Grafika Komputerowa Projekt						
Rok akademicki	Termin	Rodzaj studiów	Kierunek	Prowadzący	Grupa	Sekcja
2017/2018	poniedziałek 8:15 - 9:45	SSI	INF	MP	GKIO1	5

Raport końcowy

Data oddania raportu:

23.05.2018

Tytuł gry:

Mastermind

Skład sekcji:

Grzegorz Nowak
Magdalena Bajerska

1.Treść zadania:

Tematem zadania jest stworzenie logicznej gry desktopowej - Mastermind. Do stworzenia gry zostanie wykorzystany silnik graficzny Unity. Zadaniem użytkownika będzie odgadnięcie szyfru znajdującego się w skrzyni, przy pomocy odpowiednich wskazówek. Gracz na ekranie będzie widział planszę do gry, a zaraz obok niej znajdować się będą pionki w odpowiednich kolorach, które posłużą mu do odgadywania szyfru. Wskazówki pomagające w przejściu gry będą wyświetlane na specjalnie stworzonej do tego celu tablicy. Celem gry będzie odkrycie szyfru w jak najmniejszej ilości ruchów. Docelowo przyjmujemy, że maksymalna ilość wynosiła będzie 10, jeśli gracz przekroczy tą liczbę – przegrywa.

2.Analiza zadania:

W pierwszej kolejności program będzie generował szyfr składający się z 4 kolorów. Będą one wybierane z puli 6 kolorów dostępnych na ekranie. Ukryty kod zamknięty zostanie w skrzyni i nie będzie widoczny dla gracza. Użytkownik kolejno wybiera kolory i układa z nich kombinacje na planszy, a następnie zatwierdza kombinację. Program przetwarza podane przez użytkownika kolory i porównuje je z tymi zamkniętymi w skrzyni. Następnie uzyskane wyniki prezentuje na tablicy. W zależności od tego czy użytkownikowi udało się zgadnąć jakiś kolor w konkretnym miejscu lub zgadnąć kolor, nie znajdujący się w odpowiednim miejscu wyświetlane są wskazówki.
Podpowiedzi są prezentowane w formie kształtów o odpowiednich kolorach:

- ✓ czarny – kolor został trafiony i znajduje się na właściwym miejscu,
- ✓ szary – kolor został trafiony, ale nie znajduje się na poprawnym miejscu,
- ✓ biały – kolor nie został trafiony.

Wykorzystywane zagadnienia z laboratorium grafiki komputerowej:

- ✓ wykrywanie kolizji,
- ✓ efekty cząsteczkowe,
- ✓ animacja komputerowa.

Wykorzystywane biblioteki i narzędzia programistyczne:
Do stworzenia gry wykorzystaliśmy silnik graficzny Unity. Jest on wygodny do pisania gier zarówno dwuwymiarowych jak i trójwymiarowych. Zaletą Unity jest możliwość działania na wielu systemach operacyjnych i pozwala na tworzenie aplikacji zarówno na komputery jak i na urządzenia mobilne oraz konsole. Skrypt do naszej aplikacji będzie pisany w języku C# w środowisku Visual Studio.

Algorytmy, struktury danych, ograniczenia specyfikacji:
Algorytm zaimplementowany w grze będzie opierał się na standardowej wersji gry Mastermind, w której dostępne jest 6 kolorów, a szyfr będzie się składał z 4 nich. Cały program będzie podzielony na dwie zasadnicze części:

- ✓ część logiczna gry, w której znajdować będzie się cały algorytm odpowiedzialny za poprawne działanie gry,

- ✓ część graficzna – odpowiedzialna za prezentację danych na ekranie i odpowiednie efekty pojawiające się w przypadku określonych reakcji użytkownika.

3. Status gry, zmiany.

Gra Mastermind osiągnęła status złoty. Wszystkie założenia zostały zrealizowane, poprawiono również błędy wykryte w czasie testów wersji beta. W celu lepszego odbioru wizualnego gry postanowiliśmy usunąć animowaną postać. Nie pasowała ona do charakteru tworzonej przez nas gry przez co mogłaby niepotrzebnie zakłócić immersję gracza. W celu wyrównania ilości pracy stworzyliśmy bardziej potrzebny element - menu pauzy. Dzięki któremu możemy wstrzymać grę i potem do niej powrócić bez utraty uprzedniego stanu rozgrywki.

