UNIWERSYTET GDAŃSKI
Wydział Matematyki, Fizyki i Informatyki

Mateusz Kleina, Adrian Podlawski

nr albumu: 231 068, 236 232

Sieciowa fabularna gra komputerowa – The Trinity

Praca magisterska na kierunku:
INFORMATYKA
Promotor:
dr W. Bzyl

Gdańsk 2017

Streszczenie

Zaimplementowaliśmy wieloosobową grę w wersji Beta, która bazuje na silniku graficznym Unity.

Tryb multiplayer zbudowaliśmy na podstawie frameworka Photon Unity 3D Networking, który został pobrany ze sklepu Asset Store, a następnie zmodyfikowany i zaprogramowany zgodnie z wymaganiami gry.

Scenę po której poruszają się postaci utworzyliśmy dzięki silnikowi gry Unity. Krajobrazy oraz budynki zostały pobrane z Assets Store, a następnie zmodyfikowane. Wszystkie skrypty odpowiadające za sterowanie postaci, łączenie z serwerem, czy walkę z przeciwnikiem, zostały utworzone przez nas w języku programowania C Sharp. Postaci oraz ich animacje zostały utworzone przy pomocy storny http://www.mixamo.com, a następnie odpowiednio zaimplementowane przy pomocny skryptów.

W trakcie rozwijania aplikacji, pliki były przechowywane we wspólnym repozytorium na stronie http://www.github.com. Zbudowana wersja gry, którą można pobrać oraz uruchomić znajduje się na repozytorium Git (https://www.github.com/kirin1994/ug-projekt-gra-rpg). Projekt był testowany manualnie poprzez sterowanie postaciami oraz przy użyciu logów przesyłanych do silnika Unity. Za środowisko testowe posłużył nam osobny serwer, który nie miał wpływu na wydaną wersję projektu, znajdującą się na repozytorium.

Słowa kluczowe

Gra, RPG, Multiplayer, Unity, PhotonUnityNetwork

Spis treści

Wprowadzenie

Podczas minionych semestrów na uczelni poznaliśmy wiele nowych technologii, Framework-ów oraz języków programowania. Mieliśmy do czynienia z wieloma ich odmianami, poznaliśmy ich różne zastosowania. Niektóre z nich służyły nam do pisania prostych programów tekstowych, które korzystały tylko i wyłącznie z wiersza poleceń. Z drugiej strony były też obszerne aplikacje, połączone z bazą danych oraz utworzone według wzorców projektowych przy użyciu języków programowania takich jak Ruby, czy Java.

Chcąc rozwijać swoją wiedzę oraz poszerzać swoje umiejętności, jako programiści postanowiliśmy napisać nasz projekt w języku programowania, z którym nie mieliśmy styczności na uczelni. Uważaliśmy, że będzie to dla nas dobra okazja do sprawdzenia tego, czego do tej pory się nauczyliśmy, gdyż nauka programowania, to nie tylko nauka składni danego języka. W wielu przypadkach to głównie nauka logicznego myślenia, wiązania ze sobą faktów, szukania oraz korygowania własnych błędów i wyciągania z nich wniosków.

Szukaliśmy języka programowania, który jest wszechstronny i może być wykorzystany w różnych projektach min. przy budowaniu dużych stron internetowych, gier, serwisów. Po namyśle zdecydowaliśmy się na język C Sharp. Jest on zorientowany obiektowo, dzięki temu mogliśmy wykorzystać nasze doświadczenie min. z języka Java. Ma on bardzo szerokie zastosowanie, może być wykorzystany do tworzenia aplikacji desktopowych, Windows Service, dzięki niemu możemy również tworzyć rozbudowane strony internetowe przy użyciu technologii ASP.NET opartej na .NET Framework. Wybierając jednak nowy język programowania, chcieliśmy również skorzystać z technologii, która umożliwi nam przetestowanie naszych umiejętności zupełnie na innej płaszczyźnie.

