SPRAWOZDANIE

PROGRAMOWANIE GIER KOMPUTEROWYCH – PROJEKT

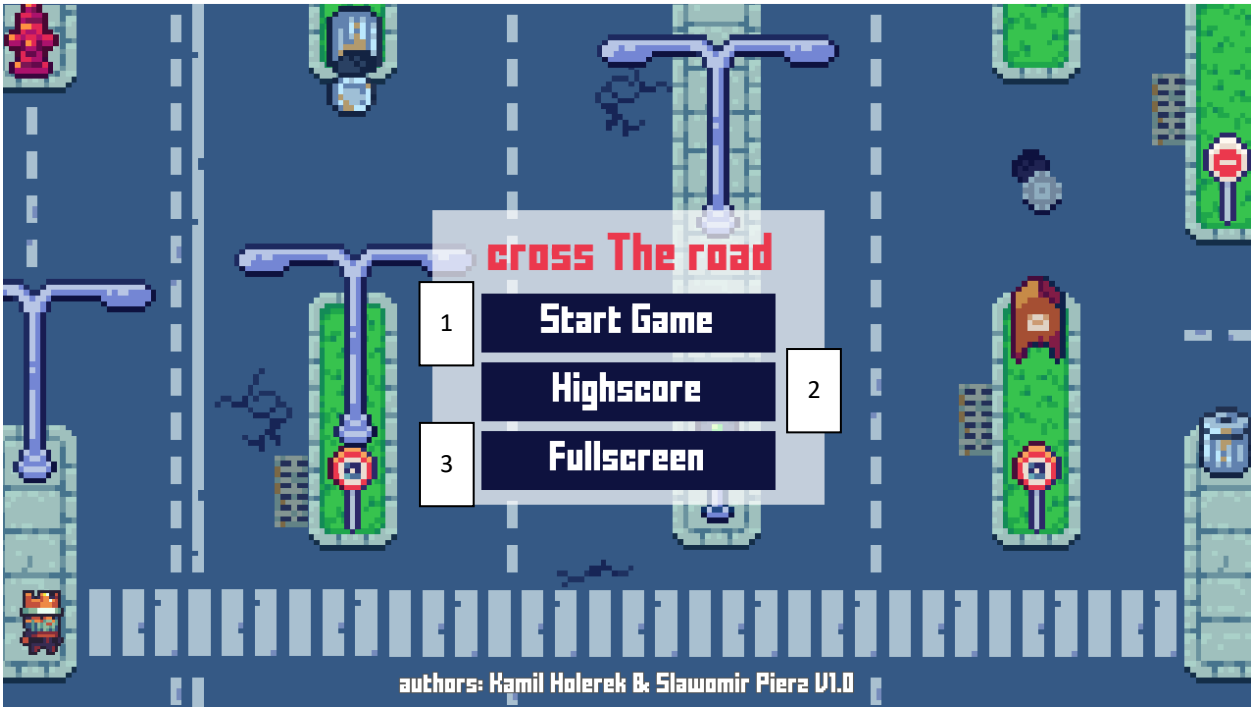
GRA CROSS THE ROAD

AUTORZY: KAMIL HOLEREK I SŁAWOMIR PIERZ, D1, SEMESTR 7

SPIS TREŚCI

OPIS GRY, CEL I MOŻLIWE ZAKOŃCZENIA

Gra polega na przechodzeniu przez ruchliwą ulicę, gracz kieruje postacią jedynie wydając jej polecenie przejścia przez ulicę za pomocą klawiszu spacji, kiedy postać zacznie ruch nie można go przerwać, ruch przerywa jedynie kolizja, koniec czasu lub przejście na drugi kraniec jezdni, za każde przejście całej jezdni gracz dostaje 1 punkt oraz zyskuje dodatkowo 5 sekund do czasu gry. Po uruchomieniu gry gracz ma to wyboru start gry (Screen 1.1), ekran rekordów (Screen 1.2) oraz możliwość włączenia pełnego ekranu (Screen 1.3), po ekranie menu można nawigować za pomocą strzałek na klawiaturze bądź myszki, potwierdzenie za pomocą entera bądź spacji. Interfejs jak i kod aplikacji zostały wykonane w języku angielskim tak by oswajać się ze standardami pisania oprogramowania obecnymi na rynku.



