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”, który oferuje między innymi unikalny język skryptowy GML. Napisano osiem testów, które użytkownik ma za zadanie rozwiązać w określonym czasie.

Zaimplementowane zostało również przesyłanie danych z testów i ankiet na zewnętrzny serwer do dalszego przetworzenia. Użyte zostało rozszerzenie GameMaker Server, za pomocą którego dane są przechowywane w plikach o rozszerzeniu .ini. Dla każdego użytkownika tworzy się osobny plik.

Większość grafik potrzebnych do funkcjonowania aplikacji została wykonana przeze mnie w programie Paint.NET. Wszystkie zapożyczone grafiki zostały uwzględnione w bibliografii.

Aplikacja została wdrożona i umieszczona w repozytorium GitHub. Jest dostępna do pobrania ze strony: https://tinyurl.com/ydd98sny . Kod źródłowy całej aplikacji także znajduje się w tym repozytorium i jest dostępny pod adresem: https://tinyurl.com/ycrsr5uu.

Dla ułatwienia dostępu do aplikacji przygotowana została wersja testowa, możliwa do zainstalowania na urządzeniu z systemem Android jak i Windows. Jako że aplikacja jest przystosowana do systemu Android, używanie jej pod systemem Windows może nie dawać oczekiwanych rezultatów. W pliku README.md znajdują się wymagane do instalacji informacje. https://tinyurl.com/y8t733ev .

W zaprojektowaniu testów brał udział magister psychologii Bartosz Tomkiewicz. Po zaimplementowaniu i wdrożeniu aplikacji, została ona przedstawiona grupie osób, które miały wykonać test. Wyniki zostały zanalizowane i przedstawione na końcu pracy.

Słowa kluczowe

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

Spis treści

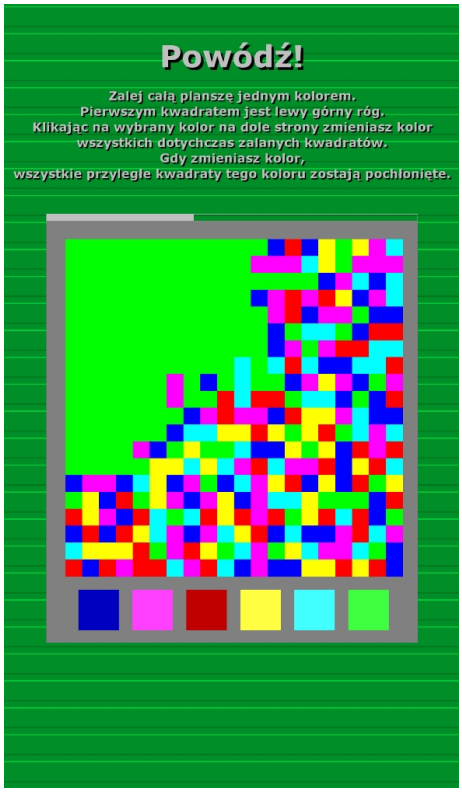
Wprowadzenie

Zamiar niekoniecznie oznacza z góry określoną okazję do jego zrealizowania, lecz potrzebę lub tymczasową potrzebę, która stwarza taką okazję. Bluma 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. Zeigarnik[1] w 1927 roku. Objaśnia on pojęcie psychologiczne związane z zagadnieniami pamięci i motywacji psychologii ogólnej oraz w rezultacie wykazuje, że ludziom łatwiej jest przypomnieć sobie czynności, które zostały im przerwane niż te, które wykonali bez żadnych problemów do końca. Przykładem tego efektu i jednocześnie tym, co zainspirowało Blumę Zeigarnik do badań są kelnerki w restauracjach, które potrafią zapamiętać na raz zamówienia nawet kilku obsługiwanych w danym momencie osób, ale mają znaczne problemy z przypomnieniem sobie klientów, którzy opuścili już lokal. Pomysł na aplikację, która pomaga badać ten efekt pojawił się podczas rozmowy ze znajomym, Bartoszem Tomkiewiczem, który ukończył studia magisterskie na kierunku psychologia i który postanowił pomóc mi z przygotowaniem zadań oraz testów, które najlepiej będą nadawać się do badania Efektu Zeigarnik. Uczestniczył również w analizie wyników uzyskanych przy pomocy aplikacji.