Aktualnie na świecie miliony osób grają w gry przeróżnego typu, a my należymy do tej bardzo licznej społeczności. Do tej pory jednak nie mieliśmy przed sobą zadania, by stworzyć jedną z nich od podstaw. Chcieliśmy podjąć się

tego wyzwania i za silnik graficzny obraliśmy Unity, który współgra z językiem C Sharp, a do tego obsługuje aż 22 platformy sprzętowe i co raz częściej wykorzystywany jest również w przeglądarkach internetowych oraz wirtualnej rzeczywistości.Wiedzieliśmy, że będzie do dla nas wyzwanie, ale zarazem wielki krok w początkach naszej kariery, jako młodych programistów.

Unity daje możliwość tworzenia gry bez wdrażania się w szczegóły techniczne dotyczące renderowania obrazu i obsługi różnych kart graficznych, obliczania skomplikowanych równań fizycznych czy korzystania z protokołów sieciowych. Tworzenie gier nie wymaga już niskopoziomowego kodu, stało się przyjaźniejsze i bardziej dostępne dla programistów chcących skupić się na samej mechanice gry.

Skłoniło nas to do stworzenia nieco odmiennego projektu, w którym rozgrywka polega na wspólnym pokonywaniu przeszkód i odkrywaniu dalszego ciągu tej samej historii, a gracze zamknięci zostają we wspólnej ramie czasowej. Dotąd gry tego typu występowały głównie w postaci gier jednoosobowych, takich jak Far Cry, Tomb Raider, czy Grand Theft Auto.

W odróżnieniu od innych gier wieloosobowych takich jak np. Diablo 3, gdzie całą fabułę możemy ukończyć sami, w The Last Trinity położyliśmy główny nacisk na współpracę. Do ukończenia rozgrywki niezbędne są wszystkie postaci, które sterowane są przez różnych graczy.

Aktualnie gry Multiplayer opierają się głównie na mechanice zbierania co raz lepszego ekwipunku, jak np. w World of Warcraft. Często doprowadza to również do wielu konfliktów pomiędzy graczami. W naszej stawiamy na wspólną zabawę, nie będzie w niej ciągłego zbierania ekwipunku, a jedynie pokonywanie rozmaitych i bardziej skomplikowanych przeszkód razem ze swoją drużyną.

Początkowym założeniem naszej pracy było utworzenie gry wieloosobowej, której świat jednocześnie będzie mogła eksplorować trójka graczy. Każdy z nich sterowałby inną, unikatową postacią. Gra miała posiadać przykładowe misje oraz stanowić fundament do dalszego jej rozwoju na tle fabularnym. Wszystkie powyższe założenia udało nam się zrealizować.

ROZDZIAŁ 1

Projektowanie oraz przygotowanie środowiska

1.1. Budowa terenu

Budowa podstawowej mapy gry jest stosunkowo nieskomplikowanym procesem, dzięki czemu mogliśmy się tutaj skupić głównie na projektowaniu dróg, rozłożeniu obiektów i kształtowaniu mapy na potrzeby rozgrywki.

1.1.1. Przygotowanie terenu

Do przygotowania terenu został użyty specjalnie do tego przeznaczony typ obiektu 3D o nazwie **Terrain**. Po utworzeniu takiego obiektu w hierarchii obiektów na ekranie ukaże się nieoteksturowana płaszczyzna.

Rysunek 1.1: Płaszczyzna terenu

1.1.2. Formowanie kształtu

Na tak przygotowanej płaszczyźnie uformowane zostały nierówności przy użyciu palety narzędzi terenu. Używając opcji **Raise / Lower Terrain** utworzone zostały wypiętrzenia nadające kształt mapie gry. Charakter wypiętrzeń dostosowany został używając odpowiedniego pędzla z panelu **Brushes**, natomiast promień zniekształceń oraz siła efektu za pomocą parametrów kolejno **Brush Size** oraz **Opacity**.

Rysunek 1.2: Modelowanie nierówności terenu

1.1.3. Nakładanie tekstur

Kolejnym krokiem było utworzenie tekstury służącej do nadaniu naszemu terenowi koloru oraz faktury. Do tego celu pobrane zostały odpowiednie tekstury z wbudowanego sklepu assetów **Asset Store**. Assety są rodzajem pakietów zawierających różnorakie obiekty, skrypty oraz tekstury, dostępne do pobrania z serwerów Unity.