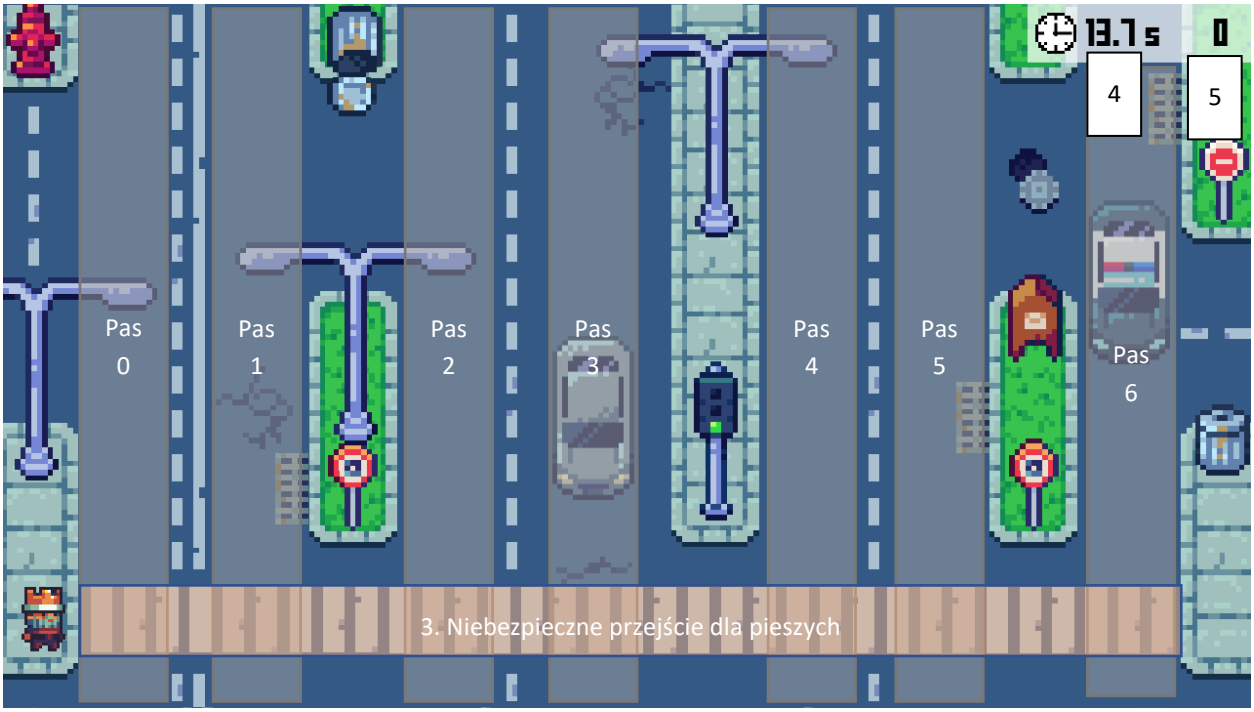
Screen 1 – Gra tuż po uruchomieniu

Mapa gry składa się z 7 pasów na których losowo generują się pojazdy, których jest 9. Początkowo gracz otrzymuje 15 sekund gry by dodać dynamikę grze, za każde przejście przez jednię otrzymuje punkt oraz dodatkowe 5 sekund gry. Celem gry jest wystartowanie postaci za pomocą klawisza spacji w odpowiednim momencie tak by ominął on wszystkie nadjeżdżające pojazdy i uniknął kolizji mieszcząc się równocześnie w czasie gry. Mechanizm czasu gry został dołożony celowo by gracz nie zwlekał zbytnio z przejściem. Poziom trudności rośnie wraz z narastającą ilością punktów na koncie gracza.

