

UNIWERSYTET GDAŃSKI
Wydział Matematyki, Fizyki i Informatyki

Szymon Filipowicz

nr albumu: 224 621

**Rozwój umiejętności awatara w grze
wideo na podstawie rozgrywek
z innym graczem**

Praca magisterska na kierunku:

INFORMATYKA

Promotor:

dr Wiesław Pawłowski

Gdańsk 15.09.2018

Streszczenie

Na przestrzeni kilku dekad gry wideo przeszły długą drogę dynamicznego rozwoju, od Pong'a do symulacji podróży po kosmosie z proceduralnie generowaną przestrzenią i spektakularnymi, realistycznymi bitwami dla tysięcy osób biorących udział w tym samym czasie.

Wraz z wymaganiami odbiorców wzrasta także poziom zaawansowania technologicznego. Zaczynają powstawać gry wykorzystujące sieci neuronowe, nauczanie maszynowe czy też przetwarzanie ogromnych zasobów danych w czasie rzeczywistym. Próbą wykorzystania nowoczesnych mechanik jest projekt stworzony na potrzeby tej pracy.

Słowa kluczowe

gra wideo, C#, Unity3D, nauczanie maszynowe, strumienie danych

Spis treści

Wprowadzenie	5
1. Wprowadzenie do gier wideo	6
1.1. Cechy gier wideo	8
1.2. Gry wideo dzisiaj	10
1.3. Gry wideo jutro	11
1.4. Silniki do tworzenia gier i ich możliwości	12
2. Dobór narzędzi	15
2.1. Założenia i wymagania	16
2.2. Silnik i język	16
2.3. Grafika i dźwięk	17
3. Implementacja	19
3.1. Scenariusz	22
3.2. Wykorzystanie sztucznej inteligencji	24
3.3. Rankingi Echo	28
3.4. Sterowanie i menu	30
Zakończenie	35
A. Gra „Fighty”	36
Spis tabel	39
Spis rysunków	40
Oświadczenie	41

Wprowadzenie

Gry powstały wraz z możliwościami technologicznymi. Początkowo programy rozrywkowe musiały dostosowywać się do sprzętu. Wiele lat później, związek ten stał się dużo bardziej równorzędny. Z jednej strony zaczął powstawać i rozwijać się niezależnie sprzęt ściśle przeznaczony do grania. Natomiast z drugiej strony, nie zapominając o swoich korzeniach, gdy tylko pojawiają się nowe technologie, firmy z branży gier wideo starają się jak najszybciej implementować je w swoim oprogramowaniu. Gdy gry starają się wykorzystywać nowy sprzęt i technologie, inne branże zaczęły starać się wykorzystywać właśnie gry.

Można to zauważyć w zupełnie odmiennych niż IT branżach. Rozwiązania wymyślane na potrzeby gier mają coraz szersze zastosowanie w życiu codziennym. Przykładem może być oprogramowanie edukacyjne czy systemy nagród oparte o tzw. grywalizację w firmach. W niniejszej pracy szczególna uwaga zostanie zwrócona na zastosowaniu sztucznej inteligencji w grach.

ROZDZIAŁ 1

Wprowadzenie do gier wideo

Uznawana za pierwszą grę wideo „Pong” (1972) rozpoczęła zupełnie nowy nurt w programowaniu. Nurt, który podobnie jak branża filmowa miał głównie za zadanie dostarczanie rozrywki i edukowanie. Pomimo ograniczonych możliwości, zarówno softwarowych jak i sprzętowych, wiele firm postanowiło zaryzykować. Krokiem milowym dla gier było wprowadzenie na rynek sprzętu dedykowanego, zwanego *konsolami*. W tym gronie znajdują się *Atari* (1977), *Pegasus*¹ (1990), czy *NES* (1983), który sprzedał się w sumie w 60 milionach egzemplarzach. Jedną z charakterystycznych elementów wymienionych konsol są obsługiwane przez nie nośniki gier. Głównym mankamentem był brak możliwości zapisu nowych danych, co powodowało niemożliwością stworzenia zapisu stanu gry. Natomiast ogólna pamięć przeznaczona na gry wynosiła średnio 35 *Mbit*. Nie powstrzymywało to deweloperów przed szukaniem nowych i kreatywnych rozwiązań prowadzących do wydawania coraz to wymyślniej zaprogramowanych gier. Mechaniki i algorytmy wymyślone w tamtych czasach nadal są powielane w dniu dzisiejszym. Firma *Nvidia*, uznawana za numer jeden w sprzedaży kart graficznych, ogłosiła, że ich najnowsza seria flagowego sprzętu będzie wykorzystywać **algorytm śledzenia promieni** z roku 1979.

Co zaskakujące, wraz z rozrostem rynku konsol, w roku 1983 nastąpił kryzys branży gier wideo. Wielka ilość sprzedanych konsol nie przeliczyła się na ilość sprzedanych gier. Wiązało się to także z ogromnym rynkiem „pirackim”, polegającym na sprzedawaniu nielegalnie kopiowanego oprogramowania. Dynamika rozwoju rynku krytycznie zwolniła (szacuje się, że rynek skurczył się aż o 93 procent). Na długi czas gry odwróciły się od niesienia wartościowych treści i zaczęły wykorzystywać brutalność, by kontrowersyjnością przyciągnąć ludzi. Prostota mechanizmów wynikała ze względu na kurczące się budżety przeznaczone na proces deweloperski. Takie decyzje spowodowały krytykę (podobnie jak niegdyś na branżę filmową), których

¹popularny jedynie w państwach układu Warszawskiego

skutki są zauważalne do dnia dzisiejszego. Gry są obarczane winą przy incydentach z użyciem broni, szczególnie na terenie USA. Ze względu na utarte stereotypy, prawdziwe problemy są bagatelizowane. Pomimo załamania wizerunku, kryzys ekonomiczny nie trwał długo. Rozwiązania legendarnej „Contry” (1987) jak rozgrywka kooperacyjna „CO-OP” i wyśrubowany poziom trudności, w którym każdorazowe otrzymanie obrażeń oznacza przegraną, przeżywają w dniu dzisiejszym drugą młodość. Sytuacja zmienia się na lepsze niemal z dnia na dzień. Polska gra – Wiedźmin 3 (2017), która jako pierwsza, stworzona w całości przez polski zespół deweloperski, dostała najwyższą nagrodę w branży za najlepszą grę roku 2017. Ukazuje ona słowiańską mitologię i uczy tradycji naszego kraju (np. misja „Dziady” jest wariacją odnośnie do twórczości Mickiewicza). Japońskie dzieła takie jak „Persona 5” (2016) podejmują próbę krytyki rzeczywistości i konsumpcyjnego społeczeństwa w którym żyjemy na co dzień. Gry odważnie wchodzą wiek w dojrzały i coraz bardziej promują do myślenia krytycznego. Co może być zaskakujące, brutalność i prostota także mają swoją rację bytu i jest na nie zapotrzebowanie. Wyładowanie frustracji i złych emocji dla osób, które nie potrafią bądź nie mogą tego zrobić za pomocą np. aktywności fizycznej, bywa cenne. Osoby takie mogą znaleźć pomoc we wszelkiego rodzaju grach, szczególnie tych z wykorzystaniem technologii rozszerzonej rzeczywistości. Zauważyło to stowarzyszenie weteranów w USA, które aktywnie wykorzystuje gry do leczenia zespołu stresu pourazowego, z pozytywnymi skutkami. Mała ilość profesjonalnych badań nad efektami gier powoduje brak zrozumienia ich wpływu na społeczeństwo. Zaskakujące może być, jak strach przed takim oprogramowaniem może prowadzić do cenzury czy ograniczenia dostępu do poszczególnych programów w całych krajach. Przodują w tym Chiny, ale też często wykorzystuje to także Australia i Niemcy. Wspomniane Chiny wymuszają, by wiele gier przeszło zmiany wizualne i ograniczenia trybów, w których można uczestniczyć, ze względu na brak spójności z panującym tam ustrojem politycznym. Dotknęło to między innymi „Players Unknown: Battleground”.

1.1. Cechy gier wideo

Gry wideo dzielą się na wiele kategorii w zależności od tego, które cechy są brane pod uwagę jako priorytetowe. Podstawowy podział można utworzyć na podstawie ilości graczy w danej sesji rozgrywki. Jeśli jest to gra dla jednej osoby, można mówić o sesji offline lub inaczej *solo*. Są to rozgrywki, które przeważnie stawiają na wciągającą fabułę i rozbudowane światy. Często pozwalają na dowolne zwiedzanie i odkrywanie nowych lokacji (open-world). Do niedawna takie oprogramowanie nie wymagało połączenia z Internetem, jednak w ostatnich czasach gry z tej kategorii nabrały charakteru serwisów z ciągle aktualizowaną zawartością, przede wszystkim fabularną.

