AKADEMIA GÓRNICZO-HUTNICZA

im. Stanisława Staszica w Krakowie

WYDZIAŁ INŻYNIERII MECHANICZNEJ I ROBOTYKI

Praca dyplomowa
inżynierska

Paula Karbowniczek
Imię i nazwisko

Inżynieria Akustyczna
Kierunek studiów

Wersja demonstracyjna silnika audio do gier RAYAV
Temat pracy dyplomowej

dr inż. Bartosz Ziółko
Promotor pracy

……..
Ocena

Kraków, rok 2013/2014

Kraków, dn……..

Imię i nazwisko: Paula Karbowniczek

Nr albumu: 241160

Kierunek studiów: **Inżynieria Akustyczna**

OŚWIADCZENIE

Świadomy/a odpowiedzialności karnej za poświadczanie nieprawdy oświadczam, że niniejszą inżynierską pracę dyplomową wykonałem/łam osobiście i samodzielnie oraz nie korzystałem/łam ze źródeł innych niż wymienione w pracy.

Jednocześnie oświadczam, że dokumentacja praca nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz. U. z 2006 r. Nr 90 poz. 631 z późniejszymi zmianami) oraz dóbr osobistych chronionych prawem cywilnym. Nie zawiera ona również danych i informacji, które uzyskałem/łam w sposób niedozwolony. Wersja dokumentacji dołączona przeze mnie na nośniku elektronicznym jest w pełni zgodna z wydrukiem przedstawionym do recenzji.

Zaświadczam także, że niniejsza inżynierska praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów wyższej uczelni lub tytułów zawodowych.

……..

podpis dyplomanta

Kraków, ……..

Imię i nazwisko: Paula Karbowniczek
Adres korespondencyjny: ul. Iłżecka 139, 27-400 Ostrowiec Św.
Temat pracy dyplomowej inżynierskiej: Wersja demonstracyjna silnika audio do gier RAYAV
Rok ukończenia: 2014
Nr albumu: 241160
Kierunek studiów: Inżynieria Akustyczna

OŚWIADCZENIE

Niniejszym oświadczam, że zachowując moje prawa autorskie , udzielam Akademii Górniczo-Hutniczej im. S. Staszica w Krakowie nieograniczonej w czasie nieodpłatnej licencji niewyłącznej do korzystania z przedstawionej dokumentacji inżynierskiej pracy dyplomowej, w zakresie publicznego udostępniania i rozpowszechniania w wersji drukowanej i elektronicznej[1].

Publikacja ta może nastąpić po ewentualnym zgłoszeniu do ochrony prawnej wynalazków, wzorów użytkowych, wzorów przemysłowych będących wynikiem pracy inżynierskiej[2].

Kraków, ...… ……..
data *podpis dyplomanta*

[1] Na podstawie Ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (Dz.U. 2005 nr 164 poz. 1365) Art. 239. oraz Ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz.U. z 2000 r. Nr 80, poz. 904, z późn. zm.) Art. 15a. "Uczelni w rozumieniu przepisów o szkolnictwie wyższym przysługuje pierwszeństwo w opublikowaniu pracy dyplomowej studenta. Jeżeli uczelnia nie opublikowała pracy dyplomowej w ciągu 6 miesięcy od jej obrony, student, który ją przygotował, może ją opublikować, chyba że praca dyplomowa jest częścią utworu zbiorowego."

[2] Ustawa z dnia 30 czerwca 2000r. – Prawo własności przemysłowej (Dz.U. z 2003r. Nr 119, poz. 1117 z późniejszymi zmianami) a także rozporządzenie Prezesa Rady Ministrów z dnia 17 września 2001r. w sprawie dokonywania i rozpatrywania zgłoszeń wynalazków i wzorów użytkowych (Dz.U. nr 102 poz. 1119 oraz z 2005r. Nr 109, poz. 910).

Kraków, dnia .....

AKADEMIA GÓRNICZO-HUTNICZA

WYDZIAŁ INŻYNIERII MECHANICZNEJ I ROBOTYKI

TEMATYKA PRACY DYPLOMOWEJ INŻYNIERSKIEJ
dla studenta IV roku studiów stacjonarnych

Paula Karbowniczek
imię i nazwisko studenta

TEMAT PRACY DYPLOMOWEJ INŻYNIERSKIEJ:

Wersja demonstracyjna silnika audio do gier RAYAV

Promotor pracy: dr inż. Bartosz Ziółko

Recenzent pracy: [Tytuł, imię i nazwisko Recenzenta] .....
Podpis dziekana:

PLAN PRACY DYPLOMOWEJ:

1. Omówienie tematu pracy i sposobu realizacji z promotorem.
2. Zebranie i opracowanie literatury dotyczącej tematu pracy, wybór narzędzi.
3. Praca nad projektem, konsultowanie rozwiązań z promotorem.
4. Testowanie gotowej gry i zatwierdzenie przez promotora.
5. Opracowanie redakcyjne.

Kraków, ...… ……..
data *podpis dyplomanta*

TERMIN ODDANIA DO DZIEKANATU: ..... **20**..... **r.**

.....
podpis promotora

Akademia Górniczo-Hutnicza im. Stanisława Staszica Kraków, ...
Wydział Inżynierii Mechanicznej i Robotyki
Kierunek: Inżynieria Akustyczna

Paula Karbowniczek
Praca dyplomowa inżynierska
Wersja demonstracyjna silnika audio do gier RAYAV.
Opiekun: dr inż. Bartosz Ziółko

STRESZCZENIE

Niniejsza praca dyplomowa zawiera opis procesu projektowania gry komputerowej, będącej wersją demonstracyjną możliwości silnika audio RAYAV, tworzonego przez Zespół Przetwarzania Sygnałów Akademii Górniczo-Hutniczej. Udostępniona technologia pozwala na urozmaicenie rozgrywki poprzez nietypowe zastosowanie efektów dźwiękowych. Autor stawia pytanie, czy w grze komputerowej możliwe jest wykorzystywanie sygnału audio jako głównego źródła informacji o tym, co dzieje się na scenie. W opracowaniu opisywane są przyjęte założenia oraz wykorzystywane metody ich realizacji. Prezentowana jest również specyfika indywidualnego sposobu pracy nad stworzeniem w pełni funkcjonalnej gry komputerowej, od etapu odpowiedniego przygotowywania stanowiska pracy, przez rozwój koncepcji, po testowanie kolejnych wersji. Szczególną uwagę poświęcono roli dźwięku w grach komputerowych, a także procesom związanym z obróbką nagrań oraz metodom umieszczania przygotowanych efektów dźwiękowych w grze, przy użyciu silnika audio RAYAV. W finalnej wersji projektu udało się osiągnąć zamierzony cel, dlatego też zaproponowano dalsze opcje jego rozwoju.

AGH University of Science and Technology Kraków, the.....
Faculty of Mechanical Engineering and Robotics
Field of Study: Acoustic Engineering

Paula Karbowniczek
Engineer Diploma Thesis
Demo version of RAYAV audio engine for games.
Supervisor: dr inż. Bartosz Ziółko

SUMMARY

This diploma thesis contains a description of a process of designing a video game that presents the capability of RAYAV audio engine, a new solution developed by Digital Signal Processing Team from University of Science and Technology (AGH). With this technology it is possible to improve gameplay through a creative use of sound effects. By this project, author is trying to answer the question, if the player can use sounds to estimate his location in the virtual world, without getting any other additional information. The description consists of a number of ideas, as well as methods of their implementation. It also documents each particular stage of work as an individual game designer in order to create fully functional and playable game, starting from organizing workspace, through developing ideas, ending up with testing different versions of the game. Much attention is focused on the subject of editing and customizing samples, importance of sound in video games and methods of using RAYAV audio engine to play prepared sound effects in the game. The goal of the project was achieved with its final version, therefore the author suggested some additional ideas for further improvements.

