JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

PŘÍRODOVĚDECKÁ FAKULTA

Katedra: Ústav aplikované informatiky **Obor:** Aplikovaná informatika

Vývoj hry Tower Defense

The development of game Tower Defense

Bakalářská práce

Autor práce:

Jan Tuček

Vedoucí bakalářské práce:

RNDr. Jaroslav Icha

České Budějovice 2015

Bibliografické údaje

Tuček J., 2015: Vývoj hry Tower Defense.
[The development of game Tower Defense. Bc. Thesis, in Czech.] – 39 p.,
Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Cílem této práce je vytvořit tutoriál, který bude popisovat vývoj hry typu Tower Defense. Tento tutoriál bude sloužit jako výukový příklad pro všechny, kteří se chtějí vydat na cestu a stát se budoucím programátorem počítačových her. V tomto tutoriálu bude popsáno, co všechno musí obsahovat tento typ hry, čeho se vyvarovat a jak řešit některé problémy při psaní her. Hra použitá v tutoriálu bude implementována v jazyce Java. Cílem je tedy poskytnout osnovu a inspiraci při psaní obdobné hry. Jsou zohledněny různá úskalí v problematice programování, které mají vliv na efektivitu her.

Hlavní přínos práce spočívá v popisu a analýze psaní počítačových her a následném pochopení psaní počítačových her. Tyto znalosti lze aplikovat i na jiné aplikace než hry. Přinos praktické části práce spočívá v prozkoumání nejdůležitějších částí hry, nejzajímavějších algoritmů a funkcí, které jsou zde použity, jež je možné následně použít v jiných hrách nebo aplikací. Pro zájemce o programování her je užitečný návrh vlastní aplikace.

Klíčová slova

Desktopová aplikace, Hra, Vývoj, Java

Abstract

The aim of this work is to create a tutorial that will describe the development of a game, which is the type of Tower Defense. This tutorial will serve as an educational example for all who want to go on a journey to become a computer games programmer in the future. In this tutorial I will describe what must contain this type of game, what to avoid and how to solve some problems in game programming. The game used in the tutorial will be implemented in Java. The aim is to provide an outline and inspiration when writing similar games. They take into account various difficulties in programming issues that have an impact on the effectiveness of games.

The main contribution of this thesis is to describe and analyze writing computer games and subsequent understanding of writing computer games. This knowledge can be applied to applications other than games. Benefits of the practical part is to examine the most important parts of the game, the most interesting algorithms and functions that are used here, which can then be used in other games or applications. For those interested in game programming is a useful proposal for their own applications.

Keywords

Desktop application, Game, Development, Java

Čestné prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 24. dubna 2015

……

Jan Tuček

Poděkování

Rád bych zde poděkoval panu RNDr. Jaroslavu Ichovi za odborné vedení mé bakalářské práce, věcné připomínky, dobré rady a vstřícnost při konzultacích a vypracovávání bakalářské práce.

OBSAH

Slovníček zkratek

OOP – Object-Oriented-Programming či objektově orientované programování

UML – Unified Modeling Language je jazyk, který slouží pro vizualizaci návrhů projektů

JVM – Java Virtual Machine

JRE – Java Runtime Environment, běhové prostřední (virtuální stroj) potřebné ke spuštění Java aplikací

JDK – Java Development Kit, sada nástrojů (knihovny, překladače, debugger, javadoc, …), která je nezbytná k vývoji aplikací, zároveň obsahuje běhové prostředí JRE

IDE – Integrated Development Environment, neboli vývojové prostředí, které slouží k vývoji aplikací (IDE k vývoji Java aplikací vyžaduje JDK)

GUI – graphical user interface, v překladu grafické uživatelské rozhraní, umožňuje ovládat aplikaci pomocí klávesnice, myši a dalších ovládacích prvků

TD – Tower Defense

1 Úvod

Tower Defense dále jen TD zažívají v několika posledních letech rychle stoupající popularitu jak mezi „Free Online Games“(většinou flashové provedení) v internetových browserech, tak i jako klasické kupované hry. Velké procento dnešních mladých lidí v České republice a ve všech státech, kde je internet běžný, dnes zná anebo hrál nějakou tu Online hru, mezi kterými je typ TD velice rozšířený. Mnoho lidí, kteří se např. nudí ve škole na přednáškách, tak brouzdají po internetu nebo hrají tyto hry, aby si ukrátili dlouhou chvíli při čekání na konec, nebo v práci když zrovna nemají náladu pracovat a šéf se nekouká anebo hrají prostě jen pro zábavu či odlehčení nálady.

Je tedy nesporný fakt, že v dnešní době prakticky každý z nynější generace hrál nebo alespoň slyšel o různých online hrách ať už při procházení internetem nebo od kamaráda. Trend ve hraní těchto her je stále vzrůstající o čemž svědčí i počet webů, které provozují tyto FOG. Počet hráčů se každým rokem zvyšuje a nutnost psát nové a zajímavé hry roste. Vývoj her je limitován jen technologiemi a fantazií.

Důvodem pro výběr tohoto tématu je výše uvedená stoupající popularita TD mezi „Free Online Games“. Již mnoho let hraju tyto TD hry a vím, jak moc jsou návykové. Před pár lety jsem se začal zajímat o psaní her a přemýšlet o tom, jak by vypadala samotná realizace vytvoření hry. Rozhodl jsem se, že se pokusím vytvořit TD a díky tomu jsem začal studovat a zjišťovat co je vše potřebné pro napsání hry.

Cílem práce je tedy ukázka a popsání vývoje TD, prozkoumání technologií, algoritmů, metod a vyvození závěrů, které budou aplikovatelné jak už pro vlastní řešení, tak pro jiné hry či programy související s touto oblastí. Práce poskytne přehled o technologiích použitelných pro vývoj her, včetně návrhu na jejich uplatnění. Praktická část práce se bude zabývat realizací samotné hry typu TD.

K dosažení stanovených jsou v teoretické části práce zpracovávány především různé statistiky a studie zabývající se programováním her. Ke zpracování odborných témat, která se týkají technologií, jsou použity oficiální stránky ORACLE a přidružených standardů. Ostatní technologie jsou popisovány na základě vlastních znalostí a zkušeností.

Pro správné pochopení této práce jsou potřebné alespoň minimální znalosti v oblasti informačních technologií a programování v jazyce Java.

2 Cíle

Cílem této práce je vytvořit funkční hru typu Tower Defense a tutoriál, který bude popisovat vývoj této hry. Tento tutoriál bude sloužit jako výukový příklad pro všechny, kteří se chtějí vydat na cestu a stát se budoucím programátorem počítačových her. V tomto tutoriálu bude popsáno, co všechno musí obsahovat tento typ hry, čeho se vyvarovat a jak řešit některé problémy při psaní her. Hra použitá v tutoriálu bude implementována v jazyce Java. Cílem je tedy poskytnout osnovu a inspiraci při psaní obdobné hry. Jsou zohledněny různá úskalí v problematice programování, které mají vliv na efektivitu her.

Hlavní přínos této práce by měl spočívat v popisu a analýze psaní počítačových her a následném pochopení psaní počítačových her. Tyto znalosti lze aplikovat i na jiné aplikace než hry. Přínos praktické části práce spočívá v prozkoumání nejdůležitějších částí hry, nejzajímavějších algoritmů a funkcí, které jsou zde použity, jež je možné následně použít v jiných hrách nebo aplikací. Pro zájemce o programování her je užitečný návrh vlastní aplikace.

3 Přehled současného stavu řešené problematiky

3.1 Zmapování historie her typu Tower Defense

Mezi první hry, které by se dali považovat za „Tower Defense“ patří hra „Rampart“[1] od společnosti Atari Games Corporation, která vznikla roku 1990 a se významně lišila od dnešních TD. V této hře se totiž hráč musel starat i o ruční zaměřování věží, kdežto v dnešních tradičních TD věže zaměřují nepřátelské jednotky automaticky.

Hry, které se dají považovat za prarodiče dnešních TD, jsou „StarCraft“ a „Warcraft III“, ve kterých se kolem roku 2000 začali objevovat uživatelsky vytvořené mapy na téma TD později známé jako „WC3 TD“. Tyto mapy začali být velice populární a výsledkem bylo vytváření „Flash Elements TD“ v kategorii známé jako „Online Flash Games“. Poté následovali další nejznámější „Online Flash TD“ a to např: „Desktop TD“, „Protector“ série, „Xeno Tactics“, „Warzone tower“, „GemCraft“ série a mnoho dalších.

V poslední době se díky popularitě těchto TD se začali dostávat na trh klasické hry typu TD, např.: „Defense Grid“ od Hidden Path Entertainment, „Dungeon Defenders“, „Orc Must Die“ nebo „Anomaly“ dostupné např. na Steamu[2].

[1] Historie Tower Defense. [online]. [cit. 2010-12-29]. Dostupné z http://towerdefenseheaven.com/content/history-tower-defense-games

[2] Steam, the ultimate entertainment platform. [online]. [cit. 2014-12-10]. Dostupné z http://store.steampowered.com/

3.2 Problematika vývoje her obecně

V dnešní době vytvořit špičkovou hru je věc extrémně náročná a problematika jejich vývoje je poměrně složitá záležitost, proto se nejdříve podíváme na finanční stránku, o které můžeme hovořit jako o **Herním průmyslu**[3]. A pak na obecné problémy při vývoji her.

3.2.1 Finance

Finanční stránka vývoje her je poněkud rozsáhlá, proto ji rozdělíme na dvě části.

Technologie

Koupě technologií potřebných pro vývoj hry: návrhové a programovací prostředí, aplikace na tvorbu grafiky, animací, zvuků, což v profesionální sféře není levné. Ceny licencí těchto profesionálních aplikací se mohou pohybovat dohromady i v řádech statisíců korun. Pro naši hru nám stačí 6 těchto technologií a aplikací, a to Visual Paradigm na vytváření UML návrhů hry, neplacený programovací jazyk Java, vývojové prostředí Netbeans, Corel PHOTO-PAINT X6 na editeci obrázků, Texture Maker na vytváření a editaci textur, particleIllusion na vytváření animací. Podrobně viz str. 16

Lidské zdroje

Jelikož vývoj her je věc časově náročná, viz Tabulka č. 2, je zapotřebí mít dostatečný obnos financí na zaplacení všech lidí, kteří na daném projektu pracují. To může čítat i více jak několik stovek lidí složených z různých profesí, viz Tabulka č. 1 a měsíčně pak může těmto lidem odcházet finance v množství x milionů korun.

Profese	Činnost profese při vývoji her
Manažer	Řídí jednotlivé týmy a komunikaci mezi nimi
Programátor	Tvorba zdrojových kódů hry
Konceptář	Tvorba předloh (kreativně tvůrčí činnost), které pak zpracují grafici
2D grafik	Tvorba 2 rozměrné grafiky
3D grafik	Tvorba 3 rozměrné grafiky
Animátor	Tvorba animací
Zvukaři	Tvorba hudby a zvuků
Designér	Tvorba scénářů, úrovňových systémů, způsobu hratelnosti, atd.
Další profese	Různé

Tabulka č. 1: Profese podílející se na vývoji her

Design hry	Preprodukce hry	Produkce hry:		
		Alfa verze	Beta verze	Gold finální verze
1-6 týdnů	Max. 1 rok (ideální případ)	1-5 let		

Tabulka č. 2: Časový rozvržení vývoje her

[3] Herním průmyslu ČR. [online]. [cit. 2015-02-10]. Dostupné z http://cs.wikipedia.org/wiki/Český_videoherní_průmysl

Vcelku pak vývoj her činí obecně 0,1 až 200 milionů dolarů. Proto je zapotřebí mít zajištěný investory, aby se hra vůbec mohla začít vyvíjet. Toto není snadné, jelikož si investoři své peníze chrání a hru financují na základě tzv. „milestones“ neboli milníků. To jsou vlastně určitý termíny s požadavky, co musí být hotovo, a jen při splnění dostaneme požadovaný peníze a můžeme pokračovat ve vývoji. V tomto tak mají obrovskou výhodu studia přímo pod vydavatelem, která dostávají finance průběžně přesně dle potřeby.

3.2.2 Obecné problémy

Přes rozpočet

Když vývojářskému týmu trvá projekt moc dlouho a dostane se do situace, kdy narazí na maximální limit sponzorů. V takovém případě pak buď musí snížit platy, nebo si zajistit dodatečný rozpočet.