Gry służące do pojedynków 1 na 1 (*versus*), czyli dokładnie dla 2 użytkowników, których zadaniem jest rozstrzygnięcie, który z zawodników jest lepszy. Rozgrzywka taka jest w pełni oparta na rywalizacji, co widać także dzięki publicznym rankingom, w których każdy może sprawdzić swoją pozycję względem innych (często z podziałem na kategorie/tryby). Taki sposób prowadzenia rozgrywek najczęściej można spotykać w bijatykach lub pojedynkach sportowych.

Gry przeznaczone dla minimum 2, a maksimum 4–5 są to sesje kooperacyjne („**CO-OP**”) i stawiają na współpracy. Wiele problemów postawionych przed graczami wymagają współpracy, by je rozwiązać. Takie rozwiązania przeważnie wspierają granie na jednym ekranie dla całej drużyny (*split-screen*) albo poprzez internet.

Gry przeznaczone od 2 do około 100 użytkowników (górną granicą nie jest dokładnie określona), w których rozgrywki polegają na pojedynkowaniu się i wyłanianiu najlepszych zawodników bądź najlepsze drużyny (*multiplayer*). Aktualnie jest to najpopularniejszy i najbardziej eksploatowany tryb. Cały czas powstają coraz to bardziej niekonwencjonalne pomysły na podziały graczy i stawiane przed nimi cele. Można założyć, że kategoria ta w najbliższych latach będzie należeć do grupy z największą dynamiką rozwoju (*Battle Royal*).

Rodzaj gier, który nie ma górnej granicy co do ilości użytkowników uczestniczących w sesji w tym samym czasie (*MMO*), jeszcze nie dawno był najbardziej popularnym (*World of Warcraft*). Gry te często cechują się okresowym abonamentem. W przeciwieństwie do większości gier, za które płaci się raz, najbardziej popularnym sposobem uczestniczenia w rozgrywce było wykupowanie dostępu do serwerów na wskazaną ilość dni. Zabieg ten, w przypadku firmy „Blizzard”, dzięki stałemu do-

plywowi środków finansowych, zarabiając do dzisiaj około 10 miliardów dolarów, umożliwił ciągły rozwój swojej gry po premierze. Nie chodziło tylko o naprawianie błędów w implementacji bądź balansowaniu rozgrywki. Gra cechowała się ciągłym rozwijaniem fabuły i dynamicznymi zmianami środowiska, w którym poruszają się gracze. Dynamiczny rozwój za pomocą płatnych usług serwisowych coraz bardziej przenika do innych rodzajów gier. W wielu wypadkach jest to odbierane negatywnie.

Ważnym elementem jest także grafika. Podział może odbywać się także ze względu na ilość wymiarów. Mogłoby się wydawać, że dwa wymiary (2D) ograniczają immersję i możliwości środowiska jak i samej mechaniki w rozgrywce. Nic bardziej mylnego. Przy dobrym doborze rzutowania grafiki i stylu artystycznego można wręcz pogłębić doznania użytkownika. Dwuwymiarowe gry powinny posiadać rzut izometryczny, gdy ważna jest obserwacja otoczenia wokół postaci gracza, a zarazem zależy nam na dynamicznym charakterze rozgrywki. Możemy też mieć widok z góry pod kątem prostym, który najczęściej jest wybierany w grach strategicznych ze względu na przejrzystość i rozróżnianiu elementów graficznych oraz. Widok od boku znany np. z Mario² idealnie współgra z szybkimi grammi akcji, w których liczy się dokładność i tempo wykonywania akcji. Jak widać, grafika 2D nie musi ograniczać i spłaszczać rozgrywki, jeśli jest wykonana i przedstawiona odpowiednio z duchem rozgrywki. Jako ciekawostkę można wspomnieć Castle Wolfenstein³, która jako jedna z pierwszych wykorzystała przestrzeń trójwymiarową z teksturami dwuwymiarowymi. Mechanika renderowania wykorzystania w tej grze dała podwaliny pod prawdziwe 3D oraz jako jedna z pierwszych wykorzystała algorytm widoczności elementów w celach optymalizacyjnych (algorytm widoczności). W grach 3D mamy jeszcze więcej możliwości. Oprócz tak zwanego 2.5D hybrydy pozwalającej na przedstawienie wszystkich rzutów opisanych wyżej, za pomocą trójwymiarowych obiektów, istnieją także dodatkowe perspektywy. Można obserwować akcje jak w prawdziwym życiu – z perspektywy pierwszej osoby. Taki wybór najczęściej jest dokonywany dla zwiększenia dynamiki w rozgrywce, np. wojennych bądź z ele-

²Popularna seria gier z Japonii opowiada o poszukiwaniach porwanej księżniczki przez włoskiego hydraulika.

³Castle Wolfenstein (1981) stworzony przez firmę id Software jest grą akcji, która utworzyła zupełnie nowy kierunek w programowaniu gier.

mentami pojedynków np. rycerskich. Możemy obserwować także postać zza pleców (TPS) co jest najczęściej wykorzystywane na konsolach ze względu na specyfikę kontrolerów, w których taki rzut jest najbardziej naturalny przy sterowaniu. Oczywiście gry można podzielić jeszcze na wiele innych kategorii, biorąc pod uwagę np. tempo rozgrywki czy model sterowania. Na potrzeby zrozumienia oprogramowania dołączonego do pracy magisterskiej wszystkie cechy i przynależność do kategorii będą opisane w rozdziale 3.1.

1.2. Gry wideo dzisiaj

Wszystko wskazuje na to, że gry wideo nie tylko powracają do swojej świetności, ale według statystyk, nigdy nie miały się lepiej. Ilość wydawanego oprogramowania jest ogromna. Wiele z nich nie zostaje nigdy zauważonych, znikają w cieniu gigantów, tak zwanych gier AAA⁴, których częstotliwość wydawania również się zwiększyła. Nad dziełami od takich firm jak Blizzard (Overwatch, Diablo) czy EA (Fifa, Star Wars Battlefront) pracują setki ludzi – od grafików poprzez muzyków no i oczywiście programistów. Dziesiątki, a nawet setki milionów sprzedanych egzemplarzy pozwalają na rozwijanie coraz to większych możliwości. Mamy gry z proceduralnie tworzonymi światami, scenariuszami tworzonymi przez hollywoodzkich reżyserów czy ścieżkami dźwiękowymi nagrami przez legendy sceny muzycznej. Nie tak dawno Polacy z CD-Project RED odbierali nagrodę za najlepszą grę roku 2017 jako pierwsi w historii naszego kraju. Coraz śmielej powstają oprogramowania do rozrywki wykorzystujące najnowsze osiągnięcia technologiczne jak neuroewolucja czy nauczanie maszynowe. Do takich gier zaliczają się np. „Echo”⁵ czy „Blitzkrieg 3”⁶. Pomimo swojej prawie 70-letniej historii, pole do popisu pozostaje nadal duże. W roku 2017 królował nowy tryb rozgrywek nazwany Battle Royal (Players Unknown), który polega na pojedynku 100 osób na pomniejszającym się obszarze, zmuszając ich do ciągłej rywalizacji o terytorium. Wygrywa osoba lub zespół, który zostaje ostatni na mapie. W takich sesjach uczestnik nie posiada żadnego ekwipun-

⁴Oprogramowanie z dużym budżetem i promocją, nastawiony na wysoką sprzedaż.

⁵Echo (2017) – gra akcji, w której przeciwnicy kopiuje ruchy gracza.

⁶Blitzkrieg 3 (2017) – gra strategiczna opowiadająca o drugiej wojnie światowej z perspektywy generałów.

ku na początku, a jedyny sposób na zwiększenie swoich szans na przetrwanie jest poszukiwanie losowo rozmieszczonych przedmiotów.

Grafika coraz bardziej upodabnia się do świata rzeczywistego, takie rozwiązania jak dynamiczne zmiany pogody czy żywe miasta, w których mieszkańcy sterowani przez sztuczną inteligencję mają swoje obowiązki i problemy pozwalają na budowanie niespodziewanych sytuacji. Jak można się domyślać, skrypty, które kontrolują wszystko, co się dzieje na ekranie wymagają coraz to większych zasobów. A końca nie widać!

