Politechnika Śląska w Gliwicach

Wydział Matematyki Stosowanej

Kierunek: Informatyka

Specjalność: Programowanie Aplikacji Mobilnych

Dokumentacja techniczna

|  |  |
| --- | --- |
| **Przedmiot:** | Aplikacje mobilne – projekt zespołowy |
| **Prowadzący:** | dr inż. Adam Zielonka |
| **Lider zespołu:** | Marek Jędrzejewski ( [mjedrzejewski92@gmail.com](mailto:mjedrzejewski92@gmail.com) ) |
| **Wykonawca 1:** | Paweł Moniewski |
| **Wykonawca 2:** | Łukasz Wsół |
| **Tytuł projektu:** | Zdzichu Runs |
| **Technologia** | Android, LibGDX |
| **Użyte narzędzia** | Android Studo, Synfig Studio, GIMP, Hydrogen, Audacity, LibGDX TexturePacker |
| **Czas realizacji** | 24.02.2016 – 15.06.2016 |
| **Repozytorium:** | <https://github.com/marekjedrzejewski/zdzichu-runs> |
| **Cel projektu:**  Stworzenie gry typu „runner”. Nauka wykorzystania silnika graficznego, zaznajomienie się z typowymi mechanizmami gier czasu rzeczywistego. Adaptacja gry na urządzenia mobilne z systemem Android. | |
| **Krótki opis projektu:**  Gra w stylistyce animacji  <https://www.youtube.com/watch?v=_BkQUgvnANE>  osadzona w świecie rzeczywistym, gdzie na bohatera czyhają przeszkody, z którymi mógłby się spotkać na swej drodze każdy (rozpędzone samochody, kaktusy, jastrzębie), choć w natężeniu ponadprzeciętnym. Bohater nie może się zatrzymać, a każdy napotkany na jego drodze obiekt jest zwiastunem śmierci, którą musi oszukać. | |

**Historia zmian dokumentu**

**Tabela zmian dokumentacji:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Osoba modyfikująca** | **Data zmiany** | **Co uległo zmianie** | **Opis zmiany** | **Uwagi** |
| Adam Zielonka | 16.03.2016 | Strona tytułowa | Przygotowanie szablonu strony tytułowej |  |
| Marek Jędrzejewski  Paweł Moniewski  Łukasz Wsół | 23.03.2016 | Strona tytułowa,  Opis projektu | Opisy projektu, konkretyzacja założeń i wymagań |  |
| Łukasz Wsół | 30.03.2016 | Założenia | Wpisanie oraz poprawki do założeń ustalonych na portalu trello.com |  |
| Paweł Moniewski | 06.04.2016 | Projekt gry | Dodanie schematu koncepcyjnego klas |  |
| Łukasz Wsół | 26.04.2016 | Technologia,  Opis projektu | Opisanie wybranej technologii, drobne poprawki stylistyczne |  |
| Paweł Moniewski | 04.05.2016 | Technologia | Rozszerzenie opisu technologii |  |
| Marek Jędrzejewski | 11.05.2016 | Technologia |  |  |
| Łukasz Wsół | 18.05.2016 | Technologia,  Opis Projektu |  |  |
| Paweł Moniewski | 25.05.2016 | Podział prac | Aktualizacja |  |
| Marek Jędrzejewski | 07.06.2016 | Lista dodatków | Dodanie nowych grafik |  |
| Paweł Moniewski | 07.06.2016 | Technologia | Dodanie informacji o dźwiękach w grze |  |
| Łukasz Wsół | 07.06.2016 | Technologia |  |  |
| Paweł Moniewski | 14.06.2016 | Instrukcja obsługi |  |  |
| Paweł Moniewski | 14.06.2016 | Testy |  |  |
| Marek Jędrzejewski | 20.06.2016 | Testy | Dodanie nowych urządzeń |  |