Po pobraniu odpowiedniej tekstury trawy, zostaje ona zaimportowana do folderu **Assets** znajdującego się w głównym katalogu projektu.

Po wybraniu narzędzia **Paint Texture** ukazuje się panel **Textures** pozwalający na skonfigurowanie używanej przez narzędzie tekstury. Znajduje się tam przycisk **Edit Textures...** po kliknięciu którego otwiera się okno konfiguracyjne tekstury pozwalające na wybór tekstury podstawowej oraz tekstury przechowującej dane o chropowatościach. Po skonfigurowaniu tekstur i nałożeniu ich na teren, całość prezentuje się następująco.

Rysunek 1.3: Nakładanie tekstur na mapę terenu

1.1.4. Rozmieszczanie obiektów

Kolejnym krokiem w tworzeniu świata gry było umieszczenie na mapie sprefabrykowanych wcześniej obiektów (tzw. *Prefabs*, o tym później). Wszystkie obiekty użyte w pracy są dostępne za darmo w bibliotece obiektów Unity (**Asset Store**). Po pobraniu, obiekty znajdują się w podfolderze **Prefabs** zainstalowanej paczki. Obiekty umieszcza się na mapie metodą przeciągnij-upuść, dostosowując ich koordynaty używając narzędzi transformacji dostępnych w pasku narzędzi znajdującym się w górnej części okna edytora.

Rysunek 1.4: Umieszczanie obiektów na mapie

1.1.5. Oddziaływanie terenu na inne obiekty

Kluczowym elementem tworzenia mapy świata gry są takie elementy jak oddziaływanie na postacie ograniczeń nachylenia terenu typu wzgórza, drzewa, czy budynki, uniemożliwiających dostanie się w niektóre miejsca. Unity w celu uproszczenia obliczeń posiada możliwość wygenerowania uproszczonej mapy dróg (tzw. *NavMesh*) na bazie modelu terenu, pozwalającej na dynamiczne omijanie przeszkód przez wroga (o tym później w sekcji **??** na stronie **??**).

Aby utworzyć taką mapę, należy przejść do zakładki **Navigation**, gdzie w panelu **Bake** znajduje się lista parametrów dotyczących maksymalnego kąta nachylenia terenu, czy maksymalnej wysokości uskoku, którą obiekty sterowane przez komputer mogą pokonać.

Rysunek 1.5: Parametry mapy dróg

Po ustaleniu parametrów i kliknięciu przycisku **Bake**, mapa zostaje wygenerowana, natomiast w oknie widoku sceny, obszary dostępne do przemierzania oznaczone zostają niebieskim kolorem. Operację można powtarzać do uzyskania optymalnych efektów.

Rysunek 1.6: Poprawnie wygenerowany *NavMesh*

Tak przygotowana mapa posłużyła nam do projektowania dalszej części gry.

ROZDZIAŁ 2

Narzędzia i standardy pokrewne

Systemy SGML, ze względu na mnogość funkcji jakie spełniają i ich kompleksowe podejście do oznakowywania i przetwarzania dokumentów tekstowych, są bardzo skomplikowane. Możemy wyróżnić dwa podejścia do budowy takich systemów. Z jednej strony, buduje się systemy zindywidualizowane, oparte o specyficzne narzędzia tworzone w takich językach, jak: C, C++, Perl czy Python. Edytory strukturalne, filtry do transformacji formatów czy parsery i biblioteki przydatne do konstrukcji dalszych narzędzi, tworzone są według potrzeb określonych, pojedynczych systemów.