Nazwa gry	Rok produkcji	Typ	Ocena metacritic
Blitzkrieg 3	Strategia	2017	68
Echo	2017	Akcja	72
Forza Motorsports	2017	Wyścigi	86

Tabela 1.1. Rozwojowa sztuczna inteligencja w grach

Źródło: <http://www.metacritic.com> – strona bierająca oceny gier z różnych źródeł i wyznaczająca ich średnią

1.3. Gry wideo jutro

Jak już zostało wspomniane, nadal wiele rzeczy pozostaje do zaimplementowania. Powodem takiego stanu rzeczy są często ograniczenia w sprzęcie. Popularnym efektem, którego gracze bardzo nie lubią jest tak zwany „downgrade”, czyli spłykanie szczegółowości obiektów, by ograniczyć wykorzystywanie zasobów komputera. Jak zostało wspomniane, coraz częściej można zauważyć próby zaimplementowania stałej wymiany informacji z serwerem gry skupiających się na rozgrywkach w pojedynkę. Celem tego jest zbieranie informacji, by dostosować rozgrywkę bądź wykryć zachowania użytkowników. Spotyka się to jednak często z protestem ludzi posiadających problemy ze stałym połączeniem. Sztuczna inteligencja zawarta w grach jest coraz bardziej skomplikowana i efekty znane ze starych gier, gdzie przeciwnicy potrafili w nieskończoność biec w ścianę zostały już wyeliminowane. Dynamicznie rozwijające się zagadnienie sztucznej inteligencji może otworzyć zupełnie nowe

możliwości dla gier. To, w jaki sposób nowinki technologiczne wpłyną na gry, może okazać się już niedługo. Firmy zapowiadają wielkie premiery i wiele ciekawych nowinek może ujrzeć światło dzienne. Duży nacisk, ze względu na presję graczy, został właśnie położony na zachowanie przeciwników sterowanych przez komputer. Wyścig rozpoczął się wiele lat temu, ale właśnie teraz jesteśmy świadkami zaostrzenia rywalizacji na tle zaawansowania technologicznego. Patrząc na to można mieć wrażenie, że rewolucja jeszcze się nie skończyła, a dopiero zaczyna. Ogromne ilości pieniędzy wydawane na koszty deweloperskie zwracają się wielokrotnie, co sprzyja warunkom do inwestowania w innowacyjne technologie, także w przestrzeni gier wideo.

Nazwa gry	Koszt produkcji	Przychód z gry
Wiedźmin 3	46 mln	250mln
GTA V	137 mln	6 mld
Minecraft	brak danych	>280mln
Half Life 2	49 mln	34mln

Tabela 1.2. Koszt produkcji w porównaniu do zarobków

Źródło: Wikipedia

Dodatkowo Minecraft został sprzedany przez swojego twórcę za kwotę 2 miliardów dolarów, czyli 1/3 tego co zarobiło GTA V, do firmy *Microsoft*.

1.4. Silniki do tworzenia gier i ich możliwości

Wraz z rozwijaniem poziomu zaawansowania gier można też zauważyć wysyp narzędzi do ich tworzenia. Kompleksowe oprogramowanie ułatwiające tworzenie gier wideo nazywamy *silnikami do gier*. Pozwalają one na implementacje grafiki, często wraz z ich animacją, dynamiczną obsługą dźwięków (często ze źródłem i zasięgiem słyszalności). Wiele silników ułatwia tworzenie bądź udostępnia już stworzoną częściową symulację świata. Do takich elementów zalicza się grawitacja, obsługa kolizji

czy podstawowe prawami Newtona. Poniżej przedstawiono krótką charakterystykę najpopularniejszych silników:

Unity3D

Jest aktualnie jednym z dwóch najpopularniejszych silników do tworzenia gier zarówno 3D jak i 2D. Przystępność w użytkowniku gotowa fizyka do implementacji w obu wymiarach i najważniejsze – jeden język jest kompilowany na wszystkie najważniejsze platformy – Windows, Linux, iOS, PS3/PS4, Xbox360/XboxOne oraz wraz z dodatkowym API na konsole Nintendo takie jak Switch czy WiiU. Do programowania służy C#, JavaScript bądź mniej popularny Boo. Ciekawe jest to, że na przestrzeni całego projektu języki można mięszać. Najpopularniejszymi grami stworzonymi na tym silniku to Hearthstone: Heroes of Warcraft (70 mln) czy Rust (średnio 35tys. użytkowników codziennie). Model biznesowy wykorzystywany przez Unity3D jest przyjazny dla deweloperów gier niezależnych. Samo używanie tego silnika jest darmowe w celach komercyjnych tak długo, jak nie przynosi przychodów powyżej 100 tysięcy dolarów. Pomocna może się okazać wielka baza assetów (zasobów) tworzona przez niezależne podmioty. W zależności od twórców mogą być darmowe lub dostępne za opłatą.

Unreal Engine

Na równi z nim można zestawić Unreal Engine, czyli najpopularniejszy silnik do tworzenia gier na PC z systemem Windows, który jest to jedynym systemem, na który można skompilować kod z poziomu tego silnika (pisany w C++). Silnik jest stworzony przede wszystkim dla gier z zaawansowaną grafiką 3D. Jest najczęściej wybierany przez duże firmy ze względu na olbrzymie możliwości przy dużym nakładzie pracy. Szczególnie trzeba zwrócić uwagę na obsługę grafiki na niesamowicie wysokim poziomie i szczegółowości. Jedne z najpopularniejszych gier stworzonych na tym silniku to „Batman: Arkham City” (6 mln sprzedanych kopii) czy „BioShock Infinite” (11 mln sprzedanych kopii).

3. WebGL – silnik, a tak naprawdę rozszerzenie języka Javascript do tworzenia gier na przeglądarki. Sam w sobie udostępnia tylko obsługę grafiki 2D i 3D na bazie

systemu Canvas. Ze względu na naturę przeglądarek możliwości *Canvas* i *WebGL* w *HTML5* są ograniczone. Z pomocą przychodzą dodatkowe API (np. *Phaser.io*) tworzenie gier skierowanych tylko i wyłącznie na przeglądarki internetowe staje się łatwiejsze a implementacja przejrzystsza i łatwiejsza w zrozumieniu. Jedną z pierwszych technologii rysowania kształtów na obszarach strony internetowej jest *Canvas*. Poważną wadą jednak był brak obiektowości. Zmuszało to do odświeżania całego obszaru roboczego nawet w wypadku przemieszczania pojedynczego elementu.

Dobór narzędzi

Podczas projektowania gier wideo ważnym aspektem jest odpowiedni dobór narzędzi, które powinny ułatwić proces deweloperski, a co ważniejsze dla wielu firm, skrócić jego czas. Podczas projektowania trzeba wziąć pod uwagę implementację mechaniki, grafiki i dźwięku. Dokonywanie wyboru powinno odbywać się poprzez dopasowanie wymagań do specyfikacji dostępnego oprogramowania. W rozdziale 1.4 zostały opisane najpopularniejsze silniki. Biorąc pod uwagę grafikę, najważniejszym aspektem jest, czy świat będzie zamknięty w trzech, czy dwóch wymiarach. Jest wiele darmowych i bardzo rozwiniętych narzędzi zarówno do grafiki 2D, jak i 3D. Dla przestrzeni trójwymiarowej najczęściej wykorzystywanym, a przy tym darmowym oprogramowaniem jest „Blender”. Biorąc na warsztat tylko dwa wymiary, aplikacje są dużo bardziej ukierunkowane. Istnieją silniki do tworzenia grafiki komiksowej, 8/16/32 bitowej, czyli tak zwanej bitowej/pikselowej oraz klasycznej, która wymaga najwięcej nakładów pracy. O ile większość silników oferuje wsparcie w implementacji dźwięków/muzyki musimy też ją dostarczyć. Możemy stosować aplikacje do nagrywania dźwięków i za pomocą mikrofonów zapisać na naszym stanowisku muzykę graną na żywo. Jeśli jednak nie mamy dostępu do profesjonalnego sprzętu nagrywającego, często lepszym rozwiązaniem jest tworzenie muzyki elektronicznej bezpośrednio na komputerze. Oba sposoby wymagają jednak zdolności muzycznych lub chociaż dobrego słuchu dla niezbyt skomplikowanych linii dźwięków. Warto pamiętać, że zarówno grafiki, jak i wszelkiego rodzaju dźwięki można dobrać z szerokiego asortymentu tak zwanych assetów w internecie. Wiele stron oferuje swoje zasoby wraz z licencją na komercyjny użytek za darmo bądź te bardziej złożone za konkretną opłatą.