Interfejs gry składa się z następujących elementów (Screen 2):

1. Pasy ruchu, w sumie 7 (nie zawsze na każdym pojawia się pojazd ale zagadnienie zostanie to poruszone w rozdziale „Mechanizm poziomów”,
2. Chodniki do oczekiwania na kolejne przejście,
3. Niebezpieczne przejście dla pieszych
4. Pozostały czas gry – w przypadku gdy pozostało nam mniej niż 10 sekund napis zmienia kolor na czerwony,

5. Liczba zdobytych punktów.



Screen 2 – Gra po rozpoczęciu rozgrywki

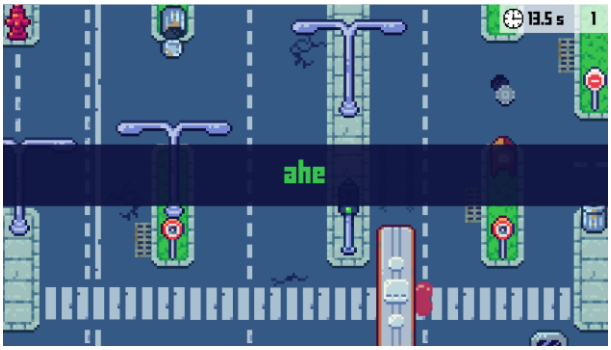
Gra została zaprojektowana w taki sposób by umożliwić praktycznie nieograniczoną rozgrywkę i nie jest możliwa do przejścia. Mechanizm poziomów został napisany do 999 punktów (o czym więcej w dalszej części sprawozdania). Możliwe są jedynie dwa możliwe scenariusze zakończenia rozgrywki – jest to śmierć postaci spowodowana kolizją (Screen 3) lub skończenie się czasu (Screen 4).

Screen 3 – Koniec gry, kolizja

Screen 4 – Koniec gry, czas się skończył

Niezależnie od zakończenia gry sprawdzany jest wynik gracza i jeżeli jest on większy od aktualnego rekordu dodatkowo gra poprosi o wpisanie imienia gracza (rekordy są zapisywane lokalnie na komputerze w %appdata%\Godot\app_userdata\crossTheRoad\save.res rozszerzenie celowo zostało wybrane w ten sposób aby uniemożliwić edycje rekordu). W Screenach 5 do 7 przedstawiono proces wpisywania nowego rekordu.

Screen 5 – Nowy rekord, podaj imię



Screen 6 – Nowy rekord, wpisano imię, potwierdzamy enterem

Screen 7 – Nowy rekord na zaktualizowanym ekranie „highscore”