Zmiany wprowadzane w wersji beta:

- ✓ zmieniono szatę graficzną gry – zmieniono tło na pasujące do reszty elementów, dostosowano skrzynię do stylistyki i charakteru gry,
- ✓ zmodyfikowano oświetlenie, dzięki któremu poprawiono widoczność kluczowych elementów,
- ✓ zrefaktoryzowano kod gry – powtarzające się fragmenty przeniesiono do metod co pozwoliło na znaczne skrócenie i uproszczenie kodu,
- ✓ Dodano komentarze wyjaśniające znaczenie poszczególnych metod,
- ✓ Poprawiono wiele drobniejszych bugów.

4. Podział prac nad projektem.

Rzeczywisty podział prac nad projektem był niemal identyczny do tego wstępnego podziału opisanego w dokumencie projektowym (http://tiny.pl/g2z59) (podpunkt 4). Różni się jedynie zamianą animacji postaci na stworzenie menu pauzy.
Rzeczywisty podział prac:

Zagadnienie	Grzegorz Nowak	Magdalena Bajerska
Efekty cząsteczkowe	X (po wygranej)	X (po przegranej)
Menu gry	X	
Menu pauzy	X	
Menu końca gry		X
Wykrywanie kolizji		X
Algorytm gry	X (porównywanie szyfru z kodem od użytkownika)	X (generowanie wskazówek na podstawie wprowadzonych danych)
Animacje	X	X