**Rozdział 1. Opis projektu**

Celem projektu jest realizacja gry czasu rzeczywistego, z którą to wiąże się implementacją rozwiązań, niespotykanych w aplikacjach klienckich czy grach turowych. Decyzja napisania gry w stylu “Runner” została podjęta po wcześniejszej analizie występujących w niej mechanizmów oraz z rozpatrzeniem propozycji prowadzącego. Podział prac został ustalony w pierwszych tygodniach, po rozważeniu wcześniejszego doświadczenia w tworzeniu poszczególnych elementów przez poszczególnych członków grupy. Silnik graficzny został wybrany na drodze popularności oraz stosowanego języka (Java), dzięki czemu istnieje lepszy dostęp do przykładów oraz źródeł.

Grą jest platformówką typu auto-runner, podczas której wydarzenia obserwujemy z perspektywy oczu głównego bohatera Zdzicha, pędzącego na złamanie karku przez trasy pełne przeszkód. Naszym zadaniem jest unikanie kolizji. Sterowanie przypomina tradycyjne gry typu runner – możemy skakać lub kucać. Droga jest prosta i pozbawiona zakrętów. Nie da się zmieniać kierunku ruchu, więc w praktyce zajmujemy się głównie przeskakiwaniem lub prześlizgiwaniem się pod przeciwnikami. Nie znaczy to jednak, że rozgrywka jest łatwa - trasy są zapełnione jastrzębiami, kaktusami oraz samochodami.

**Podział prac**

Marek Jędrzejewski:

* tworzenie grafik
* tworzenie dźwięków
* programowanie głównego bohatera
* komunikacja z resztą zespołu związana z poprawną implementacją animacji

Paweł Moniewski

* obsługa gestów sterowania
* zarządzanie stanem i wynikiem gry
* wyświetlanie plansz startowych i końcowych
* implementacja muzyki i dźwięków

Łukasz Wsół

* tworzenie świata i dynamicznego tła
* nałożenie tekstur i animacji na bohatera
* dynamiczne tworzenie przeciwników
* balansowanie grywalnością rozgrywki

**Wymagania**

* menu główne
* dynamicznie generowanie przeszkód na planszy
* kontrola ruchów bohatera w płaszczyźnie pionowej (skok lub ślizg)
* wykorzystanie własnej grafiki
* wykorzystanie dźwięków w grze
* brak błędów krytycznych

**Założenia**

**Mapa:**

Grawitacja - wyznaczenie optymalnej grawitacji odbędzie się po testach

**Główny bohater (Zdzich):**

Wymiary (Grid):

1x2 lub 2x1

Sterowanie:

- górna połowa ekranu obsługuje skok

- dolna połowa ekranu obsługuje ślizg

Ruch:

- statyczny w poziomie

- dynamiczny w pionie

a) skok - zawsze tej samej wysokości

b) ślizg - trwa tak długo, jak długo wykonywany jest gest na ekranie,  
 zamiana z 2x1 na 1x2

**Przeszkody:**

1. Jastrząb:

Wymiary:

1x1

Ruch:

- statyczny

- dynamiczny (opcjonalnie)

Typy kolizji:

lewa, dół

2. Maluch:

Wymiary:

3x2

Ruch:

- statyczny

- dynamiczny

Typy kolizji:

lewa

3. Śmietnik

Wymiary:

1x1

Ruch:

- statyczny

Typy kolizji:

lewa

4. Kaktus

Wymiar:

1x1, 2x1, 1x2

Ruch:

- statyczny

Typy kolizji:

lewa, góra

**Rozdział 2. Technologia**

W projekcie wykorzystany jest silnik graficzny LibGDX wraz z biblioteką fizyki Box2d. Biblioteka jest wieloplatformowa i pozwala dystrybucje gier na wiele systemów operacyjnych - m. in. Android, Windows, Linux, iOS, Mac OS. Wszystko to zapewnione jest dzięki Java Runtime (wirtualna maszyna Javy). Dzięki temu jeden kod może być skompilowany na wszystkie wyżej wymienione platformy.