MECHANIZM POZIOMÓW

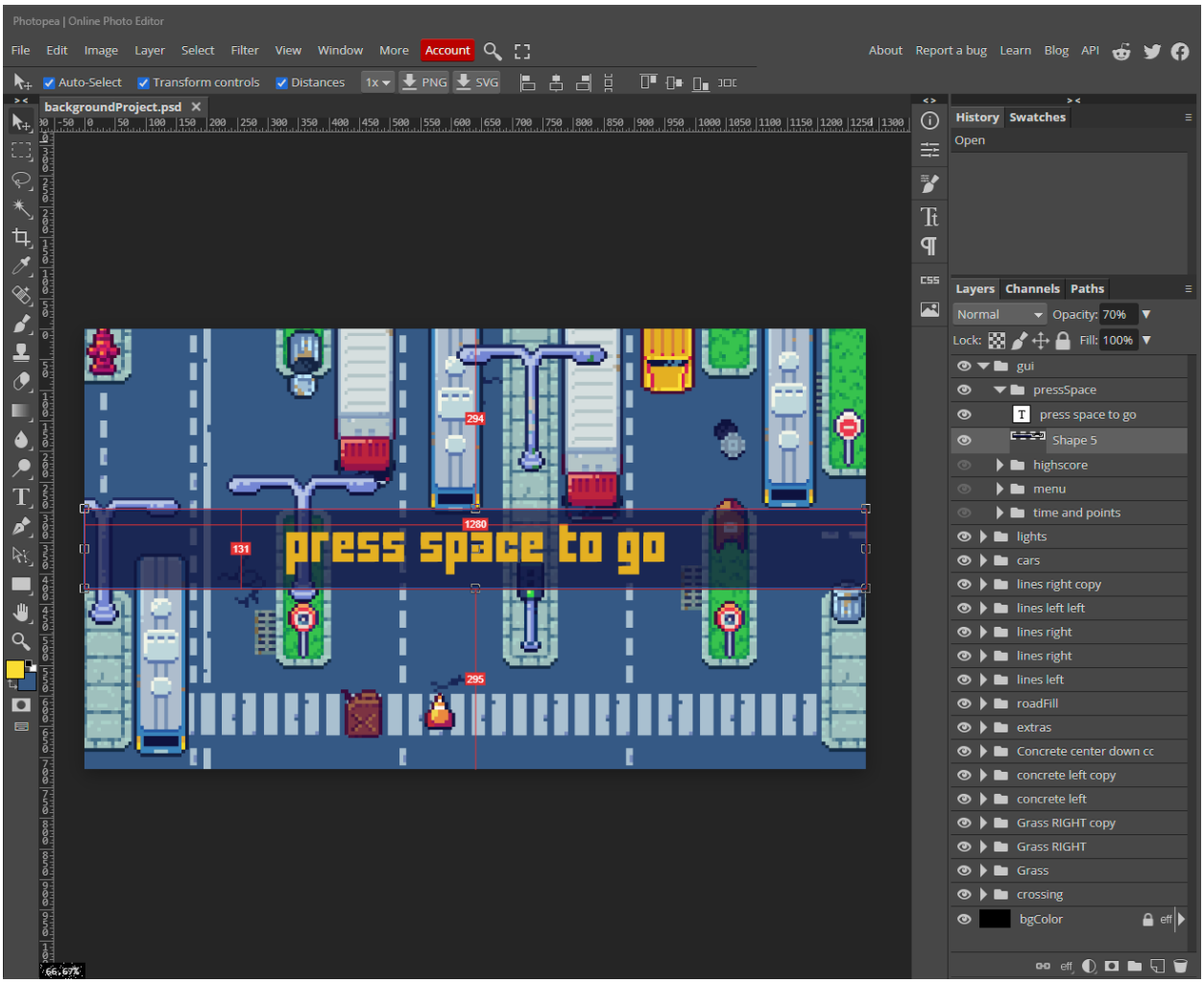
```
extends Node

var maxCarsOnTheRoad = 2
var minimumTravelTime = 4

func setLevelNumber(points):
    if points in range(0,6):
        maxCarsOnTheRoad = 2
        minimumTravelTime = 4
    elif points in range(7,10):
        maxCarsOnTheRoad = 3
        minimumTravelTime = 3.5
    elif points in range(11,15):
        maxCarsOnTheRoad = 3
        minimumTravelTime = 3.0
    elif points in range(16,20):
        maxCarsOnTheRoad = 4
        minimumTravelTime = 3.0
    elif points in range(21,25):
        maxCarsOnTheRoad = 5
        minimumTravelTime = 3.0
    elif points in range(26,30):
        maxCarsOnTheRoad = 5
        minimumTravelTime = 2.5
    elif points in range(30,40):
        maxCarsOnTheRoad = 6
        minimumTravelTime = 2.5
    elif points in range(50,999):
        maxCarsOnTheRoad = 7
        minimumTravelTime = 2.5

func getMaxCarsOnTheRoad():
    return maxCarsOnTheRoad

func getMinimumTravelTime():
    return minimumTravelTime
```

Listing 1 - res://assets/scripts/levelParameters.gd

Gra została zaprojektowana w taki sposób by poziom trudności był narastający wraz ze wzrostem punktów u gracza. Zmiennymi determinującymi trudność jest przede wszystkim maksymalna ilość pojazdów będąca aktualnie w grze (pojazdy dla ułatwienia zawsze generują się na różnych pasach ruchu tak więc górną granicą ilości pojazdów jest ilość pasów czyli 7. Drugą zmienną generującą trudność jest maksymalna możliwa prędkość nadjeżdżających pojazdów, słowo maksymalna jest użyte nie bez przyczyny ponieważ przed każdym wygenerowaniem pojazdu losowana jest jego prędkość w zakresie od do gdzie maksymalna wartość jest zależna właśnie od poziomu. Prędkość nie jest wyrażana w jednostkach fizycznych do tego właściwych tylko w czasie przejazdu pojazdu z góry ekranu na sam dół, stąd też wraz ze wzrostem trudności minimalna wartość czasu przejazdu spada. Dzięki zastosowaniu tego mechanizmu mimo wzrostu poziomu nadal czasami trafiają się powolne pojazdy a także pojazdy poruszają się z różnymi prędkościami. Funkcją odpowiedzialną za zwrot tych wartości jest getMaxCarsOnTheRoad() oraz getMinimumTravelTime() z pliku obok (Listing 1). Jak widać gra jest zaprojektowana do poziomu 999, którego osiągnięcie wydaje mi się nie możliwe – najlepszy tester gry osiągnął na czas pisania tego dokumentu 17 punktów.