2.1. Założenia i wymagania

Na potrzeby tworzonej gry należy podjąć pewne decyzje odnośnie implementacji. Ze względu na nacisk na rozwój umiejętności **awatara** i mechaniki gry, można uprościć aspekty wizualne. Wybór pada na grafikę dwuwymiarową z prostymi modelami postaci i jednolitym tłem. W grach można zaimplementować dźwięk na dwa sposoby. Pierwszym sposobem jest ustalenie źródła dźwięku, a następnie jego rozprowadzanie z danego punktu. Innym podejściem, wybranym w tym projekcie, jest rozprowadzanie dźwięku równomiernie na całej przestrzeni. Dzięki takim zabiegom można skupić się na mechanice i pogłębieniu interakcji z rozgrywką. Podczas doboru narzędzi najważniejszy jest silnik do gier, a następnie kompatybilne z nim narzędzia do grafiki. Oprogramowanie do dźwięku musi jedynie umożliwiać odpowiednią kontrolę nad ramami czasowymi. W silniku do gier musimy mieć maksymalną swobodę odnośnie do implementacji. Kodowanie musi odbywać się bez żadnych ograniczeń wynikających ze specyfikacji silnika. Poza zakres dostępnych narzędzi wychodzi oprogramowanie płatne i nie umożliwiające wykorzystywania pracy do celów komercyjnych.

2.2. Silnik i język

Unity3D jako silnik do gier jest w tej sytuacji naturalnym wyborem. Brak jakichkolwiek ograniczeń w kodowaniu oraz kompleksowe wsparcie grafiki 2D są najważniejszymi cechami popierającymi wybór. Dodatkowo Unity3D jako platforma do programowania udostępnia nam wiele dodatkowych bibliotek bezpośrednio do zaimportowania w plikach ze skryptem. Wiele z nich będą wykorzystywane. Dużym wsparciem są moduły do obliczeń matematycznych, udostępniające struktury znane z SQL czy dynamiczne odczytywanie zawartości plików. Podczas tworzenia programu nie będą wykorzystywane żadne zewnętrzne biblioteki oprócz domyślnie wbudowanych w Unity. Ponadto, jako jeden z niewielu silników, Unity3D jest kompatybilny z systemem Linux oraz udostępniona wersja za darmo w pełni zapewnia funkcjonalności potrzebne do stworzenia gry na potrzeby pracy.

Do implementacji kodu zostanie wykorzystany C# jako język z kompleksowym wsparciem w wybranym silniku. Język C# jako język obiektowy i nowoczesny udo-

stępnia wszystko to, co będzie potrzebne do zaprojektowania dynamicznie rozwijającego się awatara. Ważne są tutaj struktury pseudo SQL i listy z wyznaczoną kolejnością.

2.3. Grafika i dźwięk

Uproszczona grafika 2D pozwala na wybór nieskomplikowanego narzędzia graficznego. Możliwość zapisu plików w rozszerzeniu kompatybilnym z wybranym silnikiem do tworzenia gier oraz dostęp do rysowania i modyfikowania podstawowych kształtów to priorytetowe cechy przy dokonywaniu wyboru. Ze względu na szeroki wybór dostępnych narzędzi spełniających zarówno podstawowe wymagania z rozdziału 3.1 oraz powyżej opisane cechy wybór pada na popularne i doceniane oprogramowanie „Inkscape”. Jest ono kompatybilne zarówno z systemem Windows jak i Linux. Program pozwala na wykorzystanie efektów pracy w celach komercyjnych. Rozszerzenie faworyzowane przez wybrany program – „.png” jest w pełni kompatybilne z silnikiem Unity. Program ten zostanie wykorzystany do stworzenia wszystkich widocznych elementów graficznych w grze, takich jak obie postacie i elementy otoczenia. Wizualne efekty zostały ograniczone do minimum. Elementy graficzne postaci składają się z trzech głównych części. Górna i dolna część ciała oraz twarz. Twarz jest przede wszystkim wskaźnikiem życia. Wyraz twarzy awatara jest wyznacznikiem ilości otrzymanych obrażeń. Szczeka posiada natomiast dwa rzędy zębów. Ilość utraconych zębów w górnym rzędzie wskazuje na procentową stratę kontroli nad górną częścią ciała właśnie. Dolny rząd zębów wskazuje na to samo dla dolnej części ciała. Obie części ciała są zależne względem siebie. Jednak użytkownikowi zostają oddane narzędzia pozwalające na ich niezależną kontrolę. Dzięki temu możliwe są nie tylko uniki polegające na zręczności ale też, dodano możliwość skakania, na różne podejścia do ataku.

Do dźwięku zostanie wykorzystana aplikacja „Audacity”. Jest to oprogramowanie ze wsparciem zarówno dla systemu Windows jak i Ubuntu z wykorzystaniem do celów komercyjnych. Umożliwia on kontrolę długości nagranych dźwięków jak i konwersję na najpopularniejsze, a co najważniejsze wspierane przez Unity typy plików dźwiękowych jak „mp3” czy „wav” bez znaczącej utraty jakości. Nagrania

odbyły się za pomocą mikrofonu „SamsonGo”. Jednak za pomocą wspomnianego oprogramowania można spodziewać się wystarczającej jakości bez odczuwalnych szumów. Do zmiany dźwięku głównie został wykorzystany pogłos i zmiana tonu na niższy. Efekty te powodują wrażenie walki na ringu co jest wskazane w grach typu bijatyka. Pliki dźwiękowe zostały zapisane w rozszerzeniu „.mp3” ze względu na najlepszy stosunek jakości do rozmiaru. Muzyka zarówno w menu jak i ta odgrywana podczas pojedynku może zostać zmieniona w opcjach. Wybrać można tylko te utwory, które znalazły się w odpowiednie folderze „menu-music” oraz „fight-music”.

ROZDZIAŁ 3

Implementacja

Ze względu na przetwarzanie dużej ilości danych w czasie rzeczywistym, gra została zaimplementowana z naciskiem na wielowątkowość. Zarówno dodawanie, aktualizowanie, jak i wybieranie odpowiednich ruchów z zapamiętanej listy odbywa się równolegle. Oznacza to, że program wykorzystuje, włącznie z głównym, cztery wątki na raz. Kontrola nad ilością wykonywanych wątków odbywa się za pomocą zaimplementowania zmiennych flagowych. Wynikiem równomiernie rozłożonego obciążenia jest przede wszystkim płynność rozgrywki, ale też reaktywność w odpowiedziach przeciwnika podczas sesji. Nie ma obawy, że logika postaci będzie zablokowana przez wydłużony czas wykonywania którejś z głównych akcji, czyli dodawanie, wybieranie i aktualizowanie. By stworzyć podstawy zachowań wszelkich elementów w grze, zostały wykorzystane wyłącznie wbudowane komponenty w silniku Unity3D. Warunki świata w grze ograniczają się do grawitacji i fundamentalnej interakcji pomiędzy obiektami.

Innym wbudowanym modulem, jest komponent odpowiadający za wychwytywanie kolizji pomiędzy obiektami. Dzięki tym mechanizmom proces tworzenia gier jest znacznie ułatwiony i przyspieszony. Gra jako jeden z głównych elementów mechaniki wykorzystuje grawitację. Sprawia to, że każdy obiekt musi mieć określoną masę. Obiekty takie jak podłoga mogą zostać ustawione jako obiekt statyczny, dzięki czemu grawitacja na niego nie wpływa. Masa ogona postaci jest pięciokrotnie większa od reszty ciała w celach „uziemia” postaci przy rotacji. Kolizje są wyliczane na podstawie intersekcji pomiędzy wyznaczonymi obszarami. Jeśli obiekty nachodzą na siebie, aktywują się odpowiednie flagi wraz z wymaganymi parametrami. Dzięki temu system może mieć kontrolę nad blokowaniem, zadawaniem obrażeń i przesuwaniem awatarów w czasie rzeczywistym w optymalny sposób. Mechanizm „Joint” – w tym przypadku wykorzystywany tylko typu „Distance”. „Joint” można traktować jako obiekt, który przy odpowiednich ustawieniach zapewnia nam pewne stałe fizyczne. Gra wykorzystuje „Distance Joint” w celu połączenia obu części ciała awa-

tara w sposób dynamiczny. Dzięki temu mechanika gry, w tym grawitacja i kolizje, wpływają niezależnie na obie części ciała. Stałą fizyczną w tym wypadku jest nierozdzielność wybranego punktu dolnej części ciała od wybranego punktu górnej części ciała. Zaobserwować to można w sytuacji, w której jeden z awatarów przesuwa się po podłożu. Górna część ciała obraca się według wyznaczonej osi, ciągnąc za sobą zaczepiony punkt dolnej części. W tym samym czasie grawitacja ogona „dociska” go do podłoża uniemożliwiając reszcie ciała unoszenia się w powietrzu, co było by efektem w innym wypadku przy takiej rotacji. Na potrzeby projektu wykorzystano skomplikowane struktury danych i rozbudowane klasy. Warte wspomnienia są: List – Lista obiektów z ustaloną kolejnością. Do poszczególnych elementów można uzyskać dostęp poprzez indeks albo wskaźnik „First”. Mały koszt zmiany rozmiaru. Dictionary – Jest to zbiór obiektów, do których dostęp można uzyskać tylko i wyłącznie znając unikalne ID typu String dla poszukiwanego elementu. Cechuje się szybkim dostępem do elementów. DataTable – struktura pseudo SQL. Poszczególne parametry można zapisać do kolumn, które wcześniej trzeba zdefiniować. By uzyskać wiersz, należy stworzyć zapytanie na wzór języka SQL. Otrzymany wynik zawiera w sobie listę wierszy odpowiadających zapytaniu. Combo – zaimplementowana na potrzeby gry klasa posiadająca w sobie kompleksowy zbiór danych. Jedna instancja „Combo” posiada w pamięci takie wartości jak długość trwania całego ataku oraz