Biblioteka Box2d napisana w języku C++ została przystosowana do silnika LibGDX i pozwala na otrzymanie szybkich rezultatów, przy założeniu, iż programista ma podstawową wiedzę na temat fizyki klasycznej. Cechą charakterystyczną dla gry będzie fakt, iż wszystkie obiekty na planszy będą tworzone za pomocą prostokątów.

**Licencje:**

LibGDX, LibGDX TexturePacker - Licensed under Apache License, Version 2.0

Synfig Studio, GIMP, Hydrogen, Audacity - Licensed under GPL License.

**Marek Jędrzejewski**

* Synfig Studio
* GIMP
* Hydrogen
* Audacity

**Łukasz Wsół**

* Klasa **Game** - implementuje interface ApplicationListener. Pozwala aplikacji na proste zastosowanie wielu obiektów typu **Screen**.

Metody:

- create() - metoda wołana wtedy, kiedy aplikacja jest tworzona po raz pierwszy.

* Klasa **Screen** - reprezentuje jeden z wielu ekranów aplikacji, np. ustawienia, ekran.

Metody:

- render() - metoda wołana wtedy, kiedy obiekt **Screen** powinien się wyrenderować. Parametr delta odpowiada czasowi w sekundach od ostatniego renderu.

- show() - metoda wołana wtedy, gdy obiekt staje się głównym ekranem dla gry.

**Paweł Moniewski**

* Klasa **Stage** - jest nadrzędną klasą reprezentującą fizyczny świat gry, zawierającą świat i aktorów. Zbiera sygnały wejściowe i informacje o nich może przekazywać elementom podrzędnym.

Metody:

- act() – aktualizuje każdego aktora w obrębie sceny co określony czas,

- touchDown() – reaguje na zdarzenie dotknięcia ekranu i zwraca prawdę, jeśli aktor w danej scenie je obsłużył

- touchUp() – reaguje na zdarzenie zakończenie dotykania ekranu i zwraca prawdę, jeśli aktor w danej scenie je obsłużył

* Klasa **Actor** jest klasą reprezentującą pojedynczy obiekt sceny m.in. poprzez pozycję na ekranie oraz wymiary opisującego go prostokąta.

Metody:

- act() – wywoływana co określony czas w celu aktualizacji aktora.

- draw() - wywoływana w celu narysowania aktora na ekranie.

* Interfejs **ContactListener** deklaruje metody reagujące na zdarzenia kolizji aktorów. Wykorzystuje obiekt Contact zawierający informacje o kolizji.

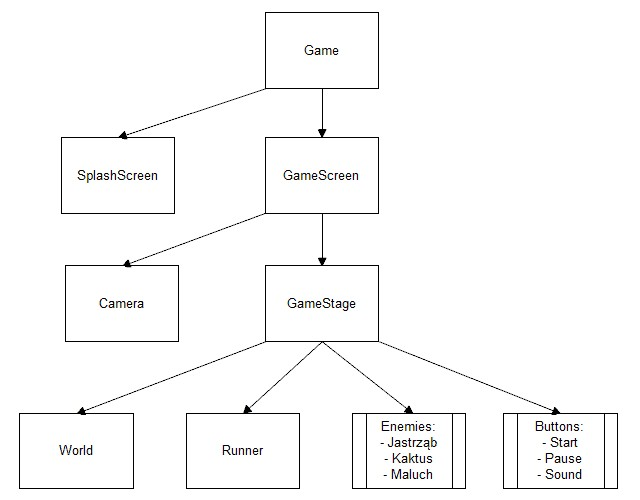
Metody:

- beginContact() – reaguje na początek kolizji,

- endContact() – reaguje na zakończenie kolizji.