PROJEKT GRAFICZNY, POSTAĆ ORAZ POJAZDY

Z założenia gra miała być prosta i w taką samą stronę poszliśmy w procesie projektowania grafiki – postawiliśmy na retro. Chcieliśmy uzyskać efekt gry sprzed 20 lat, wtedy często spotykanym typem gry była taka w której należało zdobyć jak największą ilość punktów i grywalność uzyskiwało w sposób inny niż skomplikowana budowa, fabuła czy realistyczna grafika. Całość gry miała zostać zaprojektowana w programie Photoshop ale z racji na zmianę polityki dostępu do wersji 30 dniowej wersji zastosowaliśmy darmowy odpowiednik – Photopea (Screen 8), który bardzo pozytywnie nas zaskoczył i niczego w nim nie brakowało.

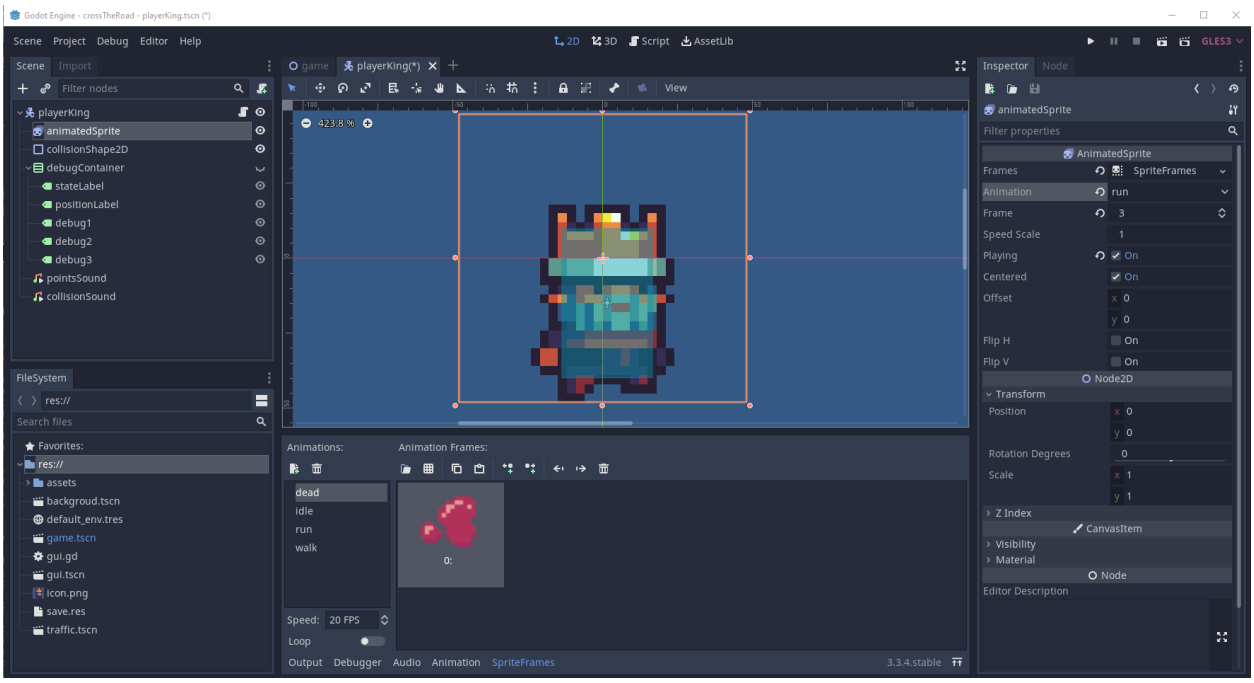


Screen 8 – Projekt gry w programie Photopea (darmowy klon Photoshop).

