**Implementacja gry platformowej**

PROJEKT DYPLOMOWY

Poznań 2020

**DANE PARTNERÓW**

**A1. Dane Opiekuna**

|  |  |
| --- | --- |
| Imię i nazwisko | **Grzegorz Nowak** |
| Stopień / Tytuł naukowy | Doktor |
| Data i podpis |  |

**A2. Dane członków Zespołu projektu**

|  |  |
| --- | --- |
| Imię i nazwisko | **Karol Krasuski** |
| Kierunek studiów | Informatyka |
| Tryb studiów | Niestacjonarny |
| Data i podpis |  |

|  |  |
| --- | --- |
| Imię i nazwisko | **Bartosz Kwiatkowski** |
| Kierunek studiów | Informatyka |
| Tryb studiów | Niestacjonarny |
| Podpis |  |

|  |  |
| --- | --- |
| Imię i nazwisko | **Jakub Łodaj** |
| Kierunek studiów | Informatyka |
| Tryb studiów | Niestacjonarny |
| Data i podpis |  |

|  |  |
| --- | --- |
| Imię i nazwisko | **Jakub Łysiak** |
| Kierunek studiów | Informatyka |
| Tryb studiów | Niestacjonarny |
| Data i podpis |  |

Spis treści

[ZAŁOŻENIA PROJEKTU 4](#_Toc30796521)

[B1. Opis projektu 4](#_Toc30796522)

[B2. Cele projektu 5](#_Toc30796523)

[REALIZACJA 5](#_Toc30796524)

[C1. Zadania w projekcie 5](#_Toc30796525)

[C2. Użyteczność projektu 9](#_Toc30796526)

[C3. Wykorzystane materiały i bibliografia związana z realizacją projektu 10](#_Toc30796527)

[C4. Spis załączników 10](#_Toc30796528)

[ZAŁĄCZNIKI 11](#_Toc30796529)

[1.1. Opis narzędzia GDevelop i jego funkcjonalność 11](#_Toc30796530)

[2.1. Problematyka projektu dyplomowego - Trudności w tworzeniu gier komputerowych 19](#_Toc30796531)

[6.1. Opis scen gry 21](#_Toc30796532)

[6.2. Opis zdarzeń zewnętrznych 29](#_Toc30796533)

[7.1. Proces testowania produktu projektowego 33](#_Toc30796534)

[7.2. Raport błędów i opis sposobu ich rozwiązania 34](#_Toc30796535)

[7.3. Wdrożenie poprawek, raport z pomyślnie przeprowadzonych testów 35](#_Toc30796536)

[8.1. Opis procesu eksportu na platformy użytkowe 38](#_Toc30796537)

# ZAŁOŻENIA PROJEKTU

## B1. Opis projektu

1. Problem badawczy

Problemem w projekcie jest stworzenie grywalnej wersji gry platformowej opartej na silniku wysokopoziomowego narzędzia programistycznego GDevelop. Wymagane jest stworzenie mechaniki samej rozgrywki, animowanej postaci bohatera głównego oraz postaci pobocznych, a także tła będącego jednocześnie grywalnym elementem gry. Dodatkowo problemem jest odpowiedni zarys fabularny przebiegu gry wpleciony w całość rozgrywki.

1. Uzasadnienie wyboru tematu

Gra jako produkt końcowy przede wszystkim ma służyć celom rozrywkowym grających, jednak w przypadku stosowania narzędzia GDevelop sam proces tworzenia gry może stanowić swoistą rozrywkę dla samych tworzących. Wybierając to narzędzie optymalizujemy wykorzystanie czasu skupiając się przede wszystkim na grywalności i głównym wątku gry jednocześnie poświęcając mniej czasu na kwestie techniczne. Duża część technicznych aspektów jest już zaimplementowana, zawarta w silniku gry i gotowa do użycia w dostarczonym przez producenta narzędzia GDevelop. Oprócz tego niewątpliwie pożądaną cechą w dzisiejszym świecie jest wieloplatformowość aplikacji.

1. Zakres podmiotowy, przedmiotowy, czasowy i przestrzenny

* przedmiot: narzędzie programistyczne GDevelop,
* zakres badawczy: implementacja gry platformowej,
* zakres czasowy: projekt realizowany od października 2019 do stycznia 2020.

1. Metodyka pracy (metody i techniki badawcze)

* język programowania: C++ (natywny dla narzędzia GDevelop),
* środowisko programistyczne: GDevelop,
* darmowe oprogramowanie do edycji grafiki wektorowej: GIMP,
* platforma do przechowywania i wersjonowania poszczególnych etapów, projektu: GitHub,
* narzędzie do kontroli cyklu życia projektu: Trello,
* narzędzie do komunikacji grupowej: Facebook Messenger.

## B2. Cele projektu

Celem projektu dyplomowego jest zaprojektowanie grywalnej wersji gry platformowej przy użyciu programistycznego narzędzia wysokopoziomowego GDevelop. Aby osiągnąć tak postawiony cel projektu, określono następujące zadania i cele szczegółowe:

* zaprojektowanie oprawy graficznej i otoczenia dla poszczególnych etapów gry,
* opracowanie fabuły i wiodącego wątku tematycznego w grze,
* zaprojektowanie, animacja głównej i poboczny postaci występujących w grze,
* implementacja zachowań (ang. events) wszystkich aktywnych elementów gry,
* eksport projektu na użytkowe platformy (Web, Mobile, Desktop).

# REALIZACJA

## C1. Zadania w projekcie

**Zadanie 1**

|  |  |
| --- | --- |
| Nazwa zadania | **Przedstawienie problematyki projektu dyplomowego** |
| Podmioty zaangażowane w realizację zadania | **Karol Krasuski, Bartosz Kwiatkowski** |
| Krótki opis zadania | Zadanie polegało na opisie problematyki projektu dyplomowego. Wskazano problemy, ich możliwe przyczyny i sposoby rozwiązania. |
| Efekty realizacji zadania | **Załącznik 1.1. Problematyka projektu dyplomowego - Trudności w tworzeniu gier komputerowych** |
| Termin rozpoczęcia realizacji zadania | Grudzień 2019 |
| Termin zakończenia realizacji zadania | Styczeń 2020 |

**Zadanie 2**

|  |  |
| --- | --- |
| Nazwa zadania | **Opis obszaru badawczego/podmiotu projektu dyplomowego** |
| Podmioty zaangażowane w realizację zadania | **Karol Krasuski, Jakub Łodaj** |
| Krótki opis zadania | Opis narzędzia GDevelop. - Przekrój funkcjonalności |
| Efekty realizacji zadania | **Załącznik 2.1. Opis narzędzia GDevelop i jego funkcjonalność** |
| Termin rozpoczęcia realizacji zadania | Grudzień 2019 |
| Termin zakończenia realizacji zadania | Styczeń 2020 |

**Zadanie 3**

|  |  |
| --- | --- |
| Nazwa zadania | **Rozpoznanie rynku gier platformowych oraz zaproponowanie zarysu fabularnego gry.** |
| Podmioty zaangażowane w realizację zadania | **Jakub Łysiak** |
| Krótki opis zadania | Badanie zapotrzebowania na rodzaj i charakter gry platformowej. Zredagowanie głównego wątku w grze |
| Efekty realizacji zadania | **Załącznik 3.1. Wniosek z rozpoznania**  **Załącznik 3.2. Opis fabuły** |
| Termin rozpoczęcia realizacji zadania | Październik 2019 |
| Termin zakończenia realizacji zadania | Listopad 2019 |

**Zadanie 4**

|  |  |
| --- | --- |
| Nazwa zadania | **Analiza literatury w celu znalezienia sposobów/metod/narzędzi do przygotowania rozwiązania projektu dyplomowego** |
| Podmioty zaangażowane w realizację zadania | **Karol Krasuski, Bartosz Kwiatkowski, Jakub Łodaj, Jakub Łysiak** |
| Krótki opis zadania | Zadanie polegało na odnalezieniu sposobów rozwiązania problemów oraz metod programistycznych w celu stworzenia aplikacji. Wykorzystano publikacje dostępne w literaturze oraz Internecie. |
| Efekty realizacji zadania | **Załącznik 4.1. Baza wiedzy GDevelop - oraz witryny tematyczne - przekrój znalezionych informacji** |
| Termin rozpoczęcia realizacji zadania | Październik 2019 |
| Termin zakończenia realizacji zadania | Grudzień 2019 |

**Zadanie 5**

|  |  |
| --- | --- |
| Nazwa zadania | **Zaprojektowanie szaty graficznej dla ostatecznej wersji gry** |
| Podmioty zaangażowane w realizację zadania | **Karol Krasuski, Bartosz Kwiatkowski, Jakub Łodaj, Jakub Łysiak** |
| Krótki opis zadania | Zadanie zawiera utworzenie projektu ostatecznej wersji oprawy graficznej gry (w tym bohaterów i ich animacje) |
| Efekty realizacji zadania | **Załącznik 5.1. Argumentacja i przedstawienie podstawowej szaty graficznej gry** |
| Termin rozpoczęcia realizacji zadania | Październik 2019 |
| Termin zakończenia realizacji zadania | Grudzień 2019 |

**Zadanie 6**

|  |  |
| --- | --- |
| Nazwa zadania | **Dostrojenie silnika gry oraz implementacja pojedynczych zdarzeń pod potrzeby projektu** |
| Podmioty zaangażowane w realizację zadania | **Karol Krasuski, Bartosz Kwiatkowski, Jakub Łodaj, Jakub Łysiak** |
| Krótki opis zadania | Zadanie polega na edycji silnika gry w taki sposób, aby odpowiednio reagował na konkretne zdarzenia podczas rozgrywki |
| Efekty realizacji zadania | **Załącznik 6.1. Opis scen zawartych w grze**  **Załącznik 6.2. Opis zdarzeń zewnętrznych zawartych w grze** |
| Termin rozpoczęcia realizacji zadania | Październik 2019 |
| Termin zakończenia realizacji zadania | Styczeń 2020 |

**Zadanie 7**

|  |  |
| --- | --- |
| Nazwa zadania | **Testy grywalnej części gry, korekcja błędów** |
| Podmioty zaangażowane w realizację zadania | **Karol Krasuski, Bartosz Kwiatkowski, Jakub Łodaj, Jakub Łysiak** |
| Krótki opis zadania | Testowanie gry pod kątem potencjalnych błędów, raportowanie znalezionych błędów, zaproponowanie i podjęcie próby naprawy napotkanych błędów. Ponowne testy w oparciu o zastosowane rozwiązania |
| Efekty realizacji zadania | **Załącznik 7.1. Proces testowania produktu projektowego**  **Załącznik 7.2. Raport błędów i opis sposobu ich rozwiązania**  **Załącznik 7.3. Wdrożenie poprawek, raport z pomyślnie przeprowadzonych testów** |
| Termin rozpoczęcia realizacji zadania | Październik 2019 |
| Termin zakończenia realizacji zadania | Styczeń 2020 |

