Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

KATEDRA INFORMATYKI STOSOWANEJ

PRACA MAGISTERSKA

ADAM RZEPKA

ANALIZA MOŻLIWOŚCI WYKORZYSTANIA NOWYCH TECHNOLOGII ZAWARTYCH W HTML5 DO REALIZACJI GRY TYPU FPS

PROMOTOR:
dr inż. Grzegorz Rogus

Kraków 2013

OŚWIADCZENIE AUTORA PRACY

OŚWIADCZAM, ŚWIADOMY ODPOWIEDZIALNOŚCI KARNEJ ZA POŚWIADCZENIE NIEPRAWDY, ŻE NINIEJSZĄ PRACĘ DYPLOMOWĄ WYKONAŁEM OSOBIŚCIE I SAMODZIELNIE, I NIE KORZYSTAŁEM ZE ŹRÓDEŁ INNYCH NIŻ WYMIENIONE W PRACY.

. . . . . .

PODPIS

AGH

University of Science and Technology in Krakow

Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering

DEPARTMENT OF APPLIED COMPUTER SCIENCE

MASTER OF SCIENCE THESIS

ADAM RZEPKA

ANALYSIS

SUPERVISOR:
Grzegorz Rogus Ph.D

Krakow 2013

podziękowania

Spis treści

1. Wprowadzenie

1.1. Cele pracy

1.2. Teza pracy

1.3. Zawartość pracy

2. Omówienie dziedziny

2.1. HTML5

HTML5 jest kolejną wersją języka HTML służącego to tworzenia stron WWW, opracowywany przez World Wide Web Consortium (W3C) i Web Hypertext Application Technology Working Group (WHATWG). Standard jest w większości ukończony i jego ostateczna specyfikacja ma być wydana do końca roku 2014, natomiast w ciągu kolejnych dwóch lat ma nastąpić wydanie wersji 5.1. Jest on pomyślany jak następca HTML 4.1 i XHTML 1.0.

HTML5 wprowadza wiele nowych tagów (np. canvas, video, audio) oraz interfejsów programistycznych wymienionych w dalszych podrozdziałach. Wraz z nowymi możliwościami HTML5 wraz z językiem JavaScript, stał się platformą umożliwiającą tworzenie rozbudowanych aplikacji oraz gier.

2.1.1. JavaScript

JavaScript jest skryptowym językiem programowania osadzony w przeglądarkach internetowych i wykorzystywany do tworzenia interaktywnych stron oraz aplikacji internetowych. Formalnie nie jest związany z HTML5, jednak jest jedynym językiem obsługiwanym przez wszystkie przeglądarki, dlatego wszystkie wymienione w dalszych podrozdziałach interfejsy programistyczne są stworzone dla tego języka.

JavaScript powstał 1995 w firmie Netscape, a następnie został ustandaryzowany przez ECMA (stąd stosowana formalnie nazwa ECMAScript). Nazwa i składnia przypomina Javę, jednak jest to podobieństwo mylące. JavaScript jest językiem dynamicznym, posiadającym wiele cech języków funkcyjnych, a obiektowość jest oparta na prototypach, a nie klasach.

Początkowo JavaScript był używany do wyświetlania prostych animacji, komunikatów i wstępnej walidacji danych w formularzach. Z w JavaScripcie czasem zaczęły powstawać coraz bardziej rozbudowane aplikacje i okazało się, że wydajność interpretowanego języka skryptowego z Garbage Collectorem (odśmiecaczem) jest coraz większym problemem. Rozpoczął się wyścig pomiędzy twórcami przeglądarek o jak najszybszy silnik skryptowy. W rezultacie, obecnie w większości przeglądarek skrypty są kompilowane do kodu natywnego przez kompilator JIT (Just In Time), z zastosowaniem wielu technik optymalizacji kodu, a Garbage Collector jest coraz szybszy. Dostępne są również rozbudowane narzędzia do profilowania kodu JavaScript.