* Klasa **Music** - służy do obsługi muzyki w grze. Plik dźwiękowy przy każdej próbie użycia jest odczytywany z pamięci masowej.
* Klasa **Sound** - służy do obsługi dźwięków w grze. Plik dźwiękowy jest ładowany do pamięci RAM.
* **LibGDX TexturePacker** - narzędzie do pakowania wielu obrazków w jedną paczkę składającą się z pliku graficznego i pliku opisu (region atlas).

**Rozdział 3. Projekt gry**



**Rozdział 4. Implementacja**

**Package: com.wmj:**

**ZdzichuRuns**

Klasa główna, dziedzicząca po Game umożliwiająca tworzenie nowych obiektów Screen.

**Package: actors**

Pakiet przechowujący klasy dla obiektów tworzonych na planszy. Obiekty te posiadają pola charakterystyczne do nakładania tekstur, metody do ich rysowania oraz mechanizmy związane z logiką ich działania.

**Background**

Klasa dziedzicząca po Actor, związana z rysowaniem oraz animowaniem tła.

**Enemy**

Klasa dziedzicząca po GameActor, związana z rysowaniem oraz animowaniem przeciwników.

**GameActor**

Klasa abstrakcyjna, która dziedziczy po klasie Actor, którą dziedziczą wszystkie inne klasy z pakietu actors. Została stworzona na potrzeby przechowywania obiektów z pakietu box2d.

**Ground**

Klasa dziedzicząca po GameActor związana z podłożem.

**Label**

Klasa dziedzicząca po Actor związana z wyświetlaniem tekstów w menu gry.

**Runner**

Klasa dziedzicząca po GameActor, związana z głównym bohaterem, nałożeniem na niego tekstury oraz logiki jego działań. Znajdują się w niej metody, które określają logikę skoku lub ślizgania.

**Score**

Klasa dziedzicząca po Actor związana z wyświetlaniem wyniku gracza.

**Tutorial**

Klasa dziedzicząca po Actor związana z wyświetlaniem podpowiedzi odnośnie sterowania.

**Package: box2d**

Pakiet przechowujący klasy charakterystyczne dla biblioteki box2d i jej fizyki potrzebne do określania m. in. szybkość poruszania się przeciwników, czy określania kąta upadku protagonisty.

**EnemyUserData**

Klasa dziedzicząca po UserData, zawierająca prędkość poruszania się przeciwników

**GroundUserData**

Klasa dziedzicząca po UserData, związana z podłożem.

**RunnerUserData**

Klasa dziedzicząca po UserData, posiadająca pola określające pozycje protagonisty podczas skoku, ślizgu oraz wektor określający przyśpieszenie, z jakim wznosi się główny bohater.

**UserData**

Klasa abstrakcyjna posiadająca pola określające wysokość, szerokość oraz typ obiektu.

**Package: enums**

Pakiet przechowujący stworzone typy na potrzeby architektury.

**EnemyType**

enum, w którym określone są typy złożone - przeciwnicy wraz z ich charakterystycznymi parametrami.

**GameState**

enum, w którym określone są typy obiektów podstawowych - stany gry (trwająca, zatrzymana, zakończona).

**UserDataType**

enum, w którym określone są typy obiektów podstawowych - m. in. Ground, Runner, Enemy.

**Package: screens**

Pakiet przechowujący ekrany gry.

**GameScreen**

Klasa dziedzicząca po interfejsie Screen odpowiedzialna za renderowanie sceny.

**SplashScreen**

Klasa dziedzicząca po interfejsie Screen odpowiedzialna za wyświetlanie grafiki początkowej.

**Package: stages**

Pakiet przechowujący sceny gry.

**GameStage**

Klasa dziedzicząca po Stage. Tak zwana scena - to w niej ustawiana jest kamera oraz pola do interakcji użytkownika z bohaterem, a także logika sterowania elementami gry (aktorami, przyciskami).

**Package: utils**

Pakiet przechowujący narzędzia dla innych obiektów.

**AnimParameters**

Kontener do przechowywania właściwości animacji.