Serdeczne podziękowania dla dr inż. Bartosza Ziółki
oraz mgr inż. Tomasza Pędzimęża
za pomoc, wsparcie i motywację

Spis treści

1. Wstęp....9
1.1 Cele pracy....9
1.2 Opis rozdziałów....9
2. Wprowadzenie....10
2.1 Realizacja efektów dźwiękowych w grach komputerowych....11
2.2 Soundtracing....12
2.2.1 Raytracing....13
2.2.2 RAYAV....15
2.3 Silnik gry....16
3. Narzędzia....17
3.1 Wybór silnika gry....17
3.2 Edycja dźwięku....19
4. Realizacja projektu....19
4.1 Opis gry....20
4.2 Grafika....20
4.3 Dźwięk....21
4.4 Mechanika....21
4.4.1 Kontrola postaci....22
4.4.2 Charakterystyka kamery....22
4.4.3 Opis trybu (nie)widzialnego postaci....23
4.4.4 Przeciwnicy (klasa Enemy)....24
4.4.4.1 Inteligentne wyznaczanie ścieżek....24
4.4.4.2 Patrolowanie trasy....25
4.4.4.3 Zasięg przeciwnika....25
4.4.4.4 Diagram stanów....26
4.4.5 System kolizji....27
4.4.6 Rozmieszczenie dźwięków....28
5. Zakończenie....29
5.1 Podsumowanie....29
5.2 Przyszłość projektu....30
Bibliografia....31

1. Wstęp

Na wybór tematu pracy inżynierskiej wpłynęły przede wszystkim zainteresowania autora szeroką dziedziną gier komputerowych, a w szczególności realizacją dźwięku w jej kontekście. Nie bez znaczenia była również perspektywa współpracy z Zespołem Przetwarzania Sygnałów Akademii Górniczo-Hutniczej przy interesującym projekcie badawczym RAYAV [1]. Wreszcie, taka forma realizacji pracy dyplomowej wydawała się być idealną okazją do sprawdzenia posiadanych umiejętności w praktyce, a jednocześnie umożliwiała zdobycie nowego doświadczenia w zakresie tworzenia gier komputerowych.

1.1 Cele pracy

Głównym celem niniejszej pracy inżynierskiej jest zademonstrowanie możliwości silnika audio do gier RAYAV poprzez prezentację jego użycia w przykładowej grze komputerowej. Istotne jest pokazanie, w jaki sposób wpływa on na poprawienie jakości odtwarzanego dźwięku oraz co wyróżnia go spośród znanych już rozwiązań. Ponadto, przyjęty sposób realizacji tematu jest niejako poszukiwaniem nowych form rozgrywki. Pozwala jednocześnie ocenić, czy opracowywana w projekcie badawczym RAYAV technologia jest wystarczająca, aby odbiorca potraktował efekty dźwiękowe jako równorzędne, a nie jedynie uzupełniające grafikę, źródło informacji.

Treść pracy dyplomowej przybliża również proces powstawania gier komputerowych, ze szczególnym uwzględnieniem realizacji dźwięku. Porusza dylematy i problemy, przed którymi staje większość początkujących twórców oraz przedstawia ich przykładowe rozwiązania.

1.2 Opis rozdziałów

Praca, pomijając rozdział wstępny, składa się z następujących części:

Rozdział 2 – *Wprowadzenie* – krótko charakteryzuje branżę gier komputerowych i opisuje w jaki sposób obecnie realizowane są w nich efekty dźwiękowe. Zawiera

również podstawowe informacje teoretyczne na temat silnika gry, ray oraz sound tracingu, a także sposobu implementacji tych metod w silniku audio RAYAV.

Rozdział 3 – *Narzędzia* – opisuje wykorzystywane przez autora oprogramowanie oraz uzasadnia jego wybór.

Rozdział 4 – *Realizacja projektu* – jest podsumowaniem zadań wykonanych w części praktycznej pracy dyplomowej. Znajduje się w nim szczegółowy opis koncepcji gry oraz zastosowanych do jej realizacji rozwiązań.

Rozdział 5 – *Zakończenie* – omawia uzyskany w wyniku pracy rezultat. Zawiera również przemyślenia autora na temat potencjalnej kontynuacji projektu.

2. Wprowadzenie

Przemysł gier komputerowych, chociaż stosunkowo młody w porównaniu z innymi dziedzinami rozrywki, stanowi wśród nich najbardziej obiecujący i najszybciej rozwijający się rynek. Łatwy dostęp do różnego rodzaju platform: konsol, telefonów, tabletów czy komputerów w połączeniu z ogromną różnorodnością gatunków gier sprawia, że grupa docelowa tej branży jest właściwie nieograniczona. Wiele dzisiejszych produkcji znacznie wykracza poza prostą interaktywną rozrywkę, proponując graczowi dopracowaną w szczegółach historię, popartą efektowną grafiką i oryginalną ścieżką dźwiękową, chcąc tym samym zapewnić mu jak największe poczucie realizmu. Twórcy gier komputerowych dużo uwagi skupili na poprawie strony graficznej, projektując coraz bardziej szczegółowe i dokładne modele obiektów i postaci, lepszej jakości tekstury i efekty wizualne. Jednocześnie zaszły też zmiany w podejściu do udźwiękowienia gier. Często korzysta ono z różnych schematów dobrze znanych już z filmów. W podobny sposób wpływa na emocje odbiorcy, a tym samym buduje klimat całej gry. Tworzenie ścieżek i efektów dźwiękowych do największych projektów przypomina coraz częściej procesy znane z produkcji kinowych. Zdarza się nawet, że dialogi nagrywane są przez znanych aktorów, a muzyka z gry wydawana jest jako osobny album. Wszystko to jednak nie wykorzystuje w pełni potencjału, jaki oferują nowe technologie.

2.1 Realizacja efektów dźwiękowych w grach komputerowych

Sposób udźwiękowienia gier komputerowych zależy od wielu czynników, takich jak gatunek, stylistyka, budżet czy platforma docelowa. Można się spodziewać, że nagrany najwyższej jakości sprzętem odgłos wystrzału z prawdziwej broni nie będzie pasował do kolorowej gry platformowej, w której główny bohater jest postacią kreskówkową. Innymi zasadami rządzą się też gry na platformy mobilne. W tym przypadku liczy się przede wszystkim szybka rozgrywka, użytkownicy rzadko skupieni są na szczegółach, a wyszukane efekty wizualne czy dźwiękowe mogą nawet czasami przeszkadzać w odbiorze. W produkcji takich gier zwraca się też uwagę na rozmiar plików dźwiękowych, co często wiąże się z nieco słabszą ich jakością.

Wspomniane wcześniej dążenie do realizmu dotyczy przede wszystkim dużych projektów, tzw. produkcji AAA. Nie jest to największa, ale zdecydowanie najbardziej dochodowa część rynku i to właśnie w tym sektorze poszukiwane są nowe rozwiązania, które mogą zapewnić najwyższą jakość produktu finalnego. Standardem dla takich gier jest przestrzeń trójwymiarowa. Oznacza to, że dźwięk musi być w niej umiejscowiony w taki sposób, aby gracz miał wrażenie jej autentyczności. W tym celu wykorzystywane są profesjonalnie przygotowane efekty dźwiękowe. Jedną z możliwości jest nagrywanie ich w warunkach jak najbardziej zbliżonych do rzeczywistości wykreowanej w grze, w podobny sposób, w jaki odbywa się to podczas produkcji filmu. W przypadku gier komputerowych zadanie jest jednak znacznie trudniejsze, ponieważ charakteryzują się one nieliniowością. Nie da się przewidzieć w jaki dokładnie sposób użytkownik będzie poruszał się w przestrzeni wirtualnej, ile czasu spędzi w danej lokacji, ani pod jakim kątem i z jakiej odległości będzie obserwował źródło dźwięku [2]. Wszystkie te zależności wiążą się z koniecznością nagrania ogromnej ilości próbek - np. odgłos kroków powinien zostać zarejestrowany w różnych pomieszczeniach i na różnych powierzchniach. Zadaniem realizatora dźwięku jest później odpowiednia edycja tych odgłosów. Większą odległość źródła dźwięku od gracza zasymulować można zmniejszając głośność efektu, a kierunek – regulując balans pomiędzy kanałami nagrania stereofonicznego.

Posuwając się o krok dalej, pewne efekty uzyskiwane na etapie rejestracji próbek można zastąpić podobnymi, generowanymi przez programy do obróbki dźwięku. W ten sposób na przykład zasymulować można kształt, wielkość i inne

aspekty pomieszczenia poprzez dodanie do oryginalnego nagrania efektu pogłosu o pasujących parametrach oraz podbicie lub tłumienie odpowiednich częstotliwości. Często uzyskuje się tym samym lepszą jakość dźwięku, chociaż jego naturalność zależy wtedy w znacznym stopniu od umiejętności realizatora. Wszystkie te czynności stanowią jednak długi proces, wymagający obróbki wszystkich używanych w grze dźwięków osobno dla każdego źródła.

Istnieje również możliwość dodawania wspomnianych efektów w czasie rzeczywistym. Oznacza to, że należy jedynie zdefiniować jakie parametry mają one przyjąć w zależności od położenia gracza i źródła dźwięku. Wówczas silnik audio wprowadzi zmiany w zarejestrowanym sygnale dopiero w momencie odtworzenia go w grze. Pierwsza z wymienionych metod nazywana jest często aktywną, druga natomiast – pasywną. Oczywiście mogą być stosowane jednocześnie lub zamiennie, w zależności od potrzeb i posiadanej bazy dźwięków [3].

