

UNIWERSYTET GDAŃSKI
Wydział Matematyki, Fizyki i Informatyki

Marcin Szpaderski

nr albumu: 195 008

Aplikacja wspierająca badania nad efektem Zeigarnik

Praca licencjacka na kierunku:

INFORMATYKA

Promotor:

dr Włodzimierz Bzyl

Gdańsk 2017

Streszczenie

W ramach pracy napisano aplikację na urządzenia mobilne o nazwie "Pułapka Zeigarnik". Do stworzenia aplikacji wykorzystane zostało środowisko do projektowania gier i programów komputerowych "GameMaker: Studio". Napisano osiem testów, które użytkownik ma za zadanie rozwiązać w określonym czasie.

Wszystkie grafiki potrzebne do funkcjonowania aplikacji zostały wykonane przeze mnie w programie Paint.NET .

Aplikacja została wdrożona i umieszczona w repozytorium GitHub. Jest dostępna do pobrania ze strony: <https://github.com/mszpaderski/Sem/tree/master/Pu%C5%82apka%20Zeigarnik> .

Słowa kluczowe

pamięć ludzka, efekt Zeigarnik, aplikacje mobilne, środowisko programistyczne GameMaker, GameMaker Server

Spis treści

Wprowadzenie	7
1. Założenia aplikacji	9
1.1. Porównanie wybranych środowisk programistycznych	9
1.1.1. Unity a GameMaker	9
1.1.2. Microsoft Visual Studio a GameMaker	10
1.2. Zadania	10
1.2.1. Najeźdźcy z kosmosu	11
1.2.2. Rozbij jajko	12
1.2.3. Powódź	12
1.2.4. Uratuj księżniczkę	12
1.2.5. Zator	13
1.2.6. Lodowy Labirynt	13
1.2.7. Dziurawy labirynt	13
1.2.8. Super Pamięć	14
2. Projekt aplikacji i wybrane technologie	15
2.1. Projekt interfejsu użytkownika	15
2.2. GameMaker: Studio	16
2.3. Gromadzenie danych - GameMaker Server	17
2.4. Pliki INI i organizacja danych	18
3. Szczegóły implementacji	21
3.1. Inicjalizacja testu	21
3.2. Pokoje z grami	22
3.2.1. Najeźdźcy z Kosmosu	22
3.2.2. Rozbij jajko	23
3.2.3. Powódź	24
3.2.4. Uratuj księżniczkę	25
3.2.5. Zator	25

3.2.6. Lodowy Labirynt	26
3.2.7. Dziurawy labirynt	26
3.2.8. Super Pamięć	26
3.3. Ankieta	26
Zakończenie	27
Spis tabel	29
Spis rysunków	31
Oświadczenie	35

Wprowadzenie

Zamiar niekoniecznie oznacza z góry określoną okazję do jego zrealizowania, lecz potrzebę lub tymczasową potrzebę, która stwarza taką okazję. Bulma Zeigarnik długo zastanawiała się nad tym stwierdzeniem, próbując zbadać jak wywołać u człowieka tę chwilową potrzebę, która wpływa na naszą pamięć. Celem niniejszej pracy jest stworzenie aplikacji, która będzie pełnić funkcję pomocniczą przy przeprowadzaniu badań nad tym, co dziś nazywamy Efektem Zeigarnik.

Efekt Zeigarnik został nazwany i opisany przez Blumę W. Zeigarnik¹ w 1927 roku. Opisuje on pojęcie psychologiczne związane z zagadnieniami pamięci i motywacji psychologii ogólnej. Efekt ten wykazuje, że czynności, które zostały nam przerwane jesteśmy w stanie lepiej sobie przypomnieć po pewnym czasie niż te, które wykonaliśmy bez żadnych problemów. Przykładem Efektu Zeigarnik są kelnerki w restauracjach, które jednocześnie pamiętają zamówienia nawet paru obsługiwanych w danym momencie osób, lecz gorzej przypominają sobie zamówienia klientów, którzy opuścili już lokal. Pomysł na aplikację, która pomaga badać ten efekt pojawił się podczas rozmowy ze znajomym, który ukończył studia na kierunku psychologia.