**Zadanie 8**

|  |  |
| --- | --- |
| Nazwa zadania | **Eksport ostatecznego projektu na konkretne platformy użytkowe** |
| Podmioty zaangażowane w realizację zadania | **Bartosz Kwiatkowski, Jakub Łodaj** |
| Krótki opis zadania | Wyeksportowanie pliku projektowego na platformy Web, Desktop Mobile |
| Efekty realizacji zadania | **Załącznik 8.1. Opis procesu eksportu na platformy użytkowe**  **Gotowy produkt projektowy w postaci wykony-walnych aplikacji lub aplikacji webowej** |
| Termin rozpoczęcia realizacji zadania | Styczeń 2020 |
| Termin zakończenia realizacji zadania | Styczeń 2020 |

**Zadanie 9**

|  |  |
| --- | --- |
| Nazwa zadania | **Przygotowanie dokumentacji końcowej i prezentacji multimedialnej projektu dyplomowego** |
| Podmioty zaangażowane w realizację zadania | **Karol Krasuski, Bartosz Kwiatkowski, Jakub Łodaj, Jakub Łysiak** |
| Krótki opis zadania | Zadanie polegało na przedstawieniu grywalnej części gry, przykładowym jej działaniu oraz omówieniu rozgrywki. |
| Efekty realizacji zadania | **Prezentacja multimedialna i dokumentacja końcowa projektu dyplomowego** |
| Termin rozpoczęcia realizacji zadania | Styczeń 2020 |
| Termin zakończenia realizacji zadania | Luty 2020 |

## C2. Użyteczność projektu

Użyteczność projektu przygotowanego przez grupę polega na przedstawieniu możliwości narzędzia programistycznego GDevelop na konkretnym przykładzie tzn.: funkcjonalnej i przystępnej użytkownikowi gry platformowej. Projekt wykazuję również, że bez poświęcania stosunkowo dużej ilości czasu można stworzyć mechanikę gry podobną do projektów, które były tworzone od początku przy użyciu niskopoziomowych narzędzi. Tym samym projekt udowadnia, że skupiając się na cechach kluczowych gry tj. fabule i grywalności, możemy uzyskać znacznie lepsze rezultaty końcowe.

Kwintesencją projektu dyplomowego jest wypuszczenie gotowego produktu tj. gry komputerowej dostępnej na najpopularniejsze platformy. Głównym aspektem użyteczności jest walor rozrywkowy, który jest niewątpliwie jednym z ważniejszych celów tworzenia gier komputerowych. Oprócz tego kod źródłowy projektu może służyć również celom edukacyjnym jako wzorzec projektowy.

Gotowy produkt będzie gotowy do publikacji na największych platformach growych np. (Steam, Google Play). Natomiast kod źródłowy został opublikowany na platformie GitHub jako projekt wzorcowy dla przyszłych produkcji i będzie dostępny dla wszystkich zainteresowanych pod adresem https://github.com/kwiat323/Praca\_Inzynierska.

## 

## C3. Wykorzystane materiały i bibliografia związana z realizacją projektu

1. *Player*, *arrow* - https://www.gamedevmarket.net/asset/fantasy-heroes-character-sprite-sheet-10156/ [dostęp 14.12.2019].
2. *Slime* - https://bevouliin.com/free-2d-game-asset-pink-monster-game-character/ [dostęp 17.01.2020].
3. *Spider* - https://bevouliin.com/furry-monster-free-2d-game-asset/ [dostęp 16.01.2020].
4. *Slime2* - https://opengameart.org/content/bevouliin-free-orange-bubble-land-monster-sprite-sheets [dostęp 16.01.2020].
5. *Fly*, *hearth*, *coin* - https://kenney.nl/assets/platformer-pack-redux [dostęp 20.12.2019].
6. *AddHearth*, *AddCoin -* https://craftpix.net/freebies/free-game-coins-sprite-sheets/ [dostęp 18.12.2019].
7. *Shuriken* - https://opengameart.org/content/shuriken-pixel-art [dostęp 17.01.2020].
8. *Boss1,Boss2,Boss3* - https://craftpix.net/freebies/free-fallen-angel-chibi-2d-game-sprites/ [dostęp 24.12.2019].
9. *Spike* - https://gamedeveloperstudio.itch.io/floor-spikes [dostęp 19.12.2019]
10. *Ledge* - https://www.kenney.nl/assets/platformer-art-extended-tileset [dostęp 20.12.2019]

## C4. Spis załączników

Załącznik 1.1. Opis narzędzia GDevelop i jego funkcjonalność

Załącznik 2.1. Problematyka projektu dyplomowego - Trudności w tworzeniu gier komputerowych

Załącznik 3.1. Wniosek z rozpoznania

Załącznik 3.2. Opis fabuły

Załącznik 4.1. Baza wiedzy GDevelop - oraz witryny tematyczne - przekrój znalezionych informacji

Załącznik 5.1. Propozycja podstawowej szaty graficznej

Załącznik 5.2. Odmienne warianty pierwotnego projektu

Załącznik 5.3. Wybór docelowej szaty graficznej - proces wyboru - argumentacja Załącznik 6.1. Propozycja scenariusza mechaniki i przebiegu gry

Załącznik 6.2. Modyfikacja oraz uzgodnienie finalnej wersji mechaniki

Załącznik 6.3. Wdrożenie zaplanowanych zmian w projekt gry

Załącznik 6.4. Test działania wprowadzanych zmian - raport z wykonanego testu

Załącznik 7.1. Proces testowania produktu projektowego

Załącznik 7.2. Raport błędów i opis sposobu ich rozwiązania

Załącznik 7.3. Wdrożenie poprawek, raport z pomyślnie przeprowadzonych testów

Załącznik 8.1. Opis procesu eksportu projektu na docelowe platformy użytkowe

# ZAŁĄCZNIKI

## 1.1. Opis narzędzia GDevelop i jego funkcjonalność

1. Wstęp

GDevelop to oprogramowanie pozwalające na tworzenie wszelkiego rodzaju gier 2D bez użycia języka programowania. Program ten szczególnie zasługuje na uwagę, ponieważ jest przystępny dla poczatkujących deweloperów gier. Projektowanie gier odbywa się za sprawą intuicyjnych i modułowych elementów mechaniki gry tj.: warunków i akcji w postaci bloków nazywanych zdarzeniami. Program jest prawie całkowicie darmowy, a gotowe gry, które eksportujemy są w pełni kompatybilne z systemami: Windows, Linux, Android, iOS oraz platformy Internetowe (na silniku HTML5). Gotowe produkty które stworzyliśmy, w świetle prawa należą całkowicie do użytkownika, który to ma jednocześnie prawo do komercyjnego korzystania z niego bez opłat licencyjnych. Jedynym znaczącym ograniczaniem w darmowej wersji programu jest limitowana ilość kompilacji projektu które można wykonać na daną platformę w ciągu dnia. Silnik i kod IDE, które składają się na środowisko GDevelop, są wolnym oprogramowaniem na licencji typu Open Source, każdy, kto chce współtworzyć środowisko GDevelop, może to zrobić w dowolnym momencie. Kod źródłowy narzędzia jest dostępny w oficjalnym repozytorium na platformie GitHub pod adresem: [*https://github.com/4ian/GDevelop*](https://github.com/4ian/GDevelop).