W części teoretycznej pracy zostaną opisane główne założenia przyjęte podczas projektowania aplikacji, sposoby rozwiązania konkretnych związanych z nimi problemów oraz możliwości samej aplikacji w zakresie przetwarzania informacji dostarczanych jej przez użytkowników.

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



Rysunek 1. Wizualizacja: Przykładowy wygląd testu w aplikacji!

Projekt został wykonany na zasadzie aplikacji mobilnej. Aplikacja została 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 z zakresu implementacji programów i gier w środowiskach do tego przystosowanych. Wymieniona technologia zostanie 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 zakończeniem.

Po wykonaniu wszystkich testów użytkownik dostanie do wypełnienia ankietę, która pozwoli ustalić, co zapamiętał. 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 uzasadnić swój wybór.

1.1.1. Unity a GameMaker

Środowisko Unity jest bardzo podobne do wybranego przeze mnie środowiska GameMaker. Służy głównie do tworzenia trójwymiarowych lub dwuwymiarowych gier komputerowych lub materiałów interaktywnych takich jak wizu-

alizacje czy animacje. Tak samo jak GameMaker, może być wykorzystany 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, do którego są bardzo podobne składnią.

Głównym aspektem, który przekonał mnie do GameMakera był fakt, że Unity jest narzędziem przede wszystkim do pracy nad elementami 3D, podczas gdy mi zależało na aplikacji w dwóch wymiarach. Nie jest oczywiście niemożliwe stworzenie dwuwymiarowej aplikacji w Unity, ale, 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 nastawiony przede wszystkim na systemy wspierające Windowsa. Co za tym idzie, zrobienie aplikacji na urządzenia mobilne z systemem Android, wymaga niestety dodatkowych frameworków oraz ograniczeń, których nie byłoby w przypadku programowania pod Windows Phone. Zaletą GameMakera jest tutaj zdecydowanie możliwość łatwej zmiany aplikacji tak, aby działała pod oboma systemami bez zbędnych utrudnień.

Podobnie jest z językiem programowania. Pomimo, że teoretycznie Visual Studio przystosowane jest do bardzo wielu języków programowania, domyślnym i najbardziej dopracowanym językiem jest tam C# (odróżnieniu od Unity i GameMakera, które skupiają się na języku JavaScript).

1.2. Zadania

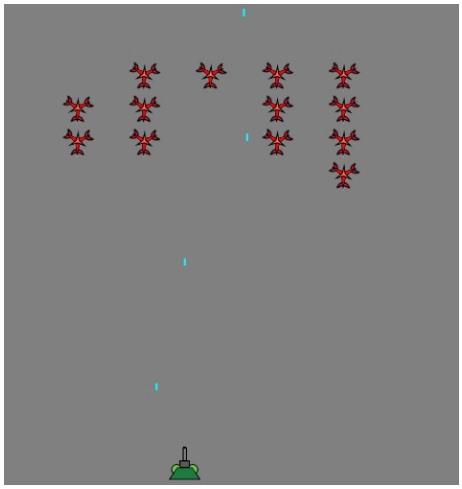
Na pierwszą wersję aplikacji zaplanowane jest osiem różnych zadań. W momencie rozpoczęcia testu wylosowana zostanie połowa testów zaokrąglona w dół, a czas dany do ich rozwiązania zostanie skrócony tak, by użytkownik

zdążył zaznajomić się z poleceniem lub nawet zacząć je wykonywać, lecz aby na pewno nie zdążył go ukończyć.