W części teoretycznej zostaną opisane główne założenia przyjęte podczas projektowania aplikacji. Przedstawione będą sposoby rozwiązania konkretnych problemów związanych z założeniami oraz możliwości aplikacji w zakresie przetwarzania informacji dostarczanych przez użytkowników.

Projekt będzie wykonany na zasadzie aplikacji działającej w trybie klient-serwer. Aplikacja zostanie zaprojektowana i wykonana w środowisku The GameMaker: Studio, które wykorzystuje unikalny dla siebie język GML, składnią zbliżony do C++ lub Pascal. Wybrałem takie środowisko ze względu na chęć rozszerzenia swojej wiedzy w zakresie implementacji programów i gier w środowiskach do tego przystosowanych. Wymieniona technologia zostanie

¹Bluma W. Zeigarnik (09.11.1900 - 24.02.1988) - rosyjska psycholog i psychiatra.

opisana w niniejszej pracy, przedstawione będą zalety jej wykorzystania oraz szczegóły implementacji.

ROZDZIAŁ 1

Założenia aplikacji

Głównym celem aplikacji jest stworzenie bazy informacji w celu przeprowadzenia badań nad Efektem Zeigarnik. Test jest jednorazowy, od użytkownika wymaga się, aby podszedł do niego tylko raz oraz bez wiedzy, co tak naprawdę wykonuje.

Założeniem badania jest sprawdzenie, czy uczestnik po wykonaniu wszystkich zadań będzie w stanie łatwiej przypomnieć sobie te, które wykonał do końca, czy te, które zostały mu przerwane przed końcem.

Po wykonaniu wszystkich testów użytkownik dostanie do wypełnienia ankietę, która pozwoli ustalić, co zapamiętał badany. Wyniki poszczególnych zadań, jak i treść ankiety wysłane zostaną na serwer.

1.1. Porównanie wybranych środowisk programistycznych

Aplikacja skierowana jest na tablety i inne urządzenia przenośne. Została stworzona w środowisku programistycznym o nazwie GameMaker: Studio. Środowisko to wybrałem spośród wielu innych dostępnych na rynku. W tej chwili najpopularniejszymi środowiskami wydają się być Microsoft Visual Studio oraz Unity. W tym rozdziale wyznaczę jakie są różnice pomiędzy nimi a tym które ostatecznie wybrałem oraz postaram się wyjaśnić dlaczego wybrałem to środowisko.

1.1.1. Unity a GameMaker

Środowisko Unity jest bardzo podobne do wybranego przeze mnie środowiska GameMaker. Służy głównie jako środowisko do tworzenia trójwymiarowych

lub dwuwymiarowych gier komputerowych lub materiałów interaktywnych takich jak wizualizacje czy animację. Tak samo jak GameMaker, służy do tworzenia aplikacji na przeglądarki internetowe, komputery osobiste, konsole gier wideo oraz wszelkie urządzenia mobilne. Oba środowiska posiadają unikalne języki skryptowe. Unity wykorzystuje UnityScript, a GameMaker swój GML. Oba te języki są bazowane na języku JavaScript, i nie różnią się wiele składnią od niego.

Główną różnicą pomiędzy tymi dwoma środowiskami, która przekonała mnie do GameMakera, do fakt że Unity jest narzędziem głównie do pracy nad elementami 3D, gdy mi zależało na aplikacji w dwóch wymiarach. Nie jest niemożliwe stworzenie zwykłej aplikacji, lecz w mojej opinii lepiej jest korzystać z narzędzi nastawionych przede wszystkim na żądany efekt.

1.1.2. Microsoft Visual Studio a GameMaker

Środowisko Microsoft Visual Studio pozwala na tworzenie aplikacji na wiele systemów, jednak jest skupiony na tworzeniu aplikacji dla systemów wspierających Windowsa. Co za tym idzie, aby zrobić aplikację nastawioną na urządzenia mobilne z systemem Android, niestety trzeba dodatkowych frameworków oraz ograniczeń których nie byłoby w przypadku programowania pod Windows Phone. Tym samym programując w GameMakerze istnieje możliwość łatwej zmiany aplikacji tak aby działała pod oboma systemami bez zbędnych formalności.