2. Opis interfejsu okna głównego

**Rysunek 1 Główne okno programu**

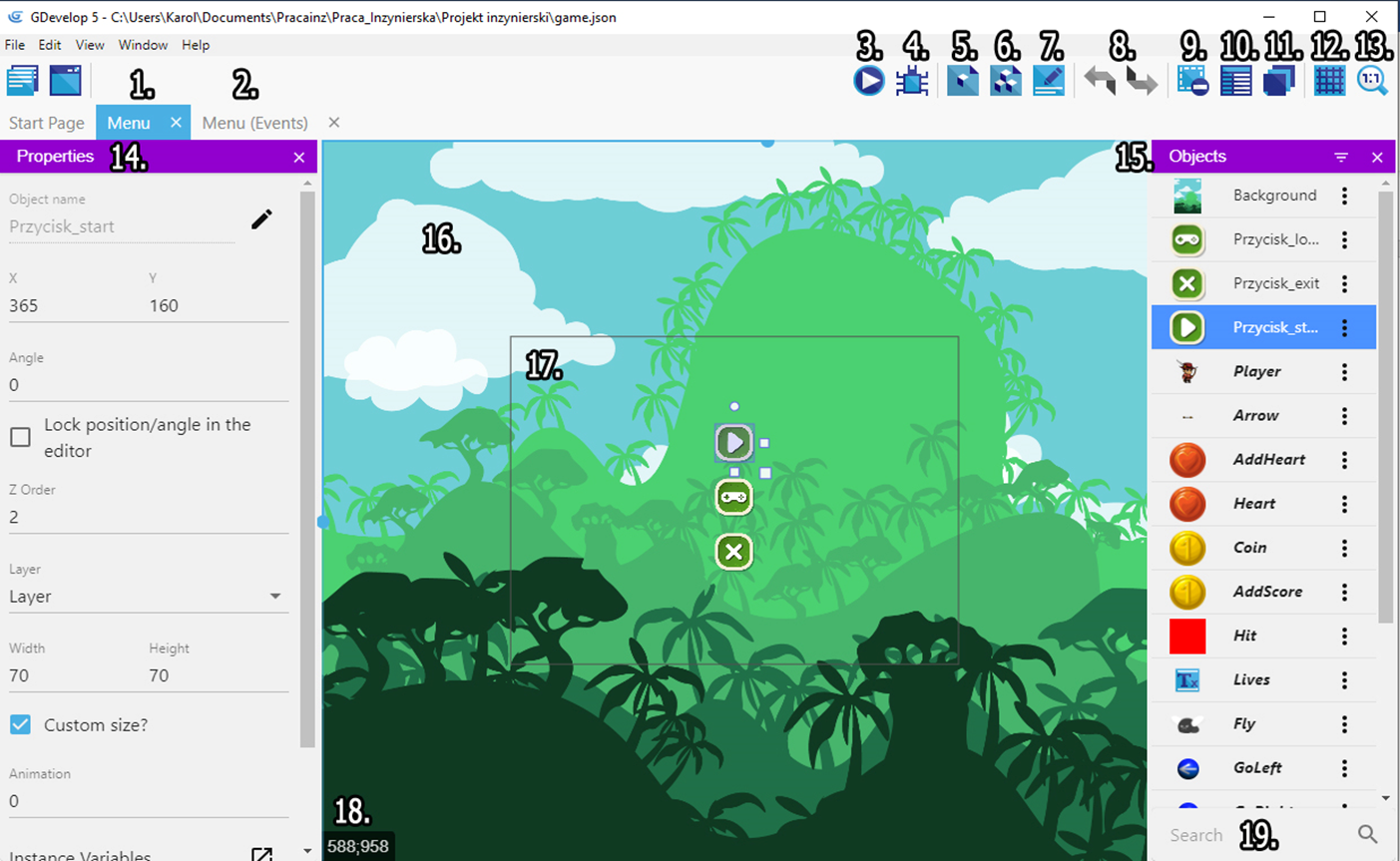


Źródło: Opracowanie własne

1. Pasek nawigacyjny składający się z pięciu rozwijanych list:
   * File: W jego zakładkach znajdują się najważniejsze opcje programu np. takie jak tworzenie nowego projektu, otwarcie istniejącego lub export projektu na konkretne platformy,
   * Edit: narzędzia pozwalające się cofnąć i przywrócić dokonane wcześniej zmiany oraz wklejać lub wycinać elementy projektu,
   * View: Zakładka z widokami uruchamiające poszczególne elementy interfejsu tj. menadżera projektu, menadżera scen, itp.,
   * Window: Pozwala zminimalizować okno programu,
   * Help: Zakładka zawierająca odnośniki do dokumentacji oraz witryn internetowych poświęconych środowisku Ikony skrótu.
2. Ikony skrótu - Odnośniki do kluczowych elementów programu tj.: menadżera projektu, kompilatora i strony głównej programu.
3. Odnośniki do poradników i forum - Odnośniki do witryn internetowych poświęconych narzędziu GDevelop jego dokumentacji (GDevelop Wiki) i forum dyskusyjnym poruszającym kwestie techniczne projektów.
4. Menadżer projektu - Przejście do widoku szkieletowego projektu z podziałem na poszczególne sceny.
5. Odnośniki do mediów społecznościowych / Zmiany języka - Łącza do mediów społecznościowych, gdzie znajdują się profile tematyczne poświęcone programowi GDevelop, oraz aktualnościami z nim związanymi. Dodatkowo obok znajduję się możliwość wyboru języka, chociaż zalecaną praktyką jest korzystanie z natywnej, anglojęzycznej wersji, gdyż cała dokumentacja jest właśnie w tym języku i nie wszystkie zwroty zostały przetłumaczone na inne języki.

3. Opis widoku sceny

**Rysunek 2 Widok sceny**

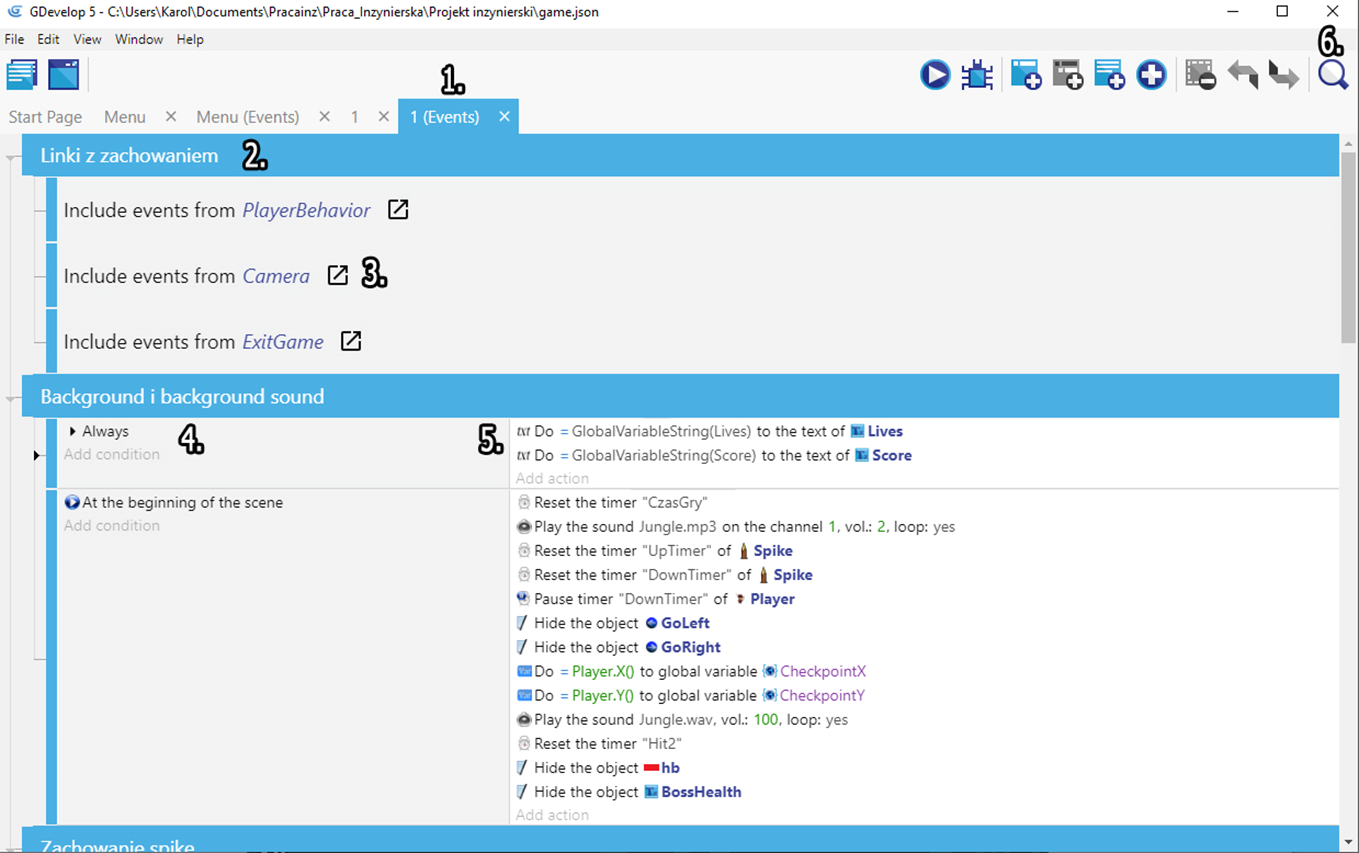


Źródło: Opracowanie własne

1. Okno główne sceny - przejście do widoku, który umożliwia edycje wizualną sceny, dodawanie obiektu etc. (rysunek 2).
2. Okno zdarzeń sceny - przejście do widoku, który umożliwia zarządzanie zdarzeniami w ramach danej sceny (rysunek 3).
3. Symulacja sceny - narzędzie umożliwiający skompilowanie bieżącej sceny w celu zasymulowania widoku sceny i mechaniki gry.
4. Debugger - narzędzie do dynamicznej analizy procedury zdarzeń i błędów w kodzie.
5. Edytor Obiektów - skrót do narzędzia pozwalającego zarządzać obiektami globalnymi oraz tymi przypisanymi do sceny (15).
6. Edytor grup obiektów - skrót do menadżera grup obiektów, umożliwiającego łączenie obiektów w grupy.
7. Edytor właściwości projektu - skrót do edytora właściwości obiektu.
8. Cofnij/Powtórz - pozwala na cofnięcie zmian dokonanych w ramach danej sceny (o jeden krok w przód lub w tył).
9. Usuń instancje obiektu ze sceny (skrót klawiszowy: delete) - usuwa instancję obiektu po ówczesnym zaznaczeniu.
10. Lista instancji - wyświetla edytor instancji obiektów z listą obiektów w bieżącej scenie.
11. Edytor warstw - wyświetla edytor warstw pozwalający na dodanie, usunięcie i manipulację warstwami.
12. Edycja widoku / widok siatki - pozwala ukryć siatkę sceny oraz ramkę maski sceny.
13. Powiększenie - pozwala manipulować powiększeniem podglądu bieżącej sceny.
14. Zakładka właściwości - pozwala edytować parametry takie jak: wielkość położenie, warstwa czy zmienne instancji, istniejących już instancji obiektów.
15. Zakładka obiekty - pozwala dodawać i edytować obiekty, czyli główne elementy projektu.
16. Podgląd sceny - okno ilustrujące aktualny wygląd scenę i rozłożenie obiektów względem współrzędnych.
17. Podgląd maski - wykadrowana scena, która będzie widoczna w oknie gry.
18. Współrzędne kursora - Pozwalają zidentyfikować konkretny punkt na siatce sceny.
19. Wyszukiwarka obiektów - Narzędzie przeszukujące istniejące w projekcie obiekty po nazwie (Szczególnie przydatne przy większej ilości obiektów).

4. Opis widoku zdarzeń

**Rysunek 3 Okno widoku zdarzeń**

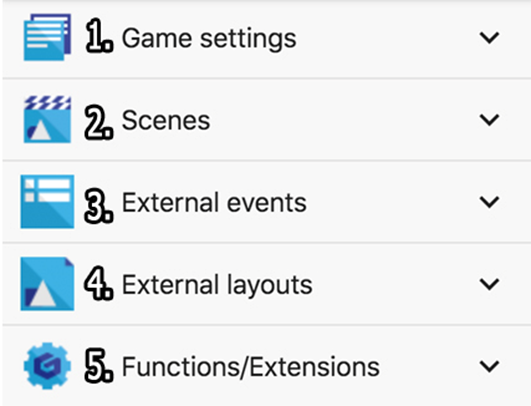


Źródło: Opracowanie własne

1. Okno zdarzeń sceny - przejście do widoku, który umożliwia zarządzanie zdarzeniami w ramach danej sceny (rysunek 3).
2. Grupy zdarzeń - element hierarchii porządkujący zdarzenia w scenie. Może zawierać zdarzenia, które również posiadają już jakąś hierarchie. Można go przywoływać również w innych scenach korzystając np. odnośników (3).
3. Odnośniki do zdarzeń - rodzaj zdarzenia który może dziedziczyć warunki i akcje już z istniejących grup zdarzeń lub całych scen.
4. Warunki - elementarna jednostka organizacyjna w sekcji zdarzenia, gdy warunek jej porządkowy zostanie spełniony wykonuje zadaną w lewej sekcji akcję (5).
5. Akcje - konkretne polecenia wykonane po spełnieniu jednego lub więcej warunków. Mogą wpływać na wszystkie elementy w grze.
6. Wyszukiwarka sekcji zdarzeń - narzędzie wyszukujące po nazwę w ramach sekcji zdarzenia (Szczególnie przydatne przy skomplikowanej hierarchii i większej ilości zdarzeń).