Velké množství závad

Jeden z nejběžnějších problémů, který nastává prakticky u každé hry a je obvykle způsobený nekorektním psaním kódu, kdy se programátor snaží obejít nějaký problém tak, že způsobí jiný. Může se jednat také o grafické závady. Nejvážnější závady, které mají vliv na chod hry, se řeší přednostně, zbylé menší závady se řeší během Beta verze hry, kdy s pomocí Beta testerů odlaďují jednotlivé závady.

Testovací problémy

Obecně testování programů nám umožnuje najít a opravit jejich případné chyby o kterých na první pohled nevíme, obzvláště pak v programovém kódu nebo v grafických vrstvách. Pokud ale máme špatné testovací nástroje, tak se nám buďto objevují neexistující chyby nebo se nám neukáží, což v prvním případě způsobí ohromnou ztrátu času nad něčím, co není, nebo v druhém případě, že hra např. spadne či se zasekne a musí dojít k restartu celého PC. To by u vydané hry způsobilo, že např. dojde ke snížení důvěry k vývojářské firmě a to jí pak může stát dost budoucích peněz.

Problémy s nástroji

Ty jsou způsobeny např. nekompatibilitou nástrojů, špatným fungováním nástrojů v dané oblasti, tím, že očekáváme od nástrojů, že umí či umožňují něco, co nezvládají.

Problémy v komunikaci

Jsou způsobené např. špatnou znalostí jazyka, ve kterém firma pracuje, např. Německá firma s anglickým programátorem. Nebo se jedná o komunikaci mezi vývojářskými týmy, kdy jeden tým něco naprogramuje, ale nesdělí to ostatním a to pak může vést ke značným chybám.

Nedostatek dokumentace

Jedná se o vážný problém, který způsobí značné zpomalení vývoje hry. Např. když se změní vývojářský tým, nebo přibere nového člověka, musí mít dostatečně propracovanou dokumentaci, aby se nový člověk nebo tým mohl dobře zorientovat. Proto je velice dobr0 naprosto vše dokumentovat.

Problémy návrhů

Špatně napsané návrhy, kde chybějí např. některé specifikace nebo jsou špatně popsány, mohou způsobit protáhnutí projektu, jelikož se při psaní hry pak musíme vracet na začátek nebo dopisovat nějaké třídy a to zabere čas, který bychom mohli věnovat jiným problémům.

Existuje mnoho dalších problémů, které zde neuvádím. V případě zájmu je dobé navštívit webovou stránku http://www.gamasutra.com/ a pročíst si některé články s nadpisem „Postmortem“, který označuje dokument, který shrnuje zkušenosti s vývojem projektu, se silným důrazem na pozitivní a negativní aspekty vývojového cyklu. Příklad konkrétních problémů u konkrétních her viz Obr. č. 1.

Game	Unreal or ambitious scope	Feature creep	Cutting features	Design problems	Delay or optimistic schedule	Technological problems	Crunch time	Lack of documentation	Communication problems	Tools problems	Test problems	Team building problems	Great number of defects	Loss of professionals	Over budget	total	%
beam runner Hyper Cross																7	46,7%
Gabriel Knights																13	86,7%
Black & White																9	60,0%
Rangers Lead the Way																7	46,7%
Wild 9																8	53,3%
Trade Empires																4	26,7%
Rainbow Six																11	73,3%
The X-Files																6	40,0%
Draconus																12	80,0%
Cel Damage																3	20,0%
Comand and Conquer: Tiberian Sun																5	33,3%
Asheron's Call																7	46,7%
Age of Empire II: The Age of Kings																5	33,3%
Diablo II																5	33,3%
Operation Flashpoint																12	80,0%
Hidden Evil																5	33,3%
Resident Evil 2																6	40,0%
Vampire: The Masquerade																7	46,7%
Unreal Tournament																7	46,7%
Tropico																4	26,7%
Occurrences	15	15	14	13	13	12	9	8	7	7	7	7	6	5	5	143	7,2
%	75%	75%	70%	65%	65%	60%	45%	40%	35%	35%	35%	35%	30%	25%	25%	47,7%	47,7%

Nenastaly problémy
Nastaly problémy

Obr. č. 1 – Procentuální zobrazení výskytů problémů při vývoji konkrétních her.

Viz http://www.ibiblio.org/winget/2009/10/what-went-wrong-a-survey-of-problems-in-game-development/

3.3 Problematika vývoje her typu TD

Tak to by bylo něco málo k obecné problematice vývoji velkých her. Nyní se ale zaměříme na malé hry a to konkrétně na TD. Pro vývoj těchto malých her není zapotřebí x miliónů korun či obrovský tým lidí. Lze je psát i sólo nebo v malém týmu do deseti lidí. Stačí mít 1 designéra, 1-2 programátory, 1 grafika, 1 zvukaře. Při vývoji hry obsažené v příloze jsem sám zastal všechny tyto lidi a čistý čas strávený nad vývojem hry se pohybuje okolo 2500 – 3000 hodin což by stočlennému týmu trvalo 3 pracovní dny. Což je taky vidět ze hry samotné když jí srovnáme s AAA hrou.

Zde se zaměříme již na konkrétní problémy při vývoji hry. Při psaní hry jsem narazil na 5 hlavních problémů, které mě doprovázeli prakticky celou dobu vývoje hry.

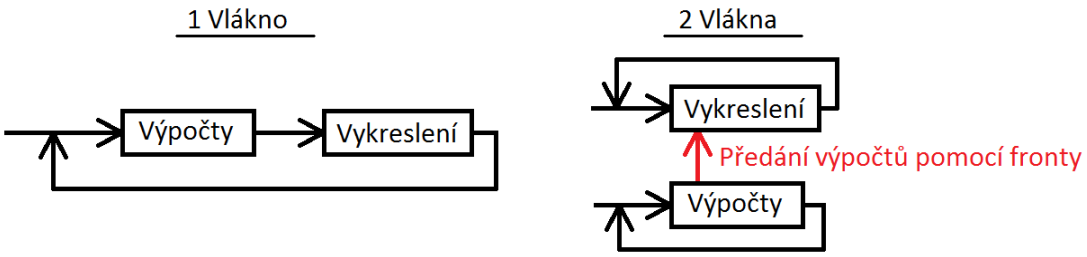
Přestavba struktury tříd

Vzhledem k tomu, že je možné TD neustále rozvíjet o nové prvky je extrémně důležité si hned na začátku vytvořit kvalitní design hry se vším, co bude hra obsahovat a co ne. V případě nedodržení vzniká problém neustálého upravování tříd objektů ve hře, viz příklad.

Př. Nedodržení designu: V původním designu hry je, že věže jsou nesmrtelný, po měsíci vývoje se rozhodneme, že chceme udělat, aby nepřátelské jednotky mohli útočit a ničit naše postavené věže. To ale znamená minimálně upravit třídy Věž, Nepřátelská jednotka, Střela. Úpravou těchto tříd můžou vzniknout další problémy ještě někde jinde a to pak způsobí značný posun v harmonogramu.

Vlákna

U her je běžné, že pracují na více vláknech, je to z důvodu, aby vykreslovací vlákno nemuselo čekat na složité výpočty z výpočetního vlákna, kde se zpracovávají jednotlivé kroky (posun nepřátelských jednotek, natáčení věží apod.) hry. V případě že hra neběží na více vláknech, tak pak dochází k tomu, že se hra jeví pomalejší a bude vytvářet dojem zasekávání se, jelikož vykreslovací metoda musí vždy čekat, než se provedou všechny výpočty. Viz Obr. č. 2. Pokud vyžijeme více vláken, tak pak musíme dbát na Synchronizaci[4].



Obr. č. 2 – Názorná ukázka fungování 1 vlákna vůči 2 vláknům

Práce s geometrií a vektory

U grafických her je samozřejmostí, že se bude pracovat s geometrií. Je proto dobré si podle zvoleného tématu hry nastudovat, základní geometrické funkce, se kterými budeme pracovat, abychom se pak nezdržovali a nehledali po internetu nebo v matematických knížkách jak správně vypočítat vektory pohybu nebo úhly v trojúhelníku a jiné. Ne vždy jsou k dispozici matematické knihovny, které to za nás počítají, a i když je máme, musíme s jejich výsledky správně naložit.

Animace

Jsou nedílnou součástí her, a proto je kladen na ně velký důraz a to zabírá velké množství času. Animace mohou být dvojího typu, a to že chování animace je počítáno přímo ve hře, běžné pro 3D modely, nebo je průběh animace uložen externě v souboru, běžné pro 2D a to

[4] Vlákna a jejich synchronizace. [online]. [cit. 2015-02-15]. Dostupné z http://www.algoritmy.net/article/39287/Vlakna-22

ve formě SPRITE, což je sekvence po sobě jdoucích obrázků uložených v jednom souboru pomocí mřížky, např. v PNG.

V našem případě používáme 2D. Pro usnadnění vytváření animací efektů je dobré použít externí program na vytváření animací „particleIllusion“ s particle systémem viz str. 21

Více rozměrná pole

Může se stát, že v implementaci hry budeme pracovat s vícerozměrnými poli a seznamy. Jedno a dvourozměrná pole jsou jednoduchá, ale u 3 a více rozměrných se můžeme ztratit v cyklech, kterými je prohledáváme, zvláště když v těchto cyklech manipulujeme z nějakého důvodu s proměnou, která slouží jako index v daném poli. V takovém to případě se může stát, že se do proměnné uloží hodnota, která je mimo rozsah daného pole a to nám vypíše chybu a může shodit program. Taková to kritická místa musíme v případě, že tam jsou, ošetřit pomocí příkazu try-catch.

4 Přehled algoritmů a technologií bezprostředně se dotýkajících řešeného úkolu

4.1 Algoritmy

Algoritmů existuje celá řada, pro vývoj této hry stačili následující.

Rekurzivní algoritmus

Jde o algoritmus, který volá sám sebe do té doby než je splněna ukončovací podmínka. Je použit v zaměřovacím systému věží

Brute-force algoritmus

Jedná se o dynamický, nejjednodušší a nejméně účinný algoritmus. Použit v metodě, která ukládá mapu do souboru, a to konkrétně na vygenerování všech kombinací řetězců pomocí regulárním výrazem, které jsou pak použity pro nalezení nejefektivnější komprese mapy.

Greedy algorithm

Jde o algoritmus, který v podstatě vyhledává z určité množiny objektů takovou podmnožinu, která splňuje předem danou vlastnost. Použit ve třídě **Shot** v metodě shot v případě že střela má parametr **Splash** – plošný účinek. Kde se prohledává seznam nepřátelských jednotek, které jsou v dosahu výbuchu střely.

Jinak jsou zde použity jednoduchý algoritmy typu Iterativní, Deterministický a nedeterministický. Složitosti použitých algoritmů se pohybují od N do N^3.

4.2 Technologie

4.2.1 Visual Paradigm

Placený profesionální program[5] na modelování aplikačních návrhů pomocí UML diagramů[6]. Nabízí propracované IDE pro vizuální návrh s možností integrace do Visual Studia .NET, Eclipse/WSAD, NetBeans/Sun ONE a JBuilder. Návrhy lze exportovat do mnoha grafických formátů i tisknout. Lze importovat/exportovat návrh tříd aplikace. Ideální program pro start vyvíjení aplikace.

UML Diagrams		Requirements Capturing		Database Modeling	
Use Case	Class	Textual Analysis	Android Phone Wireframe	Entity Relationship	
Package	Object	Requirement	Android Tablet Wireframe	ORM	
Sequence	Activity	Basic / CRC Card	Desktop Wireframe	Impact Analysis	
Component	Timing	Glossary Grid	iPad Wireframe	Matrix	Chart
State Machine		Web Wireframe	iPhone Wireframe	SoaML	
Deployment		SysML	Enterprise Modeling	Service Interface	
Communication		Block Definition	Zachman Framework	Service Participant	
Composite Structure		Internal Block	Business Motivation Model	Service Contract	
Interaction Overview		Parametric	ArchiMate	Services Architecture	
				Service Categorization	

Business Modeling					Other	
Business Process	Organization Chart		Data Flow		Overview	Brainstorm
Conversation	Fact	Decision	EPC	WSDL	Mind Mapping	Document
Process Map	Business Rule		Business Rule Grid		Grid	

Tabulka č. 3: Typy diagramů ve Visual Paradigm

Pro modelování hry Tower Defense nám ale budou stačit jen UML diagramy a to především Use Case a Class diagramy. Ty budou popsány v další kapitole.