Podobnie jest z językiem programowania. Pomimo że teoretycznie można w Visual Studio pisać w prawie dowolnym języku programowania, to domyślnym i najbardziej dopracowanym językiem w Visual Studio jest C# w przeciwieństwie do pozostałych dwóch środowisk które skupiają się na języku JavaScript.

1.2. Zadania

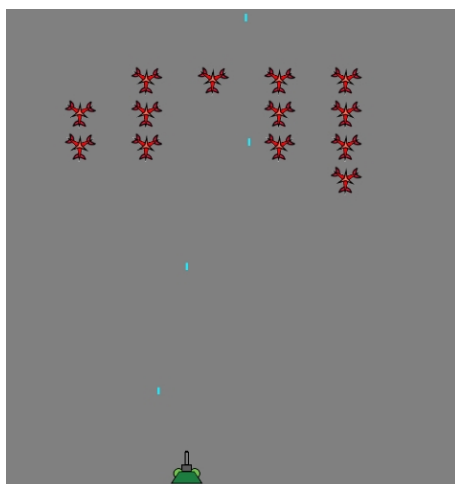
Na pierwszą wersję aplikacji zaplanowane jest osiem różnych zadań. W momencie rozpoczęcia testu wylosowana zostanie połowa testów zaokrąglona w

dół, a czas dany do ich rozwiązania zostanie skrócony tak, by użytkownik zdążył zaznajomić się z treścią zadania lub nawet zdążył zacząć je wykonywać, lecz aby na pewno nie zdążył go wykonać do końca.

Aplikacja jest przygotowana w taki sposób, że umożliwia stosunkowo łatwe dodawanie nowych testów, a co za tym idzie, jest gotowa do rozbudowy i dalszego rozwoju. W przyszłości można do niej dodać nowe testy, by zwiększyć ich różnorodność lub poszerzyć zakres badań o różne warunki ich wykonywania.

1.2.1. Najeźdźcy z kosmosu

W tej grze użytkownik porusza w lewo lub prawo, działko które strzela w określonym odstępie. Celem gry jest zestrzelenie wszystkich statków zanim skończy się czas.



Rysunek 1.1. Wizualizacja: Najeźdźcy z kosmosu!

W wersji utrudnionej, użytkownik dostaje mniej czasu, przeciwnicy szybciej się poruszają a działko strzela wolniej. Czas testu oraz odstęp między strzałami powodują że nawet bezbłędny strzelec nie zdoła zestrzelić wszystkich statków.

1.2.2. Rozbij jajko

Zadaniem użytkownika jest rozbicie jajka. Zrobi to przez stuknięcie w nie odpowiednią ilość razy.



Rysunek 1.2. Wizualizacja: Rozbij Jajko!

W przerywanym wariancie jest mniej czasu oraz potrzeba znacznie większej ilości uderzeń na każdy etap pękania. Teoretycznie w wariancie trudniejszym zadanie wciąż jest do wykonania, lecz jest to trudne do osiągnięcia.

1.2.3. Powódź

Plansza gry składa się z kwadratów których kolor jest losowy spośród sześciu barw. Gdy gracz kliknie jeden z kolorów to przemalowuje dotychczas zalany fragment planszy na dany kolor, i dołącza do powodzi te kwadraty które teraz mają ten sam kolor i stykają się z zalaną powierzchnią. Celem jest spowodowanie aby cała plansza była jednego koloru.

[Screenshot zadania][Do wstawienia]

W trudniejszym wariancie jest mniej czasu oraz znacznie zwiększona jest plansza, co powoduje że potrzeba więcej ruchów na wykonanie zadania.

1.2.4. Uratuj księżniczkę

Gracz musi przeskakiwać po trzech torach i unikać przeciwników na swojej ścieżce. Gdy trafi na jakiegoś, traci parę sekund ograniczonego czasu, tak więc jeśli wpadnie na za dużo przeciwników, nie zdąży dotrzeć do zamku.