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 działem, które strzela w górę określonym odstępie czasowym. 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 działo strzela wolniej. Czas testu oraz odstęp między strzałami powodują, że zestrzelenie wszystkich statków jest niewykonalne.

1.2.2. Rozbij jajko

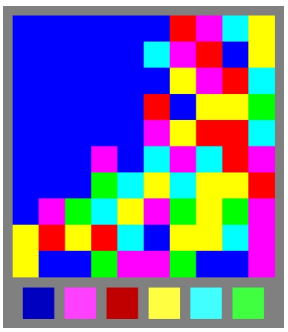
Zadaniem użytkownika jest rozbicie jajka poprzez stuknięcie w nie odpowiednią ilość razy.

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 możliwe 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, zamalowuje nim dotychczas zalany fragment planszy i dołącza do powodzi te kwadraty, które teraz mają ten sam kolor co zalana powierzchnia i się z nią stykają.

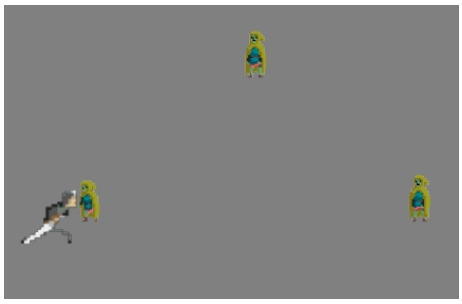


Rysunek 1.3. Wizualizacja: Powódź!

Zadaniem użytkownika jest spowodowanie, aby cała plansza była jednego koloru. W trudniejszym wariancie czas jest skrócony, a plansza znacznie zwiększona, co powoduje, że potrzeba więcej ruchów, by wykonać zadanie.

1.2.4. Uratuj księżniczkę

Gracz steruje rycerzem, który musi przeskakiwać między trzema torami, by uniknąć przeciwników na swojej ścieżce. Gdy trafi na jakiegoś, traci parę sekund ograniczonego czasu, tak więc jeśli wpadnie na zbyt wielu przeciwników, nie zdąży dotrzeć do zamku.



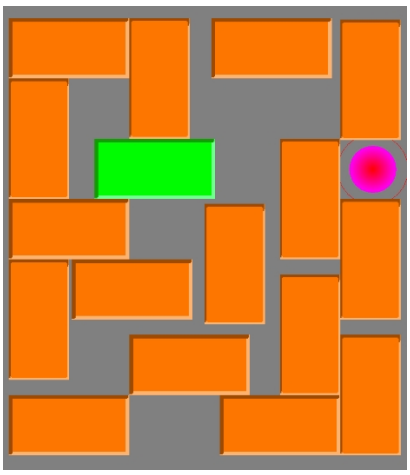
Rysunek 1.4. Wizualizacja: Uratuj księżniczkę!

Utrudniona wersja posiada skrócony czas oraz zwiększoną prędkość poruszania się postaci, przez co trudniej uniknąć zderzenia.

1.2.5. Zator

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

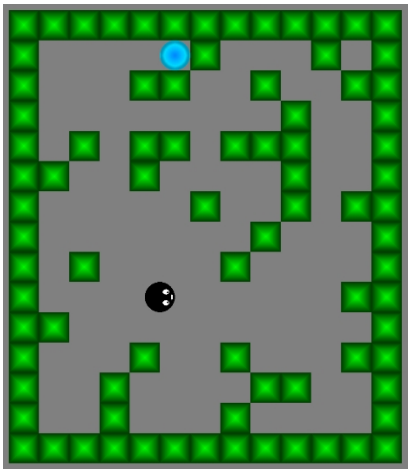
W utrudnionej wersji, poza skróconym czasem, gracz otrzymuje do rozwiązania o wiele trudniejszą wersję gry. Zagadka posiada więcej elementów i więcej zależnych od siebie akcji przesunięcia.



Rysunek 1.5. Wizualizacja: Zator!