5. Menadżer projektu

**Rysunek 4 Menadżer projektu GDevelop**



Źródło: Opracowanie własne

1. Ustawienia gry - zakładka, w której można dostosować globalne ustawienia gry. Na przykład: domyślny rozmiar okna gry lub jej nazwa. Dodatkowo w tym miejscu można ustawić ikony dla swojej gry. Wszystkie te ustawienia możesz zobaczyć w zakładce „Właściwości”.
2. Sceny - to różne widoki i poziomy w grze. W tym miejscu można wybrać bieżącą scenę, zarządzać istniejącymi lub tworzyć nowe sceny.
3. Wydarzenia zewnętrzne – w celu ułatwienia i kontrolowania rozwoju gry, można stworzyć wydarzenia w oddzielnych arkuszach wydarzeń, które nie są powiązane z określoną sceną. Nazywa się je zdarzeniami zewnętrznymi. Stworzone wydarzenia możemy połączyć z obiektami zdarzeń, kiedy ich potrzebujemy wykorzystując odnośniki zdarzeń Rysunek 3 (podpunkt nr. 3). Jeśli musimy często korzystać z kolekcji wydarzeń w różnych miejscach, możemy ustawić je jako zewnętrzne i wywoływać je ze scen zamiast wielokrotnie dodawać te same zdarzenia pozbywając się w ten sposób dwóch z tzw. *Code Smells* [[1]](#footnote-1)(*Duplicated code* [[2]](#footnote-2) oraz *Shotgun surgery*[[3]](#footnote-3)).
4. Układy zewnętrzne - podobnie jak zdarzenia zewnętrzne, możesz tworzyć generyczne poziomy i ponownie wykorzystywać część poziomów w układach zewnętrznych, a następnie ładować je do swoich scen ponownie unikając Code Smells.
5. Funkcje / rozszerzenia pozwalają tworzyć własne wydarzenia i łączyć je w funkcje dla jeszcze lepszej przejrzystości kodu czyniąc wydarzenia bardziej modułowymi.

## 2.1. Problematyka projektu dyplomowego - Trudności w tworzeniu gier komputerowych

Problemem badawczym w projekcie jest zgłębienie procesu implementacji gry platformowej. Zadanie to składa się z kilku etapów opisanych poniżej.

Pierwsza, a zarazem początkowa faza projektu powinna rozpocząć się od analizy dostępnych na rynku gier i potrzeb użytkowników. Na tym etapie należy również określić zakres i cel projektu, kierując się przede wszystkim użytecznością i zapotrzebowaniem tego typu produktów na rynku.

Następnym krokiem jest pochylenie się nad zagadnieniami teoretycznymi tj.: zdefiniowanie parametrów takich jak:

* zakres projektu,
* typ i kategoria gry,
* czas i miejsce gry,
* zaproponowanie ogólnego zarysu fabularnego.

Kolejnymi, a zarazem najtrudniejszym w realizacji elementami projektu jest faza projektowania i faza implementacji.

Pierwsza z nich: faza projektowania — polega w dużej mierze na przeszukaniu obecnego rynku developerskiego, dobraniu i przygotowaniu przydatnych w projekcie narzędzi programistycznych. Na tym etapie należałoby dopasować i wybrać najlepsze możliwe pomoce programistyczne a przede wszystkim:

* środowisko deweloperskie,
* narzędzie kontrolujące fazy życia projektu,
* platformę do przechowywania i wersjonowania poszczególnych etapów projektu,
* narzędzia do formalnej formy komunikacji (w przypadku projektów grupowych lub sponsorowanych).

Po skompletowaniu poszczególnych narzędzi projektu należy przystąpić do właściwego procesu tworzenia projektu gry.

Problemy, które można napotkać w trakcie tworzenia to w dużej mierze implementacja procesu związana z mechanika gry na przykład:

* tworzenie i konfiguracja parametrów zdarzeń (ang. events) odpowiadających za czynności postaci w zależności od zaistniałej sytuacji,
* opracowanie postaci, jej animacji, a także zachowań w otoczeniu poszczególnych elementów gry.

Oprócz tego niezwykle ważnym dla efektu końcowego aspektem jest zaprojektowanie odpowiedniej szaty graficznej, wliczając w to wszystkie dostępne dla graczy wizualne elementy gry.

Dodatkowo w momencie samego procesu tworzenia, a także po jego ukończeniu należy przeprowadzić odpowiednie dla każdego etapu testy produkcyjne. Testy te powinny zakończyć się odpowiednim raportem opisującym ich wynik, a także w razie jakichkolwiek błędów możliwe propozycje ich rozwiązania.

Końcowym etapami są:

* Eksport pliku projektowego na konkretne platformy użytkowe,
* Przeprowadzenie po raz kolejny testów wykonanych już na docelowych platformach oraz ewentualna naprawa wykrytych błędów,
* Opracowanie dokumentacji projektowej.

Opracowanie dokumentacji projektowej powinno odbyć w taki sposób, aby osoby, niezwiązane z projektem w przyszłości miały możliwość rozbudowania i kontynuacji projektu.

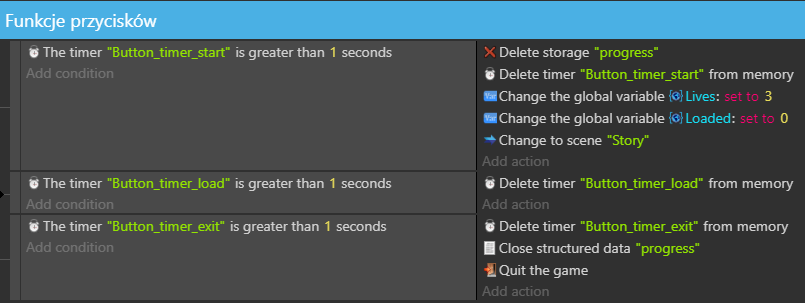
## 6.1. Opis scen gry

Projekt składa się z trzech w pełni grywalnych poziomów oraz sześciu scen które uzupełniają rozgrywkę. Sceny są podzielone na opisane sekcje z zaimplementowaną logiką gry. Poziom grywalny jest to część w której gracz ma możliwość sterowania głównym bohaterem gry i wykorzystywania utworzonych mechanik rozgrywki. Grywalne poziomy wykorzystują zdarzenia zewnętrze, które zostaną opisane w kolejnym załączniku. Sceny nie grywalne są to plansze w których m.in. gracz jest informowany o przebiegu rozgrywki, uzyskuje możliwość rozpoczęcia, wyjścia, zapisu oraz wczytania poprzednio zapisanego stanu rozgrywki. Projekt *Armin Legends* zawiera sceny:

1. Nie grywalne:
   1. *Menu*,
   2. *Story*,
   3. *Info*,
   4. *MenuInGame*,
   5. *GameOver*,
   6. *Win*.
2. Grywalne:
   1. *1* - pierwszy poziom rozgrywki,
   2. *2* - drugi poziom rozgrywki,
   3. *3* - ostatni poziom rozgrywki.

Po uruchomieniu gry jako pierwsza zostaje wyświetlona scena *Menu*. Pozwala na rozpoczęcie nowej rozgrywki, wczytanie ostatniego zapisanego stanu gry lub wyjście.





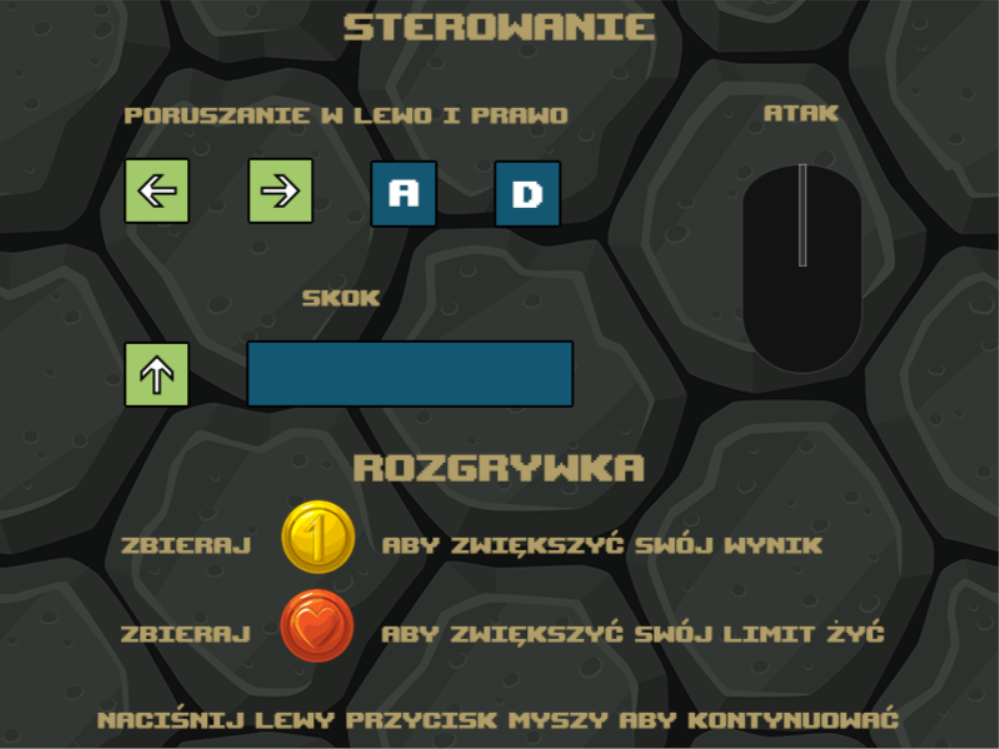
Logika planszy składa się z trzech sekcji opisanych poniżej:

* *Ustawienia i resetowanie opóźnienia działania przycisków* - uruchamia ciągłą zmianę przeźroczystości dla przycisków dostępnych na planszy. Po załadowaniu sceny zostają uruchomione czasomierze przypisane do przycisków. Odtworzona zostaje ścieżka dźwiękowa oraz wczytuje się magazyn danych potrzebny do uzyskania informacji o stanie rozgrywki,
* *Działanie przycisków* - zaprogramowano reakcje przycisków na kolizję z kursorem i ich działanie po naciśnięciu lewego przycisku myszy,
* *Funkcje przycisków* - sekcja ta opisuje reakcje gry na naciśnięcie danego przycisku.

Następna scena zawarta w projekcie to *Story* na której zostaje przedstawiona graczowi fabuła gry. Plansza wyświetlana jest tylko w przypadku rozpoczęcia gry od nowa. Scena składa się z trzech sekcji.