Z drugiej strony, twórcy oprogramowania postanowili pójść krok dalej i połączyć te różne narzędzia w jedną całość. Tą całość miał stanowić DSSSL lub jego XML-owy odpowiednik – standard XSL. Ze względu na oferowane możliwości można twierdzić, że tworzenie i używanie narzędzi implementujących standard DSSSL/XSL, jest najwłaściwszym podejściem. Przemawiają za tym różne argumenty, ale najważniejszym z nich jest to, że mamy tu możliwość stworzenia niezależnego od platformy programowej i narzędziowej zbioru szablonów – przepisów jak przetwarzać dokumenty SGML.

2.1. Przetwarzanie dokumentów SGML – standard DSSSL

DSSSL (*Document Style Semantics and Specification Language*) – to międzynarodowy standard ściśle związany ze standardem SGML. Standard ten, można podzielić na następujące części:

- język transformacji (*transformation language*). To definicja języka słu-

żącego do transformacji dokumentu oznaczonego znacznikami zgodnie z pewnym DTD na dokument oznaczony zgodnie z innym DTD.

- język stylu (*style language*) opisujący sposób formatowania dokumentów SGML.
- język zapytań (*query language*) służy do identyfikowania poszczególnych fragmentów dokumentu SGML.

Opisane główne części składowe standardu DSSSL dają obraz tego, jak wiele aspektów przetwarzania zostało zdefiniowanych i jak skomplikowany jest to problem. Jest to głównym powodem tego, że mimo upływu kilku lat od zdefiniowania standardu nie powstały ani komercyjne ani wolnodostępne aplikacje wspierające go w całości. Istnieją natomiast nieliczne narzędzia realizujące DSSSL w ograniczonym zakresie, głównie w części definiującej język stylu, który odpowiada za opatrzenie dokumentu czysto strukturalnego w informacje formatujące. Daje to możliwość publikacji dokumentów SGML zarówno w postaci elektronicznej, hipertekstowej czy też drukowanej.

2.2. Przetwarzanie dokumentów XML – standard XSL

Tak jak XML jest uproszczoną wersją standardu SGML, tak XSL jest uproszczonym odpowiednikiem standardu DSSSL. W szczególności, wyróżnić można w tym standardzie następujące części składowe:

- język transformacji (XSLT) To definicja języka służącego do transformacji dokumentu.
- język zapytań (XPath) służy do identyfikowania poszczególnych fragmentów dokumentu.
- język stylu deefiniujący sposób formatowania dokumentów XML.

ROZDZIAŁ 3

Przegląd dostępnych narzędzi

W celu wykorzystania standardu SGML do przetwarzania dokumentów, niezbędne jest zebranie odpowiedniego zestawu narzędzi. Narzędzi do przetwarzania dokumentów SGML jest wiele. Są to zarówno całe systemy zintegrowane, jak i poszczególne programy, biblioteki czy skrypty wspomagające.

3.1. Narzędzia do przeglądania dokumentów SGML

Do tej kategorii oprogramowania zaliczamy przeglądarki dokumentów SGML oraz serwery sieciowe wspomagające standard SGML, przy czym rozwiązań wspierających standard XML jest już w chwili obecnej dużo więcej i są dużo powszechniejsze.

Jeżeli chodzi o przeglądarki to zarówno Internet Explorer jak i Netscape umożliwiają bezpośrednie wyświetlenie dokumentów XML; ponieważ jednak nie wspierają w całości standardu XML, prowadzi to ciągle do wielu problemów[1].

[1] Z innych mniej popularnych rozwiązań można wymienić takie aplikacje, jak: HyBrick SGML Browser firmy Fujitsu Limited, Panorama Publisher firmy InterLeaf Inc, DynaText firmy Inso Corporation czy darmowy QWeb. W przypadku serwerów zwykle dokonują one transformacji „w locie" żądanych dokumentów na format HTML (rzadziej bezpośrednio wyświetlają dokumenty XML). Ta kategoria oprogramowania ma, z punktu widzenia projektu, znaczenie drugorzędne.