1.2.6. Lodowy Labirynt

Celem zadania jest przejście przez labirynt. Utrudnieniem jest fakt, że gdy zacznie się poruszać w którymś kierunku, zmienić go może dopiero po napotkaniu ściany. Należy planować swoje ruchy pamiętając, że poruszać trzeba się do końca linii.



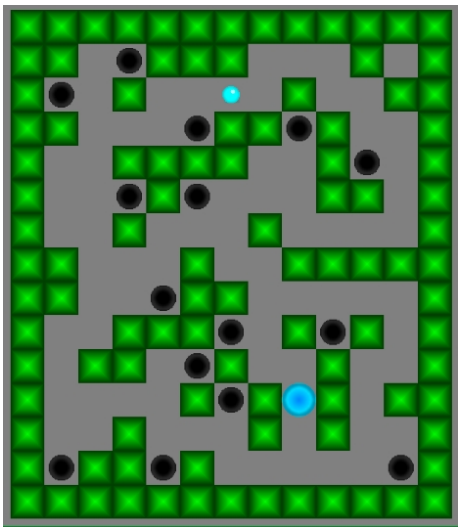
Rysunek 1.6. Wizualizacja: Lodowy labirynt!

Trudniejszy wariant, oprócz skróconego czasu, posiada inaczej umiejsco-

wione przeszkody. Prędkość gracza ulega zmniejszeniu, przez co nawet przy znajomości dokładnej ścieżki, użytkownik nie jest w stanie ukończyć zadania w wyznaczonym czasie.

1.2.7. Dziurawy labirynt

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

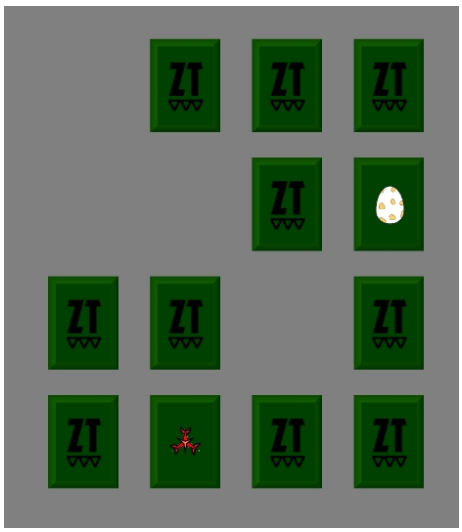


Rysunek 1.7. Wizualizacja: Dziurawy labirynt!

W trudniejszym wariancie gry, użytkownik dostaje mniejszą ilość czasu oraz większą ilość przeszkód na drodze do wyjścia.

1.2.8. Super Pamięć

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



Rysunek 1.8. Wizualizacja: Super Pamięć!

W utrudnionym wariancie czas na znalezienie par jest skrócony, a karty dłużej zostają odsłonięte. 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, a jednocześnie podawać mu obrazkowe skojarzenia z wcześniejszymi zadaniami.