* *Po załadowaniu sceny* - wyświetlona zostaje fabuła gry, mająca na celu zaciekawienie gracza i wprowadzenie do rozgrywki,
* *Efekt piszącego się tekstu* - zaimplementowano funkcję imitującą pisane tekstu w czasie rzeczywistym,
* *Przejście do następnej sceny* - po spełnieniu ustalonego warunku realizowane jest przejście do następnej sceny.

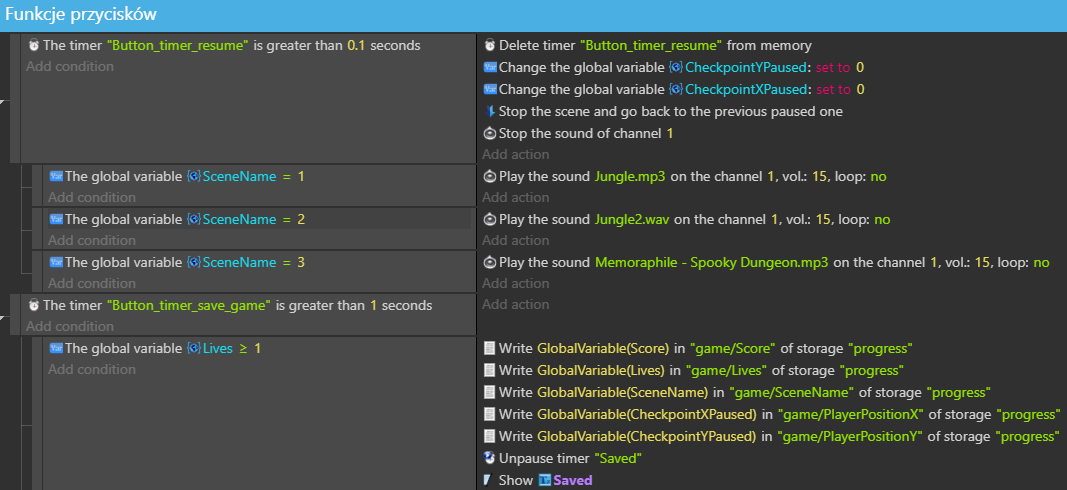
Kolejnym elementem gry jest *Info*. Scena opisuje graczowi sposób sterowania głównym bohaterem gry oraz przedstawia najważniejsze aspekty rozgrywki.



Scena składa się z trzech sekcji:

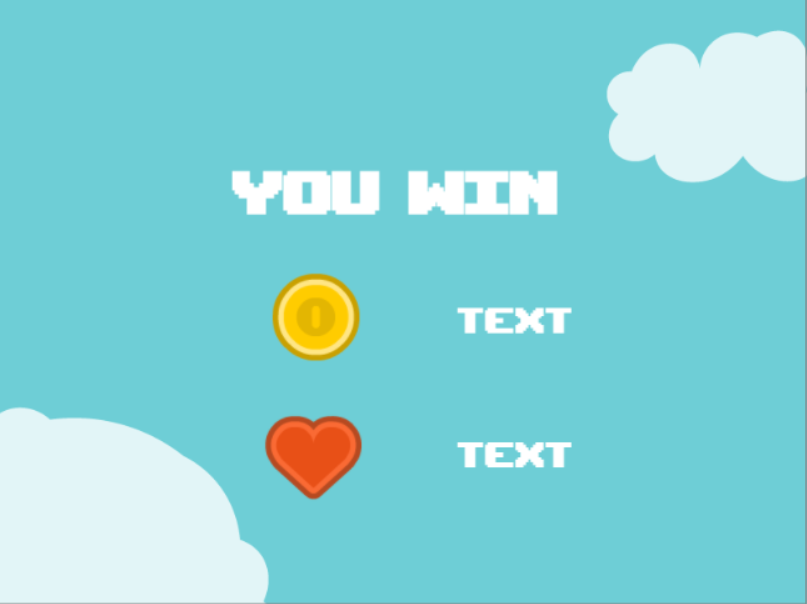
* *Odnośniki do zdarzeń zewnętrznych* - dołączone zostały zdarzenia zewnętrzne których opis znajduje się w załączniku 6.2.,
* *Po załadowaniu sceny* - wyzwolenie czasomierza odliczającego czas który musi upłynąć aby możliwe było przejście do kolejnej sceny,
* *Przejście do następnej sceny* - po spełnieniu warunków gra przenosi rozgrywkę do sceny *1*.

Plansza *MenuInGame* wyzwalana jest podczas trwania rozgrywki gdy gracz naciśnie przycisk *ESC* na klawiaturze. Pozwala ona na zapis stanu gry, przejście do planszy *Menu* lub całkowite zamknięcie gry.



Sposób zaimplementowania logiki planszy opiera się na scenie *Menu*. Dodatkowymi elementami są zdarzenia odpowiedzialne za powrót do wstrzymanej rozgrywki umieszczone w sekcji *Powrót do poprzedniej - zatrzymanej sceny* oraz obsługa zapisu stanu gry umieszczone w sekcji *Funkcje przycisków*

Plansze *Win* i *GameOver* wyświetlają się odpowiednio w sytuacji wygrania gry lub porażki.



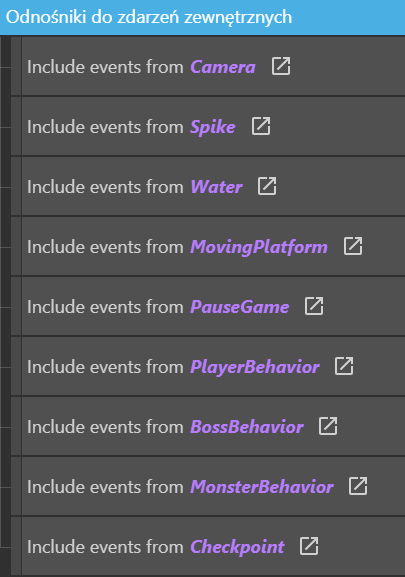
*Win* wyświetla uzyskany wynik, ilość pozostałych żyć, resetuje zmienne globalne do ustawień początkowych oraz przechodzi do *Menu*.

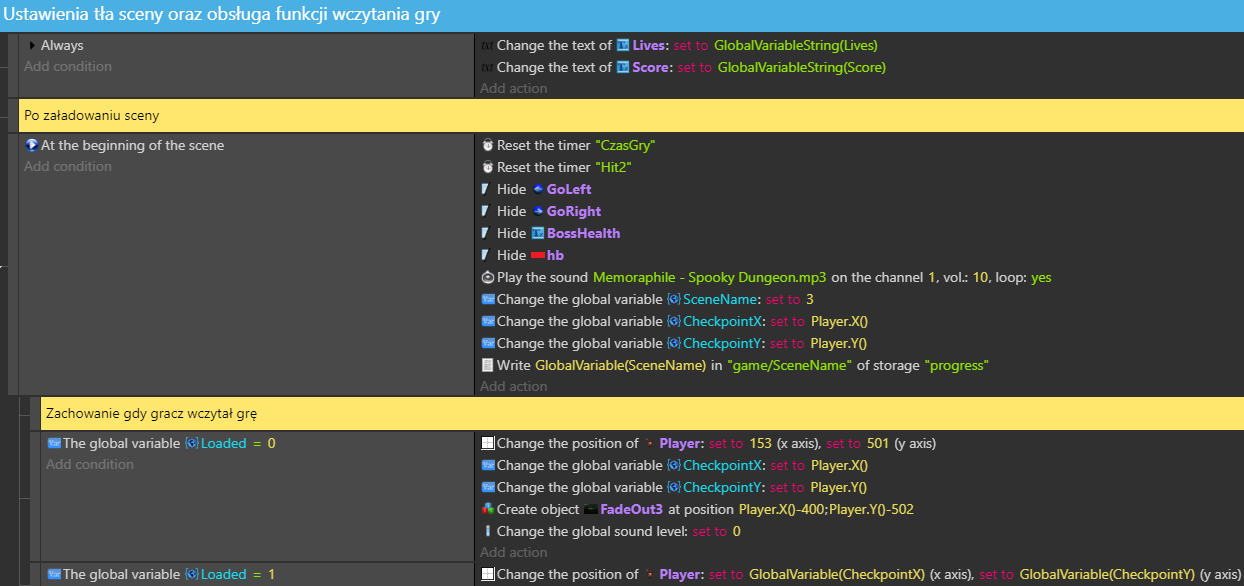
*GameOver* wyświetla uzyskany wynik w ciągu całej rozgrywki do momentu utracenia wszystkich żyć oraz przechodzi do *Menu*.

Wszystkie plansze scen grywalnych składają się z trzech podobnych do siebie sekcji. Różnice wynikają jedynie z nazewnictwa poszczególnych obiektów umieszonych w zasobach. Scena *3* posiada dodatkową sekcję obsługującą specjalne wydarzenia związane z występującym jedynie w niej pojedynku bohatera z głównym antagonistą, którego następstwem jest koniec gry. Przykładowy wygląd planszy grywalnej pokazano na Rysunek nr.

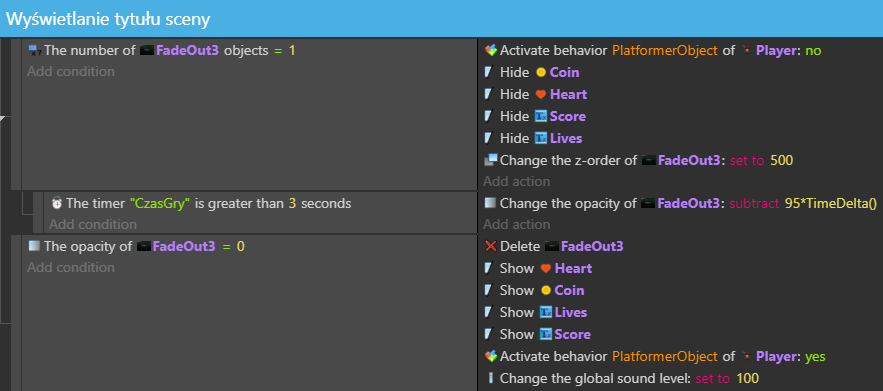


Logika plansz grywalnych opiera się na odnośnikach do zdarzeń zewnętrznych, akcji wykonywanych przez grę po załadowaniu sceny oraz funkcji zmiany sceny na kolejną po spełnieniu określonych warunków.