Ta sytuacja sprawiła, że JavaScript zaczął się nadawać do robienia wymagających obliczeniowo gier 3d. Oczywiście wciąż trzeba unikać szczególnie nieefektywnych konstrukcji tego języka i starać się

alokować jak najmniej obiektów, aby ograniczyć przestoje powodowane przez działanie odśmiecacza. Jednak przy zachowaniu tych zasad, wydajność współczesnych silników JavaScript powinna być wystarczająca dla wielu gier.

asm.js

asm.js jest to podzbiór JavaScript opracowany przez Mozillę, który zapewnia najszybsze wykonanie. Programowanie zgodnie z asm.js jest jednak bardzo żmudne, dlatego podzbiór ten jest głównie pomyślany jak cel kompilatorów innych języków do JavaScript (np. Emscripten [link/footnote]). Nie istnieje obecnie translator dowolnego kodu JavaScript do asm.js. Ponieważ gra będąca przedmiotem niniejszej pracy powstaje w JavaScript, tematyka asm.js nie będzie szerzej omawiana.

2.1.2. WebGL

WebGL jest to API dla języka JavaScript służące do wyświetlania grafiki 3d w przeglądarce. Z tego powodu jest to najbardziej istotna technologia z punktu widzenia twórców gier. Pomimo, że formalnie WebGL nie jest jeszcze częścią standardu HTML5, to jest on zaimplementowany we wszystkich znaczących przeglądarkach.

WebGL bazuje na standardzie OpenGL ES 2.0 (Open Graphics Library for Embedded Systems) przeznaczonym dla urządzeń mobilnych, który z kolei jest uproszczoną wersją OpenGL (Open Graphics Library) – otwartego API do renderowania grafiki 3d, będącym jednym z dwóch (obok Direct3D) szeroko stosowanych API do wyświetlania grafiki w grach i programach wykorzystujących 3d. Wszystkie te standardy zostały opracowane przez konsorcjum Khronos Group (wcześniej ARB), zrzeszające wszystkie (poza Microsoftem) większe firmy zainteresowane tematem grafiki 3d.

Prace nad WebGL rozpoczęła Mozilla w 2006 roku. A w następnym roku Firefox oraz Opera miały już pierwsze działające implementacje. Były one kontynuowane w grupie roboczej WebGL Working Group działającej w ramach Khronos Group z udziałem m. in. Mozilli, Opery, Google oraz Apple. W 2011 roku ukończona została pierwsza wersja standardu, a w 2013 ruszyły prace nad wersją 2.0 bazującą na OpenGL 3.0. W tym też roku Microsoft wydał Internet Explorer 11 z obsługą WebGL, pomimo początkowej niechęci do tego standardu (Microsoft promuje swoją technologię Direct3D, konkurencyjną w stosunku do OpenGL). Tym samym ostatnia z liczących się przeglądarek dodała obsługę WebGL.

WebGL zaprojektowano dla języka JavaScript, w przeciwieństwie do OpenGL i OpenGL ES, które przeznaczone są głównie dla języka C. Jednakże jest interfejsem bardzo niskopoziomowym, w zasadzie dokładnie przełożonym na JavaScript OpenGL ES. Dodano jedynie kilka usprawnień (jak wczytywanie tekstury z elementu HTML <img>) oraz dodatkową walidację danych, istotną w środowisku tekstowym. Z tego powodu programiści JavaScript mogą uznać API za trudne i nieprzystępne. Z drugiej jednak strony, taka implementacja gwarantuje maksymalną szybkość działania, a programiści znający OpenGL mogą w zasadzie natychmiast zacząć używać WebGL.

2.1.3. Web Workers

Współcześnie każdy domowy komputer posiada wielordzeniowy procesor. W tym samym kierunku podążają procesory w urządzeniach mobilnych. Aby maksymalnie wykorzystać moc obliczeniową, konieczne jest tworzenie aplikacji współbieżnych. Niestety JavaScript przez długi czas tego nie umożliwiał. Zmienia to API Web Workers należące do standardu HTML5.

Worker jest to odpowiednik wątku dla języka JavaScript. Jednakże jego sposób działania upodabnia go bardziej do procesu – poszczególne workery nie mogą współdzielić pamięci i komunikują się wyłącznie za pomocą wysyłanych wiadomości. Dzięki takiej konstrukcji można uniknąć wielu problemów związanych z klasyczną wielowątkowością, jednak odbywa się to kosztem wydajności z powodu konieczności częstego kopiowania danych. Dodatkowo API dostępne dla workera jest bardzo ograniczone (np. nie ma dostępu do elementów HTML).