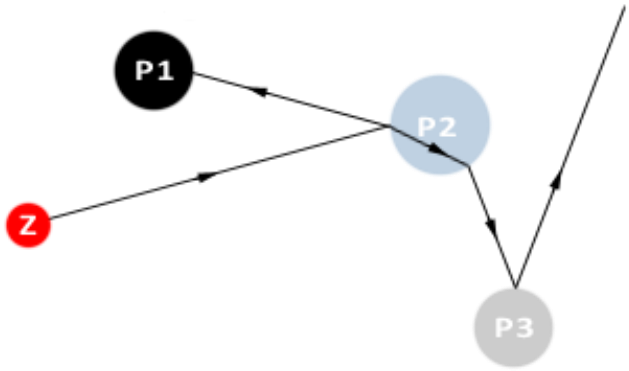
Dysponując możliwością przetwarzania dźwięku w grach w czasie rzeczywistym, poszukiwane są nowe rozwiązania, które pozwalałyby graczom na jeszcze większą interakcję ze światem wirtualnym. Przykładem może być inteligentny silnik audio, który dopasowuje parametry operacji wykonywanych na próbkach dźwiękowych w zależności od zmiennych występujących w grze, czyli na przykład wraz ze spadkiem „życia" postaci, efekty dźwiękowe mogą stawać się mniej wyraźne, dając wrażenie ogłuszenia, a tło muzyczne bardziej dramatyczne [4].

2.2 Soundtracing

Rozwiązaniem, które może w znacznym stopniu uprościć i zautomatyzować proces udźwiękawiania gier, a zarazem poprawić jakość i autentyczność odtwarzanych efektów jest soundtracing. Opiera się on na algorytmach śledzenia promieni lub ich wiązek w odniesieniu do rozprzestrzeniania się fali dźwiękowej. Obecnie, ze względu na dużą złożoność obliczeniową, są one wykorzystywane głównie do generowania obrazów trójwymiarowych, szczególnie w sytuacjach, które nie wymagają przetwarzania w czasie rzeczywistym (filmy, nieruchome grafiki). Biorąc jednak pod uwagę duże podobieństwa w propagacji fal świetlnych i akustycznych, metody te mogą być z powodzeniem stosowane także przy generowaniu dźwięku.

2.2.1 Raytracing

Raytracing to technika polegająca na śledzeniu biegu promienia. Za jego początek można przyjąć zarówno położenie obserwatora, jak i położenie źródła. Druga możliwość odzwierciedla naturalną drogę, którą pokonuje promień i chociaż daje lepsze efekty, wymaga większej mocy obliczeniowej [5]. Algorytm rozpoczyna się od znalezienia punktu, w którym dany promień napotkał na swojej drodze obiekt. W tym miejscu, w zależności od właściwości przeszkody, zachodzą odpowiednie zjawiska: refrakcja, czyli załamanie fali podczas przejścia pomiędzy ośrodkami o różnej gęstości, częściowe lub całkowite odbicie lub pochłonięcie fali, pozostała część ulega transmisji, czyli przepuszczeniu przez substancję. Oczywiście wszystkie wymienione efekty mogą występować jednocześnie, wówczas jeden promień może przy napotkaniu obiektu podzielić się na kilka o mniejszej energii. Analogicznie, powyższy algorytm zachodzi dla każdego z nich aż do momentu całkowitego (lub w przybliżeniu całkowitego) wytracenia energii. Rys. 2.1 przedstawia schemat śledzenia biegu promienia.



Rys. 2.1 Schemat śledzenia biegu promienia: Z – źródło, P1 – obiekt całkowicie pochłaniający, P2 – obiekt przepuszczający i odbijający, P3 – obiekt całkowicie odbijający

Zaletą tej techniki jest możliwość równoległego śledzenia wielu promieni. Pewną modyfikacją metody jest zastąpienie pojedynczego promienia całą ich wiązką o kształcie ostrosłupa lub stożka. Pojedynczy promień trafia w punkt na powierzchni przeszkody, więc może zdarzyć się, że ograniczona ich ilość spowoduje pominięcie jakiegoś elementu. Rozpatrywanie całych obszarów pokrywanych przez wiązki pozwala na wyeliminowanie tej niedoskonałości.

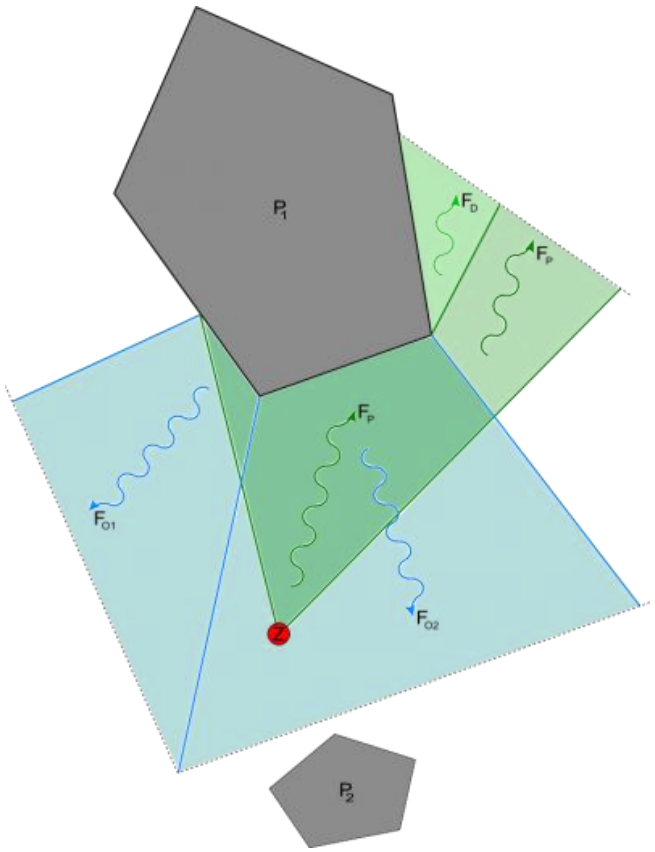
W praktyce, najbardziej efektowne rezultaty użycia raytracingu obserwować można na mocno odbijających powierzchniach obiektów trójwymiarowych, czyli wszelkiego rodzaju lustrach, powierzchniach wody czy karoseriach samochodów (rys. 2.2). Metoda ta znacznie ułatwia i przyspiesza też pracę z oświetleniem w filmach, gdyż pozwala na zredukowanie ogromnej ilości źródeł światła, kiedyś niezbędnych do imitacji naturalnej jego propagacji. Z technologii tej korzysta między innymi popularna wytwórnia filmów animowanych, *Disney Pixar* [6].

Rys. 2.2 Trójwymiarowa scena, prezentująca możliwości silnika raytracingu NVIDIA OptiX, źródło: www.nvidia.com

W odniesieniu do dźwięku, raytracing wykorzystywany jest w programach symulujących akustykę pomieszczeń, takich jak na przykład *CATT Acoustic*. Na podstawie modelu pomieszczenia, położenia odbiornika i źródeł dźwięku oraz rozmieszczenia materiałów o określonych współczynnikach pochłaniania i rozpraszania dźwięku, program wykonuje obliczenia poszczególnych parametrów akustycznych. Jedna z wykorzystywanych metod polega właśnie na tym, że promienie emitowane są ze źródła we wszystkich kierunkach, a następnie śledzone aż do momentu dotarcia do odbiornika lub całkowitego zaniku energii, uwzględniając po drodze odbicia od zdefiniowanych powierzchni. Ze względu na długi czas wykonywanych obliczeń, to rozwiązanie znajduje zastosowanie jedynie w akustyce architektonicznej.

2.2.2 RAYAV

Temat symulacji naturalnej akustyki pomieszczeń w grach komputerowych podjęty został przez Zespół Przetwarzania Sygnałów Akademii Górniczo – Hutniczej w projekcie badawczym RAYAV. Istotną dla tej pracy inżynierskiej jest część projektu, odpowiedzialna za przetwarzanie dźwięku, czyli silnik audio oparty na soundtracingu. Wykorzystuje on metodę śledzenia wiązek, co pozwala na zamodelowanie wszelkich zjawisk akustycznych towarzyszących propagacji fali dźwiękowej w ośrodku materialnym (rys. 2.3).