[Screenshot zadania][Do wstawienia]

Utrudniona wersja posiada więcej przeciwników oraz szybciej się porusza, przez co trudniej uniknąć zderzenia. Poza tym czas jest krótszy.

1.2.5. Zator

Użytkownik musi odblokować drogę dla wyznaczonego klocka tak aby ten mógł opuścić planszę. Prostokątne klocki można przesuwac tylko wzdłuż dłuższych krawędzi, tak więc tylko do góry i w dół, albo w lewo i prawo.

[Screenshot zadania][Do wstawienia]

Poza skróconym czasem, gracz otrzymuje do rozwiązania o wiele trudniejszą wersję gry. Zagadka posiada więcej elementów i więcej zależnych od siebie akcji przesunięcia.

1.2.6. Lodowy Labirynt

Celem zadania jest przejście przez labirynt. Utrudnieniem jest fakt że gdy zacznie poruszać się w którymś kierunku zatrzymać się i zmienić kierunek może dopiero po napotkaniu ściany. Trzeba planować swoje ruchy pamiętając że poruszyć trzeba się do końca linii.

[Screenshot zadania][Do wstawienia]

Trudniejszy wariant oprócz skróconego czasu, posiada większą ilość przeszkód. Prędkość gracza ulega zmniejszeniu, to oraz zmniejszony czas powodują że nawet przy znajomości dokładnej ścieżki, użytkownik nie jest w stanie ukończyć zadania w wyznaczonym czasie.

1.2.7. Dziurawy labirynt

Tym razem gracz prowadzi przez labirynt kulę, prowadzi ją poprzez przechylanie ekranu w odpowiednią stronę. Trudnością jest fakt że w planszy są dziury których należy unikać. Jeśli kula wpadnie w dół, gra zaczyna się od nowa.

[Screenshot zadania][Do wstawienia]

W trudniejszym wariantcie gry, użytkownik dostaje mniejsza ilość czasu oraz większa ilość przeszkód na drodze do wyjścia.

1.2.8. Super Pamięć

Gra polega na znalezieniu dwóch takich samych obrazków. Za każdym razem gdy odsłonięte są dwa takie same są one wykluczane z gry. Gra się kończy w momencie odkrycia ostatniej pary obrazków. W przypadku błędnego wskazania, po chwili obrazki są ponownie zakrywane.

[Screenshot zadania][Do wstawienia]

W utrudnionym wariancie jest mniej czasu na znalezienie wszystkich par. Poza tym jest o wiele więcej par do znalezienia. Ta gra jest wybrana jako ostatnia ze względu na to że wykorzystane będą w niej obrazki z poprzednich gier. Ma to na celu jednocześnie spowodować aby gracz skupił się na zapamiętywaniu innych rzeczy niż poprzednie gry, i jednocześnie podawać mu obrazkowe skojarzenia z poprzednimi zadaniami.

ROZDZIAŁ 2

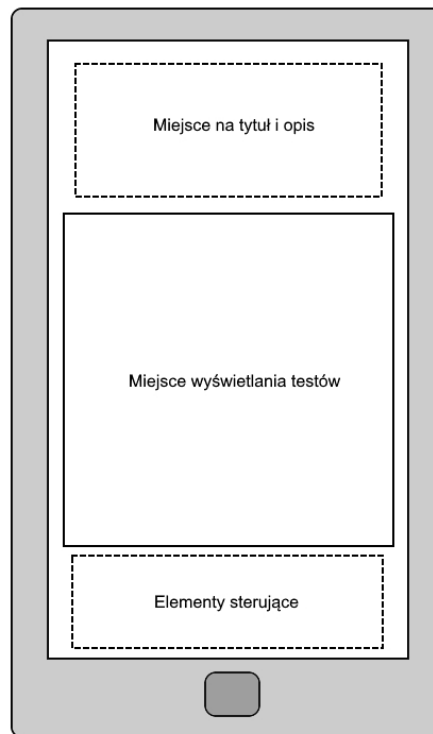
Projekt aplikacji i wybrane technologie

W tym rozdziale przedstawione zostanie jak została zaprojektowana aplikacja. Omówiony zostanie podstawowy interfejs użytkownika oraz jakie zostały użyte technologie. W tym użycie GameMaker: Studio do stworzenia interfejsu oraz oprogramowania zadań w języku skryptowym GML czy użycie zewnętrznego serwera do zapamiętywania danych o wypełnionych ankietach dzięki użyciu GameMaker Server.