[5] Program Visual Paradigm. [online]. [cit. 2014-12-19]. Dostupné z http://www.visual-paradigm.com/
[6] UML diagramy. [online]. [cit. 2014-12-19]. Dostupné z http://en.wikipedia.org/wiki/Unified_Modeling_Language

4.2.2 Java 

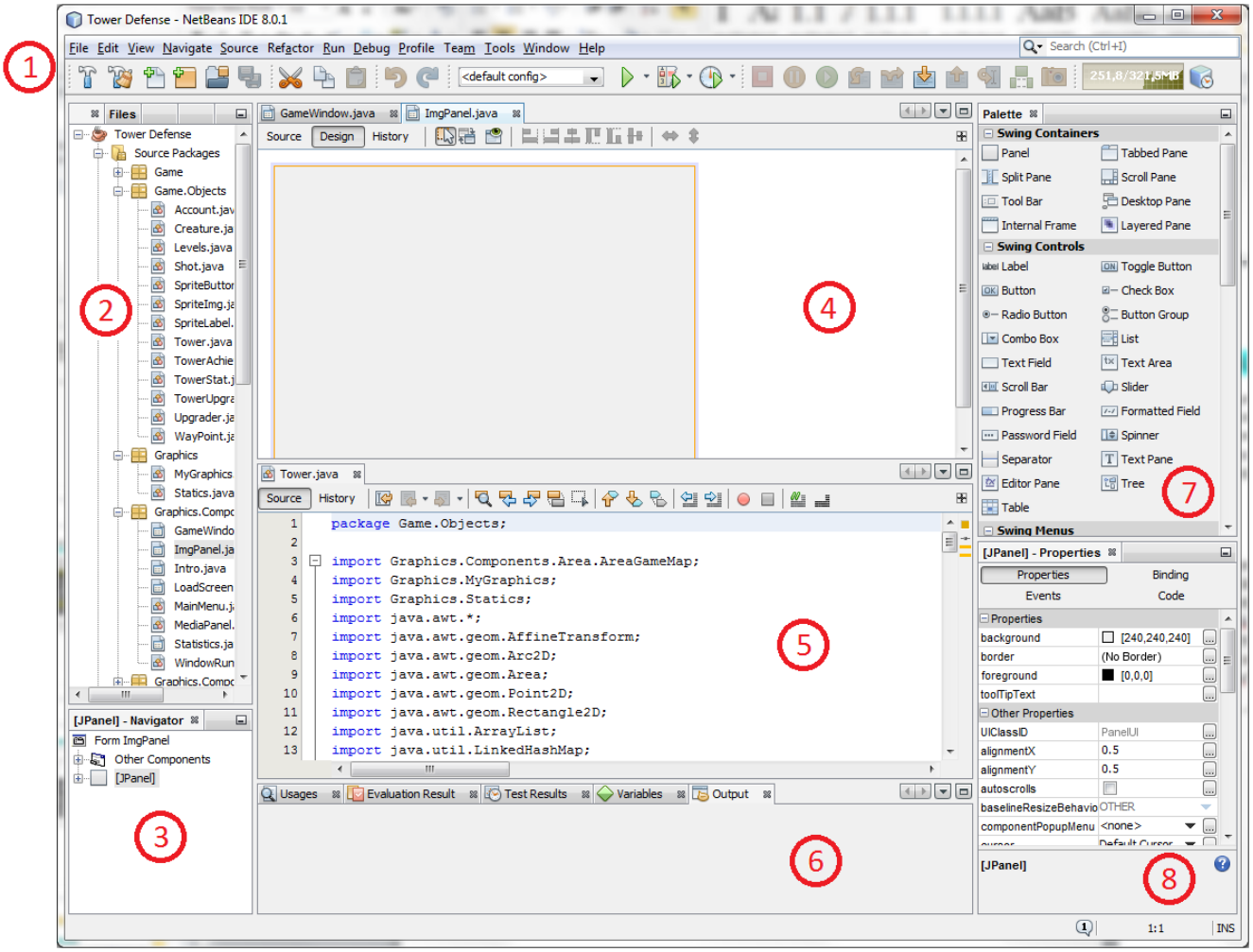
Vcelku jednoduchý OOP programovací jazyk, vyvinutý již v roce 95 vyvinutý firmou Sun Microsystems. Jeho hlavními rysy jsou:

jednoduchost	Naučení programovacího jazyku Java je jednoduché. Vývoj aplikací je velmi rychlý, protože Java už v základu obsahuje spoustu hotových řešení, které můžete při vývoji využít
objektově orientovaný	S výjimkou osmi primitivních datových typů, Java pracuje s vytvářením a manipulací objektů a jejich vzájemnou interakcí. To nám umožní vytvářet modulární programy a opakovaně použitelný kód.
distribuovaný	Java má velkou schopnost práce v sítích. Distribuované výpočty jsou v podstatě platforma, kde dva nebo více počítačů mohou pracovat společně na síti. Sítě na technologii Java, jsou až tak příliš snadné, že psaní sítových programů, je jako posílání a přijímání souborů.
interpretovaný	Místo skutečného strojového kódu se vytváří pouze tzv. mezikód (bytecode). Ten je nezávislý na architektuře počítače nebo zařízení. Program pak může pracovat kdekoliv kde je JVM.
robustní	Znamená, spolehlivé a žádný programovací jazyk nemůže opravdu zajistit spolehlivost. Java klade velký důraz na včasnou kontrolu za případné chyby, protože Java kompilátory jsou schopny detekovat mnoho problémů, které by se nejprve objevili v době spuštění v jiných jazycích. Z tohoto důvodu neumožňuje některé programátorské konstrukce, které bývají častou příčinou chyb (např. správa paměti, příkaz goto, používání ukazatelů). Používá tzv. silnou typovou kontrolu – veškeré používané proměnné musí mít definovaný svůj datový typ.
generační správa paměti	Garbage kolektor spravuje paměť za nás. Automaticky vyhledává již nepoužívané části paměti a uvolňuje je pro další použití.
bezpečný	Jazyk Javy, překladač, interpret a runtime prostření byly vyvinuty s ohledem na bezpečnost. Díky čemuž Java má vlastnosti, které chrání počítač v síťovém prostředí, na kterém je program zpracováván, před nebezpečnými operacemi nebo napadením vlastního operačního systému nepřátelským kódem.
nezávislý na architektuře	Aplikace vytvořená v Javě je platformě nezávislá a lze ji provozovat na libovolném systému s pomocí JVM
výkonný	Přestože se jedná o jazyk interpretovaný, není ztráta výkonu významná, neboť překladače mohou pracovat v režimu „just-in-time“ a do strojového kódu se překládá jen ten kód, který je opravdu zapotřebí.
více úlohový	Podporuje zpracování více vláknových aplikací. Použití více vláken je nutností ve vizuálním a síťovém programování.
dynamický	Java byla navržena pro nasazení ve vyvíjejícím se prostředí. Knihovna může být dynamicky za chodu rozšiřována o nové třídy a funkce, a to jak z externích zdrojů, převážně ze strany společnosti Oracle[7] (dříve Sun), tak vlastním programem.
elegantní	Velice pěkně se v něm pracuje, je snadno čitelný (např. i pro publikaci algoritmů), přímo vyžaduje ošetření výjimek a typovou kontrolu.

[7] Oracle. [online]. [cit. 2015-02-13]. Dostupné z http://www.oracle.com/index.html

4.2.3 Netbeans IDE

Vývojové prostředí[8] pro programovací jazyky Java, C/C++, PHP, Groovy. Jedná se o software s dvěma typy licencí a to CDDL a GPL. Podporuje platformy Windows, Linux, Mac OS a OS Independent Zip. Kromě základních verzí, lze do tohoto programu stáhnou velké množství rozšiřujících pluginů[9], které nyní čítají okolo 900. Má vcelku příjemné uživatelské prostředí a uživatel ho může měnit dle libosti, podle toho jak mu vyhovuje. Konkurentem tohoto vývojového prostředí je Eclipse[10].



Obr. č. 3 – Netbeans prostředí

(1) lišta s ovládáním	(4) vizuální prostředí	(7) seznam objektů, které lze vkládat do vizuálního návrhu aplikace
(2) souborová struktura projektu	(5) otevřená třída	
(3) navigátor pro aktivní oblast	(6) oblast s výstupem	(8) vlastnosti označeného objektu

Všechny tyto oblasti až na (1) lze libovolně přesouvat a seřazovat.

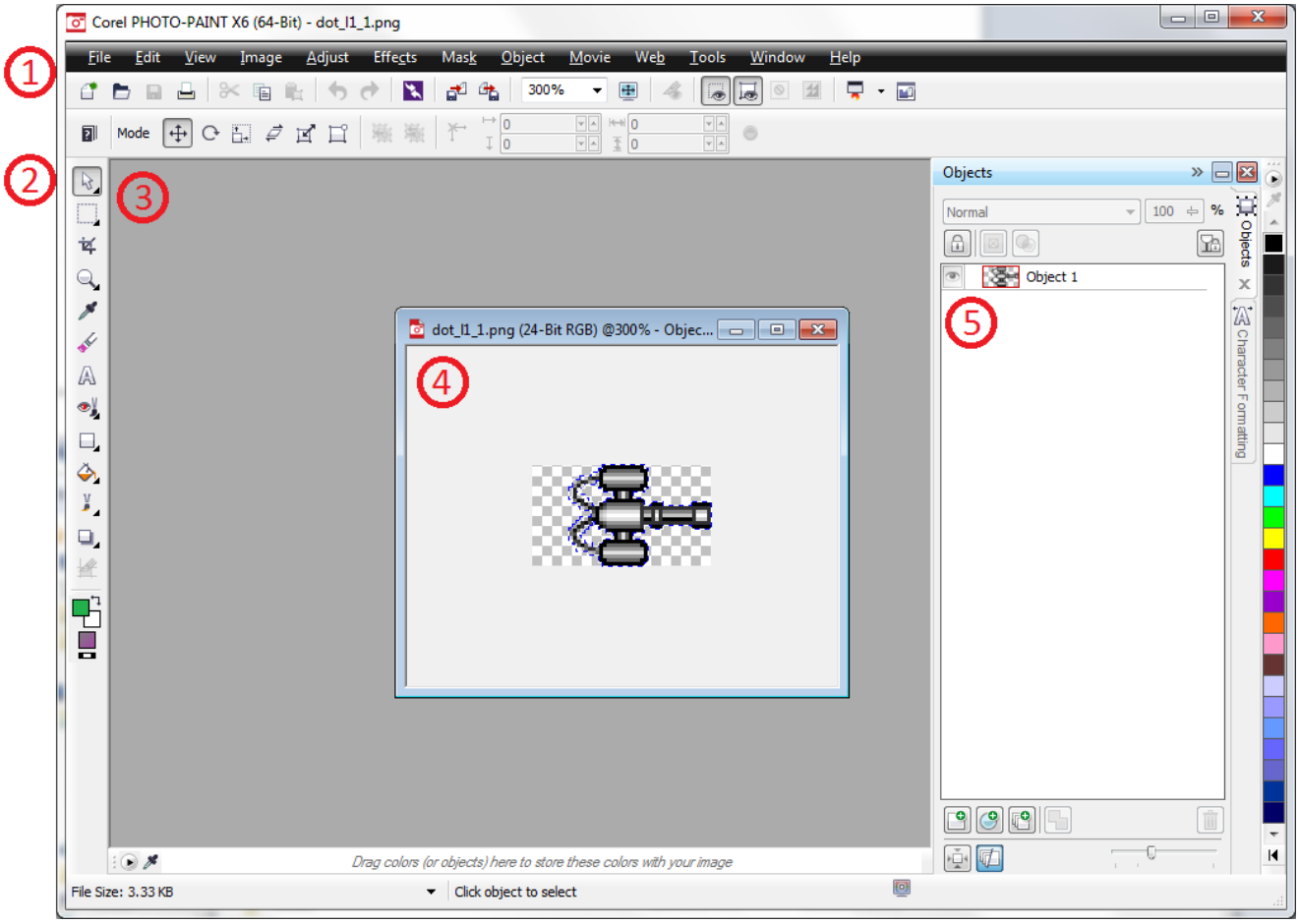
[8] Program Netbeans. [online]. [cit. 2014-12-19]. Dostupné z https://netbeans.org/
[9] Pluginy do Netbeans. [online]. [cit. 2014-12-19]. Dostupné z http://plugins.netbeans.org/PluginPortal/
[10] Eclipse. [online]. [cit. 2014-12-19]. Dostupné z https://eclipse.org/