ROZDZIAŁ 2

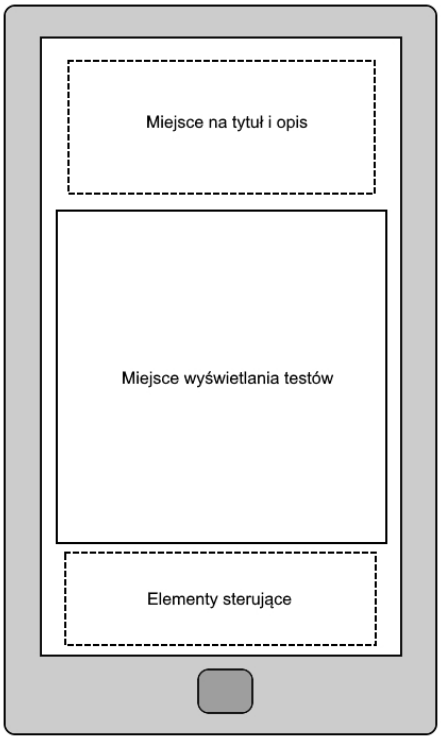
Projekt aplikacji i wybrane technologie

W tym rozdziale omówione zostanie, jak została zaprojektowana aplikacja. Przedstawiony zostanie podstawowy interfejs użytkownika oraz użyte technologie, w tym użycie GameMaker: Studio do stworzenia interfejsu i oprogramowania zadań w języku skryptowym GML, jak również użycie zewnętrznego serwera do zapamiętywania danych o wypełnionych ankietach przy pomocy 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 jego tytuł oraz 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 to w niej zawsze pojawiają się wszystkie potrzebne obiekty. Każda gra wczytuje się na swój sposób. Podczas wczytywania się każdego z pokoi, konkretne zadanie ładuje odpowiednie zmienne, które są inne w zależności od tego, czy gra ma być przez użytkownika skończona, czy też ma mu zostać przerwana.



Rysunek 2.1. Projekt interfejsu użytkownika

Pod miejscem na grę jest dodatkowa przestrzeń na elementy sterujące. Tam znajdują się przyciski pozwalające m.in. na przejście do następnego zadania w przypadku zakończenia jednego (niezależnie od tego, czy zadanie zostało ukończone, czy też przerwane przez upłynięcie wyznaczonego czasu).

2.2. GameMaker: Studio

Środowisko programistyczne GameMaker: Studio przeznaczone jest do tworzenia gier i aplikacji w technologii dwuwymiarowej. Korzysta z własnego języka skryptowego GML. Jak już pisałem wcześniej, wybrałem to środowisko ze względu na ułatwione pisanie podstawowych funkcji, dzięki czemu można bardziej skupić się na tych skryptach, które są ważne i tworzą logikę gry.

Dla każdego z zadań przeznaczony jest osobny pokój, w którym generują się wszystkie elementy niezbędne do przeprowadzenia gry. Pomimo, że jest to oprogramowanie pozwalające na bardzo dużą swobodę wstawiania elementów za pomocą trybu "złap i upuść", większość pokoi w mojej pracy posiada jedynie dwa lub trzy kontrolery, które dynamicznie uzupełniają resztę zawartości w zależności od tego, co aktualnie powinno dziać się 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, które pozwala na bezproblemowe założenie serwera i obsługę plików INI po jego stronie. Każdy użytkownik dostaje swój plik, w który automatycznie zapisują się wszystkie wyniki jego testów, dane o tym, które zadania zostały wykonane poprawnie, a przy których skończył się czas oraz adnotację do tego, które z zadań były ograniczone.

Poza tym, w pliku na serwerze zostają zapisane także informacje przekazane w ankiecie, w której to użytkownik proszony jest o przypomnienie sobie jak największej ilości tytułów zadań oraz o podanie ich krótkiego opisu.

GameMaker Server przechowuje dane w pliku INI na zasadzie klucz i wartość, które 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, np. które zadania zostały utrudnione lub czy zadanie zostało wykonane czy też skończył się czas oraz drugą, w której gromadzone są odpowiedzi z ankiety, gdzie użytkownik może wpisać dowolną liczbę odpowiedzi tytuł-opis, dopóki nie stwierdzi, że nic więcej sobie nie 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 wpisuje własne odpowie-

dzi, a co za tym idzie, powierzenie analizy tekstu skryptowi nie zawsze da adekwatne wyniki, ponieważ może on źle zinterpretować odpowiedź i przypisać ją do nieodpowiedniego zadania.

2.4. Pliki INI i organizacja danych

Plik INI to prosty format zapisu danych, który pozwala na ich przechowywanie. Postanowiłem wykorzystać go w mojej aplikacji ze względu na łatwość stworzenia osobnego pliku dla każdego użytkownika. W pliku zapisuje dane na zasadzie klucz-wartość tak, jak w przypadku zwykłych baz danych.