poszczególne akcje w odpowiedniej kolejności. Każda taka akcja ma zdeterminowaną długość i rodzaj akcji. To może być atak lub obrona. Dodatkowo każdy atak ma zapisany stan obu analogów. Klasa ta jest wykorzystywana do zapamiętywania każdego ruchu gracza i następnie do dostarczania odpowiednich ataków sztucznej inteligencji. Struktura tej klasy prezentuje się następująco:

```
class Combo
{
    List Moves
    string id
    float totalDuration
    int numberOfMoves
    int score
    bool attack
```

```
Combo() {  
    this.id = wygenerujID  
}  
addMove(newMove) {  
    numberOfMoves++;  
    Moves.Add(newMove);  
    totalDuration += newMove.duration;  
}  
}
```

Ze względu na mały rozmiar i nieskomplikowaną hierarchię plików system pozwala na pominięcie bezpośredniej instalacji gry na dysku. Wymagania są głównie związane ze specyfikacją silnika Unity3D, wyjątkiem jest procesor, który musi wspierać wielowątkowość wykorzystaną w oprogramowaniu.

Wymagania:

System: Windows Vista SP1+, Mac OS X 10.9+, Ubuntu 12.04+, SteamOS+

Karta graficzna: DX10 (shader model 4.0)

Procesor: 4 rdzenie (mogą być wirtualne) lub wsparcie działania minimum 4 niezależnych wątków na raz oraz wsparcie obsługi zestawu instrukcji „SSE2”.

Program działa na wspomnianych trzech systemach. Proces deweloperski poza grafiką i dźwiękami odbywał się na systemie „Ubuntu 16” i na nim był głównie testowany z pomocą kontrolera od „Playstation 3”.

Specyfikacja sprzętu:

Procesor: Intel(R) Core(TM) i7-3517U CPU 1.90GHz

Karta graficzna: 3rd Gen Core processor Graphics Controller

Na opisanym powyżej sprzęcie, klatki na sekundę („FPS”) wynoszą:

Średnio – 120 FPS Min – 110 FPS Max – 130 FPS

Dodatkowo gra została przetestowana na komputerze z systemem „Windows 10” i kontrolerem od konsoli „Xbox One”.

Jak widać, dzięki zastosowaniu wielowątkowości i zoptymalizowaniu kodu, amplituda klatek na sekundę jest mała, co jest efektem pożądanym dla płynności gry. Efekt ten został uzyskany także dzięki wykorzystaniu specyficznych i pasujących

do implementacji typów zmiennych jak „słownik” („dictionary”), posortowane listy („sorted list”) czy zmienna typu tablicowego („data table”) pozwalająca na tworzenie szybkich zapytań SQL z wykorzystaniem warunków logicznych „i” i „lub”. Samo testowanie zgodnie z metodami w branży gier odbywało się za pomocą tak zwanych „beta testów”. Polegają one na ciągłym odgrywaniu tych samych elementów gry w celach wyszukania nieścisłości z założeniami. Pomocna w tym przypadku jest znajomość kodu źródłowego i pełna znajomość możliwości silnika. Zdecydowana większość błędów, lub nieścisłości wynika z wykonywania akcji niezgodnych lub nieprzewidzianych z założeniami. W takim przypadku przychodzi z pomocą udostępnienie programu osobom niezwiązanym z procesem deweloperskim. Opisywana gra została udostępniona zamkniętej grupie osób posiadających doświadczenie w testowaniu gier co pozwala na zdefiniowanie zachowania oprogramowania na odmiennych specyfikacjach sprzętowych komputera. Dzięki niskim wymaganiom sprzętowym i wykorzystaniu w mechanice czas rzeczywisty procesora zamiast klatek na sekundę, program działa płynnie i poprawnie niezależnie od mocy sprzętu tak długo, jak spełnia wymagania minimalne.

3.1. Scenariusz

Rozgrywka polega na sprowadzeniu kontroli przeciwnika nad jego postacią do zera. Osiąga się to poprzez zadawanie obrażeń w odpowiednie elementy ciała przeciwnej postaci. Wyznacznik kontroli jest to liczba od 0 do 100, która jest opisana funkcją $f(x, y) = (x + y)/2$, gdzie x to wyznacznik kontroli nad górną częścią ciała a y to wyznacznik kontroli nad dolną częścią ciała. Zarówno x jak i y określa ilość pozostałej kontroli w odpowiadającej im części ciała. Gra jest ograniczona poprzez ilość czasu przydzieloną na każdą rundę lub nokaut. Mianowicie, jeśli któremukolwiek z awatarów uda się zmusić przeciwnika do leżenia na podłodze przez więcej niż 5s, rozgrywka zostaje przerwana na korzyść stojącego. Punkty życia (kontrole) traci się na skutek otrzymanych uderzeń w określoną część ciała. Im mniej punktów życia (im mniejszy wyznacznik kontroli), tym ciężiej (mniej dokładniej oraz wolniej) odpowiada awatar na odpowiednie przyciski. Oznacza to, że im mniejszy wyznacznik górnej części ciała, tym dany awatar słabiej wykonuje uderzenia za pomocą pięści,

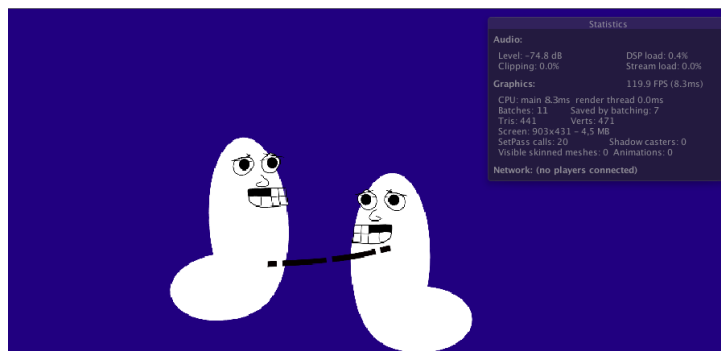
natomiast mniejszy wyznacznik dolnej części ciała utrudnia wykonywanie kopnięć i poruszanie się po powierzchni. Co więcej, główny wyznacznik kontroli określa trudność w wykonywaniu rotacji oraz wstawaniu z ziemi. Przy głównym wyznaczniku wynoszącym o dany awatar nie reaguje w żaden sposób na wprowadzany input.

Dodatkowo, natychmiastowa utrata dużych ilości punktów (dla górnej części ciała zoglównego wyznacznika) powoduje chwilową utratę przytomności (brak re-sponsywności awatara) na rzecz poszkodowanej postaci. Scenariusz dzieli się na dwa etapy:

1. Pierwsza gra – jeśli rozpoczynamy rozgrywkę po raz pierwszy system gry nie posiada wiedzy o żadnych ruchach, w tym jak się poruszać ani jak atakować. W takim wypadku rozgrywka rozpoczyna się od pierwszego ruchu gracza. Podczas gdy użytkownik klika przyciski po raz pierwszy, awatar kontrolowany przez system zaczyna nasłuchiwać i tym samym zapisywać zserializowany strumień z kontrolera użytkownika. Zapis będzie odbywać się za pomocą mechaniki „Echo” opisaną później. Gdy pierwszy ciąg ruchów (opisany także jako „combo”) zostanie zapisany, awatar przeciwnika przystępuje do akcji. Podczas gdy nasłuchiwanie trwa od tej pory bez przerwy, awatar sztucznej inteligencji będzie wykorzystywał zaobserwowane do tej pory ruchy zgodnie z panującymi warunkami w świecie gry.
2. Rozgrywanie kolejnych gier – od tej pory przechodzimy do trybu, w którym system gry posiada już co najmniej zapisany jeden ruch, jednak można założyć, że jest ich wiele. Program będzie dokonywać ciągłej oceny wybranych ciągów akcji i na podstawie utworzonego rankingu będzie prowadzić swoją strategię co do wykonywanych ataków.