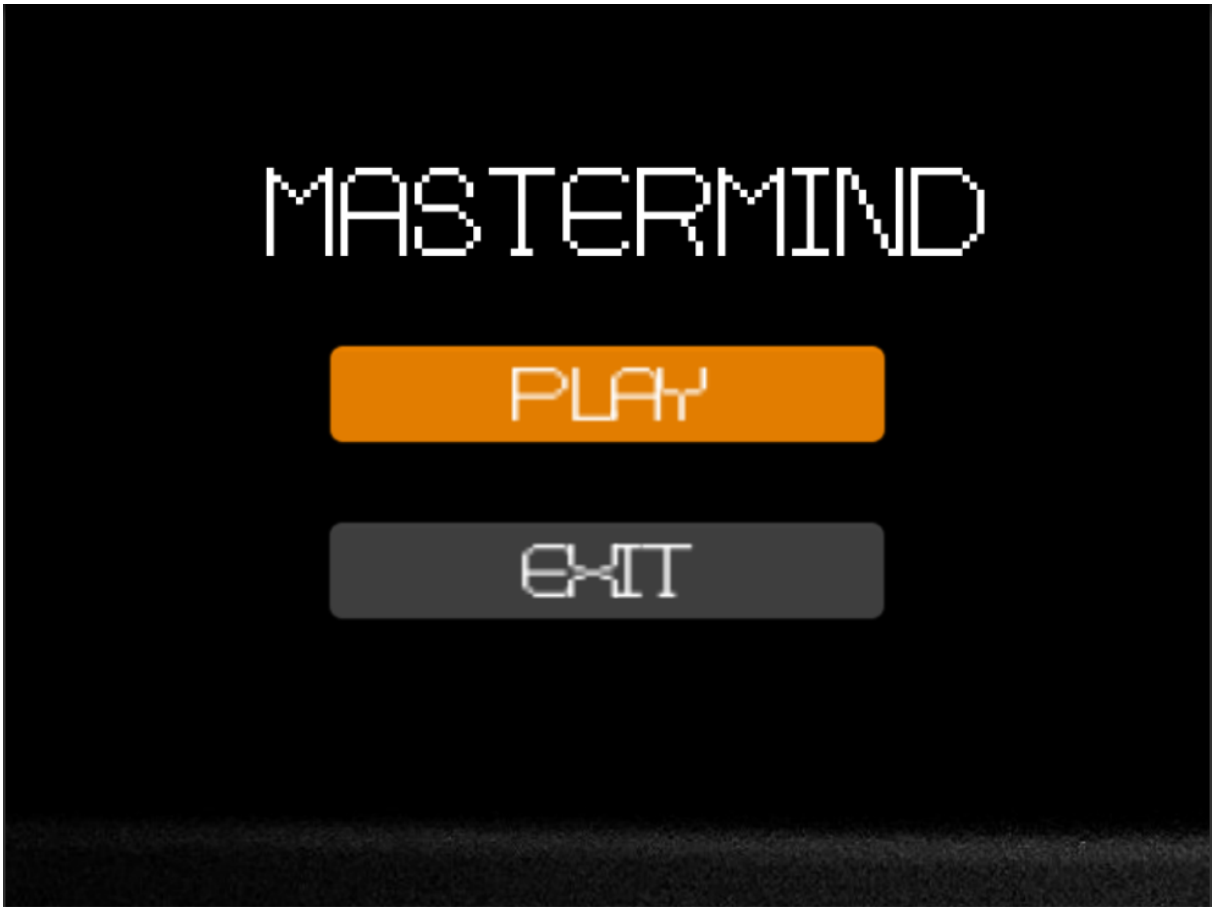
5. Specyfikacja zewnętrzna

Po uruchomieniu gry ukazuje się nam menu główne. W menu tym jest możliwość uruchomienia nowej gry lub wyjścia z programu, jeżeli został on uruchomiony przypadkiem. Po rozpoczęciu nowej gry zostanie załadowana główna scena z rozgrywką. Po lewej stronie ekranu wyświetlana jest liczba kroków oraz czas trwania aktualnej rozgrywki. Poniżej widzimy skrzynię ze skarbami, która otworzy się tylko w przypadku wygranej. Pośrodku ekranu możemy zobaczyć planszę gry Mastermind, która wypełnia się sukcesywnie w trakcie trwania rozgrywki. Poniżej planszy znajduje się pole z 4 diamentami. Zadaniem gracza jest zabarwienie ich wybranymi przez siebie kolorami. Dostępnych jest 6 kolorów widocznych poniżej. Zabarwienie polega na przeciągnięciu koloru na wybrany diament. Dokonany wybór należy zatwierdzić przyciskiem Check Color. Operacja ta powoduje aktualizację stanu wskazówek. Wspomniane wskazówki znajdują się po prawej stronie ekranu i są one aktualizowane po każdym ruchu, zgodnie z zasadami gry Mastermind. Chcąc chwilowo wstrzymać rozgrywkę należy przejść do menu pauzy. Aby uruchomić menu pauzy, należy wcisnąć Escape na klawiaturze lub przycisk Pause znajdujący się pod sekcją wyświetlającą czas gry. W menu Pauzy mamy możliwość powrotu do zatrzymanej przed chwilą rozgrywki, możemy również wyjść do menu głównego jak i do pulpitu systemu operacyjnego. Kolejnymi dwoma ekranami są ekrany przegranej i wygranej gry. Ekran przegranej gry zawiera wybuchy, które symbolicznie niszczą gracza oraz deszcz, który wprowadza nastrój porażki. Ekran wygranej gry charakteryzuje się radośniejszą atmosferą, możemy zauważyć fajerwerki oraz otwarta zostaje skrzynia ze skarbami. Oba te ekrany mają możliwość rozpoczęcia nowej gry lub powrotu do menu głównego.