Specjalnie do projektu kupiliśmy paczkę graficzną https://groovymcgee.itch.io/car-game-pixel-art i korzystając z grafik samochodów oraz elementów ulicy zostało stworzone tło gry a następnie każdy z pojazdów. Wszystkie grafiki zostały przeskalowane. Natywną rozdzielczością gry jest 1280x720 (HD), początkowo pomysł wydawał się dobry z racji na łatwą edycje na monitorach Full HD – myśleliśmy, że gra po przeskalowaniu na FULL HD albo i więcej będzie wyglądać dobrze z racji na bardzo prostą grafikę. Niestety ale gra wygląda dobrze tylko w rozdzielczości natywnej a w większej trochę gorzej. Kolejnym krokiem było znalezienie odpowiedniej postaci do gry – miała być na tyle charakterystyczna by można było zrobić z niej ikonę gry – znaleźliśmy odpowiedni darmowy projekt tutaj

https://opengameart.org/content/a-platformer-in-the-forest. Animacje postaci wykonaliśmy już w środowisku Godot. (Screen 9). Postać została zanimowana za pomocą AnimetedSprite, dodano do niej 4 animacje:

- Dead – Czyli nasza postać bezpośrednio po kolizji,
- Idle – Oczekiwanie na przejście przez drogę, w zależności od kierunku ruchu postać zostaje jeszcze obrócona (Flip H),
- Run – Przejście przez drogę, w zależności od kierunku ruchu postać zostaje jeszcze obrócona (Flip H),
- Walk – Nieużywana, pomysł został umorzony.



Screen 9 – Animacja postaci.

Kolejnym elementem graficznym i oczywiście najważniejszym są pojazdy, których w grze mamy 9 modeli. Graficzne przedstawienie pojazdu nie ma korelacji z prędkością bo ta jest losowana niezależnie od jego typu. Poniżej lista samochodów stosowanych w grze (Screen 10):

- Taxi
- Radiowóz
- Żółty samochód
- Szary samochód
- Czerwony samochód
- Niebieski samochód
- Autobus szkolny
- Ciężarówka
- Autobus

Po każdej kolizji a co za tym idzie końcu gry gracz otrzyma informacje jaki pojazd go potrącił.

Screen 10 – Wszystkie pojazdy.

MUZYKA I EFEKTY DZWIĘKOWE

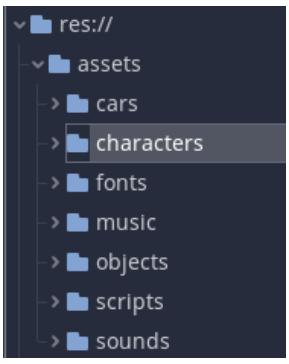
Ścieszka dźwiękowa również wpasowuje się w retro-klimat gry. Całość zamyka się w 4 plikach dźwiękowym. Mamy osobną ścieżkę do menu gry, do samej rozgrywki (w tym przypadku dźwięk jest zapętlony) jak i również dźwięki efektów. Wspomniane efekty to dźwięk zdobycia punktu oraz dźwięk hamującego pojazdu tuż przed kolizją. Do wszystkich tych dźwięków użyliśmy AudioStreamPlayer. Dzięki nie są naszego autorstwa, był tego typu pomysł ale niestety brakłoby nam czasu z racji tego, że to ostatni semestr. Poniżej źródła z których pozyskaliśmy wyżej wymienione dźwięki.

Muzyka – https://mbardin.github.io/PDM-resources/media/sound_samples/music/BossMain.wav