Ułatwia to dalszą interpretację dostarczanych danych ze względu na trudność w interpretacji odpowiedzi graczy. Oprócz podstawowego podsumowania, czy konkretne zadanie zostało ukończone lub nie, najważniejszym aspektem jest to, które zadania użytkownik był w stanie zapamiętać.

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

Listing 2.1. Fragment pliku INI

Powyższy przykład pokazuje, że podstawowo w pliku zapisany jest ciąg zerojedynkowy określający, które zadanie według kolejności posiadały utrudniony wariant, a które były normalne. Następnie aplikacja zapisuje wynik każdego zadania, gdy tylko użytkownik je zakończy.

Wpisując odpowiednio nazwę pokoju, w którym aktualnie znajduje się użytkownik oraz wartość „prawda”, jeśli zadanie zostało przez niego ukończone w czasie lub „fałsz”, jeśli czas się skończył (tak jak w ukazanym poniżej fragmencie skryptu wykonywanego w przypadku, gdy czas się skończy). Ja-

ko, że ważne jest, jak zostało wykonane zadanie, do wszystkich gier użyto tych samych skryptów kończących rozgrywkę.

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

Listing 2.2. Fragment skryptu w przypadku końca czasu

ROZDZIAŁ 3

Szczegóły implementacji

W następującym rozdziale zajmę się sposobem implementacji poszczególnych pokoi. Głównie można podzielić je na trzy kategorie:

-Pokoje inicjujące grę, w których należy zalogować się na serwer w celu przekazania danych oraz odczytania ostrzeżenia na temat limitu czasowego poszczególnych zadań i informacji, iż muszą one zostać wykonane w całości w jednym ciągu.

-Pokoje testów, z których każdy ma zaimplementowaną w sobie inną grę. Oprogramowanie jest tak przystosowane, żeby pokoje były od siebie niezależne, a wszystkie potrzebne wyniki zapisywały się bez konieczności dopisywania dodatkowych skryptów. Tym samym, dodanie nowego pokoju z grą wymaga jedynie wpisania go na listę startową oraz zaimplementowania samej gry w tym pokoju. Pokoje końcowe, przeznaczone na ankietę, którą użytkownik wypełnia 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 limicie czasowym zadań oraz przycisk rozpoczynający test. W momencie kliknięcia na niego, program losuje połowę pokoi (zaokrągloną w dół) i zmienia ich poziom trudności tak, aby nie dało się ich ukończyć w wyznaczonym czasie.

```
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;
        }
    }
}
```

Listing 3.1. Fragment skryptu scr_make_times

Za wylosowanie pokoi odpowiedzialny jest skrypt o nazwie scr_make_times, którego fragment widzimy powyżej. Fragment ten pokazuje sposób, w który losowane 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.

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 i, 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 wraz z zestawem zmiennych potrzebnych do jej działania. 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, a sama wieża strzela automatycznie w określonych odstępach czasu. Najważniejszym aspektem gry jest sterowanie wieżą w lewo i w prawo.

```
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;
}
```

Listing 3.2. Skrypt scr_direction

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 względem siebie. Następnie, zależnie od kąta przesunięcia palca, wieża może zmienić swój kierunek. Zawarty powyżej skrypt ukazuje sprawdzenie kąta przesunięcia oraz wybór, 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, pęka coraz bardziej, zmieniając się wizualnie.

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

Listing 3.3. Fragment kodu obiektu obj_egg

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

3.2.3. Powódź

Ta gra polega na używaniu sześciu przycisków w celu zmienienia koloru 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 pomalowana została reszta. W przypadku kiedy znajdzie taki 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.

```
//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);
    }
}
```

Listing 3.4. Fragment skryptu scr_retrue_colors

Powyższy fragment ukazuje sprawdzenie, czy nad aktualnie pochłoniętym kwadratem nie znajduje się inny o tym samym kolorze. Analogicznie wyglądają sprawdzenia dla kwadratu pod spodem, jak i z prawej i z 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 przo-

du, to nie bohater się porusza lecz jego przeciwnicy. W przypadku dojścia do lewego brzegu planszy, przeciwnik pojawia się na losowym torze na prawym brzegu, tak jak demonstruje to poniższy kod.