Te cechy powodują, że aby osiągnąć zysk wydajnościowy przy pomocy Web Workers, trzeba od początku projektować aplikację pod kątem tego API – tak aby ograniczyć ilość przesyłanych danych.

Należy zaznaczyć, iż taki sposób tworzenia aplikacji wielowątkowych jest nieobcy programistom gier, gdyż podobny model wymuszał procesor Cell w konsoli PlayStation3. Procesor ten składa się z jednego głównego rdzenia (tzw. Power Processing Element) oraz kilku rdzeni wspomagających (tzw. Synergistic Processing Element), które również nie mogą współdzielić pamięci. Podobnie jak Web Workerze, tak w programie działającym na SPE, dostępne API jest bardzo ograniczone.

2.1.4. Web Sockets

Wiele współczesnych gier umożliwia wspólną rywalizację wielu graczy za pośrednictwem internetu (tzw. multiplayer). W celu implementacji gry sieciowej konieczne jest API do komunikacji internetowej.

Interfejs Web Sockets jest odpowiednikiem systemowych gniazd sieciowych i umożliwia komunikację w czasie rzeczywistym pomiędzy aplikacją JavaScript, a serwerem za pośrednictwem protokołu TCP. Jest on częścią HTML5 i dostępny we wszystkich znaczących przeglądarkach.

2.1.5. WebRTC

Wprowadzenie interfejsu Web Sockets było istotnym krokiem z punktu widzenia twórców gier sieciowych, ma on jednak pewne ograniczenia. Jest oparty o protokół TCP, którego niezawodność powoduje opóźnienia w transmisji danych (przesyłanie potwierdzeń, retransmisje itp.). Dodatkowo cały ruch musi być kierowany przez serwer, gdyż bezpośrednia komunikacja pomiędzy przeglądarkami jest niemożliwa. To sprawia, że Web Sockets nie nadają się do bardzo dynamicznych gier.

Rozwiązanie tego problemu nadeszło ze strony interfejsu WebRTC (od Real Time Communication), mającego służyć przede wszystkim do przesyłanie wideo i dźwięku w czasie rzeczywistym pomiędzy przeglądarkami, dzięki czemu możliwe byłoby zbudowanie komunikatora wideo w przeglądarce.

WebRTC umożliwia również przesyłanie dowolnych danych tekstowych i binarnych, a ponieważ działa w oparciu o protokół UDP i łączy bezpośrednio przeglądarki (peer-to-peer), idealnie nadaje się do

szybkich gier sieciowych. Oczywiście użycie bezpołączeniowego protokołu UDP powoduje, że aplikacja sama musi zapewnić poprawność działania w przypadku błędów przesyłu.

WebRTC był początkowo tworzonym przez Google, lecz wkrótce prace przejęła grupa robocza w W3C, złożona m.in. z Google, Mozilli i Opery. Niestety Microsoft odmówił wsparcia dla WebRTC i – jak to miało miejsce już wielokrotnie – zaczął tworzyć swój alternatywny standard: CU-RTC-WEB (Customizable, Ubiquitous Real-Time Communication over the Web). Dlatego WebRTC nie jest obsługiwany w Internet Explorer i nie ma żadnych planów jego implementacji. Co więcej, żadna z przeglądarek nie posiada jeszcze pełnej i stabilnej implementacji (stan na początek 2014 roku). Jednakże w niektórych (Chrome, Firefox) API już na tyle dojrzało, że można go z powodzeniem używać.

2.1.6. Web Audio API

Poza grafiką, w grach bardzo ważna jest również oprawa dźwiękowa. O ile odtwarzanie muzyki nie było problemem od czasu wprowadzenia taga <audio> to udźwiękowienie efektów w grze wymagało bardziej zaawansowanego API.

W tym celu do HTML5 zostało wprowadzone Web Audio API. Jest to wysokopoziomowy, zaawansowany interfejs do przetwarzania i odtwarzania dźwięków w JavaScript. Umożliwia on wszystko co potrzebne do udźwiękowienia gry – dźwięk przestrzenny, nakładanie wielu efektów itp.

Aktualnie (2014) Web Audio API jest wspierane przez wszystkie większe przeglądarki oprócz Internet Explorer.