2.1. Projekt interfejsu użytkownika

Interfejs użytkownika został wykonany na podstawie pokoi między którymi przemieszcza się badany podczas robienia testu. Wszystkie zadania które trzeba wykonać mają ten sam główny interfejs. Tak jak to przedstawiono na poniższej ilustracji, każde zadanie ma wyznaczone miejsce w którym pojawia się tytuł zadania oraz jego krótki opis. W opisie mieszczą się przede wszystkim warunki zwycięstwa oraz wytłumaczenie sterowania.

Większość ekranu jest przeznaczona na teren gry. Szara strefa wyznacza granice gry, i w tej strefie zawsze pojawiają się wszystkie potrzebne obiekty. Każda gra wczytuje się na swój sposób, podczas wczytywania każdego z pokoi konkretne zadanie ładuje odpowiednie zmienne które są inne w zależności czy gra ma być przez użytkownika skończona czy też ma być mu przerwana.



Rysunek 2.1. Projekt interfejsu użytkownika

Pod miejscem na grę znajduje się dodatkowa przestrzeń na elementy sterujące. Tam się znajdują przyciski pozwalające chociażby przejście do następnego zadania w przypadku ukończenia zadania, niezależnie od tego czy skończył mu się czas czy też zdołał przejść zadanie.

2.2. GameMaker: Studio

Środowisko programistyczne GameMaker: Studio przeznaczone jest do tworzenia gier i aplikacji w technologii dwuwymiarowej. Korzysta z własnego języka skryptowego GML. Tak jak już pisałem wcześniej, wybrałem to środowisko ze względu na ułatwienie w tworzeniu podstawowych funkcji, dzięki temu można lepiej skupić się na tworzeniu tych skryptów które są ważne i tworzą logikę gry.

Dla każdego z zadań jest przeznaczony osobny pokój w którym generują się wszystkie niezbędne do przeprowadzenia gry, rzeczy. Pomimo że jest to oprogramowanie które pozwala na bardzo dużą swobodę wstawiania elementów za pomocą trybu "złap i upuść" w mojej pracy większość pokoi posiada jedynie umieszczone w nich dwa lub trzy kontrolery które dynamicznie uzupełniają resztę zawartości w zależności od tego co się dzieje z aplikacją.

2.3. Gromadzenie danych - GameMaker Server

Głównym celem tej aplikacji jest zbieranie danych w celu badania efektu Zeigarnik. Wyniki każdego użytkownika są spisywane po stronie serwera. W tym celu wykorzystywane jest rozszerzenie do GameMaker: Studio o nazwie GameMaker Server, który pozwala na bezproblemowe założenie serwera i obsługę plików INI po stronie serwera. Każdy użytkownik dostaje swój plik w który automatycznie wpisują się wszystkie wyniki testów, to czy został wykonany poprawnie czy skończył się czas, oraz adnotację do tego które z zadań były ograniczone.

Poza tym, w pliku na serwerze zostają zapisane także informacje przekazywane z ankiety. W ankiecie użytkownik proszony jest o przypomnienie sobie jak największej ilości tytułów oraz podania krótkiego opisu co musiał wykonać w tych zadaniach.

GameMaker Server przechowuje dane w pliku INI na zasadzie klucz i wartość oraz mogą być posegregowane w odpowiednie sekcje. Tak więc wyniki są podzielone na dwie sekcje, pierwszą w której mieszczą się informacje wypełniane na bieżąco przez aplikację, takie jak które zadania zostały wybrane jako te które należy przerwać, oraz po każdym zadaniu informacja czy zadanie zostało wykonane czy też skończył się czas. Druga sekcja jest wydzielona na odpowiedzi w ankiecie, gdzie użytkownik może teoretycznie wpisać dowolną ilość odpowiedzi tytuł-opis dopóki nie stwierdzi że to wszystko co sobie przypomina.