Rys. 2.3 Schemat przedstawiający metodę śledzenia wiązek: Z – źródło dźwięku, P_1, P_2 – przeszkody, F_p – wiązka pierwotna, F_d – zjawisko dyfrakcji, F_o – zjawisko odbicia, źródło: www.dsp.agh.edu.pl

Uwzględnione zostały nie tylko najbardziej oczywiste efekty, takie jak pochłanianie i odbicie dźwięku w zależności od stosowanych materiałów, ale także na przykład dyfrakcja, czyli zjawisko ugięcia fali na krawędziach przeszkód lub też transmisja, czyli zjawisko przenikania fali dźwiękowej, dzięki czemu gracz będzie

w stanie na przykład usłyszeć, co dzieje się za zamkniętymi drzwiami. Oprócz tego, silnik symuluje także efekt Dopplera, który polega na zmianie częstotliwości dźwięku docierającego do odbiorcy w sytuacji kiedy źródło względem niego jest w ruchu. Wszystkie te zjawiska w określony sposób wpływają na zmianę charakterystyki częstotliwościowej sygnału. Odgłosy które docierają do odbiorcy ze wszystkich źródeł są więc zmodyfikowane w zależności od drogi, jaką przebyła wiązka. Na koniec są one ze sobą miksowane z uwzględnieniem kierunku, z którego dotarły do słuchacza, dając w rezultacie ostateczny dźwięk, który zostanie odtworzony graczowi.

Opisany sposób działania silnika jest dość dokładnym odzwierciedleniem rzeczywistych zjawisk fizycznych towarzyszących propagacji fali dźwiękowej. Pozwala to zakładać, że oparte na nim gry komputerowe charakteryzować będą się wyjątkowo realistycznym udźwiękowieniem. Równie istotną zaletą jego zastosowania jest znaczne uproszczenie pracy realizatora dźwięku. Jego zadanie sprowadza się do rozmieszczenia na scenie poszczególnych źródeł dźwięku oraz zdefiniowania parametrów i przypisania materiałów do odpowiednich obiektów. Na tej podstawie silnik wykonuje obliczenia w czasie rzeczywistym, symulując opisane wyżej zjawiska i efekty.

2.3 Silnik gry

Silnik gry to część kodu, która zajmuje się realizacją najbardziej podstawowych funkcji. Składa się on z wielu modułów odpowiedzialnych między innymi za wyświetlanie grafiki (silnik graficzny), odtwarzanie dźwięku (silnik audio), obsługę wejść czy symulację zjawisk fizycznych, takich jak na przykład grawitacja lub kolizje pomiędzy obiektami. Wszystkie te elementy są uniwersalne i mogą być wielokrotnie wykorzystywane w różnych projektach, są pewnego rodzaju bazą, na której twórcy gier komputerowych opierają się rozpoczynając pracę nad kolejnymi tytułami.

Decyzja o tym, jaki silnik wykorzystywany będzie podczas tworzenia gry, powinna być podjęta w zależności od rozmiaru produkcji, przeznaczonego na nią czasu, budżetu i umiejętności zespołu. Jednym z rozwiązań jest stworzenie własnego silnika i chociaż pozwala ono na najlepsze dopasowanie wszelkich funkcji i zastosowań do swoich potrzeb, nie jest zbyt często wybieraną opcją, ponieważ proces ten jest dość czasochłonny. Nawet największe studia gier komputerowych nie zawsze decydują się

na tę możliwość. Na rynku istnieje wiele gotowych rozwiązań, które oferują użytkownikom rozwinięty, często intuicyjny interfejs, proste w obsłudze narzędzia i możliwość pisania kodu gry za pomocą skryptów. Niektóre z nich wymagają więcej programowania, inne sprowadzają się do korzystania z oferowanych przez aplikację narzędzi. Część z nich przeznaczona jest do projektowania gier jedynie w przestrzeni dwuwymiarowej. Różnią się też między innymi funkcjonalnością czy jakością grafiki [7]. Taka różnorodność znacznie ułatwia proces projektowania gier komputerowych, szczególnie osobom, które nie mają w tym jeszcze doświadczenia. Nie oznacza to jednak, że ogólnodostępne rozwiązania skierowane są tylko do amatorów. Najlepszym przykładem może być darmowy silnik *Unreal Engine (UDK),* dzięki któremu powstały między innymi takie serie gier jak *Mass Effect*, *Bioshock* i *Gears of War* [8].

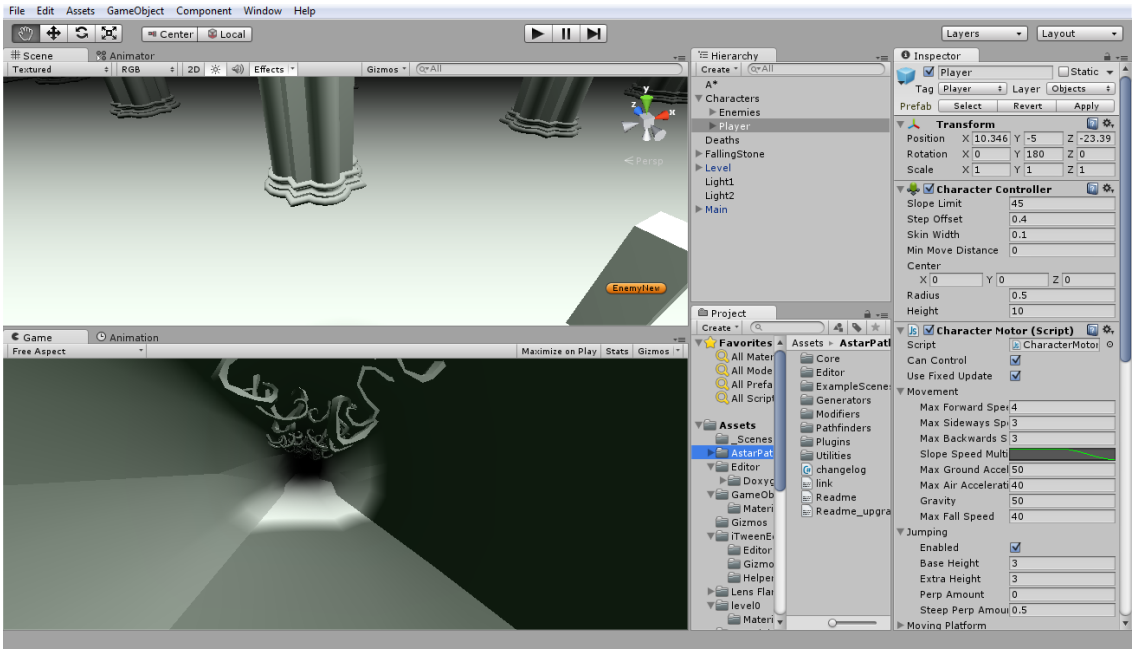
3. Narzędzia

Przed przystąpieniem do właściwej pracy nad projektem, niezbędne było określenie preferowanych narzędzi do jego wykonania. Opisywane działania i ich rezultat są wynikiem własnej pracy autora, przeprowadzonej w porozumieniu i z wykorzystaniem narzędzi udostępnionych przez zespół pracujący nad projektem badawczym RAYAV. Ponieważ silnik audio nie był jeszcze gotowym produktem, przede wszystkim należało się zastanowić nad możliwościami jego implementacji w kontekście tworzonej gry komputerowej.

3.1 Wybór silnika gry

Biorąc pod uwagę charakter projektu, odpowiednią decyzją wydawało się skorzystanie z jednego z gotowych silników. W temacie gier trójwymiarowych, najbardziej obiecującymi i najszybciej rozwijającymi się rozwiązaniami są *Unity 3D* oraz wspomniany już wcześniej *UDK*. Po porównaniu obu propozycji, wybór padł na *Unity 3D* (rys. 3.1), ponieważ jest to oprogramowanie o bardziej przystępnym i intuicyjnym interfejsie, łatwym do obsługi nawet bez wcześniejszego doświadczenia z tego typu programami. Dodatkowo nie bez znaczenia okazała się dostępna

w Internecie rozbudowana dokumentacja i duża ilość materiałów do nauki. *Unity* oferuje także wiele gotowych rozwiązań w postaci wbudowanych komponentów i możliwych do pobrania dodatków. Umożliwia też pisanie własnego kodu w popularnych językach programowania, takich jak C# i JavaScript. Chociaż *UDK* cechuje się lepszej jakości grafiką i bardziej dokładną symulacją zjawisk fizycznych, stawia użytkownikom dość duże wymagania sprzętowe [8], przez co uważane jest za narzędzie skierowane raczej do profesjonalistów. Głównym problemem wybranego oprogramowania jest licencja. W darmowej wersji silnika, *Unity Free*, niektóre opcje, w tym też takie, które później okazały się kluczowe dla projektu, są zablokowane [9]. Pełna funkcjonalność dostępna jest w wydaniu *Unity Pro*, która wiąże się z zakupem kosztownej w porównaniu do innych silników licencji. Gra ostatecznie powstała przy użyciu wersji próbnej *Unity Pro.*



Rys. 3.1 Okno programu *Unity 3D*

W celu zintegrowania silnika audio RAYAV z *Unity 3D*, Zespół Przetwarzania Sygnałów dostarczył biblioteki, pozwalające na uruchamianie poszczególnych funkcji z poziomu *Unity*, poprzez skrypty pisane w języku C#. Właśnie na tym etapie wersja *Free* oprogramowania okazała się być niewystarczająca, ponieważ nie pozwalała na korzystanie z bibliotek zawierających kod natywny (C++) [10].