**AudioUtils**

Narzędzia do sterowania dźwiękiem w grze. Klasa w oparciu o wzorzec Singleton.

**BodyUtils**

Narzędzia do kontroli kolizji i uzyskiwaniu informacji o rodzaju aktora.

**Constants**

Interfejs przechowujący stałe ustawienia gry, ścieżki do plików, informacje o autorach.

**GameStateManager**

Klasa do kontrolowania stanu gry. Klasa w oparciu o wzorzec Singleton.

**RandomUtils**

Klasa do generowania losowych przeciwników.

**WorldUtils**

Statyczna klasa przechowująca metody do tworzenia obiektów na scenie.

**Ciekawsze implementacje**

**Poruszanie tła:**

Poruszanie się tła polega na zmianie pozycji dwóch prostokątów z nałożoną teksturą. Kiedy jeden osiąga pozycję, w której to znajduje się całkowicie za lewą granicą ekranu, tworzony jest nowy po prawej stronie i analogicznie przesuwany w lewą stronę. Stary prostokąt jest usuwany.

**if** (leftBoundsReached(delta)) {

resetBounds();

} **else** {

updateXBounds(-delta);

}

**private void** updateXBounds(**float** delta) {

**textureRegionBounds1**.**x** += delta \* **speed**;

**textureRegionBounds2**.**x** += delta \* **speed**;

}

**private void** resetBounds() {

**textureRegionBounds1** = **textureRegionBounds2**;

**textureRegionBounds2** = **new** Rectangle(Constants.***APP\_WIDTH***+5, 0, Constants.***APP\_WIDTH***+5, Constants.***APP\_HEIGHT***);