Prostota tych danych pozwala na łatwą analizę odpowiedzi. W przyszłości

można stworzyć skrypt analizujący dane wejściowe gdy tylko zostają dostarczone. Jedynym problemem jest to że użytkownik może wpisywać własne odpowiedzi, co za tym idzie, czasem trudno jest powierzyć analizę tekstu skryptowi który może źle zinterpretować o które zadanie chodziło użytkownikowi.

2.4. Pliki INI i organizacja danych

Plik INI jest to prosty format zapisu danych który pozwala na przechowywanie danych. Postanowiłem wykorzystać go w mojej aplikacji ze względu na prostotę stworzenia osobnego pliku dla każdego badanego. W pliku zapisuje dane na zasadzie klucz-wartość, jak w przypadku zwykłych baz danych. Ułatwia to jednak dalszą interpretację dostarczanych danych ze względu na trudność w interpretacji odpowiedzi graczy. Oprócz podstawowego podsumowania czy ukończono lub nie, konkretne zadanie, najważniejszym aspektem jest to które zadania użytkownik był w stanie zapamiętać.

Listing 2.1. Fragment pliku INI

```
[test_made]
worse = 00110101
space_invaders = true
egg_smasher = true
flood_it = false
```

Powyższy przykład obrazuje że podstawowo w pliku zapisany jest ciąg zero-jedynkowy który określa które zadanie według kolejności posiadały utrudniony wariant, a które były normalne. Następnie aplikacja wpisuje wynik każdego zadania gdy tylko użytkownik je zakończy. Wpisując odpowiednio nazwę pokoju w którym aktualnie się znajdujemy oraz wartość prawdy, jeśli ukończyliśmy zadanie w czasie, lub fałsz, jeśli skończył nam się czas. Tak jak to przedstawia poniżej ukazany fragment skryptu wykonywanego w przypadku gdy czas się skończy. Jako że ważne jest jak zostało wykonane zadanie, do wszystkich gier zostały użyte te same skrypty kończące rozgrywkę.

Listing 2.2. Fragment skryptu w przypadku końca czasu

```
room_name = room_get_name(room);  
gms_ini_player_write("test_made", room_name, false);
```


ROZDZIAŁ 3

Szczegóły implementacji

W następującym rozdziale zajmę się sposobem implementacji poszczególnych pokoi. Głównie można podzielić wszystkie pokoje na pokoje inicjujące grę, w których należy się zalogować na serwer w celu przekazywania danych i ostrzeżeniu że cały test jest robiony na czas więc trzeba znaleźć wystarczająco czasu aby wykonać go w całości w jednym ciągu. Drugą kategorią pokoi są te w których znajdują się testy, każdy z nich ma zaimplementowaną w sobie inną grę. Oprogramowanie jest tak przystosowane żeby pokoje były od siebie niezależne, a wszystkie potrzebne wyniki zapisywały się bez potrzeby dopisywania dodatkowych skryptów. Tym samym dodanie nowego pokoju z grą wymaga jedynie wpisania go na listę startową oraz zaimplementowanie samej gry w tym pokoju. Trzecim i ostatnim rodzajem pokoju są pokoje końcowe, przeznaczone na ankietę po wykonaniu ostatniego zadania.

3.1. Inicjalizacja testu

Pierwsze trzy pokoje służą do zalogowania się do serwera dzięki czemu możliwy będzie zapis informacji o teście w pliku zdalnym. W trzecim pokoju znajduje się informacja o tym że czas na wykonanie każdego zadania będzie ograniczony, oraz przycisk rozpoczynający test. W momencie kliknięcia na ten przycisk, program losuje połowę pokoi zaokrągloną w dół i zmienia ich poziom trudności tak aby nie dało się ich ukończyć w czasie.