Efekty (Hamowanie) - https://opengameart.org/content/car-tire-squeal-skid-loop

Efekty (Punkty) - https://opengameart.org/content/8-bit-platformer-sfx

STRUKTURA PROGRAMU



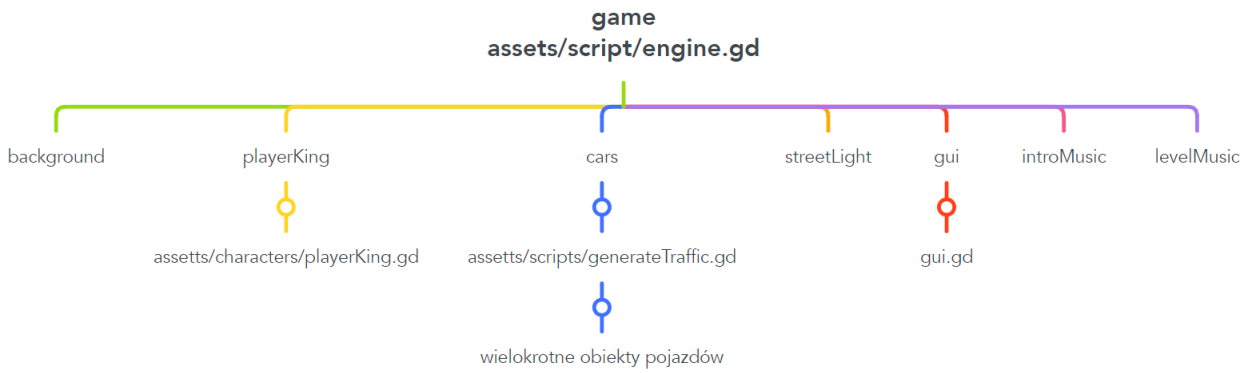
W programie staraliśmy się zachować porządek i ład. Od początku stworzyliśmy katalogi, które podpatrzyliśmy w grach bardziej doświadczonych programistów (Screen 11):

- Cars – W tym katalogu posiadamy wszystkie pojazdy zarówno w pliku .png jak i później w pliku sceny .tscn
- Characters – postać i elementy niezbędne do animacji postaci
- Fonts – ciekawa czcionka używana wszędzie w grze
- Music – tło muzyczne

Screen 11 – Drzewo katalogów.

- Objects – obiekty graficzne
- Scripts – najciekawszy z wszystkich katalogów, w nim znajdują się prawie wszystkie skrypty w grze,
- Sounds – wspomniane wcześniej efekty dźwiękowe

Główną sceną startową jest scena game.tscn (typu Node2D dołączony jest do niej skrypt assets/scripts/engine.gd), znajdują się w niej następujące podsceny:



- background – scena tła, jej zawartość to TextureRect oraz dwa collisionPolygony (zbędne po zmianach animacji postaci),
- playerKing – scena wyświetlająca oraz animująca postać (są w niej zawarte także efekty dźwiękowe generowane przez samą postać takie jak dźwięk zdobycia punktu czy kolizji,
- cars – scena domyślnie bez obiektów. W niej za pomocą skryptu assets/scripts/generateTraffic.gd generowane są pojazdy, scena odpowiada także za odpowiedni poziom trudności korzystając dodatkowo ze assets/scripts/levelParameters.gd. Scena wywołuje wielokrotnie inne obiekty samochodów, każdy z nich ma podpięty ten sam skrypt assets/scripts/car.gd, który odpowiada za poruszanie samochodu a także detekcje ewentualnej kolizji – emituje sygnał. Po kolizji za pojazdem dodawane są także ślady hamowania (assets/cars/tireMarks.tscn),
- streetLight – scena czysto graficzna tak by latarnie umiejscowić wyżej na stosie warstwy Z i uzyskać poprawny efekt graficzny,
- gui – scena odpowiedzialna za komunikacje z użytkownikiem, wyświetlanie menu, wyświetlanie ilości punktów oraz czasu, zapis i odczyt najlepszego wyniku,
- introMusic – muzyka menu,
- levelMusic – muzyka grana podczas rozgrywki.

Każda z podanych scen posiada sporo funkcji, podczas pisania kodu staraliśmy się zachować czystość i przejrzystość tak by zarówno edycja jak i debugowanie było przyjemne.

PODSUMOWANIE

Tworzenie gry mimo jej prostoty było dla nas ciekawym doświadczeniem, różnym od wcześniejszych zadań algorytmicznych. Pochłonęła sporo pracy ale także dała więcej satysfakcji niż wcześniejsze projekty. Podczas prac używaliśmy GITa – poniżej umieszczam link do repozytorium:

https://github.com/kholerek/crossTheRoad

Ponadto gra została wyeksportowana do środowiska Windows oraz HTML5 – możliwość gry w przeglądarce można sprawdzić pod poniższym adresem:

https://holerek.com/crosstheroad/crossTheRoad.html

Planowaliśmy także dodanie mechanizmu znajdziek czy inne postaci odblokowywane wraz z osiąganiem wyższych wyników ale z racji tego, że w tym semestrze czeka nas jeszcze praca inżynierska postanowiliśmy na tym zakończyć.