2.1.7. Pozostałe przydatne API

HTML5 zapewnia wiele interfejsów programistycznych przydatnych w wielu aplikacjach i grach w zależności od potrzeb. Wśród bardziej przydatnych należałoby wymienić:

- Gamepad API – obsługa gamepadów, joysticków, kierownic komputerowych itp.
- Mouse Lock API – możliwość ukrycia i zablokowania kursora myszy,
- Fullscreen API
- Web Storage – przechowywanie danych po stronie klienta (np. w celu cache'owania danych gry),
- File API – operowanie na plikach,
- XmlHTTPRequest – asynchroniczne pobieranie plików z serwera (standard de facto od dłuższego czasu),
- Device Orientation i Touch Events – dla urządzeń mobilnych.

2.2. Gry Komputerowe

Gry komputerowe to w dzisiejszych czasach ogromny rynek zapewniający rozrywkę milionom graczy na całym świecie. Historia gier zaczęła się niedługo po wynalezieniu komputerów. Początkowo

były to bardzo proste programy, jednak z czasem stawały się coraz bardziej skomplikowane, zarówno pod względem technicznym (grafika, dźwięk), jak i pod względem mechaniki (zasad gry).

[jakiś skrin?]

Gry opanowały wiele platform sprzętowych. Wcześniej utożsamiane głównie z automatami i konsolami, wkrótce odniosły wielki sukces na komputerach osobistych, a współcześnie na telefonach i tabletach. Co prawda gry działające w przeglądarce pojawiały się od początku istnienia języka JavaScript, czy platformy Adobe Flash (wcześniej Macromedia Flash), jednak były one utożsamiane z bardzo prostą rozgrywką i ubogą warstwą audiowizualną. Wraz w nadejściem ery HTML5 i technologii z nim związanych (przede wszystkim WebGL), zaistniała możliwość ekspansji bardziej zaawansowanych gier na ten – stosunkowo jeszcze młody – rynek.

W następnych podrozdziałach zostanie przedstawiony gatunek gier FPS oraz dwie gry tego typu, szczególnie związane z niniejszym opracowaniem, a także silnik gry, na którym są oparte. Następnie wprowadzone zostaną koncepcje związane z architekturą gier sieciowych (mutliplayer).

2.2.1. Gry FPS

Przedmiotem pracy jest stworzenie gry typu FPS działającej w przeglądarce.

Skrót FPS oznacza First Person Shooter. Jest to strzelanina w której obserwujemy świat oczami bohatera. Gry typu FPS należą do najstarszych i najczęściej występujących gier 3d. Z jednej strony mają one dość prostą mechanikę i zasady, z drugiej natomiast, często są bardzo zaawansowane technologicznie i bywają pionierskie w dziedzinach takich jak realistyczna grafika, czy rozgrywka sieciowa. Dlatego właśnie taki gatunek został wybrany do implementacji w HTML5.

[screen ze współczesnej gry]

2.2.2. Quake III Arena

W roku 1996 ukazała się gra Quake, która odniosła wielki sukces, głównie dzięki rewolucyjnej grafice i dopracowanej rozgrywce sieciowej. Twórcą gry była firma id Software. Rok później powstała kolejna część - Quake II.

Z racji ogromnej popularności trybu dla wielu graczy, kolejna część – Quake III Arena – zupełnie porzuciła tradycyjny tryb fabularny i skupiła się tylko na rozgrywce sieciowej. Została wydana w roku 1999. Gra również odniosła sukces i do dziś pozostaje popularna pośród graczy. Profesjonalne turnieje gier sieciowych (tzw. e-sport) bardzo często w swoim programie mają rozgrywki w Quake III Arena. W roku 2000 ukazał się dodatek do gry pod nazwą Quake III Team Arena.

[screen]

Siedem lat później, w roku 2007 id Software rozpoczął prace nad wydaniem Quake III Arena i Team Arena na przeglądarki internetowe pod nazwą Quake Live. Kod gry uruchamiany był poprzez specjalnie stworzoną wtyczkę do przeglądarki. W związku z tym rozgrywka była możliwa tylko na wybranych systemach operacyjnych i przeglądarkach, konieczna też była instalacja wtyczki przez gracza.

w międzyczasie, w roku 2005 ukazał się Quake IV, który kontynuował wątki fabularne z Quake II. Gra pomimo dobrych ocen w prasie nie przyjęła się tak dobrze wśród graczy, głównie ze względu na brak znaczących ulepszeń w trybie sieciowym.