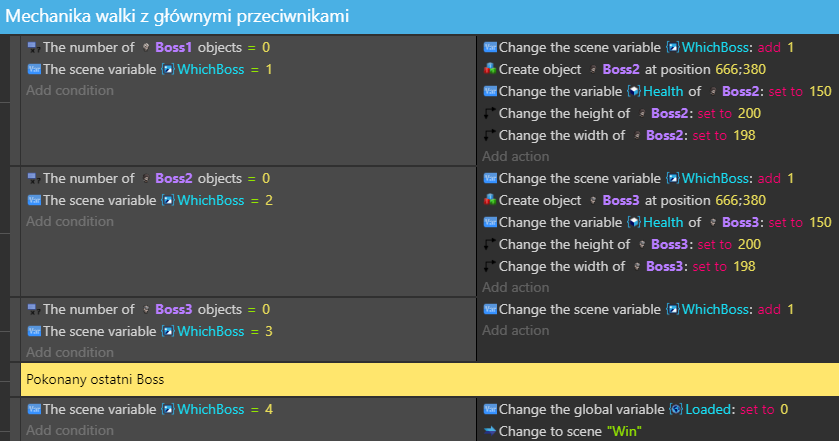


Po rozpoczęciu planszy ukrywane są elementy służące do sterowania poruszaniem się przeciwników. Następnie gra sprawdza czy użytkownik wykorzystał funkcję załadowania poprzednio zapisanego stanu gry. Rozpoczynane jest odtwarzanie podkładu muzycznego. Drugim elementem jest *FadeOut*, który służy do wyświetlenia nazwy sceny po jej załadowaniu. W planszach grywalnych zastosowano tylko część mechaniki *FadeOut*, główna definicja znajduje się w zdarzeniach zewnętrznych.



Trzecim elementem jest mechanika zmieniania aren na kolejne. Jest ona analogiczna do mechanik wykorzystywanych w scenach nie grywalnych.

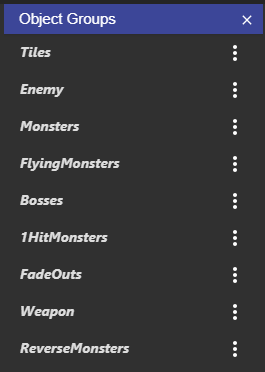
W arenie *3* dodatkową sekcją jest *Mechanika walki z głównymi przeciwnikami.*



Podczas rozgrywania ostatniej planszy gracz ma za zadanie pokonać trzech przeciwników głównych. Antagoniści zostają przyzywani pojedynczo tj. gdy jeden zostaje pokonany, następny staje do dalszej walki.

## 6.2. Opis zdarzeń zewnętrznych

Załącznik ten ukazuje efekty stworzenia mechaniki zdarzeń zewnętrznych wykorzystywanych w projekcie „Armin Legends”. W celu ujednolicenia kodu aplikacji, który wykorzystywany jest w różnych scenach utworzono zdarzenia zewnętrze. Działanie to pozwoliło na ominięcie błędnego zachowywania się elementów umieszczonych w grze, które mogło być spowodowane powielającym się kodem w wielu miejscach. Taki zabieg zapewnił uzyskanie większej przejrzystości kodu, co ułatwiło pracę oraz zmniejszyło liczbę powiązanych ze sobą w skomplikowany sposób problemów. Dodatkowo utworzono specjalne grupy obiektów o tych samych właściwościach, aby w kodzie posługiwać się za pomocą nazw grup, a nie pojedynczych elementów.

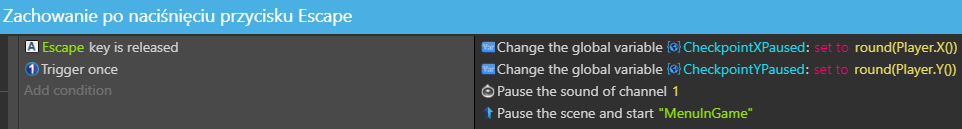
.

Zdarzenia zewnętrzne zaimplementowane w projekcie to:

* *PauseGame*,
* *PlayerBehavior*,
* *Camera*,
* *BossBehavior*,
* *Checkpoint*,
* *MonsterBehavior*,
* *Water*,
* *Spike*,
* *MovingPlatform*.

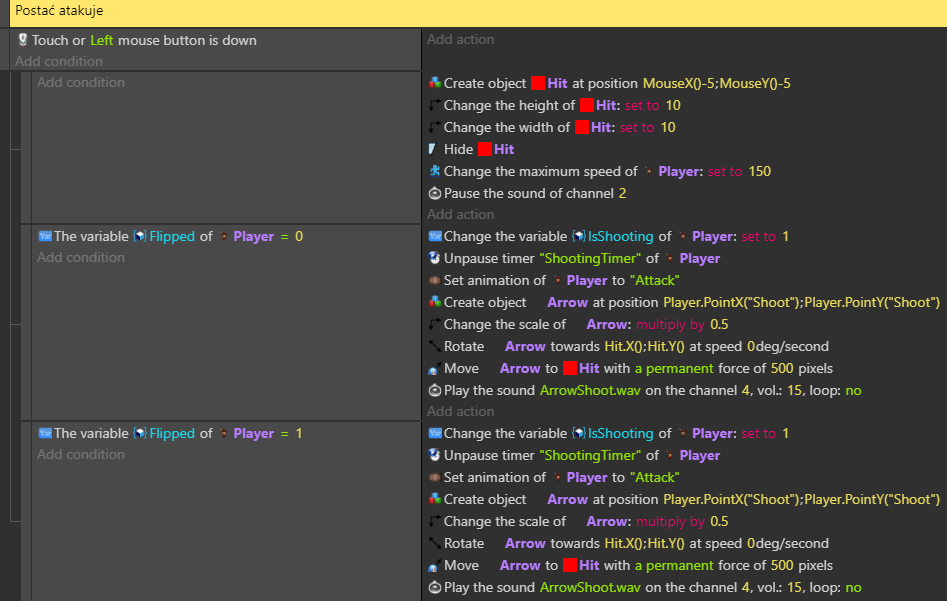
Można je podzielić na zdarzenia powiązane z otoczeniem i na zdarzenia powiązane z postaciami występującymi w grze.

1. *PauseGame* służy do wstrzymywania aktualnie rozgrywanej sceny i przechodzenia do planszy *MenuInGame*.

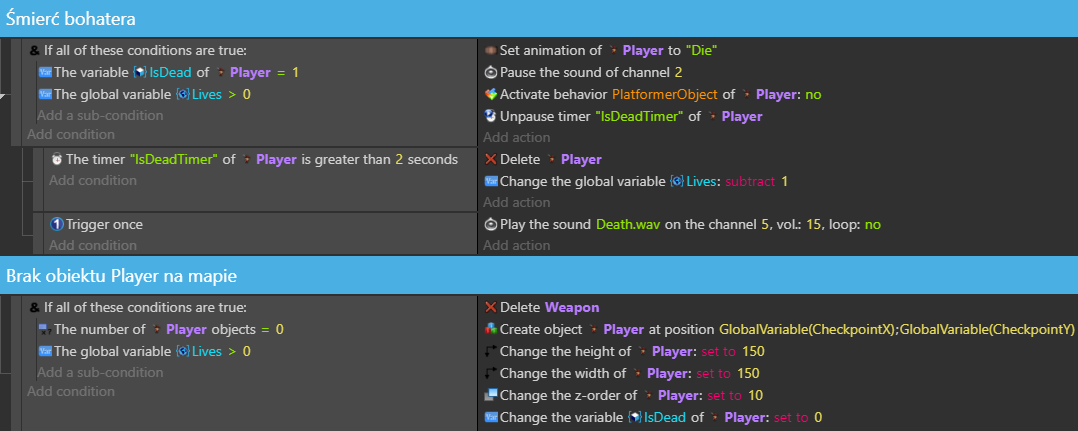


1. *PlayerBehavior* to rozbudowane zdarzenie zewnętrzne zawierające wszystkie mechaniki dotyczące zachowania postaci oraz jej interakcji z otoczeniem. Zdarzenie to zawiera dziewięć sekcji:

* *Zachowanie po rozpoczęciu sceny* -wyzwalane są czasomierze odpowiedzialne za śmierć bohatera oraz czas jaki musi upłynąć, aby można było ponownie zaatakować,
* *Orientacja grafiki player oraz implementacja poruszania się bohatera gry* -wyświetlanie wszystkich animacji zaimplementowanych dla obiektu bohatera. Dodatkowo umieszczono mechanikę ataku z łuku głównej postaci,



* *Śmierć bohatera* -obsługa animacji śmierci bohatera gry, po której następuje zmniejszenie ilości dostępnych żyć,
* *Brak obiektu player na mapie* - opisuje sytuację w której na mapie brakuje obiektu *Player*,



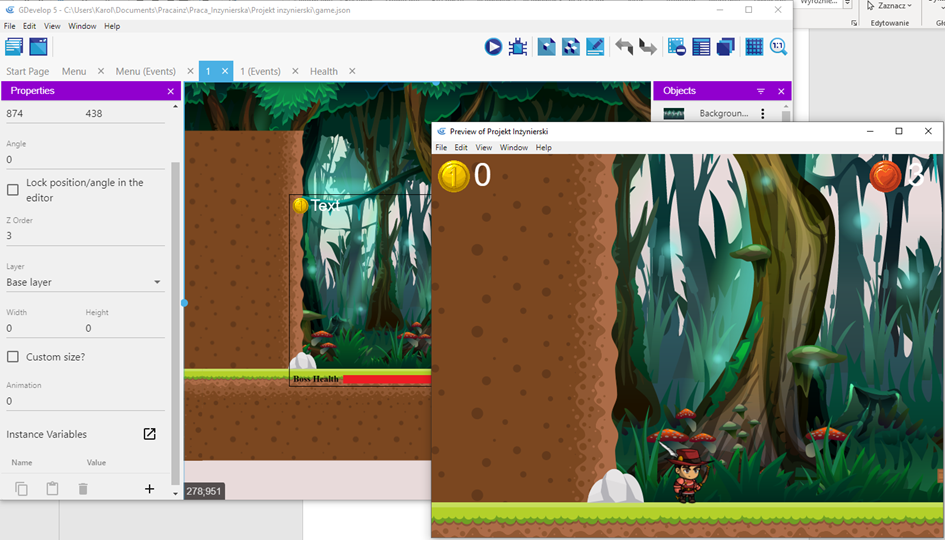
* *Kolizja bohatera z sercem lub monetą* - podczas rozgrywki możliwe są do zdobycia dodatkowe punkty życia i punkty wyniku, które przedstawione są w postaci serc i monet. Podczas kolizji obiektu *Player* z jednym z tych elementów do zmiennych globalnych dodawane są określone punkty,
* *Kolizja bohatera z przeciwnikami* - przy kolizji z przeciwnikiem odjęty zostaje punkt życia, a postać aktywuje wydarzenia z sekcji *Śmierć bohatera*,
* *Zmienna globalna Lives jest równa 0* - sprawdzanie czy aktualna wartość zmiennej globalnej *Lives* jest mniejsza niż zero. W przypadku potwierdzenia następuje zmiana sceny na *GameOver*,
* *Zachowanie strzały po kontakcie z otoczeniem* - obsługa przedmiotu za pomocą którego atakuje postać główna. W przypadku kolizji z otoczeniem, obiekt zostaje usunięty,
* *Zabijanie przeciwników poprzez upadanie* - implementacja przypadku w którym obiekt *Player* jest w trakcie opadania i w tym czasie nastąpi kolizja z przeciwnikiem. W tym przypadku przeciwnik zostanie uśmiercony.