3.2 Edycja dźwięku

Do odpowiedniego przygotowania efektów dźwiękowych wykorzystany został program *Samplitude Pro X Suite*, czyli cyfrowa stacja robocza do obróbki dźwięku (DAW). Ze względu na rozmiar projektu zadecydowano, że wystarczające będą posiadane efekty dźwiękowe, pochodzące z bibliotek i zbiorów internetowych, toteż pominięty został etap ich rejestracji. Aby uzyskać dopasowane do potrzeb projektu nagrania, konieczna była edycja i obróbka wybranych próbek.

4. Realizacja projektu

Główną ideą projektu było jak najlepsze przedstawienie potencjału silnika audio RAYAV. Postawiono więc pytanie, czy wykreowana akustyka przestrzeni wirtualnej będzie na tyle zbliżona do naturalnej, że gracz będzie potrafił bez pomocy innych zmysłów lokalizować poszczególne źródła dźwięku i na tej podstawie poruszać się po planszy. Chociaż istnieją już gry oparte na podobnych założeniach (bez grafiki, gdzie jedyną informacją jest odpowiednio przetworzony dźwięk), to są to produkcje skierowane przede wszystkim do osób niewidomych, które o wiele sprawniej potrafią określić swoje położenie na podstawie docierających do nich dźwięków. Należy oczywiście pamiętać, że nie wszystkim przychodzi to z łatwością. Początkowa realizacja pomysłu, czyli stworzenie labiryntu, z którego gracz miałby znaleźć wyjście nic nie widząc, okazała się więc dla przeciętnego odbiorcy mało atrakcyjna i wręcz nudna. Koncepcja została nieco zmodyfikowana, pozwalając graczowi w pewnym stopniu na podglądanie sceny, dzięki czemu łatwiej jest mu przypisać w wyobraźni odpowiednie efekty dźwiękowe do poszczególnych obiektów.

Ponieważ projekt jest wersją demonstracyjną, przygotowana została jedna scena, dzięki której zaprezentowany zostanie sposób działania poszczególnych funkcji. Dla uzyskania najlepszego efektu, dźwięk podczas gry powinien być odtwarzany na słuchawkach.

4.1 Opis gry

Wake Up!, czyli grę będącą przedmiotem projektu, można zaliczyć do gatunku „skradanek” [11]. W przeciwieństwie do pokrewnych gier akcji, zadaniem gracza nie jest pokonanie przeciwników, ale raczej ich omijanie i unikanie. Użytkownik ogląda świat wirtualny z perspektywy pierwszej osoby, wcielając się w postać uwięzioną w swoim własnym śnie. Wybór tak nierealistycznej przestrzeni jest celowy i pozwala na duże zróżnicowanie stylistyczne poszczególnych poziomów, bez konieczności logicznego ich połączenia. Dzięki temu możliwe będzie też użycie wielu oryginalnych efektów dźwiękowych. Mimo że planowany klimat gry nie mógł być osiągnięty w tak wczesnej wersji programu, wiele koncepcji i rozwiązań zostało zaprojektowanych mając na uwadze dalszy rozwój pomysłu.

Zadaniem gracza jest odnalezienie umieszczonego na planszy wyjścia, omijając jednocześnie wszelkiego rodzaju pułapki oraz unikając przeciwników. Nie ma on możliwości z nimi walczyć, ale posiada za to inną umiejętność – potrafi stać się niewidzialny. Znacznym utrudnieniem jest to, że w tym samym czasie sam widzi zniekształcony i o wiele mniej wyraźny obraz, a nieuniknione kolizje z różnymi obiektami odkrywają jego położenie. Najlepszym rozwiązaniem jest wtedy uważne wsłuchiwanie się w dźwięki otoczenia, które mogą bardzo ułatwić wyznaczenie bezpiecznej trasy do celu, pomimo mocno zaburzonej wizji.

Zaprojektowany poziom *Wake Up!* składa się z długiego tunelu, prowadzącego do przestronnej komnaty. Pierwsza część sceny pozwala graczowi na oswojenie się ze sterowaniem, a dopiero przejście do drugiego pomieszczenia utrudnione jest przez pułapkę – spadający kamień. W komnacie rozmieszczeni są przeciwnicy, patrolujący określone ścieżki w taki sposób, że odnalezienie wyjścia bez używania trybu niewidzialnego jest praktycznie niemożliwe. Gdy gracz zostanie złapany, wraca do punktu startowego. Poziom kończy się w momencie przejścia przez portal.

4.2 Grafika

Początkowa koncepcja przewidywała brak albo znikomą ilość grafiki. W związku ze wprowadzonymi zmianami, konieczne było jednak przygotowanie przynajmniej modelu planszy. Został on stworzony przez grafika 3d zgodnie ze wskazówkami autora

i udostępniony w formacie FBX, który pozwala silnikowi *Unity* na bezpośredni import obiektu. Ponieważ w grze wykorzystywana jest kamera pierwszoosobowa, nie było potrzeby modelowania głównego bohatera, a za prototyp przeciwników posłużyły zwykłe prostopadłościany. Przyjęte rozwiązania nie są oczywiście ostateczne, ale są w pełni wystarczające do zaprezentowania idei i schematu gry.

4.3 Dźwięk

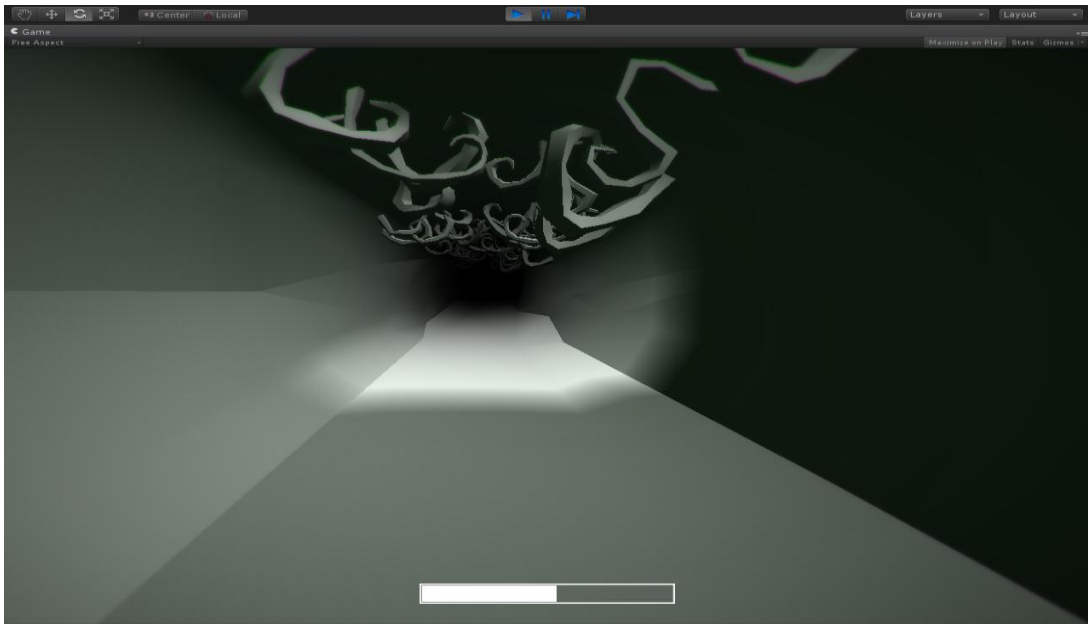
Mając w planach zastosowanie silnika audio RAYAV, wybrane do projektu nagrania nie wymagały wiele pracy. Poddane zostały między innymi edycji w celu usunięcia niepotrzebnych fragmentów, w niektórych z nich zmieniono wysokość dźwięku lub stłumiono pewne częstotliwości. Przygotowane pliki zawierają efekty dźwiękowe odpowiednie dla poruszającej się postaci (kroki, uderzenie w obiekt), przeciwników (przemieszczanie się, oddech) oraz otoczenia (toczący się kamień, kolizja kamienia z drzwiami, portal, ambient). Należało wziąć pod uwagę, że niektóre z tych odgłosów będą odtwarzane jednokrotnie, a inne powtarzać się będą przez całą rozgrywkę. Dla tych właśnie dźwięków wskazane było przygotowanie kilku alternatywnych wersji tego samego efektu. Dzięki temu można stosować je zamiennie i uniknąć sytuacji, w której gracz staje się znudzony powtarzalnością. Ważny okazał się też wybór tła muzycznego dla ekranu startowego. Daje ono użytkownikowi informacje o tym, czego powinien się spodziewać w dalszej części rozgrywki, wstępnie buduje klimat, wprowadza odpowiednie emocje, a przez to diametralnie zmienia odbiór gry.