Quake III oraz Quake Live jest oparty na silniku gry o nazwie id Tech 3, o którym będzie mowa w podrozdziale 2.2.4.

2.2.3. OpenArena

W roku 2005 główny programista id Software, John Carmack, ogłosił uwolnienie kodu źródłowego silnika id Tech 3 na licencji GNU GPL. Uwalnianie źródeł swoich starszych technologii jest już tradycją w tej firmie.

Wkrótce rozpoczęły się prace nad wieloma grami open source korzystającymi z tego silnika. Jedną z najbardziej dojrzałych jest OpenArena.

OpenArena jest właściwie klonem Quake III Arena. Ponieważ z pierwowzoru został udostępniony tylko kod, fani utworzyli potrzebne zasoby gry (modele, tekstury itp.), również na licencji GPL. Wprowadzono też wiele usprawnień do kodu gry, dodając np. obsługę nowszej wersji OpenGL i shaderów [1]. Tym samym OpenArena stała się pełnoprawną i całkowicie otwartą grą.

[skrin]

2.2.4. id Tech 3

Każda bardziej zaawansowana gra posiada w kodzie źródłowym mniej lub bardziej rozgraniczony rdzeń nazywany silnikiem gry. Odpowiada on za wczytywanie i wyświetlanie świata 3d, odtwarzanie dźwięków, obsługę sieci oraz wiele innych zadań. Często jeden silnik jest wykorzystywany przy tworzeniu wielu gier.

id Software zawsze wykorzystywało swoje autorskie silniki, których źródła były publikowane po utworzeniu nowej wersji. Quake III Arena, a potem OpenArena wykorzystywały id Tech w wersji 3.

Silnik ten był bardzo zaawansowany technologicznie jak na swoje czasy. Do wyświetlania grafiki konieczny był sprzętowy akcelerator graficzny. Pośród ciekawszych możliwości tego silnika należy wymienić:

– wyświetlanie powierzchni opartych na krzywych sklejanych,

– system tzw. skryptów shaderów, dzięki którym graficy mieli dużą kontrolę nad wyglądem każdej powierzchni [2],

– wyświetlanie generowanych wcześniej map oświetlenia, zapewniających bardziej realistyczny wygląd otoczenia,

– przestrzenna mgła,

[1] Shader - niewielki program działający na karcie graficznej, opisujący właściwości wierzchołków i pikseli.

[2] Nie należy mylić tego systemu nieistniejącymi jeszcze shaderami działającymi na karcie graficznej. Skrypty z id Tech3 były jednak pewną namiastką shaderów w dzisiejszym rozumieniu.

– odbicia lustrzane,

– zaawansowany system synchronizacji stanu gry przez sieć,

– QVM – wbudowana wirtualna maszyna wykonująca kod gry.

2.2.5. Mulltiplayer

Multiplayer jest to tryb gry, w którym uczestniczy wielu graczy, zazwyczaj za pośrednictwem Internetu. Tryb ten jest często spotykany w grach FPS, ale także w grach wyścigowych, czy strategiach. Wiele gier i graczy skupia się wyłącznie na grze sieciowej. Zdarza się, że w jednej rozgrywce uczestniczy na raz tysiące graczy [3]

Tryb multiplayer wymaga, aby wszyscy gracze widzieli ten sam stan gry. Synchronizacja gry pomiędzy komputerami w sieci jest zagadnieniem nietrywialnym, szczególnie w przypadku gier bardzo dynamicznych i posiadających złożoną mechanikę (a przez to złożony opis stanu gry).

Z punktu widzenia organizacji w sieci, możliwe są dwie architektury:

– Peer-to-peer – nie ma głównego komputera, każdy komputer rozsyła do wszystkich pozostałych informacje o każdej akcji wykonanej przez gracza lokalnego. Architektura ta jest rzadko używana, ze względu na trudność z synchronizacją i problemy z nieuczciwymi graczami.

– Klient - serwer – najczęściej używana. Jeden z komputerów jest serwerem gry. Na nim odbywa się cała symulacja, a wszyscy klienci służą jedynie jako terminale, wysyłając informacje o naciśniętych przez gracza przyciskach i otrzymując migawkę (snapshot) aktualnego stanu gry do wyrenderowania.