1. Zdarzenie zewnętrzne *Camera* wprowadza do projektu funkcję sztucznego tła trójwymiarowego. Efekt ten został opisany w załączniku 5.1.
2. *BossBehavior* sekcja opisująca zachowanie głównych przeciwników. Definiuje ona w jaki sposób postacie będą atakować głównego bohatera.
3. *Checkpoint* służy do obsługiwania zapisu stanu gry do utworzonej komórki pamięci *progress*. Wywoływana jest w trakcie kolizji głównego bohatera z obiektem *Checkpoint*.
4. *MonsterBehavior* opisuje sposób poruszania się wszystkich przeciwników występujących w grze, a także ich kolizje z obiektem *Arrow* i wiążącą się z tą kolizją śmierć przeciwnika.
5. *Water* to zdarzenie opisujące sytuację w której dochodzi do zderzenia się dwóch obiektów *Player* oraz *Tile10*.
6. *Spike* określa sposób zachowania wysuwającego się kolca tj. czas w którym jest wysunięte i czas w którym jest schowane.
7. *MovingPlatform* definiuje zachowanie poruszających się drewnianych bali, służących do przemieszczania się postaci m.in. nad wodą lub żeby dostać się do wyżej, niedostępnych dla gracza miejsc.

## 7.1. Proces testowania produktu projektowego

Testy oprogramowanie odbywały się na każdym etapie fazy implementacji gry. Każda funkcjonalność w grze po jej zaimplementowaniu została przetestowana przez wbudowane w środowisko GDevelop narzędzie symulatora. Symulator umożliwia skompilowanie bieżącej sceny do aplikacji wykonywalnej w której to jesteśmy w stanie sprawdzić funkcjonalność podczas rozgrywki w ramach danej sceny (Rysunek 5).

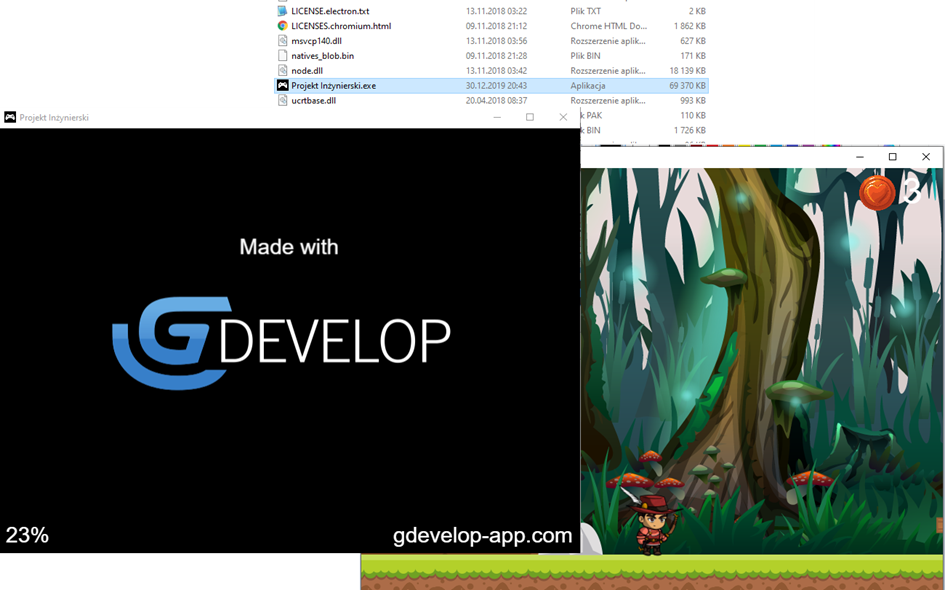
**Rysunek 5 Testowanie działania konkretnej sceny przy pomocy narzędzia do symulacji**



Źródło: Opracowanie własne

Oprócz tego po ukończeniu większych faz implementacji gra była eksportowana na platformy docelowe, aby ponownie przeprowadzić testy funkcjonalne tym razem na „żywym organizmie” czyli docelowym środowisku gry (Rysunek 6).

**Rysunek 6 Test funkcjonalny konkretnej wersji gry na platformie Windows**



Źródło: Opracowanie własne

## 7.2. Raport błędów i opis sposobu ich rozwiązania

1. Założenia

Błędy które zostały wychwycone w odpowiednich fazach testów zostały opisane biorąc pod uwagę kryteria takie jak:

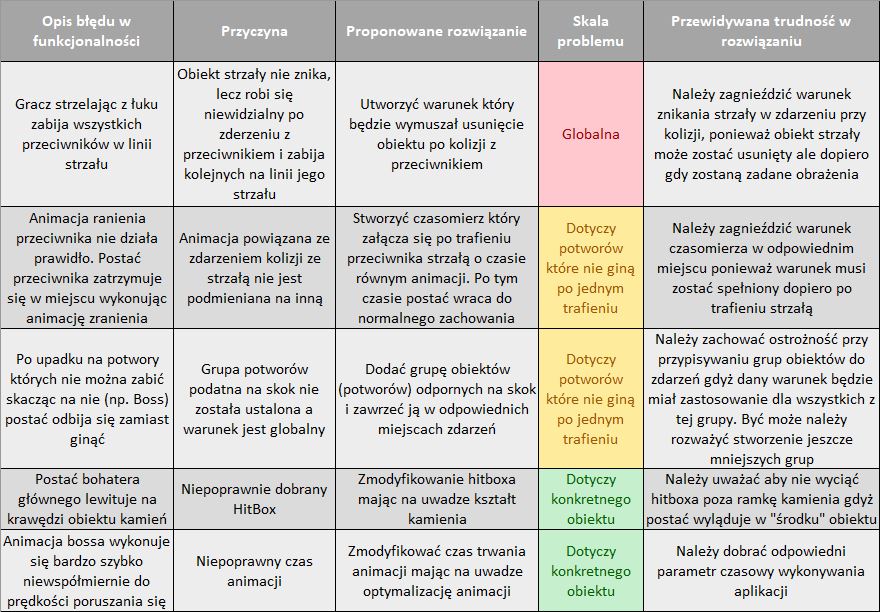
* opis błędu,
* przyczyna błędu,
* proponowane rozwiązanie,
* skala problemu,
* przewidywane trudności w rozwiązaniu.

Dodatkowo, skala problemu która wyznacza rangę problemu może przybrać następujące wartości:

* problem globalny - (kolor: czerwony) - najpoważniejsze problemy, wpływające na całą rozgrywkę. Należy je rozwiązywać w pierwszej kolejności,
* problem o zakresie grupy obiektów lub typu obiektów - (kolor: żółty) - problemy średniej skali których zakres działania obejmuje grupy obiektów lub dany typ, którego łączy wspólna cecha lub zmienna np. wszystkie elementy otoczenia,
* problem o zakresie pojedynczego obiektu - (kolor: zielony) - często mało znaczący problem (w zależności od instancji danego obiektu) odnoszący się do konkretnego obiektu lub jego konkretnej instancji powinien zostać rozpatrzony po uprzednim uporaniu się z problemami o wyższych rangach.

2. Raport z przeprowadzonych testów i wstępna analiza

**Rysunek 7 Raport i Analiza znalezionych błędów**



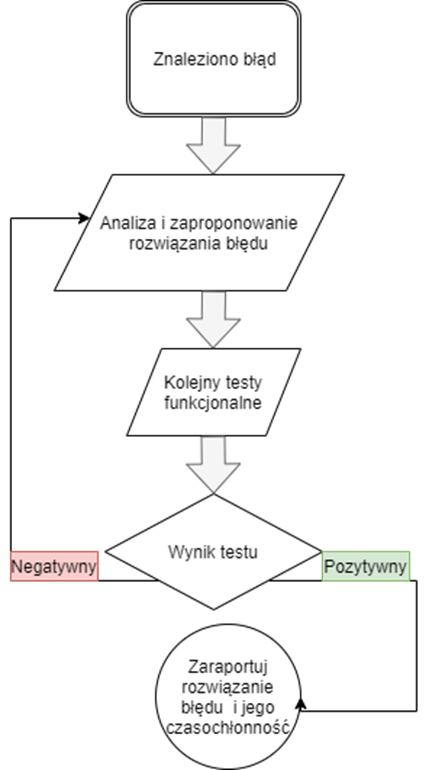
Źródło: Opracowanie własne

## 7.3. Wdrożenie poprawek, raport z pomyślnie przeprowadzonych testów

Po zdiagnozowaniu i analizie problemów w oprogramowaniu należy przystąpić do ich rozwiązywania. Po implementacji rozwiązania należy je przetestować, aby sprawdzić, czy rozwiązanie przyniosło oczekiwany rezultat. Dalsze kroki są zależne od wyniku prze-prowadzonego testu tj.:

* jeżeli wynik testu jest pozytywny należy oznaczyć problem jako rozwiązany i oszacować czasochłonność rozwiązywania problemu,
* jeżeli wynik testu jest negatywny należy zaplanować i wdrożyć nowe rozwiązanie i ponownie przetestować. (krok powinien być powtarzany do momentu pozytywnego zakończenia testu).