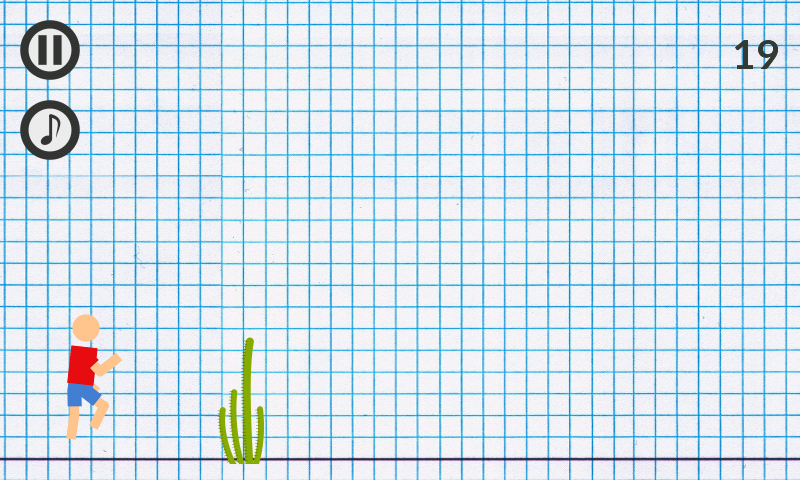
}

**Rozdział 5. Testy**

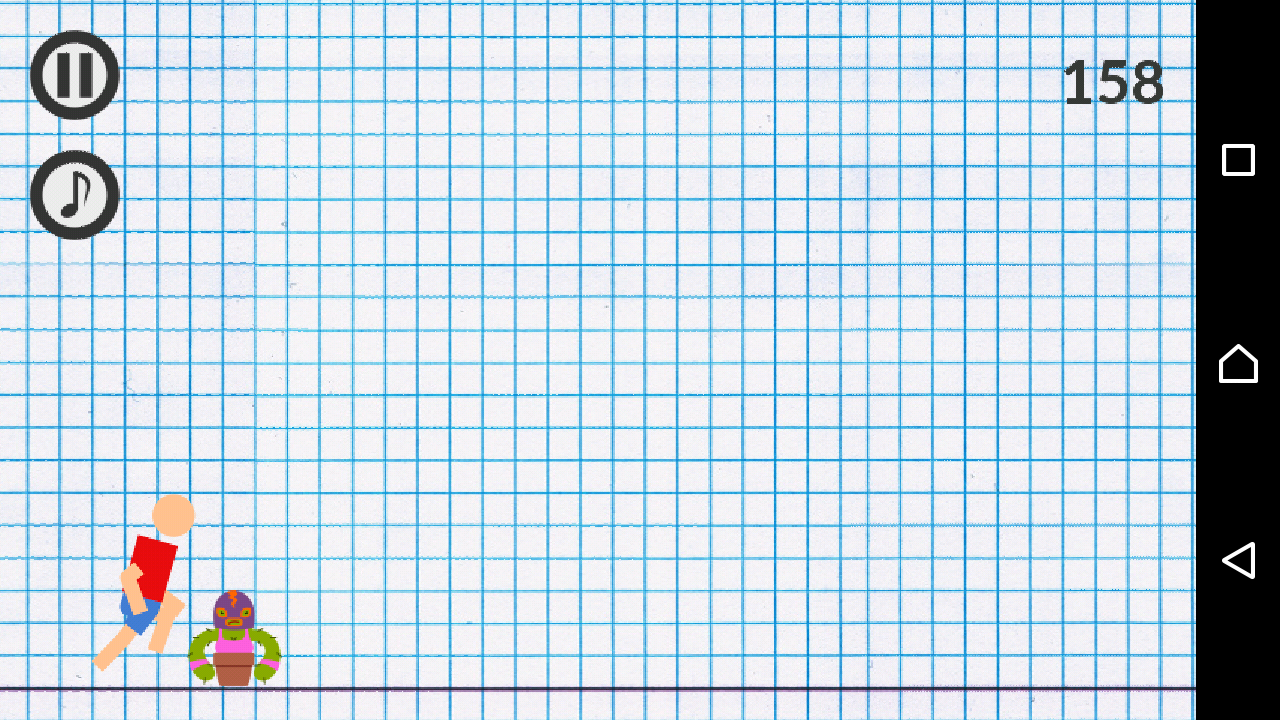
Gra podczas procesu jej tworzenia testowana była głównie w ramach projektu desktopowego, dzięki czemu udało się wyeliminować większość błędów jeszcze przed jej wdrożeniem i uruchomieniem na urządzeniach mobilnych. Niestety na części komputerów członków zespołu występował problem przy restartowaniu gry, który powodował wyłączenie się aplikacji. Problemu tego nie stwierdzono przy testowaniu na urządzeniach mobilnych.

Lista urządzeń, na których testowano grę:

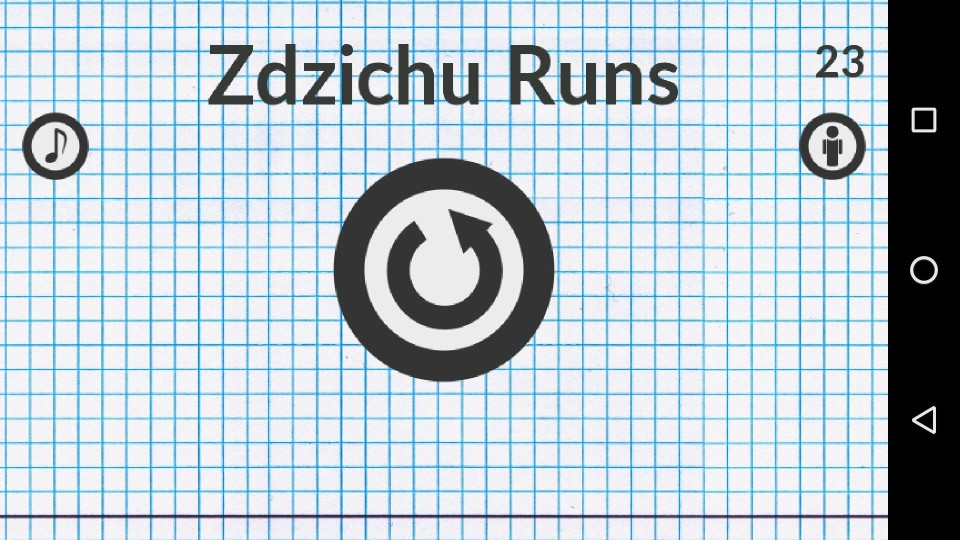
* Samsung Galaxy S (Android 4.4.4)



