzenia pojazdu jest dodanie modelu, model należy ustawić jako dziecko Empty Car Prefab i model tak aby jego środek odpowiadał środkowi prefabrykantu. Kolejnym etapem jest przypisanie kół w tym celu należy model poszczególnych kół przypisać do odpowiedniego skryptu Wheel prefabrykantu w pole Wheel Model a następnie ustawić w poprawnej pozycji i rozmiarze Wheel Colidery poprzez naciśnięcie przycisku Align to assigned wheel model uzyskując efekt jak na rys. 11. Ze względów wydajnościowych kolizje nie są wykrywane siatką całego modelu lecz przybliżeniem składającym się z trzech prostopadłościanów odwzorowujących ogólną bryłę samochodu.



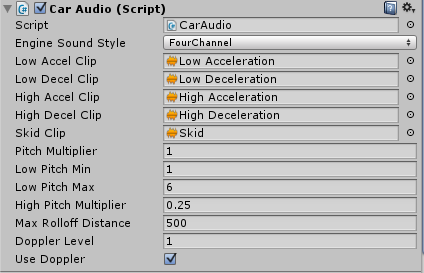
rys. 13 Komponent Auto Cam elementu Multipurpose Camera Rig

Taki model nadaje się już do jazdy jednak nie ma żadnej możliwości śledzenia go, w tym celu wykorzystana została kamera Multipurpose Camera Rig z Unity Sample Assets, pozwala ona na swobodne poruszanie kamerą dookoła danego punktu przy pomocy myszki. Podstawa kamery ustawiona jest w centrum samochodu efekt ten uzyskany jest poprzez przypisanie pola Target skryptu Auto Cam, pivot point wyciągnięty jest ponad samochód tak aby szło kamerą zobaczyć co się dzieje przed modelem a ramie kamery wyciągnięte jest za samochód ostateczny efekt widoczny jest na rys. 13.



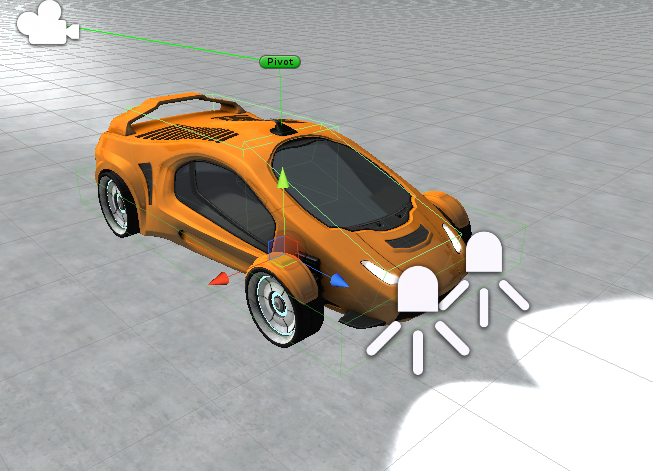
rys. 14 Multipurpose Camera Rig

W celu poprawy wyglądu samochodu oraz nadania mu realizmu dodany został jeszcze kolor karoserii w tym wypadku żółty, dołożone zostały także 2 światła typu spotlight symulujące reflektory samochodowe do opcji Skid Trail Prefab przypisany został prefabrykant z komponentem trail renderer dający wrażenie śladów zostawionych po poślizgu samochodu. Dodany został także komponent Car Audio wraz z przykładowymi dźwiękami jego ustawienia można zaobserwować na rys. 14.



rys. 15 Komponent Car Audio

Ostateczny wygląd obiektu samochodu można zaobserwować na rys. 15



rys. 16 Ostateczny wygląd pojazdu

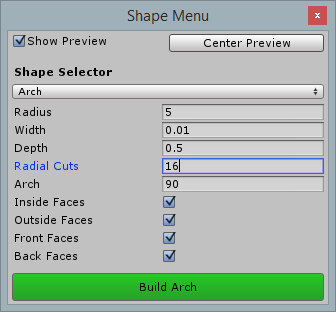
## Łuk

Wygląd sceny łuku został opisany w rozdziale 3.4, żeby uzyskać specyficzny kształt linii użyte zostało rozszerzenie od unity od firmy ProCore o nazwie Prototype pozwala ono na tworzenie prostych kształtów wewnątrz unity bez konieczności korzystania z edytorów grafiki, lista kształtów które można tworzyć za pomocą prototype znajduje się na rys. 17



rys. 17 Lista kształtów Prototype

aby utworzyć linie proste tworzony został cube o odpowiednich wartościach boków, do tworzenia zakrzywień łuku wykorzystano tworzenie arch przedstawione na rys. 18



rys. 18 Prototype łuk

wartość konta ustalona została na 90 stopni a ilość cięć dla uproszczenia ustawiona została na 16 pozwala to nie uzyskanie dobrego efektu krzywej bez generowania dużej ilości trójkątów. Aby kolizja linii była lepiej wykrywana kopia linii została umieszczona powyżej linii umieszczonych na ziemi, linie na ziemi są tylko ozdobami pozbawionymi kolizji natomiast ich kopia powyżej jest triggerem do zakończenia rozgrywki klasy Lost (listing 1).