Gdy walka się kończy, sesja jest resetowana, natomiast sztuczna inteligencja od razu przystępuje do walki po odliczaniu. Z programu można wyjść w każdym momencie, wybierając opcje „Save and Quit” lub „Quit”. Pierwsza opcja będzie nadpisywać nowy, przekształcony na podstawie przeprowadzonych walk ranking ataków. Druga opcja natomiast wychodzi z gry bez zapisywania jakichkolwiek zmian. W opcjach można ustawić jak wiele poprzednich wersji umiejętności awatara zostanie zapisywanych w postaci kopii zapasowych. Domyślnie jest to jedna wersja wstecz, czyli

plik posiadający zapisane umiejętności, zanim użytkownik włączył program, przeprowadził dowolną ilość rozgrywek i wyłączył program zapisując postępy. Dzięki unikatowym walkom i ciągle rozwijającej się sztucznej inteligencji sterującej przeciwnikiem, gra potencjalnie ma długą żywotność. Unikalne i zaskakujące pojedynki pozwolą na utrzymanie zainteresowania u graczy. Wymiana wyuczonych umiejętności awatarów i potencjalne rozgrywki jeden na jednego z innym użytkownikiem na tym samym komputerze to kolejne argumenty przemawiające za utrzymaniem zainteresowania.



Rysunek 3.1. Główne menu

Źródło: Opracowanie własne

3.2. Wykorzystanie sztucznej inteligencji

Nauczanie maszynowe dzieli się na kilka podstawowych typów. Są to:

1. Analityczne myślenie – polega na wprowadzaniu dużej ilości oznakowanych danych często z opisanymi parametrami i wymuszanie nauki na ich podstawie. W tym modelu komputer podczas nauczania poszukuje zależności i połączeń pomiędzy wprowadzonym danym. Następnie relacje są nałożone na dołączone parametry. Przykładem takich implementacji są sieci neuronowe.
2. Nienadzorowana nauka – w tym przypadku dane są nieoznakowane. Model

taki wykorzystuje się do poszukiwania wzorów lub ulepszania już istniejących. Przykładem takiego algorytmu jest „k-means clustering”.

3. Zasilanie – metoda ta pozwala na naukę poprzez obserwacje. Głównie polega na interakcji z otoczeniem i wycenianiu stosunku ryzyka do nagrody. Taka nauka rozwija umiejętności sztucznej inteligencji w sposób iteracyjny na zasadzie akcja–reakcja. Taka implementacja jest stosowana głównie do wyspecjalizowania rozwiązania do specyficznych warunków. Problem „Reinforcement Learning” jest uznawany za oddzielny problem w dziedzinie algorytmów i aktywnie rozwijany. Właśnie ten typ nauczania jest eksploatowany w branży gier wideo i uznawany jako przyszłościowy. Główne etapy tego algorytmu można podzielić na następujące punkty:

- (a) Obserwacja
- (b) Decyzja i akcja
- (c) Nagroda i kara
- (d) Wnioski

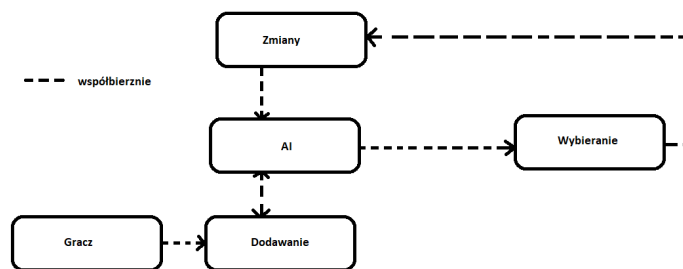
Popularnymi algorytmami stosującymi zasilanie jest Q-learning lub „Temporal Difference (TD)”.

Wybór modelu powinien być zależny od problemu. Algorytmy nadzorowane wymagają dużej ilości danych startowych, by rozpocząć naukę. Przy braku takowych warto zastanowić się nad algorytmami zasilanymi. Wymagają one jednak interaktywnego środowiska. Określonych systemów nagradzania i karania. Gdy problem nie spełnia takich wymagań, pozostają algorytmy nienadzorowane, które pomagają przy dużych ilościach danych nieopisanych lub gdy poszukiwane są modele/wzorze.

Rozwój umiejętności awatara polega na wykorzystaniu utworzonych rankingów i ciągłym odświeżaniu pozycji ataków na liście poprzez obserwacje. Czynnikiem decydującym o wartościowaniu ataków jest różnica pomiędzy dysproporcją kontroli nad postaciami przed i po ich wykonaniu. Gdy wartość ataku zostanie ponownie obliczona poprzez wykorzystanie średniej arytmetycznej z nowej i starej wartości, zostanie on ponownie umiejscowiony w rankingu. Dzięki systemowi „Echo”, ten

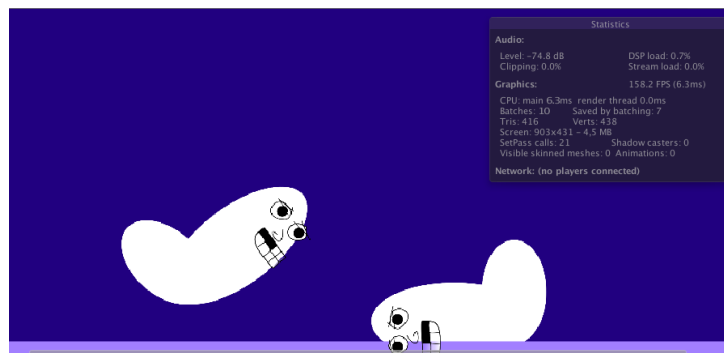
sam atak, zależnie w którym rankingu się znajduje, może mieć inną pozycję w rankingu. Pozwala to sztucznej inteligencji na samodzielne wyspecjalizowanie wyuczonego ataku na poszczególne sytuacje. W przypadku braku ataku dla danej sytuacji, wybierany jest atak eksperymentalny z innych rankingów. Przeciwnie do wykonywania ataków przez komputer, które odbywają się w głównej pętli, obserwacja jest wykonywana równoległe do głównego wątku gry. Współbieżnie działają także aktualizowanie, dodawanie i wybieranie ataków przez system gry. Priorytetyzacja wykonywania ataków przez system na równi z graczem i delegowanie mocniej obciążających prace procesora zadań na współbieżne wątki zapewnia nie tylko płynność rozgrywki, ale także daje gwarancje dynamicznego i reaktywnego pojedynku. Taka implementacja systemu kontroli awatara niesie ze sobą ciekawe korzyści i pozytywne cechy. Przeciwnik będzie posiadać tylko te ataki, które zostaną zaobserwowane u użytkownika. Sprowadza to zaistniałą sytuację do pozycji „mistrz–uczeń”. Wymiana wiedzy pomiędzy graczem a sztuczną inteligencją jest tutaj ściśle powiązana z tym, jakie umiejętności posiada żywy uczestnik walki. Ważne jest, że mechanika ta nie implikuje bezpośrednio sytuacji, w której przeciwny awatar nigdy nie przekroczy umiejętności gracza. Dzięki rozwijanej umiejętności dobierania coraz lepiej ataków do zaistniałej sytuacji może dojść do momentu, w którym system gry będzie lepiej dostosowywał się niż użytkownik. Ze względu na to, że przeciwnik jest wyspecjalizowany do pojedynkowania z poszczególnym użytkownikiem, program została zaprojektowana tak, by wymiana plików z zapisanymi umiejętnościami pomiędzy użytkownikami była możliwie najłatwiejsza. Dzięki temu gracze mogą łączyć się w grupy i trenować swoje awatary nawzajem. Elementy angażujące wymianę doświadczenia z graczami nie tylko werbalnie, ale też fizycznie jest ważnym elementem utrzymania popularności gry, który jest coraz częściej doceniany przez różne firmy. Wspomniane ułatwienie wymiany danych odbywa się poprzez sprowadzenie całej lokalnej logiki sterowania awatarem przeciwnika w grze do dwóch plików posiadających zoptymalizowane dane, który jest łatwy w odczytywaniu i zapisywaniu przez system. Co więcej, „sztucznych inteligencji” albo „przeciwników” gracz może posiadać wiele i w opcjach dokonujemy aktualnej konfiguracji. Wszystko to by wspierać łączące się grupy. Rankingi i turnieje wyuczonych zawodników przez użytkowników na serwerach online to tylko kilka dodatkowych propozycji na rozwój. Powodem wyboru systemu polegającego na obliczaniu różnicy kontroli nad postaciami

jest prosty do obserwacji i daje wymierne korzyści. Natomiast sama mechanika pozycjonowania ataków w rankingach może zostać połączona z innymi pomysłami. Na przykład dzięki neuroewolucji, postać mogłoby dobierać następne ataki względem dokonanych przed chwilą decyzji, a nie na stałe względem panującej sytuacji.