4.2.4 Corel PHOTO-PAINT X6

Placený profesionální program[11] na tvorbu a editaci rastrových/bitmapových obrázků. Podporuje více jak 20 těchto grafických formátů[12]. Je možné si ho vyzkoušet pomocí 30 - denní trial verze. Plnou verzi lze buďto koupit, to je ale velice drahé (okolo 20 tis. Kč), nebo si předplatit a to vyjde na 800 Kč/měsíc či 6,5 tis. Kč/rok. Další možností jsou free programy[13]

+ Výhody	– Nevýhody
Součástí Suite CorelDRAW	Není k dispozici samostatně
Velké množství nástrojů pro editaci	Upřednostňuje Windows PC
Automatizované příkazy	Strmá křivka učení pro začátečníky

Tabulka č. 4: Corel PHOTO-PAINT výhody a nevýhody, viz http://www.coreldraw.com/cz/



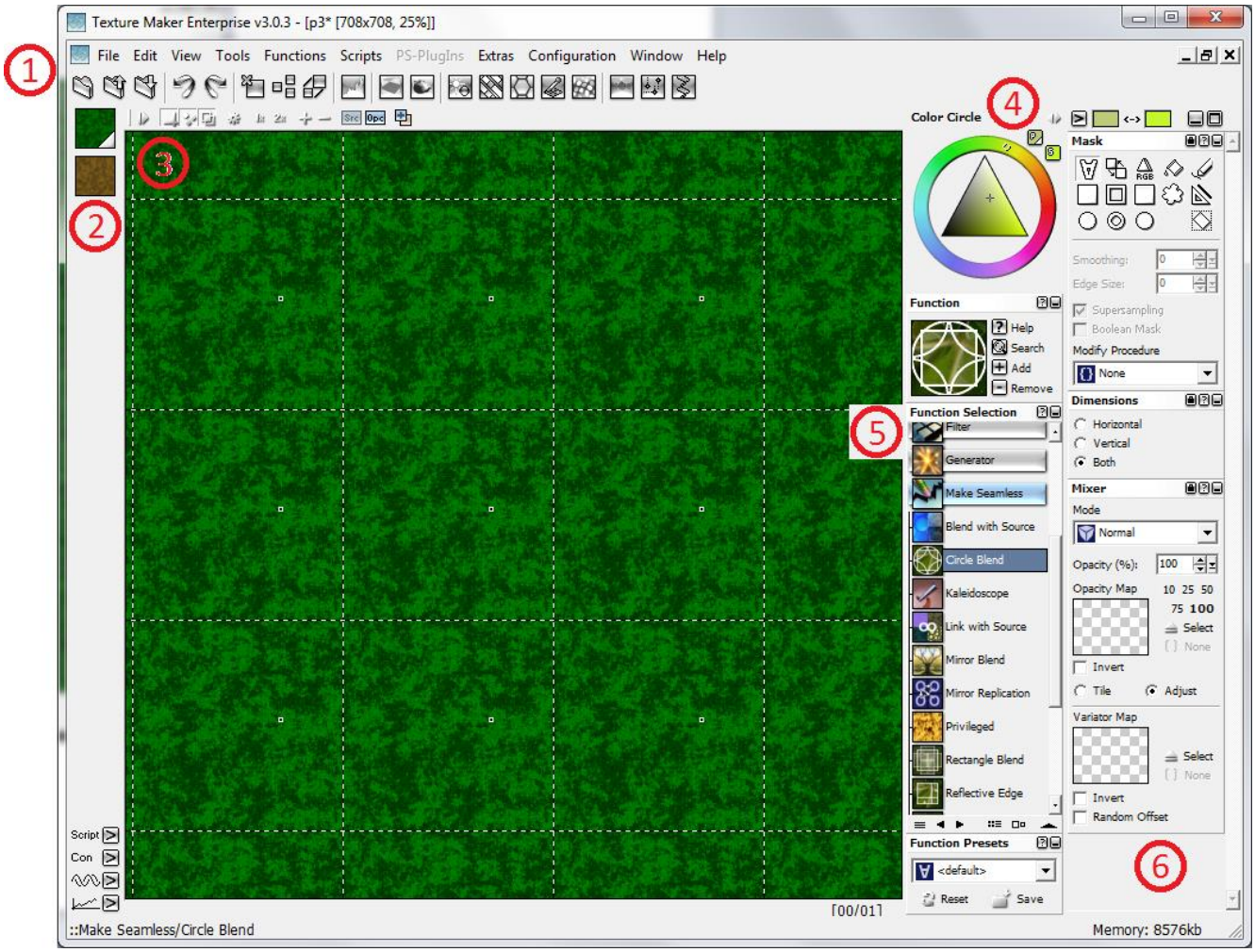
Obr. č. 4 – Corel PHOTO-PAINT X6 prostředí

(1) lišta s ovládáním	(4) okno s obrázkem
(2) lišta s nástroji	(5) vrstvy obrázku
(3) prostor s otevřenými obrázky	

[11] Program Corel PHOTO-PAINT X6. [online]. [cit. 2015-02-15]. Dostupné z http://www.coreldraw.com/cz/
[12] Grafický formát. [online]. [cit. 2015-02-15]. Dostupné z http://en.wikipedia.org/wiki/Image_file_formats
[13] Srovnání rastrových grafických editorů dle dostupnosti. [online]. [cit. 2015-02-15]. Dostupné z http://en.wikipedia.org/wiki/Comparison_of_raster_graphics_editors

4.2.5 Texture Maker

Placený program[14] na vytváření textur[15] pomocí mnoha různých funkcí. Je možné si ho vyzkoušet pomocí 30 - denní trial verze. Plnou verzi lze zakoupit pomocí licence. Množství zpřístupněných funkcí je vázáno na typ licence[16].



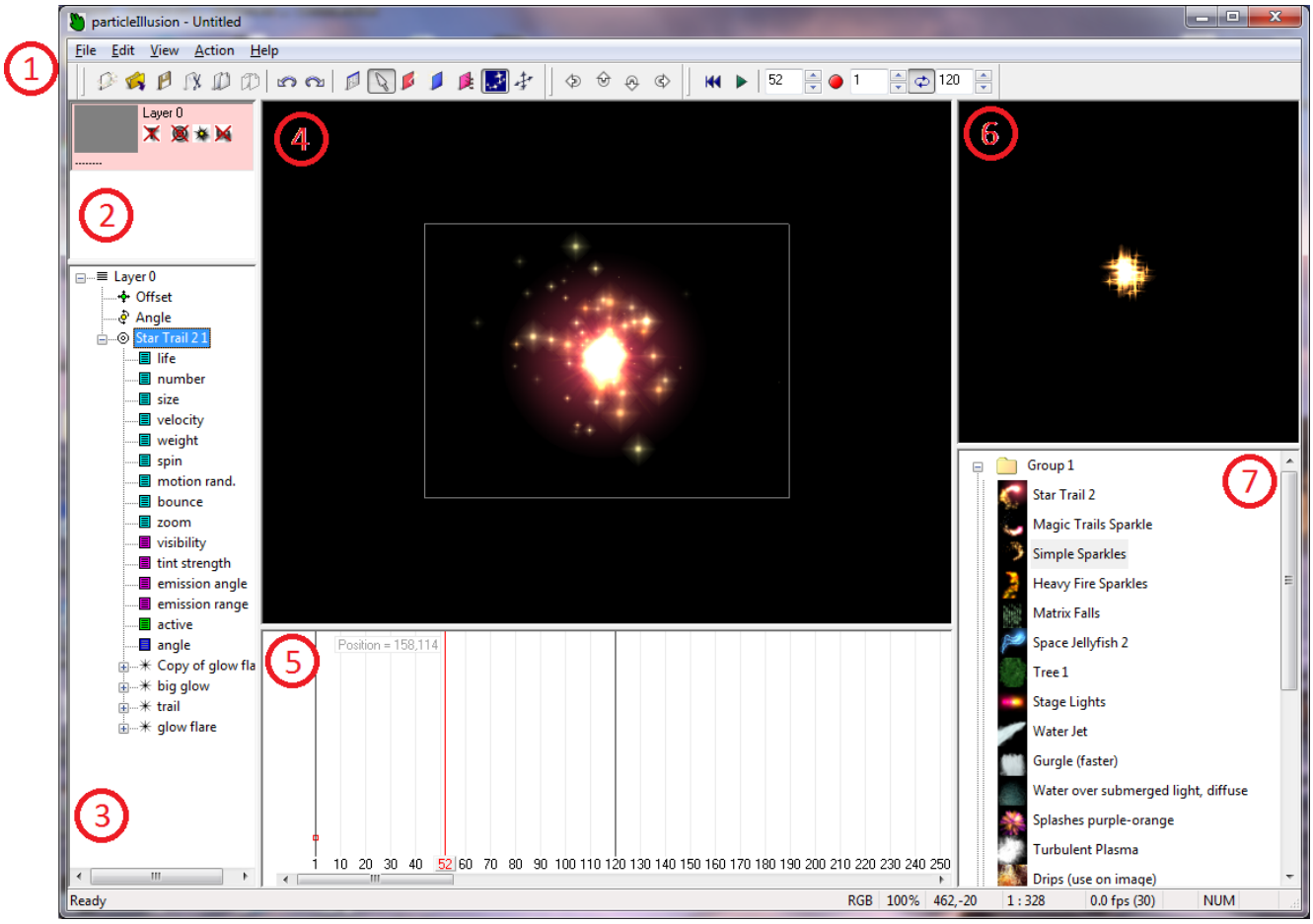
Obr. č. 5 – Texture Maker prostředí

(1) lišta s ovládáním	(4) nastavení barev
(2) seznam otevřených textur	(5) editační funkce
(3) editovaná textura	(6) nastavení vybrané funkce

[14] Program Texture Maker. [online]. [cit. 2015-02-17]. Dostupné z http://www.texturemaker.com/download.php
[15] Texture. [online]. [cit. 2015-02-17]. Dostupné z http://en.wikipedia.org/wiki/Texture_(visual_arts)
[16] Typy licencí. [online]. [cit. 2015-02-17]. Dostupné z http://www.texturemaker.com/help/misc/features.htm

4.2.6 particleIllusion

Placený program[17] na vytváření animací pomocí particle systému[18]. V případě, že bychom nemohli použít tento program, lze si napsat buďto podobný program v javě s pomocí knihovny processing core[19], která obsahuje particle systém, a následně animaci uložit do souboru nebo tento systém přímo použít v aplikaci[20]. To ale má značný vliv na výkon aplikace, jelikož tyto systémy poměrně značně zatěžují procesor, z tohoto důvodu je lepší si danou animaci nejprve vytvořit a uložit do nějakého vhodného formátu např. jako SPRITE[21] v .PNG nebo .GIF či .AVI a následně ji v aplikaci použít, tím ušetříme náročnost na procesor.