listing 1 Klasa Lost

using UnityEngine;

using System.Collections;

using UnityEngine.UI;

public class Lost : MonoBehaviour {

public string comment;

public GameObject lost;

public Text text;

CarUserControl player;

void Start(){

player = GameObject.FindGameObjectWithTag("Player").GetComponent<CarUserControl>();

}

void OnCollisionEnter(Collision collision){

if (collision.gameObject.tag == "Player") {

execute();

}

}

void OnTriggerEnter(Collider collision){

if (collision.gameObject.tag == "Player") {

execute();

}

}

void execute()

{

lost.SetActive(true);

text.text = comment;

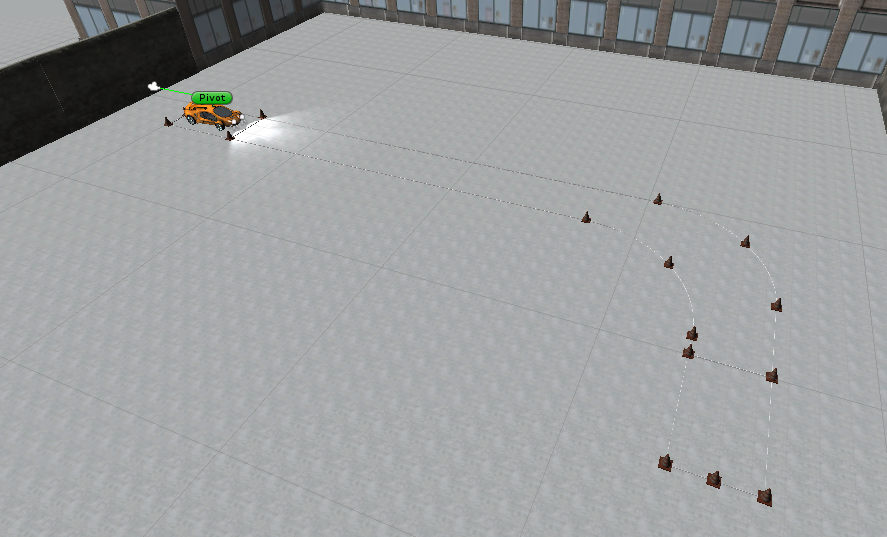
player.enabled = false;

}

}

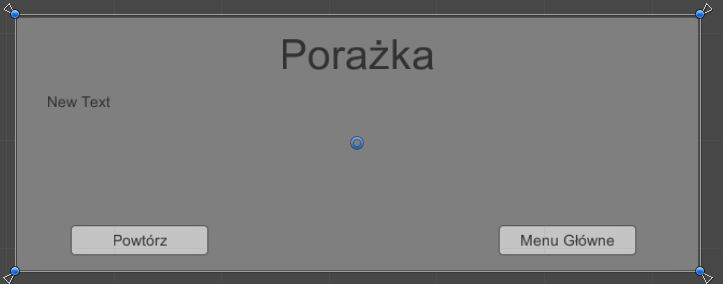
Klasa Lost (listing 1) gdy zostanie przypięta do obiektu powoduje, że jeśli zostanie przypięta do obiektu każda kolizja z nim (jeżeli ma Collider) lub wejście w jego strefę (jeśli ma Trigger) powoduje wywołanie metody execute która wyświetla menu przegranej, zmienia tekst na tym menu na powód przegranej ustawiony w klasie oraz odbiera graczowi możliwość kontrolowania pojazdu.

Słupki uliczne są modelami 3D z nałożonym rigidbody aby można je było wywracać, na słupki nałożony jest mesh collider dla dobrego odwzorowania siatki, oraz dodany jest jak w przypadku linii komponent Lost. Gotowy tor jest zaprezentowany na rys. 19

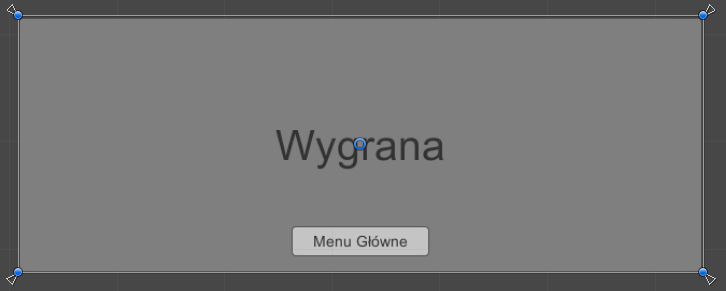


rys. 19 Tor sceny Łuk

Jako menu wygranej oraz przegranej wykorzystane są elementy UI zaprezentowane na rys. 20 oraz rys. 21.



rys. 20 UI przegrana



rys. 21 UI wygrana

# Instrukcja użytkownika

# Podsumowanie i wnioski

Cel projektu został wykonany

# Bibliografia

|  |  |
| --- | --- |
| [1] | O. s. k. prawko. [Online]. Available: http://www.prawko.com/. [Data uzyskania dostępu: 6 Styczeń 2014]. |
| [2] | „Unity manual,” [Online]. Available: http://wiki.unity3d.pl/. [Data uzyskania dostępu: 12 Styczeń 2015]. |
| [3] | W. Goldstone, Projektowanie gier w środowisku Unity 3.X, Helion, 2013. |