* Sony Xperia Z1 Compact (Android 5.1.1)



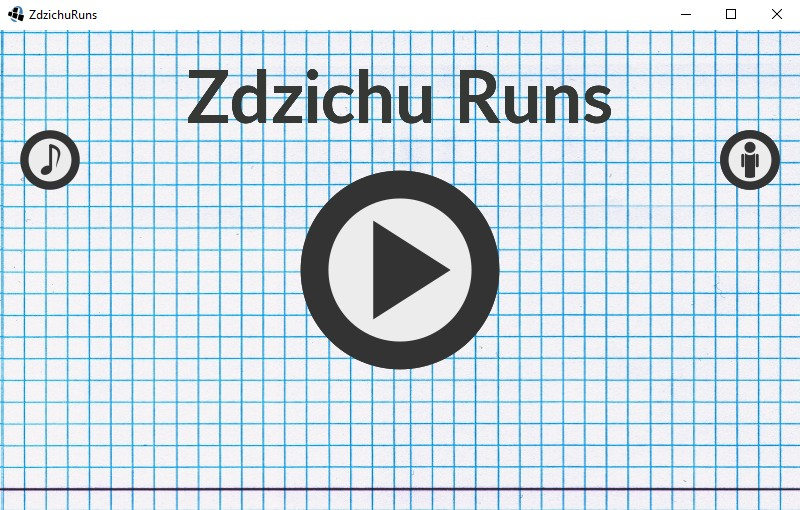
* Motorola Moto X (Android 6.0.1)



* Huawei Honor Holly (Android 4.4.2)
* Samsung Galaxy Note 3 (Android 5.0.0)
* Samsung Galaxy Tab 4 7.0 (Android 5.0.2)

**Rozdział 6. Instrukcja użytkownika**

Po uruchomieniu aplikacji pojawia się plansza powitalna, która znika po 2 sekundach, by ustąpić miejsca głównemu menu.



Po lewej stronie znajduje się przycisk Sound, który służy do kontroli dźwięków w grze.

Po prawej stronie znajduje się przycisk Authors do wyświetlenia informacji o autorach.

Aby rozpocząć rozgrywkę, należy przycisnąć przycisk Start znajdujący się na środku ekranu.

Po uruchomieniu gry należy unikać pojawiających się przeciwników.

Można to wykonywać za pomocą dwóch gestów:

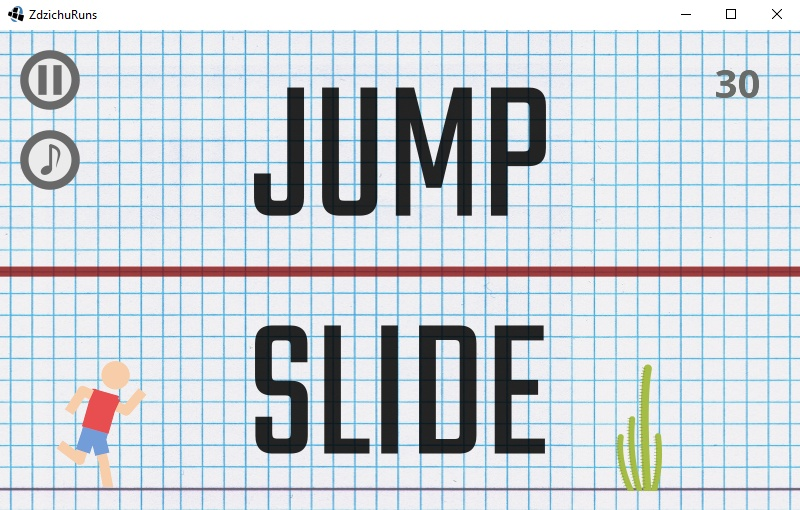
* skoku, poprzez naciśnięcie na górną połowę ekranu,
* ślizgu, poprzez naciśnięcie na dolną połowę ekranu.