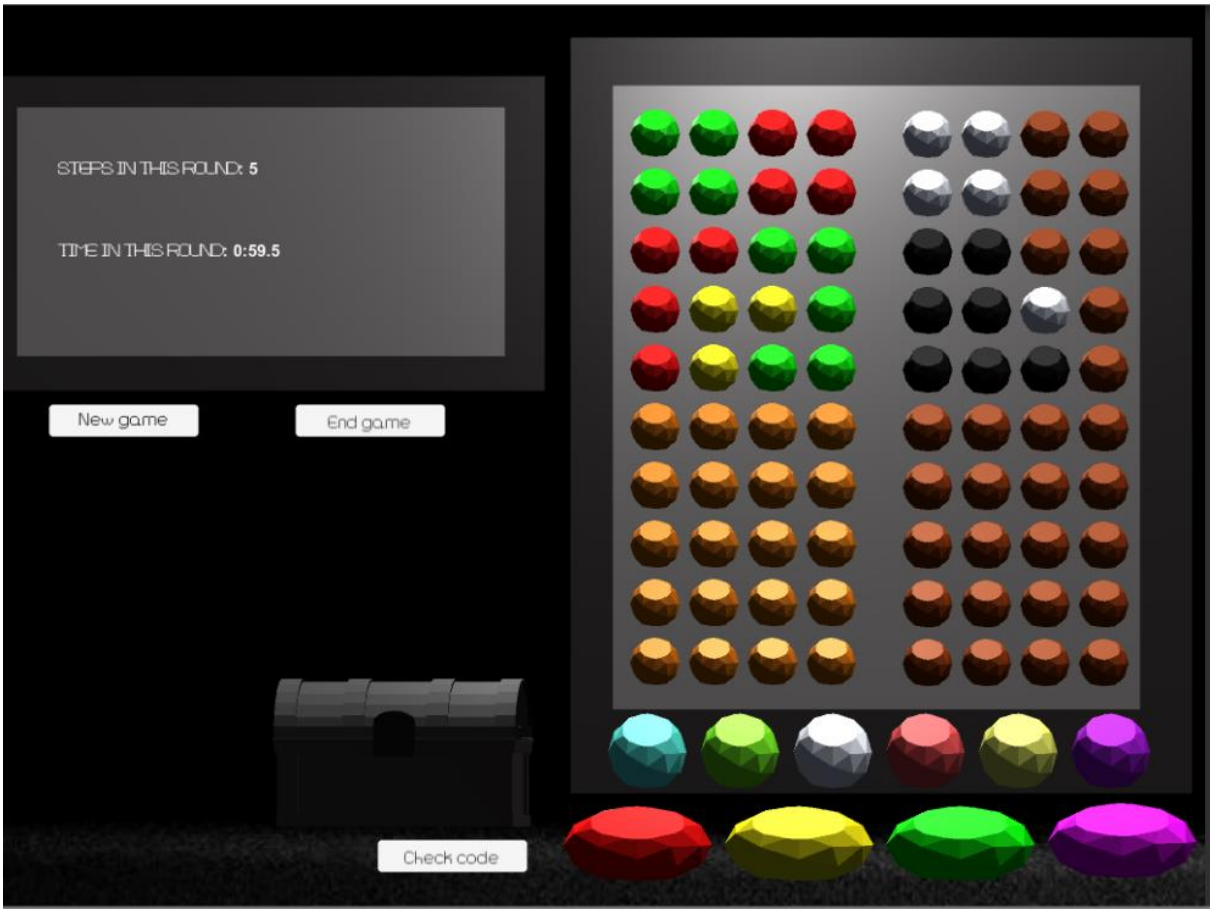
6. Przykłady działania

Poniżej zaprezentowano 5 screenów z gry:

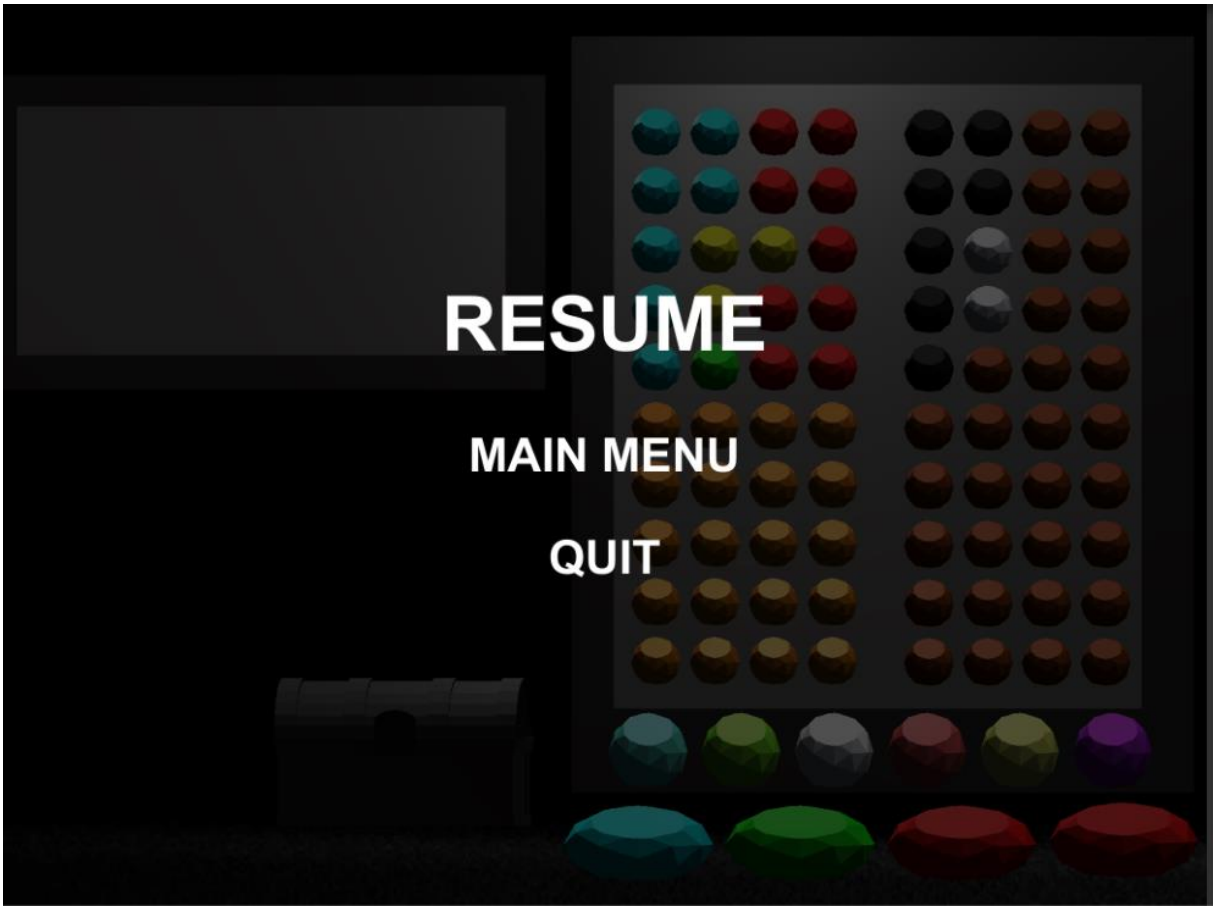
a) Menu główne,
b) Główna gra,
c) Menu Pauzy,
d) Ekran końca gry w przypadku wygranej,
e) Ekran końca gry w przypadku przegranej.



Rysunek 1. Menu główne gry.