[obrazek z porównaniem]

Należy zaznaczyć, że serwer niekoniecznie musi być dedykowaną maszyną utrzymywaną przez producenta gry. W przypadku gier w niewielką liczbą uczestników, zazwyczaj jeden z komputerów graczy pełni rolę serwera. Dodatkowo, bardzo często producent zapewnia tzw. lobby server, czyli główny serwer, który służy m. in. do odnajdywania toczących się właśnie rozgrywek i inicjalizacji połączenia. W związku z tym pojęcie serwer może mieć dwa różne znaczenia i wymaga doprecyzowania, dlatego na potrzeby dalszej części pracy wprowadzimy następujące pojęcia:

– serwer główny lub lobby – serwer utrzymywany przez dewelopera, zbierający informacje o toczących się grach i ułatwiający inicjalizację połączenia,

– serwer gry – instancja gry która zarządza konkretną rozgrywką

W przypadku gry przeglądarkowej, serwer gry działa w przeglądarce, co może wprowadzić dodatkowe nieporozumienie, gdyż w aplikacjach webowych przeglądarka jest tożsama z klientem.

2.3. Poprzednie prace

W niniejszym rozdziale zaprezentowane zostaną projekty, mające związek z tematyką pracy.

[3] Są to tak zwane gry MMO – Massive Multiplayer Online

2.3.1. Quake II w przeglądarce

W roku 2010 dwaj pracownicy Google, Joel Webber i Ray Cromwell udostępnili swój projekt – port Quake II na przeglądarki z wykorzystaniem WebGL. Była to jedna z pierwszych prac, które udowodniły możliwość zaistnienia gier 3d w przeglądarkach.

quake2-gwt-port, bo tak został nazwany projekt, powstał poprzez skompilowanie w GWT [tu ref/footnote/whatever] kodu Jake2 – portu Quake II do Javy. Projekt ten nie powstał bezpośrednio w JavaScript, przez co nie integruje się zbyt dobrze ze środowiskiem webowym, a do uruchomienia konieczne było wiele łat i rozwiązań prowizorycznych. Dodatkowo w roku 2010 nie istniało jeszcze Mouse Lock API, przez co sterowanie grą jest bardzo trudne.

Z tych powodów projekt ten należy uznać raczej za demo technologiczne niż grę. Mimo to był to ważny krok milowy dla technologii WebGL.

[strona projektu]

2.3.2. Quake III level viewer

W tym samym roku Brandon Jones zaprezentował swoje demo o nazwie q3bsp. Jest to level viewer[4] poziomów z Quake III Arena. W przeciwieństwie do quake2-gwt-port, q3bsp został w całości napisany w JavaScripcie, dzięki czemu mógł użyć wiele udogodnień zapewnianych przez środowisko przeglądarki (np. zapewnione wczytywanie plików graficznych, interfejs użytkownika w HTML), a jego wydajność jest dużo lepsza, pomimo bardziej zaawansowanej grafiki.

Projekt ten również zyskał duży rozgłos w środowisku deweloperów WebGL i poza nim. Można go zobaczyć na stronie [strona]. Część kodu źródłowego q3bsp (udostępnionego na licencji zlib) została wykorzystana w niniejszej pracy.

2.3.3. Banana Bread

Banana Bread jest portem gry Cube2:Sauerbraten [link] na JavaScript zrobionym przez Mozillę. Powstał przez kompilację z C++ kompilatorem Emscripten.

Jest to pełnoprawna i działająca gra, w którą można zagrać na stronie [ref].

[4] Level viewer - narzędzie umożliwiające podgląd danego poziomu gry, bez uruchamiania logiki rozgrywki

3. Projekt

3.1. Zakres projektu

3.2. Architektura systemu

3.2.1. Strona serwera

3.2.2. Strona klienta

3.3. Architektura aplikacji przeglądarkowej

3.3.1. Ogólny opis

3.3.2. Dane gry

3.3.3. Renderer

3.3.4. Klient gry

3.3.5. Serwer gry

4. Analiza

4.1. Stopień realizacji tematu

4.2. Porównanie z grami natywnymi

4.3. Przydatność HTML5 do tworzenia gier

4.3.1. Zalety

4.3.2. Wady

4.4. Możliwości dalszego rozwoju gry

5. Podsumowanie