4.4 Mechanika

Mechanika gry obejmuje stworzony system, według którego toczy się rozgrywka. Zawiera w sobie wszelkie obowiązujące gracza zasady, jego możliwości, umiejętności oraz sposoby interakcji ze światem wirtualnym. Opisuje także w jaki sposób odbiorca jest angażowany przez grę, czyli jak dokonuje w niej postępów oraz jak jest za nie nagradzany. Dobrze przemyślana i zbalansowana mechanika jest podstawowym elementem każdej ciekawej i atrakcyjnej gry.

4.4.1 Kontrola postaci

Ruch gracza kontrolowany jest przez proponowany przez *Unity* komponent o nazwie *FPS Input Controller*. Jest on zaprojektowany specjalnie do gier w perspektywie pierwszej osoby i realizuje standardowe, dobrze znane graczom sterowanie: ruch postaci wraz z dołączoną do niej kamerą do przodu, do tyłu i na boki obsługują odpowiednio klawisze *W, S, A, D,* natomiast skok możliwy jest po naciśnięciu klawisza *spacja*. Używany skrypt pozwala na kontrolowanie prędkości poruszania się niezależnie dla każdego kierunku. Dzięki temu postać nieznacznie wolniej przemieszcza się do tyłu i na boki. Do kontrolera napisano dodatkowo własny skrypt, pozwalający graczowi na chwilowe przyspieszenie, rodzaj biegu, który uaktywnia się prawym przyciskiem myszy. Aby uniknąć sytuacji, w której użytkownik stale korzysta z tej funkcji, wprowadzono ograniczenie w postaci automatycznie regenerującego się paska energii, widocznego na dole ekranu (rys. 4.1).



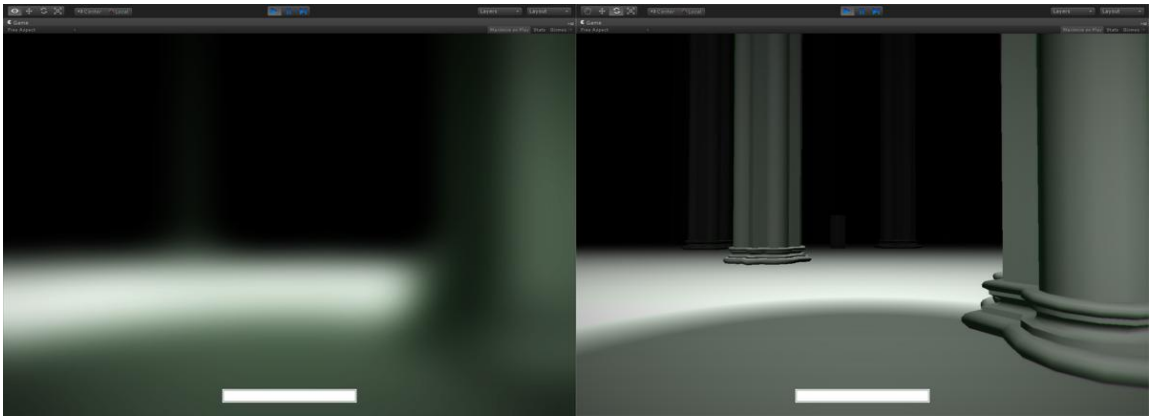
Rys. 4.1 Zrzut ekranu z programu Unity 3D, przedstawiający pasek energii

4.4.2 Charakterystyka kamery

Dla gier w których gracz ogląda świat oczami bohatera, kluczowym elementem wygodnego sterowania postacią jest dobrze opracowany algorytm poruszania kamerą. Jej rotacja w osi poziomej i pionowej zależy bezpośrednio od ruchów myszą

w odpowiednich kierunkach. W poziomie możliwe jest wykonanie całego obrotu, natomiast w pionie nałożono blokadę, która nie pozwala na rotację kamery wokół tej osi o kąt większy niż 60 stopni. Podstawowa funkcja ruchu kamery została nieco poprawiona poprzez dodanie lekkiego opóźnienia względem ruchów myszy. Dzięki temu zabiegowi obrót postaci stał się znacznie bardziej płynny i naturalny.

Wersja *Unity Pro* pozwala dodatkowo na korzystanie z efektów post-processingowych [9]. Polegają one na przetwarzaniu wcześniej już wyrenderowanego dwuwymiarowego obrazu. Zaliczają się do nich między innymi wszelkiego rodzaju operacje związane ze zmianą koloru, jasności, kontrastu, nasycenia, nakładanie warstw na pełny ekran, rozmycie itd. W kontekście gry miało to duże znaczenie przy projektowaniu obrazu wyświetlanego dla postaci w trybie niewidzialnym. Zastosowano między innymi stopniowe rozmycie obrazu, wzmocnienie kontrastu i lekkie przyciemnienie kolorów. Wynik nałożonych efektów pokazany jest na rysunku 4.2.



Rys. 4.2 Porównanie obrazu po i przed zastosowaniem efektów post-processingowych

4.4.3 Opis trybu (nie)widzialnego postaci

Charakterystyczna dla gry *Wake Up!* jest koncepcja przejścia postaci w stan niewidzialny. Gracz może korzystać z tej umiejętności naciskając klawisz *E*. Staje się wówczas niewidoczny dla przeciwników, aż do momentu celowego lub przypadkowego ujawnienia się. Postać zdradza swoją obecność wchodząc w kolizję z dowolnym obiektem na scenie (np. ścianą, kolumną, drzwiami). Powrót do trybu widzialnego spowodowany może być także znalezieniem się w zbyt bliskiej odległości od przeciwnika. Ten wówczas będzie w stanie usłyszeć obecność bohatera i natychmiast zacznie poruszać się w jego kierunku. Pozostając w stanie kolizji

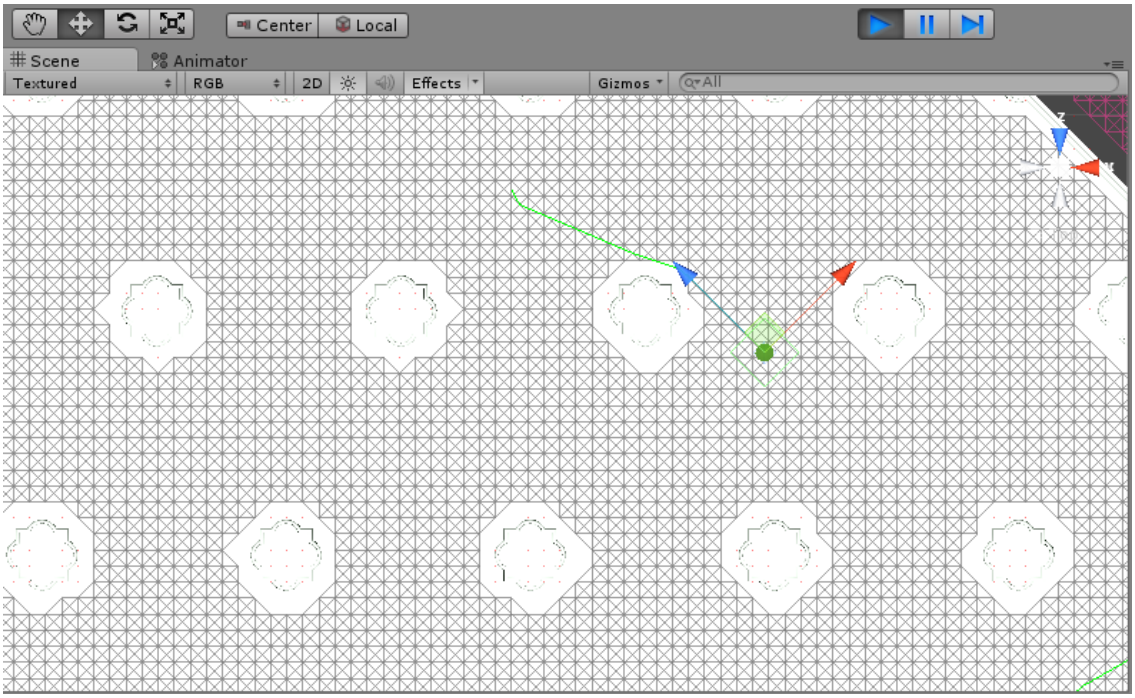
(np. opierając się o ścianę) lub w zasięgu któregokolwiek z przeciwników, gracz nie ma możliwości stać się znów niewidzialnym. Rozwiązaniem może być odsunięcie się od przeszkody lub próba ucieczki poza pole widzenia wroga. Poziomy gry powinny być zaprojektowane w taki sposób, aby wymagały od gracza sprawnego i przemyślanego korzystania z opcji niewidzialności.