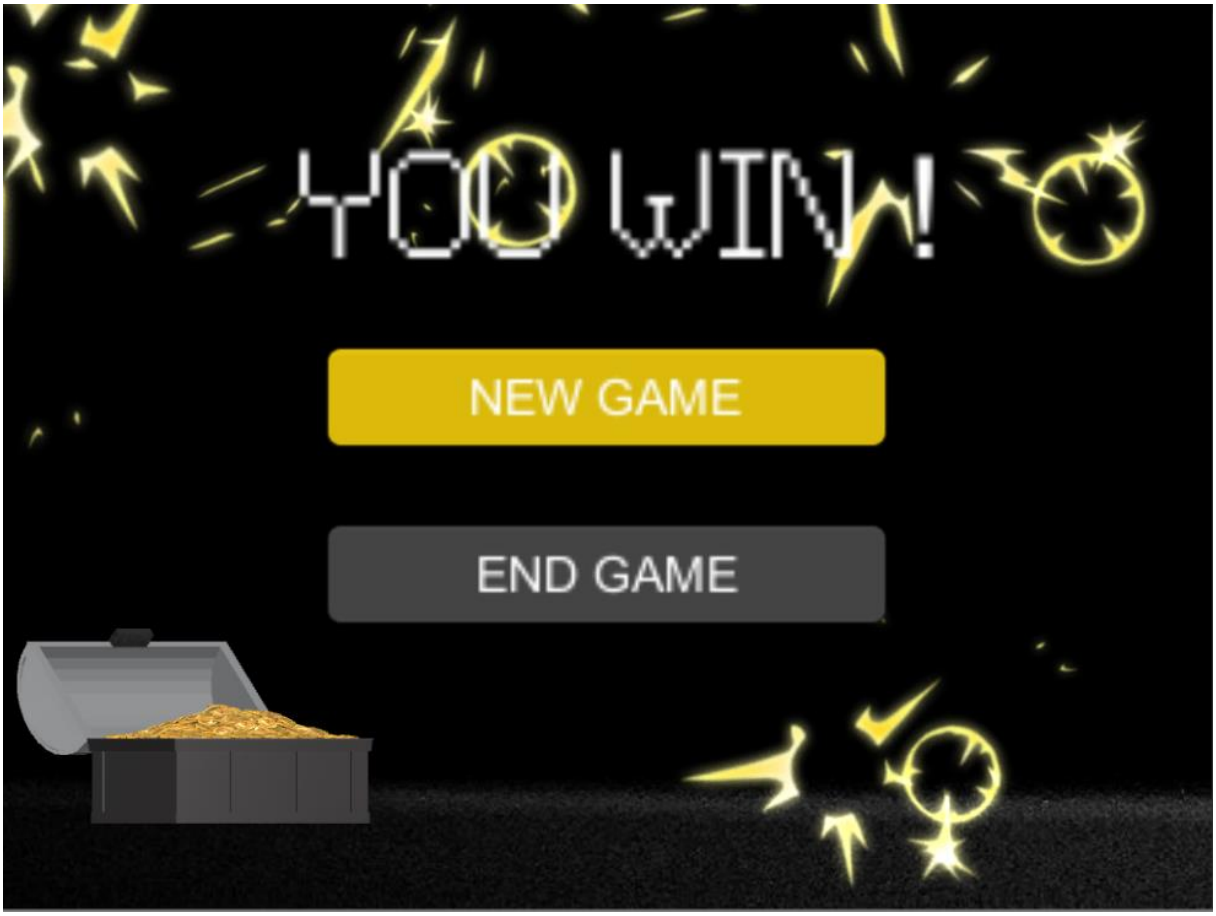


Rysunek 2. Główna gra



Rysunek 3. Menu Pauzy

Rysunek 4. Ekran przegranej gry



Rysunek 5. Ekran wygranej gry

7. Specyfikacja wewnętrzna

Dokumentacja specyfikacji wewnętrznej została wygenerowana na podstawie specjalnych komentarzy XML zawartych w kodzie gry. Zawarto tam opisy wszystkich stworzonych przez nas klas oraz zawartych w nich metod. Wspomnianą dokumentację uruchamia się w przeglądarce internetowej wybierając plik:

`.\Documentation\Specyfikacja wewnetrzna\annotated.html`

Po załadowaniu pliku naszym oczom ukaże się lista klas wraz z opisami. Po kliknięciu w nazwę klasy zostaniemy przekierowani do jej dokładniejszego opisu oraz składających się na nią metod. Metody posiadają ogólny opis, opis przyjmowanych parametrów oraz opis zwracanej wartości.

W grze wykorzystano następujące elementy pobrane z Asset Store:

- ✓ Czcionka Hana - Hana Pixel Font (MYSTERY CORGI),
- ✓ Ekran końcowy przy wygranej grze (fajerwerki) – Space Arcade - Effects,
- ✓ Ekran końcowy przy przegranej grze (wybuchy i deszcz) – 2d Flat Explosion (RED SHARK) i Rain Maker - 2D and 3D Rain Particle System for Unity (Digital Ruby (Jeff Johnson)),

- ✓ Skrzynia –Treasure chest pack22+1 (WIZAPPLY) – tylko model, kod implementowany samodzielnie,
- ✓ Diamenty – Gem Shader (UNITY TECHNOLOGIES).