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

Listing 3.5. Fragment kodu obiektu obj_orc

Jeśli bohater trafi na przeciwnika to przestaje na chwile 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 przesuwać tylko wzdłuż ich dłuższej krawędzi. Zważając na założenia tej gry, najważniejszym jej aspektem jest sprawdzanie kolizji pomiędzy klockami i zapobieżenie ewentualnemu nakładaniu się na siebie elementów.

```
if(!(place_meeting(mouse_x + xx, y, obj_vertical_one)
    or place_meeting(mouse_x + xx, y, obj_horizontal_one))
    and (mouse_x+xx &lt;850 and mouse_x+xx &gt; 230)){
        if(grab == false){
            exit;
        } else{
            x = mouse_x + xx;
        }
}
```

Listing 3.6. Fragment kodu obiektu obj_green_one

Jak widać na załączonym powyżej fragmencie kodu 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

Założenie Lodowego Labiryntu jest takie, że gdy zaczniemy się poruszać, nie możemy zmienić kierunku dopóki nie dotrzemy do ściany, która nas zatrzyma. Tym samym, aplikacja została zaprogramowana, by analizować ruch palca przy przesunięciu i na jego podstawie wyznaczyć kierunek ruchu. W momencie rozpoczęcia ruchu blokujemy skrypt odpowiedzialny za wybieranie kierunku.

```
    if(place_meeting(x-speed_ice, y, obj_lab_wall)){
        while(!place_meeting(x-1, y, obj_lab_wall)){
            x -= 1;
        }
```

Listing 3.7. Fragment kodu obiektu obj_player_ice

W przytoczonym fragmencie kodu widzimy, jak co klatkę sprawdzane jest położenie obiektu i czy nie styka się on już ze ścianą. Jest to fragment sprawdzający kolizję w przypadku ruchu w lewo. Dla każdego kierunku kod ten różni się ze względu na inne położenie na osiach.

3.2.7. Dziurawy labirynt

W przeciwieństwie do innych zadań, w tym nie sterujemy za pomocą ruchu palca, lecz poprzez przechylanie urządzenia. Program na bieżąco bada, czy i która oś jest przekręcona i dostosowuje do tego ruch kuli. W przypadku wpadnięcia w dziurę, poziom się resetuje.

```
if(place_meeting(x+xx, y, obj_lab_wall)){
    while(!place_meeting(x-device_get_tilt_x(), y,
   obj_lab_wall)){
```

```
        x += -device_get_tilt_x();
    }
}else{
    x += xx;
}
```

Listing 3.8. Fragment kodu obiektu obj_player_ball

Takie zachowanie wymagało innego algorytmu kolizji, ponieważ ważne było, aby kula przesuwała się po krawędzi w przypadku przechylenia urządzania w jej stronę. Powyższy przykład ukazuje tę funkcję dla osi X.

3.2.8. Super Pamięć

W tej grze, która jest ostatnią w całym teście, wykorzystane zostały grafiki z poprzednich zadań. Gdy gracz odkrywa pierwszą kartę, zapamiętane zostaje, jaki obrazek się pod nią krył. Gdy wybiera drugą kartę, program sprawdza zgodność przypisanego jej obrazu z tym z poprzedniej karty. Po tym sprawdzeniu decyduje się na jedną z dwóch akcji.

```
    if(obj_memory_shuffle.now_turned == 1){
        obj_memory_shuffle.now_shown = obj_memory_shuffle.
   card_deck[| id_fac];
        obj_memory_shuffle.now_shown_id = id;
    } else if(obj_memory_shuffle.now_turned == 2){
        if(obj_memory_shuffle.now_shown ==
   obj_memory_shuffle.card_deck[| id_fac]){
            alarm[0] = obj_memory_shuffle.delay_time;
        }else{
            alarm[1] = obj_memory_shuffle.delay_time;
        }
    }
```