Listing 3.1. Fragment skryptu `scr_make_times`

```
for ( i=0; i<floor ( tasks /2 ); i++){  
    checked = false ;  
    while (!checked){
```

```
x = floor(random_range(0,(tasks - 0.1)))
if (!global.worse[x]) {
    global.worse[x] = true;
    checked = true;
}
}
```

Za wylosowanie pokoi odpowiedzialny jest skrypt o nazwie `scr_make_times`, którego fragment widzimy powyżej. Fragment ten pokazuje sposób w który wylosowywane są pokoje. Tablica `worse` jest globalną zmienną i posiada tyle miejsc ile jest zadań. Na każdym miejscu, dla konkretnego zadania, ma domyślnie wpisaną wartość `false`, która oznacza że gra nie jest utrudniona. Potem algorytm losuje pokój i zmienia wartość `worse` dla tego pokoju na `true` które powoduje utrudnienie zadania.

3.2. Pokoje z grami

Domyślnie każdy pokój ładuje się w ten sam sposób, tak jak było to zaprojektowane, posiada miejsce w którym pokazuje się gra. Na górze jest miejsce na tytuł zadania oraz wyjaśnienie zasad. Każdy z pokoi w którym są gry, ładuje się z zestawem zmiennych potrzebnych do działania gry. Zależnie od wcześniej wylosowanej wartości `worse` dla danego pokoju, wczytywane są inne wartości potrzebnych zmiennych.

3.2.1. Najeźdźcy z Kosmosu

W tej grze, gracz porusza wieżą obronną. Przeciwnicy poruszają się w określony sposób, sama wieża strzela automatycznie po określonym czasie. Najważniejszym aspektem tej gry jest sterowanie wieżą która może być poruszana na lewo i prawo.

Listing 3.2. Skrypt `scr_direction`

```

if (point_direction(xstart, ystart, x, y) > 270
    or point_direction(xstart, ystart, x, y) < 90){
    obj_tower.direction_s = 1;
}

if (point_direction(xstart, ystart, x, y) > 90
    and point_direction(xstart, ystart, x, y) < 270){
    obj_tower.direction_s = 2;
}

```

Gdy użytkownik przesunie palcem po ekranie, gra zapisuje współrzędne początku oraz końca ruchu. Potem porównuje te dwa punkty, sprawdzając pod jakim kątem znajdują się one od siebie. Następnie zależnie od kąta przesunięcia palca, wieża może zmienić swój kierunek. Powyżej zawarty skrypt ukazuje sprawdzenie kąta przesunięcia oraz wyboru w którą stronę powinna przemieszczać się wieża.

3.2.2. Rozbij jajko

Rozbicie jajka polega na stuknięciu w nie odpowiednią ilość razy. Zawsze gdy otrzyma odpowiednią ilość uderzeń coraz bardziej pęka, zmieniając się wizualnie.

Listing 3.3. Fragment kodu obiektu obj_egg

```

if (current_hit == 0){
    if (image_index < 4){
        image_index += 1;
        current_hit = hit_count;
    } else {
        image_index += 1;
        depth = -150
        scr_end_win();
    }
}
}

```

Powyższy kod ukazuje sprawdzenie warunku zwycięstwa przy każdym kolejnym uderzeniu. Jeśli została spełniona ilość ciosów, należy dodać złamań. Jeśli użytkownik rozbije w końcu jajko, nastąpi ostatnia zmiana na obraz stłuczonego jajka oraz ekran końca gry.

3.2.3. Powódź

Ta gra składa się na używanie sześciu przycisków, które zmieniają kolor, dotychczas pochłoniętych kwadratów. Po pokolorowaniu kwadratów które posiadają wartość "flood_flooded

i, n

true", program sprawdza wszystkie przyległe pola(nie po skosie) w poszukiwaniu takiego samego koloru na jaki się pomalowało resztę. W przypadku kiedy znajdzie takie sam kolor, wchłania go zmieniając parametr flood_flooded z false na true i rekurencyjnie sprawdza dla tego nowego kwadratu, czy on też nie znajduje się obok innych o tym samym kolorze, które nie zostały jeszcze pochłonięte.