3.2. Parsery SGML

Program `nsgmls` (z pakietu `SP` Jamesa Clarka) jest doskonałym parserem dokumentów SGML, dostępnym publicznie. Parser `nsgmls` jest dostępny w postaci źródłowej oraz w postaci programów wykonywalnych przygotowanych na platformę MS Windows, Linux/Unix i inne. Oprócz analizy poprawności dokumentu parser ten umożliwia również konwersję danych do formatu ESIS, który wykorzystywany jest jako dane wejściowe przez wiele narzędzi do przetwarzania i formatowania dokumentów SGML. Dodatkowymi, bardzo przydatnymi elementami pakietu `SP` są: program `sgmlnorm` do normalizacji, program `sx` służący do konwersji dokumentu SGML na XML oraz biblioteki programistyczne, przydatne przy tworzeniu specjalistycznych aplikacji służących do przetwarzania dokumentów SGML.

W przypadku dokumentów XML publicznie dostępnych, parserów jest w chwili obecnej kilkadziesiąt. Do popularniejszych zaliczyć można Microsoft Java XML Parser firmy Microsoft, LT XML firmy Language Technology Group, Exapt oraz XP (James Clark)

3.3. Wykorzystanie języków skryptowych

3.4. Wykorzystanie szablonów XSL

Stosując wersję XML typu DocBook można wykorzystać szablony stylów przygotowane w standardzie XSL (autor N. Walsh). W chwili obecnej są dostępne narzędzia umożliwiające przetworzenie dokumentów XML do postaci drukowanej (Adobe PDF) oraz hipertekstowej (HTML).

Podobnie jak w przypadku szablonów DSSSL, szablony stylów XSL są sparametryzowane i udokumentowane i dzięki temu łatwe w adaptacji. Do zamiany dokumentu XML na postać prezentacyjną można wykorzystać jeden z dostępnych publicznie procesorów XSLT (por. tabela 3.1).

Nazwa	Autor	Adres URL
`sablotron`	Ginger Alliance	`http://www.gingerall.com`
`Xt`	J. Clark	`http://www.jclark.com`
`4XSLT`	FourThought	`http://www.fourthought.com`
`Saxon`	Michael Kay	`http://users.iclway.co.uk/mhkay/saxon`
`Xalan`	Apache XML Project	`http://xml.apache.org`

Tabela 3.1: Publicznie dostępne procesory XLST

Źródło: Opracowanie własne

XSL:FO jest skomplikowanym językiem o dużych możliwościach, zawierającym ponad 50 różnych „obiektów formatujących", począwszy od najprostszych, takich jak prostokątne bloki tekstu poprzez wyliczenia, tabele i odsyłacze. Obiekty te można formatować wykorzystując przeszło 200 różnych właściwości (properties), takich jak: kroje, odmiany i wielkości pisma, odstępy, kolory itp. W tym dokumencie przedstawione jest absolutne miniumum informacji na temat standardu XSL:FO.

Cały dokument XSL:FO zawarty jest wewnątrz elementu `fo:root`. Element ten zawiera (w podanej niżej kolejności):

- dokładnie jeden element `fo:layout-master-set` zawierający szablony określające wygląd poszczególnych stron oraz sekwencji stron (te ostatnie są opcjonalne, ale typowo są definiowane);
- zero lub więcej elementów `fo:declarations`;
- jeden lub więcej elementów `fo:page-sequance` zawierających treść formatowanego dokumentu wraz z opisem jego sformatowania i podziału na strony.

Zakończenie

Możliwości, jakie stoją przed archiwum prac magisterskich opartych na XML-u, są ograniczone jedynie czasem, jaki należy poświęcić na pełną implementację systemu. Nie ma przeszkód technologicznych do stworzenia co najmniej równie doskonałego repozytorium, jak ma to miejsce w przypadku ETD. Jeżeli chcemy w pełni uczestniczyć w rozwoju nowej ery informacji, musimy szczególną uwagę przykładać do odpowiedniej klasyfikacji i archiwizacji danych. Sądzę, że język XML znacznie to upraszcza.

DODATEK A

Tytuł załącznika jeden

Treść załącznika jeden.

DODATEK B

Tytuł załącznika dwa

Treść załącznika dwa.

Spis tabel

Spis rysunków

Oświadczenie

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

.....	.....
data	podpis