4.4.4 Przeciwnicy (klasa Enemy)

4.4.4.1 Inteligentne wyznaczanie ścieżek

Realizm każdej gry komputerowej zależy w znacznym stopniu od zaimplementowanych algorytmów sztucznej inteligencji. Muszą być one dobrze przemyślane i zbalansowane w taki sposób, aby postacie z nich korzystające wykazywały logiczne i naturalne zachowania. W *Wake Up!* zaprojektowano niezbędne reakcje i zależności między przeciwnikami a graczem i otoczeniem.

Podstawowym zadaniem przeciwników jest poruszanie się po scenie. Należało przy tym pamiętać, że wszelkie interakcje z graczem będą miały wpływ na nieprzewidywalną zmianę trasy ich ruchu. Realizacja wymagała więc zastosowania algorytmu, pozwalającego na inteligentne wyszukiwanie ścieżek w czasie rzeczywistym. Wykorzystane w projekcie rozwiązanie opiera się o popularny dodatek do *Unity* o nazwie *A* Pathfinding Project* [12]. Dzięki niemu wygenerowana została określonego rozmiaru siatka pokrywająca obszar po którym mogą poruszać się przeciwnicy (rys. 4.3). Uwzględnia ona także znajdujące się na scenie obiekty, z którymi mogą zachodzić kolizje. Na jej podstawie w czasie rzeczywistym szukana jest najkrótsza trasa pomiędzy przeciwnikiem a określonym punktem końcowym.



Rys. 4.3 Zrzut ekranu przedstawiający wygenerowaną przez algorytm A* Pathfinding siatkę (widok z góry) i obraną na jej podstawie ścieżką (zaznaczona kolorem zielonym)

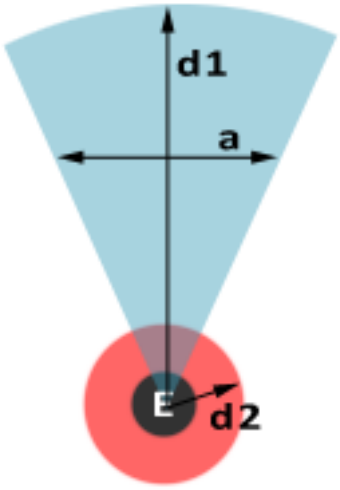
4.4.4.2 Patrolowanie trasy

Dla każdego przeciwnika zdefiniowano punkty wyznaczające patrolowaną przez niego trasę. Podczas ruchu, o ile nie zachodzi żadna interakcja z graczem, algorytm wyznacza ścieżkę do pierwszego z określonych punktów. Następnie funkcja sprawdza, czy obiekt znajduje się w miejscu docelowym. Jeśli tak, to przeliczana jest nowa ścieżka, kierująca przeciwnika do kolejnego punktu (funkcja *GetNewPoint*). Po dotarciu do końca określonej trasy, wraca on do miejsca początkowego i zaczyna cały proces od nowa. Każdy obiekt klasy *Enemy* przechowuje współrzędne punktu trasy (*path*), do którego aktualnie dąży. Dzięki temu, jeżeli w którymś momencie algorytm patrolowania zostanie przerwany przez gracza, przeciwnik później będzie mógł kontynuować poruszanie się po swojej ścieżce.

4.4.4.3 Zasięg przeciwnika

Przeciwnik reaguje na gracza dopiero gdy ten znajdzie się w jego zasięgu (funkcja logiczna *PlayerSpotted*). Sprawdzane jest to za pomocą promieni kierowanych w stronę postaci z punktu, w którym znajduje się przeciwnik. Jeżeli długość takiego

promienia jest mniejsza niż określona wartość, oznacza to, że gracz znalazł się zbyt blisko wroga i ten jest w stanie go usłyszeć. W tym przypadku nie ma znaczenia, czy postać korzysta aktualnie z niewidzialności (jeżeli tak – zostanie odkryta). Druga możliwość symuluje zmysł wzroku przeciwnika. Wtedy długość promienia porównywana jest z większą wartością, jednak wprowadzone jest ograniczenie w postaci określonego kąta widzenia. Dodatkowo postać musi znajdować się w trybie widzialnym. Zasięg przeciwników zilustrowany jest na rys. 4.4.



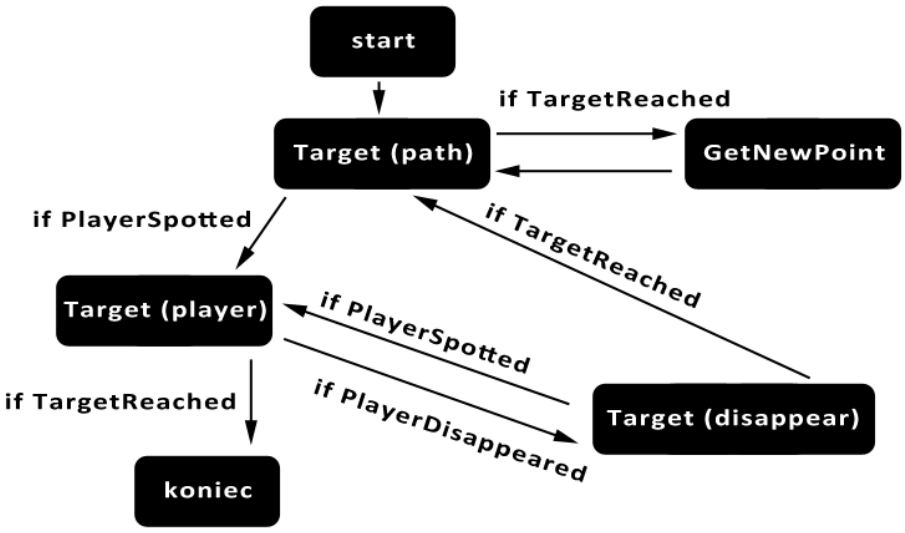
Rys. 4.4 Zasięg przeciwnika: E – przeciwnik, obszar czerwony – zasięg słuchu, obszar niebieski – zasięg wzroku, a – kąt widzenia, d – odpowiednie długości promieni

Funkcja *PlayerSpotted* pozwala na zdefiniowanie jeszcze jednego zdarzenia – sytuacji, w której postać znika z zasięgu przeciwnika (*PlayerDisappeared*, typ logiczny). W najmniejszych możliwych odstępach czasu sprawdza aktualną i wcześniejszą wartość funkcji *PlayerSpotted*. Jeżeli otrzymuje odpowiednio wyniki 0 oraz 1, oznacza to, że graczowi udało się zgubić przeciwnika.

4.4.4.4 Diagram stanów

System zmiany stanu przeciwnika korzysta z kilku funkcji, z których część została już opisana. Oprócz tego istotna dla działania algorytmu jest też funkcja *Target*, która ustala punkt docelowy, do którego w danym momencie dąży przeciwnik. Może to być jeden z punktów wyznaczających jego standardową ścieżkę (*path*), położenie, w którym aktualnie znajduje się gracz (*player*) lub współrzędne miejsca, w którym

postać zniknęła z pola widzenia przeciwnika (*disappear*). Osiągnięcie celu monitorowane jest przez funkcję logiczną *TargetReached*. Warto też wspomnieć, że od aktualnego stanu zależy prędkość, z jaką porusza się przeciwnik – zwiększa się, kiedy ten podąża za graczem i maleje, gdy bohater jest poza jego zasięgiem. Diagram zmiany stanów przeciwnika przedstawiony jest na rys. 4.5.