Obr. č. 6 – particleIllusion prostředí

(1) lišta s ovládáním	(4) zobrazení animace	(6) zobrazení vybrané animace z knihovny
(2) vrstvy animace	(5) časová přímka	(7) seznam animací otevřené knihovny
(3) seznam vlastností animací		

[17] Program particleIllusion. [online]. [cit. 2015-02-19]. Dostupné z http://www.wondertouch.com/index_2.asp
[18] Particle Systém. [online]. [cit. 2015-02-24]. http://en.wikipedia.org/wiki/Particle_system
[19] Processing Core v Javě. [online]. [cit. 2015-02-19]. https://www.processing.org/
[20] Particle Systém v Javě. [online]. [cit. 2015-02-19]. http://natureofcode.com/book/chapter-4-particle-systems/
[21] Sprite. [online]. [cit. 2015-04-17]. http://en.wikipedia.org/wiki/Sprite_%28computer_graphics%29

5 Návrh řešení

5.1 Formulace a analýza řešeného problému

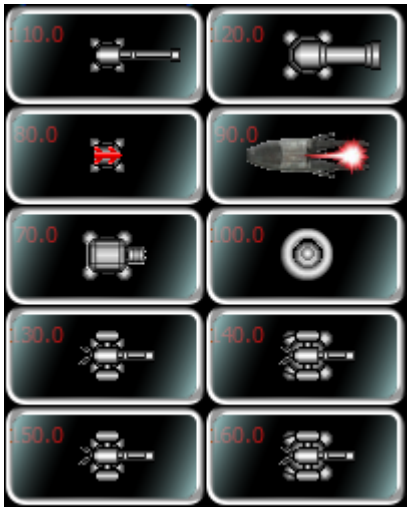
Chceme vytvořit hru pro vzdělávací účely pro budoucí programátory her. Z toho vyplívá, že návrh a implementace hry musí být jednoduchý, aby se v nich kdokoliv, kdo začíná s programováním, vyznal. Toho docílíme správným rozčleněním daného problému na jednotlivé sekce.

V prvé řadě si musíme říct, jak takový návrh řešení by měl vypadat. Naším úkolem je vytvořit hru Tower Defense. Tudíž si nejdříve musíme shrnout, co by měla všechno hra obsahovat a toho docílíme vytvořením konceptu hry, viz str. 22. Poté si vytvoříme diagram tříd a diagram použití, díky kterým získáme a ukážeme upřesnění o tom, jak by měla být hra konstruována. Poté se musíme rozhodnout o programovacím jazyce, ve kterém chceme tuto hru implementovat. Následně pak popsat implementační části programu. Ukázat zajímavé algoritmy a testy, aby viděli výsledky tohoto tutoriálu.

5.2 Stručný popis konceptu celé hry

Hlavním úkolem ve hře je zabránit nepřátelským jednotkám v postupu, jelikož se snaží odvést krystaly, které my nechceme, aby dostali. To je docíleno stavěním obrané linie pomocí obraných věží a zkoumáním technologií, které ty

K dispozici jsou základní a vyzkoumatelné věže. Každý typ má jiné vlastnosti a tím pádem jiné výhody a nevýhody čímž se každý typ hodí na jiného nepřátele. Ve hře je celkem 10 typů věží viz Obr. č. 7: kulometní, artilerie, raketová, plasmová, laserová, elektrická, ohnivá, mrazící, jedová, časová. Každá samostatná věž získává zkušenosti za zabití nepřátel a za určité množství zabití se jí zvýší úroveň. S vyšší úrovní se odemknou různá možná vylepšení, která v určitých směrech významně vylepší obranou schopnost věže.

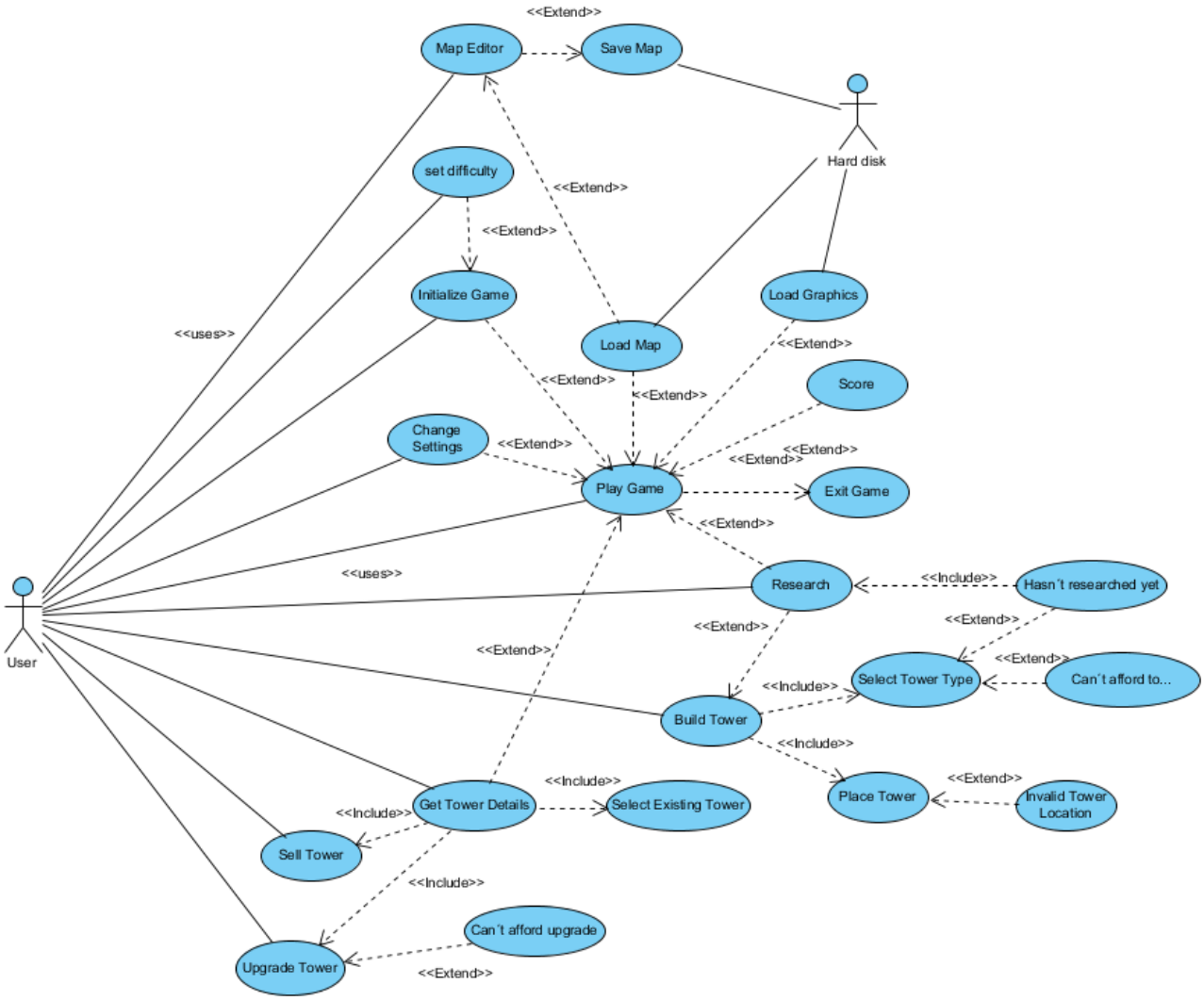


Obr. č. 7 – Seznam věží

Dále je zde výzkum, kde se vyzkoumávají technologie potřebné pro stavění pokročilých věží a které vylepšují vlastnosti některých typů věží. Každý výzkum je jinak drahý a spotřebuje jiné množství zkoumacích bodů, které získáváme za zabití nepřátelských jednotek s parametrem **Boss**. Typy výzkumu lze obecně rozdělit na 3 kategorie a to: první je „jednou vyzkoumat a hotovo“, druhou je „potřebné několikanásobné vyzkoumání a hotovo“ a poslední je „lze zkoumat do nekonečna“.

5.3 Formulace způsobu užití pomocí diagramu užití

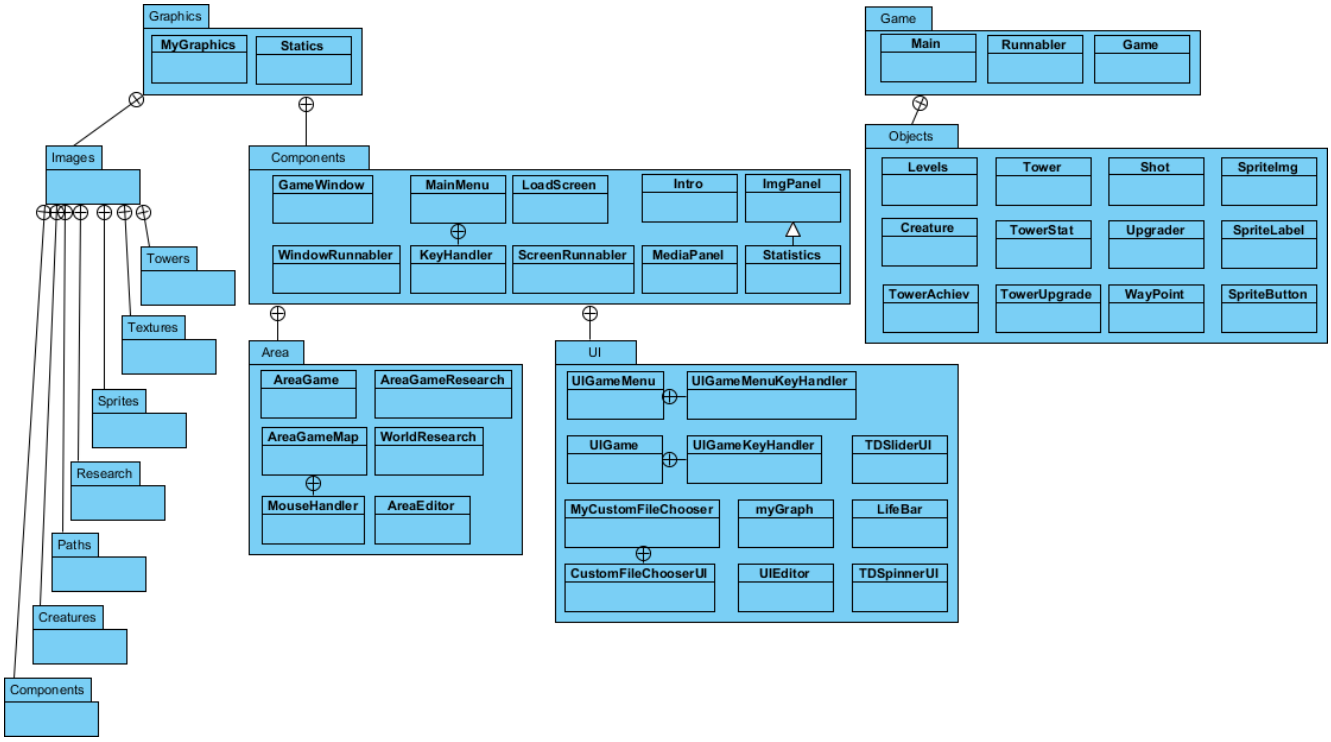
Po dokončení konceptu hry se musí vytvořit diagram užití, a to z toho důvodu, abychom viděli a věděli, jak bude uživatel s hrou pracovat. Je dobré nad tvorbou tohoto diagramu strávit nějaký čas a rozhodně ho nepřeskakovat, to by v budoucnu způsobilo problémy a zmatky. Na Obr. č. 8 je znázorněn diagram užití naší hry. Vidíme z něj, že disk pracuje s ukládáním a načítáním mapy a že se z něj načítá grafika. Uživatel naopak pracuje s map editorem, hraje hru, může nastavit obtížnost hry, změnit nastavení zobrazení, zkoumat výzkum a všemožně manipulovat s věžemi. Dále z něj vidíme, že stavění věží je omezené penězi a výzkumem.



Obr. č. 8 – Diagram užití

5.4 Objektový návrh hry pomocí diagramu tříd

Nyní když máme hotový koncept hry a diagram užití, můžeme přistoupit k tvorbě diagramu tříd. Tím si rozčleníme jednotlivé případy užití a koncept hry na třídy s konkrétními účely. Z Obr. č. 9 vidíme, že obrázky grafiky jsou uloženy ve strukturované složce Images, která má několik podsložek typy obrázků. Dále zde vidíme, že grafické komponenty jsou rozděleny do dvou skupin a to do Area, vykreslovací, a UI, ovládací. Skupina Area jsou grafické komponenty nejvyšší 4. úrovně GUI popsané dále, viz str. 26 a skupinu UI, která je na 3. úrovni v GUI. Dále ve složce Objects vidíme jednotlivé herní objekty, jako jsou věže, střely, nepřátelské jednotky a další.



Obr. č. 9 – Zjednodušený diagram Tříd

Kvůli velké složitosti tohoto diagramu a jednotlivých tříd, zde uvádím jen zjednodušenou verzi. Detailnější diagram tříd je k dispozici v příloze na CD.