Rysunek 3.2. Ogólna logika gry

Źródło: Opracowanie własne



Rysunek 3.3. Główne menu

Źródło: Opracowanie własne

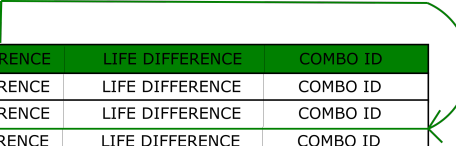
3.3. Rankingi Echo

System Echo określa w jaki sposób ataki gracza są zapisywane do rankingów. Po pierwsze awatar przeciwnika podczas nasłuchiwania obserwuje warunki panujące na obszarze gry. Na warunki takie składają się:

1. Odległość pomiędzy awatarami
2. Kąt pomiędzy awatarami
3. Różnica wektora kontroli pomiędzy awatarami

Panujące warunki w grze są obserwowane przez system, a następnie konwertowane do rozumianych przez awatara parametrów. System podczas zapisywania bierze pod uwagę wszystkie te rankingi, których choć jeden z warunków pasuje do panujących aktualnie. Powoduje to rozproszenie jednego ataku na wiele różnych sytuacji, dzięki czemu taki ruch może osiągać lepsze wyniki ze względu na wybraną ścieżkę dostępu. Podczas zapisu atak jest zapisywany na odpowiedniej pozycji w liście w każdej ze ścieżek ze względu na osiągnięty wynik, czyli ilość zadanych obrażeń minus ilość otrzymanych obrażeń podczas jego wykonywania. Na przykład dla $odlegoci = 1, kt = 2, ycie = 3$ można zaobserwować poniższą sytuację przy dodawaniu nowego ataku:

```
RankEcho [odległość=1] [kąt=dowolnie] [życie=dowolnie] = Add(combo1)
RankEcho [odległość=dowolnie] [kąt=2] [życie=dowolnie] = Add(combo1)
RankEcho [odległość=dowolnie] [kąt=dowolnie] [życie=3] = Add(combo1)
```



1	DISTANCE	ANGLE DIFFERENCE	LIFE DIFFERENCE	COMBO ID
2	DISTANCE	ANGLE DIFFERENCE	LIFE DIFFERENCE	COMBO ID
3	DISTANCE	ANGLE DIFFERENCE	LIFE DIFFERENCE	COMBO ID
4	DISTANCE	ANGLE DIFFERENCE	LIFE DIFFERENCE	COMBO ID
5	DISTANCE	ANGLE DIFFERENCE	LIFE DIFFERENCE	COMBO ID

Rysunek 3.4. Schemat rankingu

Źródło: Opracowanie własne

Przy wyborze ataku przez awatar przeciwnika mamy natomiast sytuację, w której otrzymujemy tylko jeden konkretny ranking: RankEcho[odległość=1][kąt=2][życie=3] = GetElement(o) Przy wybieraniu ataku, to pierwszy z listy w wybranym rankingu zostaje wybrany, ponieważ jest to atak z najwyższym ocenionym wynikiem.

Rankingi Echo zostały zaimplementowane jako tablice pseudo-SQL z czterema kolumnami. Pierwsze trzy kolumny służą jako opis sytuacji, natomiast ostatnia kolumna jest to lista z ustaloną kolejnością, posiadająca zapisane ID ataków. Dane te służą następnie do weryfikacji ataku z zmiennej typu „Dictionary” posiadającej wszystkie ataki. Konstrukcja ta pozwala na tworzenie zapytań SQL względem panującej sytuacji i uzyskania szybkiego dostępu do szeregu rankingów lub jednego konkretnego, w zależności od intencji. Przy uzyskaniu poszukiwanego ID ataku, wykorzystujemy go, by uzyskać dostęp do jego parametrów w czasie $O(1)$. Natomiast głównym powodem zastosowania listy w rankingach zamiast standardowej tablicy jest większa prędkość, z jaką komputer radzi sobie ze zmianą rozmiaru wybranego typu względem odrzuconych. Pseudokod implementacji rankingu „Echo”:

GetCombo ():

```
sql = Select * from Echo where
    rememberedSituation1 = situationNow1 OR
    rememberedSituation2 = situationNow2 OR
    ...
EchoTemp = Echo.Find(sql)
for i=0 to EchoTemp.Count do
    chosenCombo = max(EchoTemp[i], chosenCombo);
end
return chosenCombo;
```

end

AddCombo (newCombo):

```
sql = Select * from Echo where
    rememberedSituation1 = situationNow1 OR
    rememberedSituation2 = situationNow2 OR
```

```
...  
EchoTemp = Echo.Find(sql)  
for i=0 to EchoTemp.Count do  
    EchoTemp[i].Add(newCombo)  
end  
end  
  
ChangeCombo (oldCombo, j):  
    EchoTemp[j].Replace(oldCombo)  
end
```

3.4. Sterowanie i menu

Podczas działania gry, zarówno w menu jak i podczas walki, w celu dołączenia i przejęcia kontroli nad drugim awatarem przez drugiego gracza, musi zostać naciśnięty przycisk „start” na drugim podłączonym kontrolerze. System kontrolujący postać przeciwnika zostanie natychmiast wyłączony aż do następnego przyciśnięcia „start” na tym samym kontrolerze. Gdy tak się stanie, sytuacja powraca do poprzedniego stanu, gdzie komputer przejmuje kontrolę nad postacią przeciwnika i bierze udział zarówno w walce jak i nabieraniu doświadczenia.

Każda z postaci ma dwa ośrodki kontroli. Jeden umiejscowiony na wysokości twarzy. Odpowiada on za kontrole nad górną częścią ciała. Odnosi się to zarówno do odchyłania się jak i atakowania rękoma. Drugi ośrodek znajduje się w dolnej części ciała, na wysokości ogona. Odpowiada on za przemieszczanie się tej części ciała i za kopnięcia. Zadane obrażenie w daną część ciała obniża kontrole nad odpowiednim ośrodkiem. Im mniejsza kontrola, tym wolniejsza reakcja i mniej zadawane obrażenia. Co więcej, gdy suma kontroli nad ośrodkami wynosi poniżej 5 procent, dana postać nie będzie w stanie się podnieść po upadku. Podczas poruszania się, postać może upaść. Upadek oznacza sytuację, w której dana postać znajduje się na ziemi, a jej odchył przekroczył dozwoloną ilość stopni. W takiej sytuacji, dopóki gracz nie wykona akcji podnoszenia, nie może wykonywać żadnych innych ruchów. Im szybciej, tym lepiej, ponieważ leżąca postać, nie mogąc się bronić, jest wystawiona na

ataki przeciwnika. Powoduje to sytuacje, w której kontrola nad rotacją awatara zarówno na ziemi jak i w powietrzu, jest kluczowa dla dobrej walki. Wyskoki to ważny element gry. Przy dobrze wyuczonej obronie, walka na ziemi może nie wystarczyć. Z pomocą dochodzi umiejętność walki w powietrzu. Skok może być niski, zainicjowany za pomocą tylko jednego analogu, albo wysoki, z dwoma analogami. Im mniejsza kontrola nad postacią, tym mniejsze skoki. Podczas skakania postać może obracać się bez ograniczeń. Przy odpowiednich umiejętnościach, gracz, za pomocą odpowiednio dobranych ruchów może znaleźć się za plecami przeciwnika, zadając wiele obrażeń po drodze. Dodatkowo, postać może uzyskać zwiększoną moc wyskoku poprzez trzymanie jednego lub obu analogów (zależnie do którego rodzaju skoku) w dół. Jednak jeśli skok nie nastąpi po lądowaniu, wartość bonusu drastycznie maleje do zera. Wspomniane ośrodki odgrywają główną rolę w poruszaniu się na ziemi. Na podstawie animacji postaci przeciwnika, gracz może spodziewać się ataków na odpowiednich wysokościach (górze/środek/dół). W odpowiedzi można odchylić górną część ciała lub przemieścić dolną w celu uniknięcia ataku. Jeśli brakuje czasu, drugą opcją jest obrona. Użytkownik nadal musi wybrać wysokość, na której będzie się bronić, jednak tym razem reakcja postaci jest niemal natychmiastowa. Minusem tego rozwiązania jest to, że nadal część obrażeń jest rozproszona równomiernie na oba ośrodki ciała. Postać może się bronić na dwa sposoby. Poprzez blokowanie ataków z góry, lub te wycelowane na środku wysokości awatara. W przypadku tych ostatnich, postać obroni atak niezależnie od wysokości, jednak otrzymane obrażenia będą tylko nieznacznie ograniczone. Wyjątkiem są ataki na wysokości środka. W takim razie obrażenia są ograniczone tak samo, jak w przypadku pasującej obrony góry i ataku na tejże wysokości. Lewy trigger odpowiada za obronę lewej ręki i lewej nogi, natomiast prawy trigger prawą ręką i lewą ręką. Pozycja obrony jest determinowana na podstawie wciśniętych „triggerów”. W zależności czy strona obrony (prawa lub lewa) pasuje do strony ataku, obrażenia z ataku zostaną odpowiednio zredukowane. Jeśli nie pasuje, obrażenia zostaną w pełni zadane drugiemu awatarowi. Obronę można wykonywać obustronnie na raz. Do dyspozycji gracza zostały oddane cztery ataki. Lewa i prawa pięść oraz lewe i prawe kopnięcie. Jednak awatar może wykonywać tylko jedną akcję ataku na raz. Dodatkowo blok blokuje postać przed wykonywaniem którejkolwiek z tych czterech akcji. Jednak podczas wykonywania takiej akcji, postać może jednak dowolnie się prze-