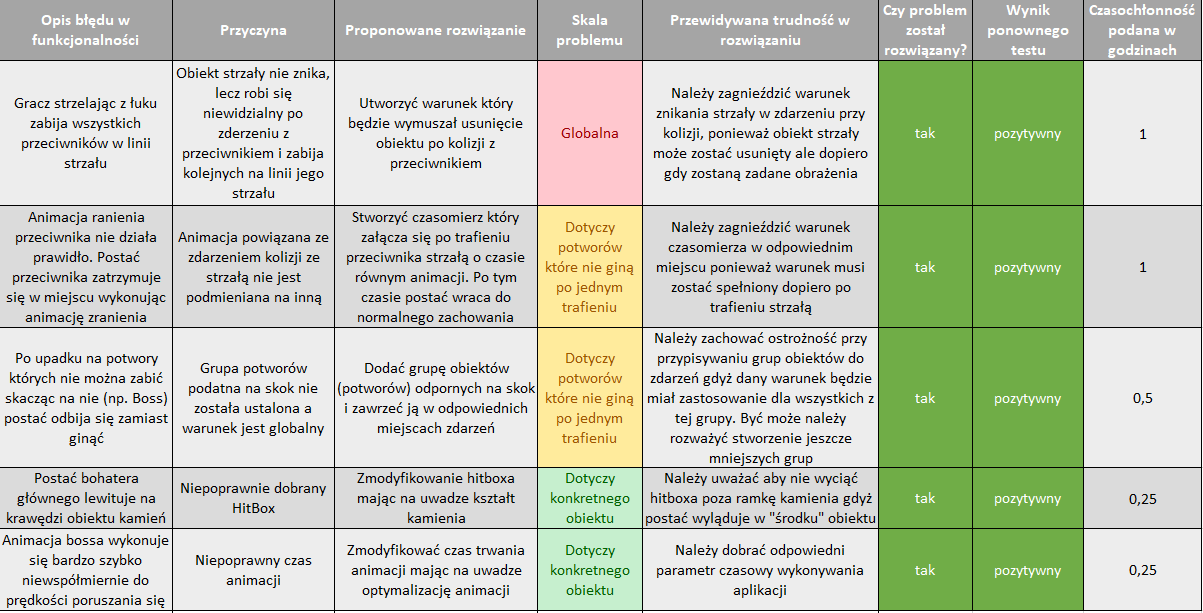
**Rysunek 8 Procedura eliminowania błędów**



Źródło: Opracowanie własne

4. Raport z rozwiązywania błędów

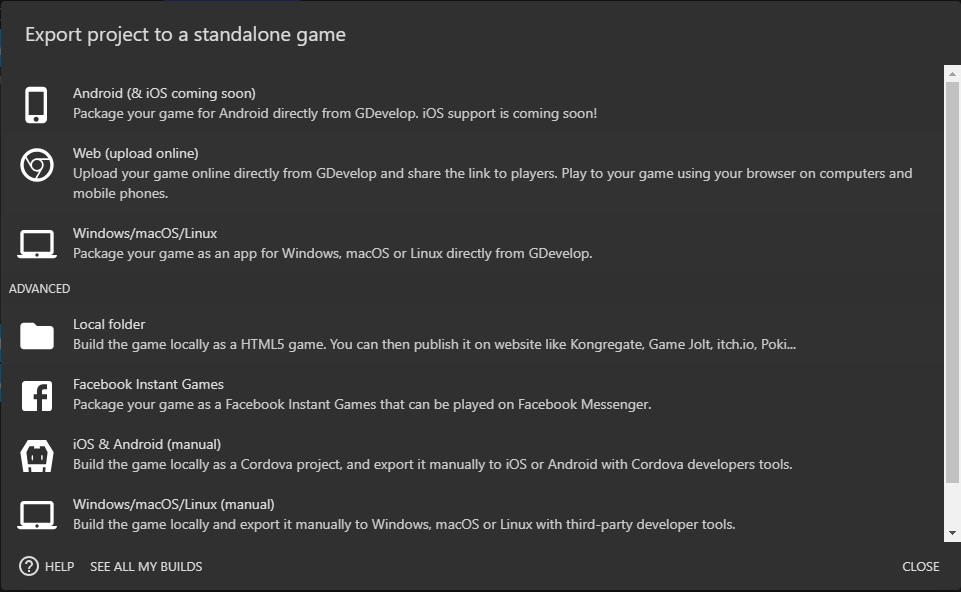
**Rysunek 9 Raport z rozwiązywania błędów**



Źródło: Opracowanie własne

## 8.1. Opis procesu eksportu na platformy użytkowe

Narzędzie GDevelop umożliwia użytkownikowi na wyeksportowanie swojego projektu na kilka różnych sposobów.

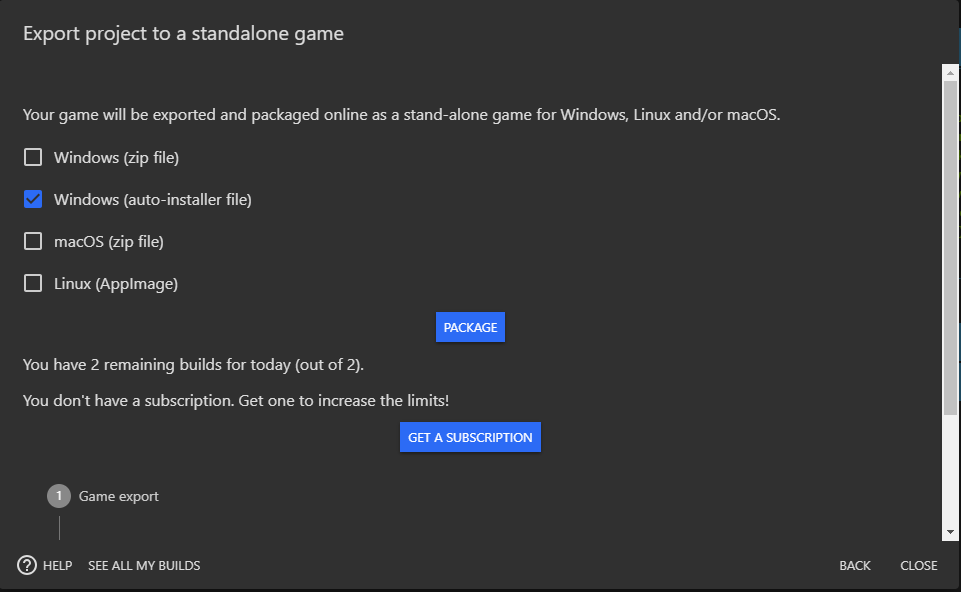


Otwierając z paska narzędzi zakładkę eksport ukazuje się widok (Rysunek nr.). Z tego miejsca możliwe jest wyeksportowanie projektu na:

* plik .apk, który można uruchomić na urządzeniach z systemem Android,
* wysłać nasz projekt na serwer Web aplikacji GDevelop. Po kilku minutach w odpowiedzi otrzymujemy link do strony internetowej na której znajduje się grywalna wersja projektu,
* eksport gry na system Windows/macOS/Linux,
* zapis projektu do lokalnego folderu,
* eksport jako pliku Facebook Instant Games, który może zostać wykorzystany do uruchomienia gry w aplikacji firmy Facebook noszącej nazwę Messenger,
* utworzenie plików źródłowych na platformy iOS/Android/Windows/macOS/Linux, które w późniejszym czasie można skompilować za pomocą zewnętrznych programów.

Projekt zostanie zapisany przy pomocy opcji eksportu gry na system Windows/macOS/Linux.

Po wybraniu wyżej opisanej opcji użytkownikowi ukazuje się następne okno wyboru (Rysunek nr.).

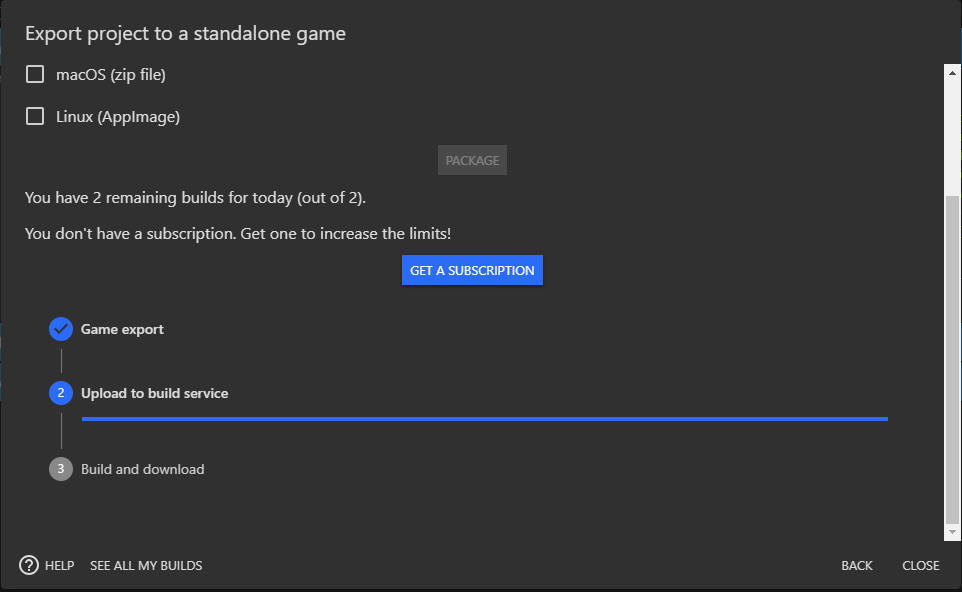


W tym oknie program umożliwia eksport gry m.in. do systemu Windows, w którym można zapisać projekt w postaci skompresowanego pliku .zip lub pliku wykonywalnego .exe. Dla systemu macOS można wyeksportować grę podobnie jak do systemu Windows do pliku .zip, a dla systemu Linux jest możliwość zapisania projektu w postaci AppImage.

Gra Armin Legends zostanie wyeksportowana do postaci pliku wykonywalnego .exe.

Wybierając opcję zaznaczoną na rysunku (Rysunek), a następnie naciskając przycisk „Package” program rozpoczyna proces eksportu gry. Proces ten składa się z trzech etapów:

1. Wysłania projektu na serwer Web aplikacji GDevelop (Rysunek nr.),



1. Wyeksportowania przez serwer gry do pliku wykonywalnego,
2. Aplikacja udostępnia link do pobrania z sieci wyeksportowanego pliku .exe.

Po pobraniu pliku wykonywalnego i jego otwarciu, można rozpocząć rozgrywkę w grze *Armin Legends*.

1. Zapach kodu (ang. Code Smells) – nawyki występujące przy wytwarzaniu oprogramowania uznawane za nieprawidłowe. Element badawczy w procesie tworzenia oprogramowania szczególnie na etapie code review – czyli analizy poprawności kodu. Bardzo często wykrywa i naprawie je się za pomocą automatycznych skanerów kodów czyniąc kod bardziej przejrzystym i zrozumiałym. [↑](#footnote-ref-1)
2. Powielony kod (ang. Duplicated Code) – jeden z zapachów kodu występuje, gdy: identyczny lub bardzo podobny kod istniejący w więcej niż jednej lokalizacji. [↑](#footnote-ref-2)
3. Shotgun surgery - Jeden z zapachów kodu występuje, gdy: jedna zmiana musi być zastosowana do wielu klas projektu jednocześnie inaczej wywoła nieprawidłowości w działaniu kodu. [↑](#footnote-ref-3)