8. Testowanie i uruchamianie

W trakcie tworzenia gry na bieżąco przeprowadzane były testy. Przyjęcie takiej strategii pozwoliło nam na szybkie wychwycenie i poprawienie niechcianych błędów. Program nie przyjmuje żadnych danych, gdyż nie ma takiej potrzeby. Jedyne dane jakie są wprowadzane przez użytkownika to wybór kolorów potrzebnych do wykonania następnego kroku. Program również nie zapisuje danych wyjściowych, wszystkie potrzebne informacje są na bieżąco wyświetlane na ekranie. Zostało przetestowanych bardzo dużo zachowań użytkowników, od standardowych zachowań do nietypowych prób znalezienia błędu. Jednym z wykrytych przez nas problemów, który pilnie należało naprawić jest nieprzywracanie ówczesnego stanu gry po powrocie z menu pauzy. Większość poważniejszych błędów i problemów została opisana w raporcie beta gry (podpunkt 2). W celu zweryfikowania działania algorytmu, który sprawdzał zgodność kolorów wybranych przez użytkownika z wylosowanym na samym początku rozgrywki szyfrem, przeprowadzono wiele różnorakich testów. Testy w głównej mierze polegały na wielokrotnym graniu w grę. Dzięki wielu próbom udało się wyeliminować wszystkie błędy. Kolejną korzyścią płynąca z tego rodzaju testów jest możliwość zbalansowania poziomu trudności, aby gra nie była zbyt łatwa ani zbyt trudna. W trakcie dłuższej rozgrywki udało się również wyeliminować irytujące elementy interfejsu użytkownika, takie jak słabo widoczne przyciski itd.

8. Wnioski

Nasze wstępne założenia do projektu okazały się niemal w 100% trafne. Nie trzeba było ich modyfikować poza drobnymi modyfikacjami opisanymi powyżej. Silnik graficzny Unity okazał się bardzo dobrym narzędziem do tworzenia gier. Ma bardzo duże możliwości co czyni go popularnym narzędziem developerskim. Początki korzystania z Unity nie należały to rzeczy prostych. Wiele rzeczy np. wykrywanie kolizji było dość problematyczne. Wielką zaletę Unity jest ogromna społeczność jaka skupiła się wokół tego środowiska, dzięki temu powstało wiele assetów i tutoriali bez których pomocy stworzenie gry wydaje się niemożliwe, a na pewno byłoby znacznie trudniejsze. Dużym wyzwaniem było dla nas poznanie samego środowiska i sposobów w jaki z niego korzystać. Każdy obiekt w Unity np. Collider czy Animator ma bardzo rozbudowaną konfigurację. Większość czasu poświęcaliśmy właśnie na poprawne skonfigurowanie tych obiektów, aby działały zgodnie z naszymi zamierzeniami. Jeśli chodzi o część programistyczną, nie stanowiła ona dla nas większego problemu, ponieważ już wcześniej wiele razy korzystaliśmy z środowiska Visual Studio i z języka C#. Jedyne problemy jakie się zdarzały w tej części projektu to kłopoty z działaniem struktur w języku C# typowych tylko dla silnika Unity. Nasze wcześniejsze projekty tworzyliśmy zazwyczaj w jednym środowisku, a w tym przypadku kluczowa

była współpraca Unity z Visual Studio, co na początku przysporzyło nam paru problemów. Dzięki temu projektowi poznaliśmy rzeczywiste sposoby tworzenia gier komputerowych. Wcześniej zaznajomiliśmy się tylko z biblioteką OpenGL, która jednak nie jest już wykorzystywana do produkcji gier, ponieważ byłoby to znacznie bardziej czasochłonne i kosztowne. Unity upraszcza wiele mechanizmów znanych z OpenGL. Było to dla nas miłym zaskoczeniem, kiedy wielogodzinna praca w OpenGL tutaj sprowadzała się do kilkunastu minut. Zgodnie uważamy, że stworzenie tej gry było dla nas owocnym doświadczeniem, które pomoże nam w przyszłej karierze zawodowej.