Listing 3.4. Fragment skryptu scr_retrue_colors

```
//checking up
if (i-1 >= 0 && obj_flood.flood_color[i-1,n] == arg_color){
    if (obj_flood.flood_flooded[i-1,n] != true){
        obj_flood.flood_flooded[i-1,n] = true;
        scr_retrue_colors(i-1,n,arg_color);
    }
}
```

Powyższy fragment ukazuje sprawdzenie czy nad aktualnie pochłoniętym kwadratem nie znajduje się taki o tym samym kolorze, analogicznie wyglądają sprawdzenia dla kwadratu pod spodem, jak i z prawej i lewej strony. Oczywiście nie sprawdzamy dalszą rekurencją tych kwadratów które zostały już pochłonięte.

3.2.4. Uratuj księżniczkę

W tej grze, gracz musi unikać spotkań z wrogiem aby jak najszybciej dostać się do księżniczki. Aby stworzyć iluzję nieskończonego poruszania się do przodu, to nie bohater się porusza lecz jego przeciwnicy. W przypadku znalezienia się na brzegu planszy, każdy przeciwnik zostaje zwrócony na drugi brzeg, wraz z losowym torem na którym się pojawia, tak jak demonstruje to poniższy kod.

Listing 3.5. Fragment kodu obiektu `obj_orc`

```
if (x < 120){
    y = choose(750, 1000, 1250);
    x = 960;
    obj_hero.win_count -= 1;
}
```

Jeśli bohater trafi na przeciwnika to przestaje na chwilę biec aby go pokonać. Jeśli trafi na za dużą liczbę przeciwników, skończy mu się czas i nie zdąży dobiec do księżniczki.

3.2.5. Zator

Zator polega na odsunięciu z drogi wszystkich klocków które zasłaniają wyjście zielonemu blokowi. Problemem jest fakt że wszystkie klocki można przesuwac tylko wzdłuż ich dłuższej krawędzi. Zważając na założenia tej gry, najważniejszym jej aspektem jest sprawdzanie kolizji pomiędzy klockami i zapobieganie ewentualnemu nakładaniu się na siebie elementów.

Listing 3.6. Fragment kodu obiektu `obj_green_one`

```
if (!(place_meeting(mouse_x + xx, y, obj_vertical_one)
    or place_meeting(mouse_x + xx, y, obj_horizontal_one))
    and (mouse_x+xx < 850 and mouse_x+xx > 230)){
    if (grab == false){
```

```
        exit ;  
    } else {  
        x = mouse_x + xx ;  
    }  
}
```

Jak widać na załączonym powyżej fragmencie obiektu zielonego bloku, w przypadku poruszenia jakimkolwiek blokiem, program wpierw sprawdza czy wykonany ruch nie spowoduje kolizji i dopiero po upewnieniu się że nic nie stoi na drodze, przesuwa blok tak jak było to zaplanowane.

3.2.6. Lodowy Labirynt

3.2.7. Dziurawy labirynt

3.2.8. Super Pamięć

3.3. Ankieta

Zakończenie

PODSUMOWANIE TUTAJ

Spis tabel

Spis rysunków

1.1. Wizualizacja: Najeźdźcy z kosmosu!	11
1.2. Wizualizacja: Rozbij Jajko!	12
2.1. Projekt interfejsu użytkownika	16

Listings

2.1. Fragment pliku INI	18
2.2. Fragment skryptu w przypadku końca czasu	19
3.1. Fragment skryptu scr_make_times	21
3.2. Skrypt scr_direction	22
3.3. Fragment kodu obiektu obj_egg	23
3.4. Fragment skryptu scr_retrue_colors	24
3.5. Fragment kodu obiektu obj_orc	25
3.6. Fragment kodu obiektu obj_green_one	25

Oświadczenie

Ja, niżej podpisany(a) oświadczam, iż przedłożona praca dyplomowa została wykonana przeze mnie samodzielnie, nie narusza praw autorskich, interesów prawnych i materialnych innych osób.

.....

data

.....

podpis