6 Implementace TD

6.1 Výběr programovacího jazyka

Výhody Javy jsou dost lákavé a svádějí člověka k dojmu, že je nejlepší. To ale není pravda, neexistuje nejlepší programovací jazyk, každý jazyk je vhodný na něco jiného.[22] Takže pokud chceme vyvíjet aplikaci, musíme si nejdříve shrnout, co to bude za aplikaci a jaký je nejvhodnější programovací jazyk na danou aplikaci. To docílíme tím, že srovnáme popis aplikace s následující tabulkou programovacích jazyků.

Vhodný pro:	**Java**	**C++**	**C#**	**Visual Basic**	**Action Script**
Hry	Platformě nezávislé	Platformě závislé	Není ideální pro hry	NE	Málé webové hry
Programování	vysoko úrovňové	nízko úrovňové	vysoko úrovňové	nízko úrovňové	vysoko úrovňové
Webové použití	ANO, Podnikové aplikace	ANO, ale náročné ladění	ANO	NE, na to je VBS	ANO
Platformy Microsoft	NE	ANO	ANO	ANO	–
malé programy	NE	ANO	ANO	ANO	ANO
střední programy	ANO	ANO	ANO	NE	NE
velké programy	ANO	NE	ANO	NE	NE

Tabulka č. 5: Typy programovacích jazyků a jejich vhodnost

Z tabulky, která je složeninou několika webových stránek a mých osobních názorů, vidíme, že na naši hru TD, která slouží jako příklad programování, je nejlepší Java a to z důvodu nezávislosti na platformě, webového použití a jedná se o středně velkou aplikaci.

Zdrojové stránky tabulky č. 5:

http://www.dadajax.net/je-java-vhodna-pro-desktop/

http://www.zive.cz/clanky/java-vs-c---ktery-jazyk-zvolit/sc-3-a-104694/default.aspx

http://www.dreamincode.net/forums/topic/27087-advantagesdisadvantages-of-programming-languages/

[22] Je Java vhodná pro desktop? [online]. [cit. 2008-05-20]. Dostupné z: http://www.dadajax.net/je-java-vhodna-pro-desktop/

6.2 Implementace TD v Javě

Nyní se dostáváme k samotné implementaci TD. Základem této hry jsou následující třídy, které tvoří samu podstatu tohoto typu her. Všechny zdrojový kódy tříd lze prozkoumat v dokumentaci na CD.

6.2.1 Game

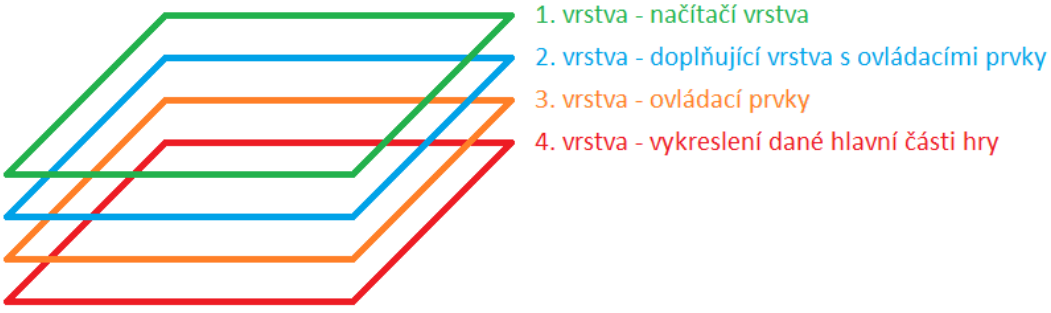
Hlavní třída, která se stará o chod hry. Běží na samostatném vlákně. Jsou zde uloženy herní objekty: nepřátelské jednotky, naše věže, střely a exploze, a to pomocí seznamů.

Nachází se zde hlavní **while cyklus**, ve kterém se počítají jednotlivé kroky hry, který se nachází v metodě **runGame**(). Na začátku tohoto cyklu se volá metoda **pause**(), ve které pokud je proměnná **pause** typu **boolean** nastavená na **TRUE** běží while cyklus, který pozastaví vlákno, na kterém tato hlavní metoda běží. V tomto hlavním cyklu se počítají veškeré akce všech herních objektů, kterými jsou: nepřátelské jednotky, naše věže, pohyb a detekce nárazů střel a exploze. Těsně před koncem každého cyklu se zašle informaci třídě, která se stará se o **GUI**, že nový krok je spočten a ta si následně vyžádá tyto nové informace, podle kterých vykreslí nový snímek.

Dále tato třída obsahuje metodu **NewGame()** se dvěma parametry, kterými jsou: **levelsValues**, která obsahují vlastnosti dané hry, jakož je např. obecný nárůst nepřátelských životů za kolo a mnoho dalších. Druhým parametrem je **loadedMap**, ve kterém jsou uloženy informace o mapě, které byly načteny ze souboru. Pomocí tohoto parametru se v této metodě zavolá metoda **createPath**(), která nám z této proměnné vytáhne data o cestě, po které kráčí nepřátelské jednotky.

6.2.2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní zkráceně **GUI**, je rozděleno do několika tříd. Všechny hlavní části hry, kterými jsou: hlavní menu, hrací plocha a map editor jsou tvořeny pomocí 3 až 4 vrstev viz Obr. č. 10. Každá z těchto vrstev představuje jednu podtřídu typu **JPanel**, ze které dědí její vlastnosti a funkce pro vykreslení. Ty mají nastavené pozadí jako průhledné a tím tvoří dojem, že se jedná o jeden vizuální panel. Hlavním důvodem takového to rozdělení je zpřehlednit kód a rozdělit funkce do tříd podle kategorií. Například pro hrací plochu je rozdělení následovné: 1. – Načítací obrazovka, 2. – Menu, 3. – Ovládací prvky (stavění věží, nastavování věží atd.), 4. – Vykreslení hrací plochy (mapa s věžemi, nepřátelskými jednotkami, střelami a explozemi).



Obr. č. 10 – ukázka vrstveného způsobu zobrazení GUI

6.2.3 MyGraphics

Velice důležitá třída, jelikož v sobě uchovává všechny grafický obrázky načtený ze souborů. Tyto obrázky jsou zde uloženy pomocí hashovací tabulky **HashMap**<String, ImageIcon>, což je struktura postavená nad polem, jež slouží k ukládání dvojic klíč–hodnota, zde jako textový řetězec–obrázek a která kombinuje výhody seznamů a polí, umožňuje vyhledat prvek pomocí průměrně konstantního počtu operací a nevyžaduje velké množství dodatečné (nevyužité) paměti.

Při vykreslování herních objektů si ve vykreslovacích metodách patřičný objekt zavolá funkci **MyGraphics.get().getImage**(název obrázku), který vrátí proměnnou typu **ImageIcon**, která obsahuje data obrázku. Je to z důvodu šetření místem v paměti, kdyby totiž tento **ImageIcon** byl jako proměnná uvnitř herního objektu tak by byl v paměti uložen N krát takže např. při 50 nepřátelských jednotek by zaplňoval místo v paměti 50x.

6.2.4 Věž

Jedná se o třídu, ve které jsou uloženy informace o postavené věži. Obsahuje mnoho proměnných, které nám říkají o jejím aktuálním stavu, např. kdy může znovu střílet, jakou má zaměřenou nepřátelskou jednotu, nastavení zaměřovacího systému a spoustu dalších parametrů.

Hlavními metodami jsou **canFire**(). Na začátku metody pokud nemá zaměřenou nepřátelskou jednotu tak zavolá metodu **targetTheEnemy**(), která zaměří podle nastaveného zaměřovacího systému nejvhodnější cíl. Poté v hlavní metodě **canFire**() pokud již věž má nastavený cíl palby, tak na základě několika parametrů a proměnných s aktuálním stavem věže určí, zda věž může či nemůže již střílet. V případě že může, vytvoří herní objekt typu **Střela** a předá jí některé své parametry, jako je např. účinnost, dolet střely a případně speciální efekty jako jsou např. plošný výbuch, zpomalení, zapálení a mnoho dalších, které si můžeme vymyslet a případně dopsat.

Dále tato třída obsahuje dvě vykreslovací metody, a to **paint**() a **drawRangeInfo**(). První vykresluje věž jako takovou a druhá vykresluje barevně oblasti dosahu palby dle priorit.

6.2.5 Nepřátelská jednotka

Třída, ve které jsou uloženy informace o nepřátelské jednotce. Opět obsahuje mnoho proměnných s aktuálním stavem jednotky, mezi nejdůležitější patří proměnné typu **boolean** s vlastnostmi jednotky. Těch je 6: **boss**, **hard**, **fast**, **air**, **shield** a **phased.** Kombinace těchto proměnných nám určí typ jednotky, celkem tedy 2^6 jednotek, ale obrázek jednotky ovlivňují pouze 4 tyto vlastnosti, takže nám stačí pouze 16 různých obrázků jednotek viz Obr. č. 11.

Obr. č. 11 – obrázky jednotek

Hlavní metody jsou **move**(), která se volá z **Game.runGame**(), a slouží ke správnému posunu po cestě během kroku hry. Dále se v ní upraví proměnné s názvy **time**, ve kterých je uloženo, jak dlouho bude jaký efekt ještě působit. Tyto proměnné se nastavují pomocí metody **activeteEffect**(), kde se nastavuje čas a hodnota účinnosti jednotlivých efektů a ta se volá ze třídy **Shot**.**shot**(), viz **Střela**. Následující metoda **hit**() se stará o to, že v případě zásahu odečte životy a případně nejdříve štít, pokud ho má, z jednotky podle jejích vlastností, kupříkladu vlastnost **Hard** změní hodnotu účinnosti střely na 0.8 původní fyzické nebo 0.95 energetické účinnosti. Poslední hlavní metodou je **draw**(), která vykresluje jednotku plus u vrtulníků rotující vrtuli a případné efekty které na jednotce jsou.

6.2.6 Střela

Třída, ve které jsou uloženy informace o střele. Opět obsahuje mnoho proměnných s aktuálním stavem střely, mezi nejdůležitější patří účinnost, dolet, startovní a cílové souřadnice, typ střely (fyzická/energetická), zda se jedná o projektil nebo souvislí paprsek, animace střely uložená pomocí třídy **SpriteImg**.

Obsahuje 3 hlavní metody a to: metoda **move**(),která se volá z **Game.runGame**() stejně jako u nepřátelských jednotek a slouží ke správnému posunu střely a po posunutí z detekuje pomocí metody **shot**() zda nezasáhla nějakou jednotku. Poslední metoda **draw**(), se stará o správné vykreslení střely a její animaci

6.2.7 Map Editor

Součástí implementace je i editor map, který je tvořen třídami **AreaEditor** a **UIEditor.** Editor funguje na principu vrstev, nyní omezeno na 8 vrstev, kde sudý vrstvy jsou posunutý o polovinu šířky stejně tak výšky jedné mapové buňky která má šířku i výšku 40px. Editace je rozdělena na dvě části: levým tlačítkem se kreslí cesta s tím, že nám editor ukazuje, kam cestu můžeme z aktuální buňky vést. Pravým tlačítkem pak kreslíme povrch mapy. Nyní je možné

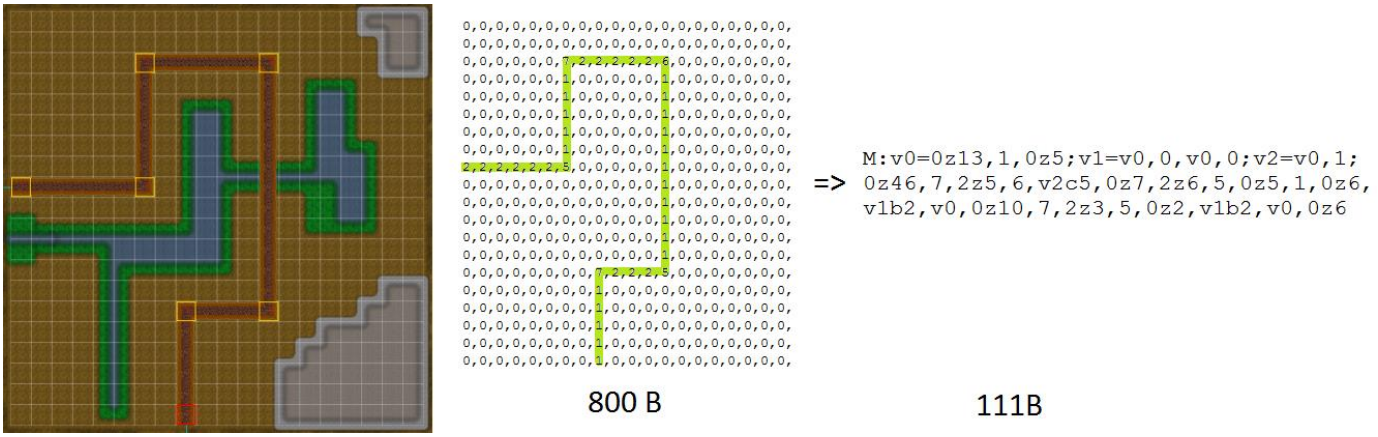
kreslit vodu, písek, trávu, nízkou skálu a vysokou skálu, lze snadno rozšiřovat o další, jelikož se typy povrchu se ukládají jako čísla od 0 do 5.