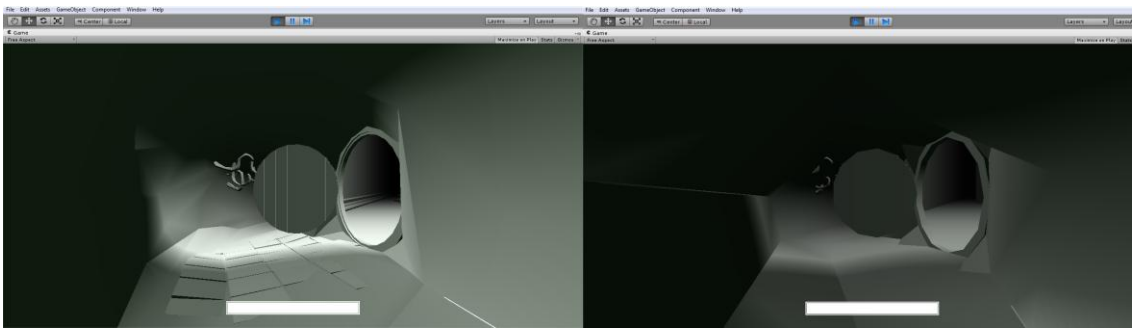
Rys. 4.5 Schemat przedstawiający algorytm zmiany stanów przeciwnika

4.4.5 System kolizji

Wszystkie występujące na scenie kolizje obsługiwane są przez silnik *Unity*. Do obiektów, które mają im ulegać, dołączono niezbędne komponenty, odpowiedzialne za nadanie im właściwości fizycznych. Dzięki temu różnego rodzaju obiekty nie przenikają się, a oddziałują na siebie zgodnie z prawami fizyki. Ponadto możliwy jest dostęp do różnego rodzaju informacji o kolizji, która nastąpiła, np. współrzędnych zderzenia albo nazw obiektów biorących w niej udział. Jest to użyteczne, ponieważ niektóre kolizje nie powinny być brane pod uwagę (np. ciągły kontakt gracza z podłożem). Funkcja pomija więc rejestrację tych zderzeń, każdorazowo weryfikując nazwę obiektu. Współrzędne punktów, w których gracz obija się o stałe elementy planszy (ściany, kolumny) wykorzystywane są do określenia miejsc, w których powinien zostać odtworzony efekt dźwiękowy.

4.4.6 Rozmieszczenie dźwięków

Unity wśród wszystkich swoich narzędzi zapewnia także podstawowy silnik audio. W projekcie wykorzystywany jest on do odtwarzania tła muzycznego, jednak wszystkie efekty przestrzenne obsługiwane są przez RAYAV. Odbiornik jest oczywiście połączony z postacią, dzięki czemu wszelkie dźwięki otoczenia pozwalają zorientować się, w którym kierunku gracz powinien się poruszać, a których miejsc unikać. Dla rozgrywki ma to szczególne znaczenie podczas korzystania z trybu niewidzialnego. Aby odtworzyć dźwięk przy użyciu silnika audio RAYAV, najpierw należy przy użyciu odpowiednich funkcji zdefiniować współrzędne źródła oraz przypisać dany plik dźwiękowy do próbki. Obsługiwane są zarówno statyczne, jak i ruchome źródła dźwięku. Wszelkie niezbędne obliczenia wykonywane są już poza *Unity*. RAYAV śledzi bieg promienia na podstawie uproszczonego modelu planszy (rys. 4.6), który posiada około 5 tysięcy trójkątów mniej niż oryginalna wersja. Biorąc pod uwagę brzmienie uzyskanego dźwięku, różnica powinna być znikoma, ponieważ ogólna geometria pomieszczeń pozostaje bez zmian. Istotne jest to jednak w kontekście wydajności, gdyż taki zabieg ma zauważalnie skrócić złożoność oraz czas potrzebny do wykonania obliczeń.



Rys. 4.6 Porównanie oryginalnego i uproszczonego modelu planszy

Realizacja tła muzycznego jest już natomiast zadaniem *Unity*. Dźwięk ten ma formę dwuwymiarową, co oznacza że odtwarzana jest oryginalna próbka, a położenie bohatera w żaden sposób nie wpływa na sygnał.

5. Zakończenie

Założeniem projektu było stworzenie gry komputerowej, która przede wszystkim demonstruje szerokie możliwości, oferowane przez silnik audio RAYAV. Wybrana koncepcja wydaje się trafna, ponieważ skupia uwagę gracza na najistotniejszych elementach. Ostatecznie uzyskano krótki, ale w pełni grywalny fragment. Gra tworzona była metodą iteracyjną, czyli od najprostszego prototypu, w każdym kolejnym etapie dodając nowe funkcje i elementy. Po każdym kroku starano się jak najdokładniej testować wprowadzone zmiany. Ze względu na oryginalność pomysłu, w trakcie realizacji niezbędne było jednak częste weryfikowanie przyjętych założeń, co nieraz skutkowało odrzucaniem efektów wielogodzinnej pracy. Dodatkowo, przez większość czasu gra tworzona i testowana była bez dźwięku, ponieważ zintegrowanie silnika audio RAYAV z *Unity* było zadaniem trudniejszym niż się spodziewano.

5.1 Podsumowanie

Realizacja *Wake Up!* wiązała się z koniecznością wykonania przez autora wielu różnych zadań, między innymi zaprojektowania poziomu i rozgrywki, przygotowania efektów dźwiękowych oraz programowania. Wymagało to wykorzystania znacznej części wiedzy i umiejętności pozyskanych w toku studiów. Mając znikome wcześniejsze doświadczenie w tworzeniu gier komputerowych, konieczne okazało się również dokształcenie w tej dziedzinie. Praca nad projektem trwała kilka miesięcy i przekładała się na wiele godzin spędzonych przed komputerem. Własnoręcznie napisane skrypty składają się ostatecznie z ponad 10 tysięcy znaków, pomijając wykorzystywane gotowe rozwiązania oraz elementy dodawane i edytowane poprzez narzędzia dostępne w *Unity*. Jak na jedną z pierwszych gier komputerowych, projekt okazał się być dużym wyzwaniem i prawdopodobnie, gdyby nie odpowiednia motywacja, nigdy nie zostałby ukończony. Mimo wszystko taki rodzaj pracy okazał się dawać zaskakującą satysfakcję, a zdobyte doświadczenie z pewnością przełoży się na wybór dalszej kariery.

5.2 Przyszłość projektu

Wake Up! składa się obecnie z jednego dosyć krótkiego poziomu. Dzięki temu sprawdzono jednak, że jest to całkiem atrakcyjna forma gry, a planowana kontynuacja projektu może dać ciekawe rezultaty. Jedną z propozycji, która znacznie poprawi pierwsze wrażenie wywierane przez grę, jest wprowadzenie tekstur pasujących do jej klimatu i przygotowanie odpowiedniego oświetlenia sceny. Poza tym, prostopadłościany służące za makiety przeciwników powinny zostać zastąpione dopracowanymi, animowanymi modelami. Oczywiście niezbędne jest też zaprojektowanie większej ilości poziomów, pułapek i przeciwników, a także dodanie w pełni funkcjonalnego menu gry. Dobrym pomysłem mogłoby okazać się też wprowadzenie dodatkowego systemu motywacyjnego, aby uniknąć znudzenia się powtarzalnością przy dłuższej rozgrywce, chociaż ta idea wymaga przetestowania przy większej ilości dostępnych poziomów.

Bibliografia

[1] Strona domowa Zespołu Przetwarzania Sygnałów AGH: *http://www.dsp.agh.edu.pl/* (22.01.2014)

[2] Marks A.: *The Complete Guide to Game Audio For Composers, Musicians, Sound Designers, and Game Developers*, Oxford, Focal Press 2009

[3] Stevens A., Raybould D.: *The Game Audio Tutorial: A Practical Guide to Sound and Music for Interactive Games*, Oxford, Focal Press 2011

[4] Collins K.: *Game Sound. An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design*, Cambridge, The MIT Press, 2008

[5] Pharr M., Humphreys G.: *Physically Based Rendering*, San Francisco, Morgan-Kaufmann 2004

[6] Strona domowa firmy NVIDIA: *http://www.nvidia.com/object/optix.html* (16.01.2014)

[7] Artykuł *What is a Game Engine?*: *http://www.gamecareerguide.com/features/529/ /features/529/what_is_a_game_.php* (17.01.2014)

[8] Strona domowa silnika Unreal Engine*: http://www.unrealengine.com/* (17.01.2014)

[9] Strona domowa silnika Unity 3D: *http://unity3d.com/unity/licenses* (17.01.2014)

[10] Dokumentacja Unity 3D: *http://docs.unity3d.com/Documentation/Manual/Plugins* (18.01.2014)

[11] Rogers S.: *Level Up! The Guide to Great Video Game Design*, Chichester, John Wiley & Sons 2010

[12] Strona domowa projektu A* Pathfinding: *http://arongranberg.com/astar/* (20.01.2014)