mieszcząc i skakać, jednak prędkość i wysokość jest zmniejszona. Input, jaki jest wprowadzany przez system ma dokładnie taką samą strukturę jak input użytkownika dzięki strukturze zapisywanych danych. Tak samo obserwacja, jaką wykonuje program na rzecz postaci przeciwnika dokonywana jest tylko na podstawie wizualnych efektów, jak odległość czy różnica kątów. Dzięki temu użytkownik może mieć poczucie wyrównanych szans i braku faworyzowania przeciwników co jest częstym problemem w innych grach, szczególnie akcji, jak i strategicznych. Oprogramowanie wspiera wszystkie kontrolery posiadające dwa analogi, cztery przyciski i dwa trigger. Oznacza to, że do grania użytkownik musi posiadać urządzenie z opisaną powyżej specyfikacją.

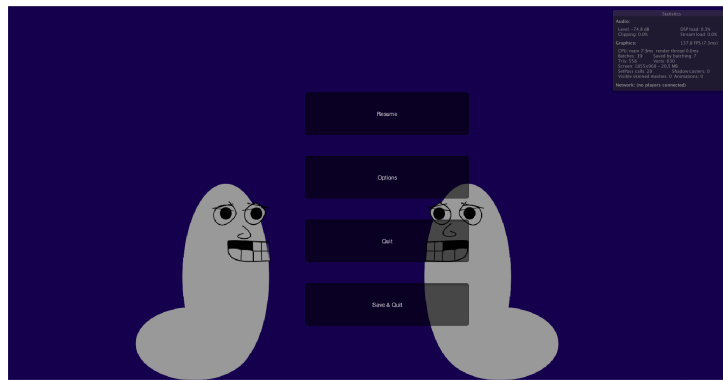
Sterowanie dla kontrolera XBOX i PlayStation 3:

- Kontrola dół – lewy analog
- Kontrola góra – prawy analog
- Lewa pięść – X
- Prawa pięść – B
- Lewa noga – Y
- Prawa noga – A
- Lewa obrona – lewy trigger
- Prawa obrona – prawy trigger
- Skok – $\text{analog.y} > 0.5$
- Wysoki skok – $\text{analogi.y} > 0.5$
- Wstawanie – $\text{analog.y} > 0$
- Ładowanie skoku – $\text{analog.y} < 0$
- Przemieszczanie w lewo – $\text{analogi.x} < 0$
- Przemieszczanie w prawo – $\text{analogi.x} > 0$

- Rotacja w powietrzu w lewo – $\text{analog1.x} > 0$ and $\text{analog2.x} < 0$
- Rotacja w powietrzu w prawo – $\text{analog1.x} < 0$ and $\text{analog2.x} > 0$

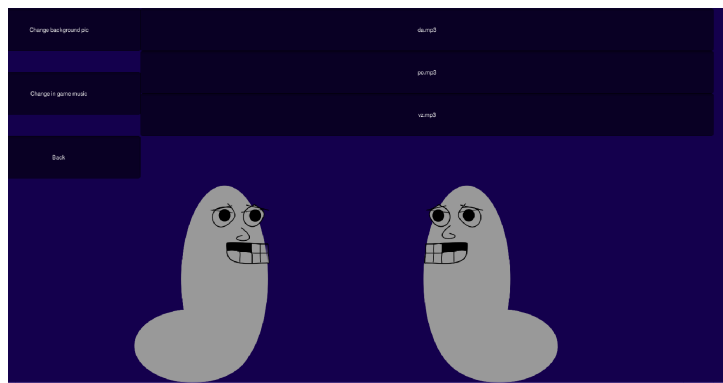
Rozgrywka została zaprojektowana tak, by gracz od razu przenosił się do pojedynku. Jeśli naciśnie jakikolwiek przycisk gra rozpocznie odliczanie i rozpocznie się rozgrywka z domyślnie włączoną sztuczną inteligencją, która od razu zacznie sterować drugim awatarem. Wykryta aktywność na drugim kontrolerze spowoduje wyłączenie systemu kontrolującego postać przeciwnika i kontrola nad drugim awatarem zostanie do niej przydzielona. Menu można przywołać za pomocą przycisku start na kontrolerze. Menu zatrzyma czas aktualnie rozgrywanej sesji oraz oferuje do wyboru:

- Resume – powrót do trwającej walki
- Options – opcje gry. Tutaj użytkownik może zmienić muzykę odgrywaną podczas rozgrywek na swoją własną za pomocą przycisku „Change game music”, może także zmienić tło gry za pomocą przycisku „Change background”. Wszystkie opcje zmuszają użytkownika do wybrania odpowiedniego pliku. Po dokonaniu wyboru użytkownik powraca do wyboru w zakładce „Options” z odświeżonym stanem gry. Natomiast ostatnim przyciskiem jest opcja „Back”, który powraca gracza do głównego menu.
- Quit – nadpisuje nową logikę dla awatara w odpowiednim pliku i wyłącza program
- Quit without save – wyłącza program bez nadpisywania nowej logiki



Rysunek 3.5. Główne menu

Źródło: Opracowanie własne



Rysunek 3.6. Opcje wybieranie plików

Źródło: Opracowanie własne

Zakończenie

Cały projekt miał za zadanie umożliwienie trenowania własnego przeciwnika. Awatar zdobywa umiejętności powoli, czego można się było spodziewać. Same walki z nim bywają ciekawe i zróżnicowane. Wbudowane menu, balans postacią i charakterystyczne sterowanie może być inspiracją do następnych projektów. Dynamicznie rozwijający się silnik Unity3D i występujące problemy z kompatybilnością z systemem Ubuntu prowadził do spowolnienia procesu deweloperskiego. Gra umożliwia także na pojedynki dla dwóch graczy, które można traktować jako oddzielny tryb rozgrywek, umożliwiający sesje dla znajomych, jak i kompetytywne turnieje. Gra będzie udostępniona na szeregu różnych platformach udostępniających oprogramowanie zarówno na konsole jak i komputery na najpopularniejszych systemach operacyjnych w tym Windows, OSX i Ubuntu.

DODATEK A

Gra „Fighty”

Spakowany plik z grą i wszystkimi niezbędnymi plikami do poprawnego działania na nośniku USB.

Słownik

- CO-OP – skrót z angielskiego od „cooperation”, oznacza gry z mechaniką zaprojektowaną do rozgrywki dla bardzo małej ilości osób naraz.
- Joint – łącznik, mechanizm popularny w silnikach z symulacją fizyki, pomagający wyznaczyć wzajemny wpływ danych obiektów na siebie.
- FPS – skrót z angielskiego od „frames per second”, oznacza ilość klatek na sekundę. Jest to główny wyznacznik płynności gry na danym sprzęcie. Na przykład, dla konsol, akceptowalna minimalna ilość klatek na sekundę to 30, a na komputerach 60. Natomiast standardem w branży filmowej to średnio 24 klatek na sekundę.
- Awatar – przedstawiciel gracza w wirtualnym świecie. W grze jest to postać, nad którą użytkownik lub system gry ma kontrolę.

Bibliografia

- [1] *Dokumentacja Unity3D* (2018), <https://docs.unity3d.com/ScriptReference/GUI.BeginScrollView.html>.
- [2] *Rozważania nad sztuczną inteligencją* (2018), <https://www.ted.com/topics/ai>.
- [3] *Historia gier wideo* (06.2018), https://en.wikipedia.org/wiki/History_of_video_games.
- [4] *Nauczanie maszynowe* (2018), https://en.wikipedia.org/wiki/Machine_learning.
- [5] *Turner Whitted „Multi-bounce Recursive Ray Tracing* (1979).

Spis tabel

1.1. Rozwojowa sztuczna inteligencja w grach	11
1.2. Koszt produkcji w porównaniu do zarobków	12

Spis rysunków

3.1. Główne menu	24
3.2. Ogólna logika gry	27
3.3. Główne menu	27
3.4. Schemat rankingu	28
3.5. Główne menu	34
3.6. Opcje wybieranie plików	34

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