Cesta se při editaci mapy ukládá do proměnných v podobě seznamu úseček, ve hře pak cesta je převedená na objekty typu **WayPoint**, která obsahuje pouze proměnné x, y, směr. Vrstvené pozadí mapy se při editaci ukládá do čtyřrozměrného pole ve tvaru [4][2][x][y], kde 4 značí počet úrovní, 2 značí, zda se jedná o lichou či sudou vrstvu, x a y značí šířku a výšku mapy. Toto velké pole s povrchem mapy a seznam s cestou se map ukládá do souboru pomocí kompresního algoritmu, který je popsán na str.29.

6.3 Popsání nejdůležitějších a nejzajímavějších algoritmů použitých při vývoji hry

6.3.1 Ukládací kompresní algoritmus map

U her je komprese ukládaných dat na disk velice důležitá, kdyby nebylo komprese tak mapy a ukládání rozehraných her u větších her poněkud náročné na paměť disku. Z tohoto důvodu jsem vytvořil jednoduchou kompresy dat jako ukázku pro zájemce, NEJEDNÁ se o nejlepší či nejefektivnější algoritmus, ale své splní, jak je vidět na Obr. č. 12.



Obr. č. 12 – ukázka komprese mapy

Tento algoritmus funguje následně: Nejprve po sobě jdoucí opakující se čísla zkrátí na výraz typu ČÍSLOzN, kde N je počet po sobě jdoucím opakování čísla. Následně pomocí regulárních výrazů hledá opakující se výrazy, v tomto případě čísla oddělené čárkou a zkrácené výrazy opakujících se po sobě jsoucích čísel. Nalezené opakující se řetězce si ukládá do HashMap <String, Integer[2]>, kde String je daný řetězec a v číselném poli jsou uloženy dvě hodnoty a to počet opakování a celkově ušetřené místo, které by se získalo zkrácením daného výrazu. To se spočítá pomocí následujícího výrazu:

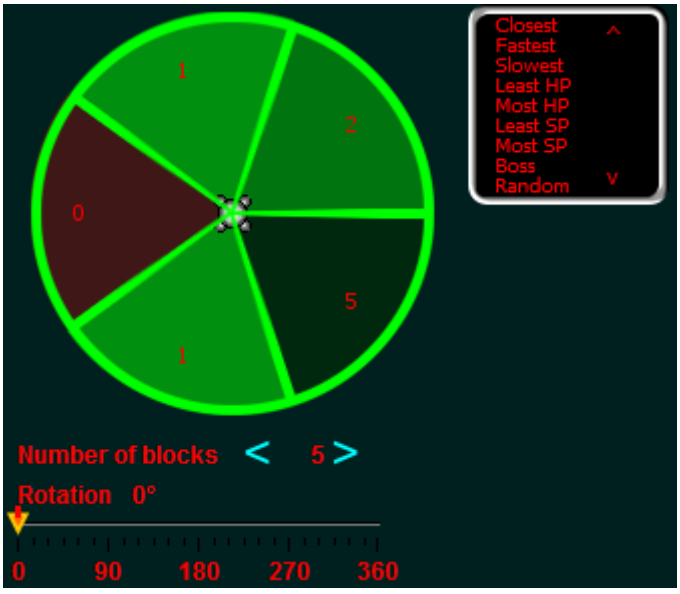
$$\text{ušetřené místo} = (\text{šířka výrazu} - \text{šířka ID}) * \text{počet opakování} - (\text{šířka výrazu} + \text{šířka ID} + 2)$$

Nyní se vybere nejvhodnější řetězec, ten se nahradí zkráceným výrazem a uloží se do seznamu použitých řetězců. Nyní se celý proces s regulárními výrazy opakuje od začátku,

jelikož zkrácením mohlo dojít ke změně hodnot pro ostatní nalezené řetězce plus přibydou nový, jak je vidět na Obr. č. 12, kde v1 obsahuje v0.

6.3.2 Zaměřovací systém věží

Jak jsem se již zmínil, věže mají zaměřovací systém. Nejprve se nastaví oblasti palby a jejich priority značící, v kterých oblastech se budou nepřátelské jednotky nejprve hledat viz **Obr. č. 13** nalevo. Priority jsou následovné: 0 je vypnutá oblast, 1 nevyšší priorita a nejvyšší číslo, kterým je počet oblastí, je nejnižší priorita. Dále se nastaví, dle kterých vlastností nepřátelských jednotek bude zaměřovací systém nejprve pátrat pomocí prioritního seznamu viz **Obr. č. 13** napravo, ve kterém platí: čím níže v seznamu tím má vlastnost větší prioritu. Pokud je zaměřovací systém nastavený tak funguje následně: Nejdříve prohledá oblasti s nejvyšší prioritou, v případě nenalezení jednotky pokračuje na oblast s nižší prioritou. Pokud ale nalezne alespoň jednu nepřátelskou jednotku, tak ji uloží do dočasné proměnné a začne danou oblast prohledávat dle seznamu prioritních vlastností, kde porovnává vždy danou vlastnost dočasně uložené jednotky s následujícími. V případě shody porovnávané vlastnosti pokračuje na další, dokud se nenajde lišící se hodnotu. Pokud se liší správným směrem: menší nebo větší, záleží na typu, tak tehdy se uloží nově nalezená jednotka do dočasné proměnné, nebo pokud narazí na vlastnost **Random**, tak ji vybere náhodně z těchto dvou porovnávaných jednotek. Takto to pokračuje v aktuální oblasti, dokud tímto způsobem neprojede celé pole s nepřátelskými jednotkami. Výslednou nejideálnější jednotku si uloží do proměnné **targetcreature**.



Obr. č. 13 – nalevo nastavení oblastí palby a napravo seznam s prioritními vlastnostmi

7 Programátorské a uživatelské testy

V konečné fázi implementace je nutné začít s testováním, aby se odhalily případné nedostatky nebo bugy. Testování aplikací bývá někdy velmi náročné, zvláště u her, a musí se rozdělit na dvě části.

7.1 Programátorské testy

Jedná se o testování funkčnosti aplikace samotnými vývojáři. Při dokončení třídy nebo skupiny tříd mezi sebou souvisejících je zapotřebí zjistit zda fungují správně tak, jak máme popsaný v návrhu a diagramech. V této implementaci hry je zapotřebí provést hned několik testování.

Načítání a ukládání map ze souborů

Protože používáme kompresní algoritmus pro mapy, je jejich testování rozděleno na dvě části: vizuální a kontrola komprimované mapy uložené v souboru.

Vizuální část je prováděna v editoru mapy ve spuštěné aplikaci. Testujeme, zda mapa po uložení a opětovném načtení vypadá v editoru map stejně. Ne vždy se totiž hned na poprvé podaří napsat správně opačný algoritmus k původnímu, a proto tímto testováním můžeme odhalit případné chyby a rozdíly mezi kompresním a dekompresním algoritmem, který jsou použity při ukládání map z a do souborů. Výsledkem tohoto testování prováděného v naší implementaci hry bylo zjištění, že námi vytvořený algoritmy komprese a dekomprese fungují totožně.

Kontrola komprimované mapy uložené v souboru je část velice náročná. Jediný způsob kontroly je ruční. Abychom se ujistily, že náš kompresní algoritmus funguje tak jak byl původně navržen, musíme ve vývojovém prostředí tento algoritmus testovat řádek po řádku a sepisovat si např. do souboru nebo na papír výsledky jednotlivých kroků tak aby bylo vidět, co jaký krok udělal s výstupním tokem dat. Takto sepsané kroky pak porovnáme s původním návrhem algoritmu, abychom viděli, zda vše souhlasí. Nyní provedeme pomocí tohoto sepsaného postupu algoritmu ruční kompresy dat a měli bychom se dostat k výsledku shodným s uloženou mapou v souboru.

Těmito dvěma způsoby testování jsem testoval více jak 200 různých map a neodhalil jsem žádné chyby. To ale definitivně neříká, že algoritmus je 100% funkční, jelikož jde vytvořit mnohem více různých map, které zatím nebyly testovány, a může se stát, že u některých kombinací bude tento algoritmus neefektivní nebo dokonce nefunkční.

Chování nepřátelských jednotek

Toto testování je velice jednoduché ale i tak důležité. Musíme otestovat, zda jednotky chodí správně po cestě mapy, dále zda při zásahu se jim odečtou životy a jestli speciální efekty věží se na ně správně aplikují. Toho testování provádíme pomocí ladícího nástroje tzv.

Debugger[23], který je součástí Netbeans[24] a umožňuje nám odkrokovat spuštěný program. Výsledkem bylo zjištění, že chování nepřátelských jednotek funguje správně.

Chování věží

Testování chování věží je obdobné jako chování nepřátelských jednotek. Pomocí ladícího nástroje testujeme po krocích všechny metody, obzvláště pak připravenost k palbě plus palba samotná, správné natáčení věže a zaměřovací systém, který je nejdůležitější částí chování věže. Výsledkem bylo zjištění špatného chování natáčení věže, kdy se v určitém úhlu věž začala otáčet opačným směrem, tato chyba byla později díky tomuto testování opravena.

Chování střel

Toto testování chování střel je též obdobné jako testování věží a nepřátelských jednotek, jelikož jsou to všechno herní objekty. Zde testujeme, zda při střelbě střela správně převezme vlastnosti věže a její pohyb v čase a strefení do nepřátelské jednotky. Výsledkem bylo zjištění, že chování střel funguje správně.

7.2 Uživatelské testy

Jde o testování z pohledu koncového uživatele, pro kterého byla aplikace navržena, v našem případě pro hráče. Některé z těchto typů testů provádí stále ještě programátoři, ale ne vždy se dokáže programátor vnést do role uživatele a je proto nutné přizvat k tomuto testování lidi, kteří nemají nic společného s programováním.

Grafické uživatelské rozhraní

Veledůležitou částí uživatelského testování je testování uživatelského rozhraní zkráceně GUI. Pokud se programátor nedokáže vžít do role uživatele, tak toto testování probíhá formou konzultací s uživatelem, díky kterému zjišťujeme případné nedostatky nebo chyby v GUI. Výsledkem bylo zjištění, že sice GUI funguje správně, ale není úplně uživatelsky příjemné a některé texty jsou hůře čitelné.

Správné vykreslování efektů

Toto testování spočívá ve vizuální kontrole, kdy při spuštěné aplikaci uživatelé sledují vykreslování efektů a případné chyby se hlásí grafikovy nebo programátorovy. Jedná se o zdlouhavé testování, protože převážná většina her je založena právě na těchto efektech, kterými se snažíme upoutat budoucí uživatele. Výsledkem tohoto testování bylo zjištění, že efekty se vykreslují správně.

Průběh hry

Jedná se o testování hry jako celku, především její hratelné části a funguje tak jak byla navržena. Testuje se např. zábavnost hry a odhalují se zde chyby, které vznikají až při hraní a nejsou zjistitelné z programátorských testů. Výsledkem tohoto testování bylo zjištění, že hra funguje tak jak byla popsána v konceptu.