Listing 3.9. Fragment kodu obiektu obj_memory_card

W przypadku kiedy rysunki się ze sobą zgadzają, wywoływany zostaje alarm, który powoduje usunięcie obu instancji odkrytych kart. W drugim przypadku, gdy karty się ze sobą nie zgadzają, wywoływany jest alarm powodujący zakrycie kart z powrotem. Niezależnie od tego, który alarm zostanie wywołany, informacje o ostatnio odkrytej karcie zostają zresetowane.

3.3. Ankieta

Na potrzeby zakończenia badania, została zaimplementowana prosta ankieta, w której użytkownik ma za zadanie wpisać tytuły oraz krótkie opisy zadań, które wykonywał. Ankieta składa się na dwa pola tekstowe oraz przyciski umożliwiające wpisanie kolejnego zadania lub zakończenie wpisywania zadań. Tutaj nie ma żadnego limitu czasowego, użytkownik ma opisać to, co pamięta i tak, jak to pamięta. Wszystkie te dane oraz informacje o tym, które zadania użytkownik zdołał ukończyć zapisują sie na serwerze, dla każdego użytkownika w osobnym pliku.

ROZDZIAŁ 4

Podsumowanie wyników testu

Aplikacja została przedstawiona grupie dwudziestu sześciu osób. Przy analizie danych brał udział Bartosz Tomkiewicz, magister psychologii. Z dostarczonych nam przez aplikację danych dowiedzieliśmy się, że stosunek zapamiętanych przez użytkowników zadań nieukończonych do ukończonych wynosi 1,78. Oznacza to, że osoby wypełniające test były w stanie przypomnieć sobie zadania przerwane o 78% częściej niż te, które udało im się dokończyć.

Taki wynik potwierdza Efekt Zeigarnik oraz to, że nasza pamięć często pozbywa się informacji o czynnościach, które wykonaliśmy do końca, aby nie zaprzątać nimi naszych myśli. Należy jednak pamiętać, że wynik ten to średnia wszystkich użytkowników. W grupie badanych znalazła się zarówno osoba, która pamiętała jedynie nieskończone zadania, jak i taka, która nie przypomniała sobie żadnego z przerwanych testów.

Co ciekawe, oryginalne badania Blumy Zeigarnik przeprowadzone na podobnej ilościowo grupie ludzi, dały zbiorczy wynik 1,9 czyli zbliżony do uzyskanego za pomocą mojej aplikacji. W przyszłości można powtórzyć badania dodając do aplikacji więcej testów lub zmieniając wymagania obecnych oraz zwiększając liczbę użytkowników, aby uzyskać więcej wiarygodnych wyników.

Bibliografia

[1] Bluma Zeigarnik, „On Finished and Unfinished Tasks”
http://codeblab.com/wp-content/uploads/2009/12/On-Finished-and-Unfinished-Tasks.pdf/

[2] GameMaker: Studio Dokumentacja
https://docs.yoyogames.com//(Dostęp: 25.05.2017)

[3] GameMaker Server Dokumentacja
http://gamemakerserver.com/en/docs//(Dostęp: 25.05.2017)

[4] Użytkownik SP1D3R z forum YoYo Games, skrypt do cieni pod tekstem.
https://forum.yoyogames.com/index.php?threads/text-shadow.23435/(Dostęp: 10.04.2017)

[5] Użytkownik bellow ze strony OpenGameArt.Org, grafika i animacja rycerza.
https://opengameart.org/content/pixel-character-0(Dostęp: 10.04.2017)

[6] Użytkownik Calciumtrice ze strony OpenGameArt.Org, grafika i animacja orków.
https://opengameart.org/content/animated-orcs(Dostęp: 10.04.2017)

Spis rysunków

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