W prawej części ekranu znajduje się stale aktualizujący się wynik.

W lewej części ekranu znajduje się przycisk Pause służący do wstrzymywania rozgrywki.

W trakcie pauzy wyświetlana plansza z instrukcjami sterowania, która znika po 4 sekundach.





**Dodatek**

Tabela obiektów graficznych.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Autor (źródło)** | **Nazwa pliku** | **Rozmiar** | **Przeznaczenie** | **Opis** | **Uwagi** |
| Marek Jędrzejewski | *car\_sheet\_96.png* | *2048x96* | Przeciwnik | Jadący samochód |  |
| Marek Jędrzejewski | falcon\_sheet\_64.png | 2048x64 | Przeciwnik | Lecący ptak |  |
| Marek Jędrzejewski | jump\_sheet\_128.png | 2048x128 | Główny bohater | Animacja skoku |  |
| Marek Jędrzejewski | run\_sheet\_128 | 2048x128 | Główny bohater | Animacja biegu |  |
| Marek Jędrzejewski | slide\_sheet\_32 | 2048x32 | Główny bohater | Animacja ślizgu |  |
| Marek Jędrzejewski | notebook\_800x480.jpg | 800x480 | Tło | Zeszyt w kratkę |  |
| Marek Jędrzejewski | about.png | 60x60 | Menu opcji | i z rączkami |  |
| Marek Jędrzejewski | pause50.png | 60x60 | Menu opcji | dwie kreski |  |
| Marek Jędrzejewski | play60.png | 60x60 | Menu opcji | strzałka |  |
| Marek Jędrzejewski | sound\_off.png | 60x60 | Menu opcji | ósemka |  |
| Marek Jędrzejewski | sound\_on.png | 60x60 | Menu opcji | przekreślona ósemka |  |
| Marek Jędrzejewski | play200.png | 200x200 | Menu opcji | strzałka |  |
| Marek Jędrzejewski | restart200.png | 200x200 | Menu opcji | zakręcona strzałka |  |
| Marek Jędrzejewski | splash.png | 800x480 | Uruchamianie gry | Typografia |  |
| Marek Jędrzejewski | tutorial.png | 800x480 | pauza | ekran podzielony z objaśnionym sterowaniem |  |
| Marek Jędrzejewski | cactus\_small1.png | 64x64 | Przeciwnik | kaktus |  |
| Marek Jędrzejewski | cactus\_small2.png | 64x64 | Przeciwnik | kaktus |  |
| Marek Jędrzejewski | cactus\_tall1.png | 64x128 | Przeciwnik | kaktus |  |
| Marek Jędrzejewski | cactus\_tall2.png | 64x128 | Przeciwnik | kaktus |  |
| Marek Jędrzejewski | cactus\_tall3.png | 64x128 | Przeciwnik | kaktus |  |
| Marek Jędrzejewski | garbage\_container\_blue.png | 96x107 | Przeciwnik | kontener na śmieci |  |
| Marek Jędrzejewski | garbage\_container\_green.png | 96x107 | Przeciwnik | kontener na śmieci |  |
| Marek Jędrzejewski | garbage\_container\_pink.png | 96x107 | Przeciwnik | kontener na śmieci |  |
| Marek Jędrzejewski | garbage\_container\_red.png | 96x107 | Przeciwnik | kontener na śmieci |  |
| Marek Jędrzejewski | garbage\_container\_yellow.png | 96x107 | Przeciwnik | kontener na śmieci |  |