[23] Debugger. [online]. [cit. 2014-02-10]. Dostupné z http://cs.wikipedia.org/wiki/Debugger
[24] Netbeans debugger. [online]. [cit. 2015-02-25]. Dostupné z https://netbeans.org/features/java/debugger.html

8 Návrh pro budoucí řešení

8.1 Uživatelské připomínky k vylepšení této hry

Vylepšení a zpřehlednění Grafické uživatelské rozhraní

Jak bylo zmíněno v uživatelských testech, viz str. 32, byly zde připomínky ke GUI. Není úplně dobře optimalizované a příjemné pro uživatele, jelikož takovéto optimalizování vyžaduje značnou praxi, porozumění požadavků hráče na GUI a vžití se do nich.
V programování her jsem sám vcelku nováčkem, ale zajímám se o něj, a proto bych se nadále chtěl rozvíjet v dané oblasti a naučit se jak GUI správně optimalizovat.

Hezčí a Lepší efekty

Mezi další uživatelské připomínky spadá připomínka o kvalitě efektů ve hře. Kvalita efektů použitých ve hře je poněkud zastaralá a nedá se srovnat s kvalitně napsanými hry v prodeji. Tudíž bylo by vhodné efekty do budoucna vylepšit. *It's all about the special effects.*[25]

3D zpracování

Jako další připomínkou je návrh 3D zpracování dané hry. V dnešní době je velké množství nejhranějších her zpracováno v 3D. Je to z důvodu, že 3D zpracování her nás dostává blíže k realistickým vjemům a požitkům z jejich hraní.

[25] Kirkpatrick, Chris. Special effects. [online]. [cit. 2015-03-10]. Dostupné z http://www.brainyquote.com/quotes/quotes/c/chriskirkp421537.html

8.2 Návrhy pro vylepšení této a i jiných verzí této hry

Možností pro rozšíření her je prakticky omezené jen naší představivostí, proto zde uvedu jen pár těch nejzajímavějších, které by dali této hře nový rozměr.

Zlepšení výkonu hry a 3D zpracování

Jedním z návrhů pro vylepšení této hry jak bylo již zmíněno v uživatelských připomínkách, viz str. 33. je převedení grafiky do 3D a zlepšení výkonu hry. K tomu bychom mohli využít knihovnu Javy processing.core[26], která má mnoho zajímavých rozšiřujících tříd, které se hodí jak pro zefektivnění výkonu aplikací tak má i třídy pro tvorbu 3D aplikací. Existuje velice zajímavý tutoriál k této knihovně a lze ho najít na webové stránce http://natureofcode.com/book/.

Dějový příběh a ukládání rozehrané hry

Další možným vylepšením je vytvořit nebo rozšířit dějový příběh a ukládání hry v průběhu rozehřátého kola. Většina prodávaných her má napsaný kvalitní příběh, který má uživatele zaujmout, protože hry bez příběhu nejsou záživné. Dala by se např. vytvořit mapa světa s misemi a dějovým scénářem. V takovém případě by se pak hodilo dotvořit ukládání rozehrané hry, aby uživatelé nemuseli pokaždé začínat od začátku.

Multiplayer

Pravděpodobně asi nejzajímavějším vylepšením by bylo napsat pro hru multiplayer – možnost více hráčů. V dnešní době hodně her multiplayer má. Je to z důvodu, že interakce s jinými uživateli během hraní dodává hře další nový rozměr zábavy.

[26] Knihovna Javy processing.core. [online]. [cit. 2015-03-12]. Dostupné z https://www.processing.org/

9 Závěr

V této práci jsem se pokusil navrhnout a vytvořit hru v programovacím jazyce Java, která by sloužila jako vodítko pro budoucí programátory, kteří by se chtěli vydat cestou programování her. K vytvoření této práce mě vedla myšlenka vyzkoušení si napsání hry. S velikým údivem jsem zjistil jak moc je to náročné, jak z hlediska návrhu tak samotné implementace, musím říct, že programátoři, kteří programují hry, to mají opravdu dost těžké. Na závěr bych chtěl shrnout, co všechno jsme se v této práci dozvěděli:

V první teoretické části práce byla zmapována historie TD a jeho vývoje, kde jsme se dozvěděli, jaké byli první hry tohoto typu a jak se s rostoucí popularitou začali vyvíjet do dnešních her. Dále jsme si řekli něco o problematice vývoje her a to jak obecný problémy tak i pár konkrétních které nastali během mé implementace.

Druhá část práce se zaměřuje na popis obecného návrhu řešení TD. Zde jsme si ukázali jak navrhnout pomocí UML diagramů základní strukturu dané hry.

Ve třetí části práce jsme se zaměřili na implementaci našeho návrhu v jazyce Java. Nedříve jsme si zhodnotili, v jakém jazyce bude nejlepší pracovat. Poté jsme probrali nejzákladnější třídy implementované hry a ukázali si pár zajímavých algoritmů, který jsou v implementaci hry použity.

Poslední část práce se zabývá testováním dané implementace TD, což jsme si ukázali na několika programátorských a uživatelských testech a jejich vyhodnocení. Dále jsme si ukázali několik případných vylepšení do budoucna.

Téma vývoje her je dnes natolik rozsáhlé a zároveň měnící, že uvedené téma popisuje jen malou část o programování her a není tímto možné vystihnout všechny aspekty dotýkajících se programování. Vhodným námětem pro další práce v této oblasti může být srovnání programovacích jazyků, technologií používaných při tvorbě her, detailnější problematika vývoje atd.

10 Seznam použitých zdrojů

Seznam literatury

1. HORSTMAN, CAY, S., CORNELL, G. Core *Java 2 Volume I - Fundamentals Eight Edition. Prentice Hall PTR, 2008.*.
2. HORSTMAN, CAY, S., CORNELL, G. *Core Java 2 Volume II - Advanced Features Eight Edition. Prentice Hall PTR, 2008.*.

Elektronická podoba:

1. Historie Tower Defense. [online]. [cit. 2010-12-29]. Dostupné z http://towerdefenseheaven.com/content/history-tower-defense-games
2. Historie Tower Defense. [online]. [cit. 2010-06-10]. Dostupné z http://mygaming.co.za/news/features/6341-tower-defense-a-brief-history.html
3. Steam, the ultimate entertainment platform. [online]. [cit. 2014-12-10]. Dostupné z http://store.steampowered.com/
4. Herním průmyslu ČR. [online]. [cit. 2015-02-10]. Dostupné z http://cs.wikipedia.org/wiki/Český_videoherní_průmysl
5. Vlákna a jejich synchronizace. [online]. [cit. 2015-02-15]. Dostupné z http://www.algoritmy.net/article/39287/Vlakna-22
6. Program Visual Paradigm. [online]. [cit. 2014-12-19]. Dostupné z http://www.visual-paradigm.com/
7. UML diagramy. [online]. [cit. 2014-12-19]. Dostupné z http://en.wikipedia.org/wiki/Unified_Modeling_Language
8. Oracle. [online]. [cit. 2015-02-13]. Dostupné z http://www.oracle.com/index.html
9. Program Netbeans. [online]. [cit. 2014-12-19]. Dostupné z https://netbeans.org/
10. Pluginy do Netbeans. [online]. [cit. 2014-12-19]. Dostupné z http://plugins.netbeans.org/PluginPortal/
11. Eclipse. [online]. [cit. 2014-12-19]. Dostupné z https://eclipse.org/
12. Program Corel PHOTO-PAINT X6. [online]. [cit. 2015-02-15]. Dostupné z http://www.coreldraw.com/cz/
13. Grafický formát. [online]. [cit. 2015-02-15]. Dostupné z http://en.wikipedia.org/wiki/Image_file_formats
14. Srovnání rastrových grafických editorů dle dostupnosti. [online]. [cit. 2015-02-15]. Dostupné z http://en.wikipedia.org/wiki/Comparison_of_raster_graphics_editors
15. Program Texture Maker. [online]. [cit. 2015-02-17]. Dostupné z http://www.texturemaker.com/download.php

16. Texture. [online]. [cit. 2015-02-17]. Dostupné z http://en.wikipedia.org/wiki/Texture_(visual_arts)
17. Typy licencí. [online]. [cit. 2015-02-17]. Dostupné z http://www.texturemaker.com/help/misc/features.htm
18. Program particleIllusion. [online]. [cit. 2015-02-19]. Dostupné z http://www.wondertouch.com/index_2.asp
19. Particle Systém. [online]. [cit. 2015-02-24]. http://en.wikipedia.org/wiki/Particle_system
20. Processing Core v Javě. [online]. [cit. 2015-02-19]. https://www.processing.org/
21. Particle Systém v Javě. [online]. [cit. 2015-02-19]. http://natureofcode.com/book/chapter-4-particle-systems/
22. Sprite. [online]. [cit. 2015-04-17]. http://en.wikipedia.org/wiki/Sprite_%28computer_graphics%29
23. Debugger. [online]. [cit. 2014-02-10]. Dostupné z http://cs.wikipedia.org/wiki/Debugger
24. Netbeans debugger. [online]. [cit. 2015-02-25]. Dostupné z https://netbeans.org/features/java/debugger.html
25. Kirkpatrick, Chris. Special effects. [online]. [cit. 2015-03-10]. Dostupné z http://www.brainyquote.com/quotes/quotes/c/chriskirkp421537.html
26. Knihovna Javy processing.core. [online]. [cit. 2015-03-12]. Dostupné z https://www.processing.org/
27. Webové stránky s různými verzemi hry Tower Defense [online]. [cit. 2014-12-08]. Dostupné z http://www.towerdefence.cz/
28. Killer Game Programming in Java [online]. [cit. 2014-12-09]. Dostupné z http://fivedots.coe.psu.ac.th/~ad/jg/
29. Basic information about Desktop Tower Defense [online]. [cit. 2015-01-16]. Dostupné z http://en.wikipedia.org/wiki/Desktop_Tower_Defense
30. Video game development. [online]. [cit. 2015-02-23]. Dostupné z: http://en.wikipedia.org/wiki/Video_game_development
31. Game programming. [online]. [cit. 2015-03-10]. Dostupné z: http://en.wikipedia.org/wiki/Game_programming
32. THE GAME PRODUCTION PIPELINE: CONCEPT TO COMPLETION. [online]. [cit. 2006-03-15]. Dostupné z: http://www.ign.com/articles/2006/03/16/the-game-production-pipeline-concept-to-completion
33. What Went Wrong? A Survey of Problems in Game Development. [online]. [cit. 2009-10-19]. Dostupné z: http://www.ibiblio.org/winget/2009/10/what-went-wrong-a-survey-of-problems-in-game-development/

34. The Process of Game Development. [online]. [cit. 2015-02-17]. Dostupné z: http://www.pearsonhighered.com/samplechapter/0672326922.pdf
35. Algoritmus. [online]. [cit. 2015-04-22]. Dostupné z: http://cs.wikipedia.org/wiki/Algoritmus
36. Java (programovací jazyk). [online]. [cit. 2015-04-15]. Dostupné z: http://cs.wikipedia.org/wiki/Java_%28programovac%C3%AD_jazyk%29
37. Java Advantages and Disadvantages. [online]. [cit. 2008-08-30]. Dostupné z: http://java-work.blogspot.cz/2008/08/java-advantages-and-disadvantages.html
38. Je Java vhodná pro desktop? [online]. [cit. 2008-05-20]. Dostupné z: http://www.dadajax.net/je-java-vhodna-pro-desktop/
39. Java vs. C#. [online]. [cit. 2002-01-19]. http://www.zive.cz/clanky/java-vs-c---ktery-jazyk-zvolit/sc-3-a-104694/default.aspx
40. Advantages/Disadvantages Of Programming Languages. [online]. [cit. 2007-04-24]. Dostupné z: http://www.dreamincode.net/forums/topic/27087-advantagesdisadvantages-of-programming-languages/
41. Terrain Texturing, Foliage and Collision. [online]. [cit. 2009-10-05]. Dostupné z: https://daggerxl.wordpress.com/2009/10/05/terrain-texturing-foliage-and-collision/
42. Free Icons. [online]. [cit. 2014-12-20]. Dostupné z: http://icons.mysitemyway.com/
43. Free images of army. [online]. [cit. 2015-01-10]. Dostupné z: http://www.juniorgeneral.org/load.php?Period=10

11 Seznam obrázků a tabulek

12 Přílohy

CD nosič

Součástí bakalářské práce je CD obsahující:

- Elektronickou podobu textu ve formátu PDF
- Projektové složky se zdrojovými kódy aplikace Tower Defense
- Spustitelný .jar soubor
- Manuál ke hře