VŠB - Technická Univerzita Ostrava
Fakulta Elektrotechniky a Informatiky
Katedra informatiky

Modul pro 3D herní plochu hry Carcassonne

3D Engine for Game Carcassonne

2015 Jan Frydrych

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jan Frydrych**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Modul pro 3D herní plochu hry Carcassonne
3D Engine for Game Carcassonne

Zásady pro vypracování:

Cílem je vytvořit 3D náhled na herní plochu hry Carcassonne v jazyce Java (za podpory 3D knihoven), kde plocha a jednotlivé figurky ožijí a budou se pohybovat dle předefinovaných pravidel.

Vlastnosti programu:
1. Rozložení herní plochy a figurek bude přebíráno z již naimplementované verze hry Carcassonne.
2. 3D pohled na herní plochu umístěnou na rovinou desku s možností natáčení kamery.
3. Pohyb figurek po herních oblastech, které jsou jimi obsazeny (cesty, města, louky, ...).
4. Dynamika krajinných prvků (růst obilí, sopka, řeky, ...).

Práce bude obsahovat:
1. Přehled dostupných knihoven pro tvorbu 3D grafiky v jazyce Java.
2. Implementaci výše popsaného programu.
3. Programátorskou dokumentaci řešení s využitím diagramů jazyka UML.

Seznam doporučené odborné literatury:

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (Gang of Four): Návrh programů pomocí vzorů. Grada. Praha 2003. ISBN 8024703025
[2] DARWIN, Ian F. Java cookbook. 2nd ed. Sebastopol, CA: O'Reilly, c2004, xxiv, 829 p. ISBN 05-960-0701-9. Dostupné z: http://it-ebooks.info/book/2249/

Dále podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Ježek, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Místopřísežné prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2015

.....

podpis studenta

Rád bych poděkoval svému vedoucímu Ing. Davidu Ježkovi, Ph.D. za odbornou pomoc, cenné rady a trpělivost při vytváření této bakalářské práce.

Abstrakt

Cílem mé bakalářské práce je vytvořit 3D náhled na herní plochu hry Carcassonne. 3D náhled je implementovaný v jazyce Java, za podpory knihovny LWJGL, která poskytuje přístup k funkcím z grafické knihovny OpenGL. Práce je rozdělena na několik částí. V první části se věnuji popisu knihoven pro podporu tvorby grafiky v jazyce Java. Další část obsahuje zmínku o tvorbě 3D modelů. Poslední část se věnuje tvorbě samotného programu, problémům které při něm vznikly a dokumentaci programu.
Má práce navazuje na existující bakalářkou práci, která řeší logiku a pravidla hry.

Klíčová slova: Bakalářská práce, hra Carcassonne, 3D grafika, knihovny a moduly pro jazyk Java, OpenGL, LWJGL, Blender

Abstract

The aim of my thesis is to create a 3D perspective on the game play area Carcassonne. 3D preview is implemented in Java, with the support of LWJGL library that provides access to the features of graphics library OpenGL. The work is divided into several parts. In the first part I describe libraries to support the creation of graphics in Java. Another section contains a reference to the creation of 3D models. The last part is dedicated to creating the program itself, the problems that occurred, and program documentation.
My work builds on existing bachelor thesis, which manages game logic and rules.

Key worlds: Bachelor thesis, game Carcassonne, 3D graphics, libraries and engines for Java programing language, OpenGL, LWJGL, Blender

Seznam použitých zkratek

API	Application Programming Interfaces
BSD	Berkeley Software Distribution
GLU	OpenGL Utility Library
IDE	Integrated Development Environment
JNI	Java Native Interface
JOAL	Java OpenAL - Audio Library
JOCL	Java OpenCL - Computing Language
JOGL	Java OpenGL - Graphics Library
JVM	Java Virtual Machine
RMI	Remote Method Invocation
SDK	Software development kit

OpenGL	Open Graphics Library
OpenGL ES	OpenGL for Embedded Systems pro vestavěné systémy, mobilní telefony, tablety, herní konzole

Obsah

1 Úvod ... 1
2 OpenGL ... 2
2.1 Implementace ... 2
2.2 Struktura OpenGL ... 2
2.3 Historie ... 2
3 Počítačová 3D grafika ... 3
3.1 Knihovny pro tvorbu 3D grafiky v jazyce Java ... 4
3.2 Základní přehled ... 4
3.3 Knihovny ... 5
3.3.1 GL4Java ... 5
3.3.2 Java 3D ... 5
3.3.3 Java OpenGL (JOGL) ... 5
3.3.4 libGDX ... 6
3.3.5 LWJGL ... 7
3.4 Moduly ... 9
3.4.1 3DzzD ... 9
3.4.2 Ardor3D ... 9
3.4.3 Auriga3D ... 9
3.4.4 Bindenlicht ... 10
3.4.5 Env3D ... 10
3.4.6 Jake2 ... 10
3.4.7 Java is Doomed ... 10
3.4.8 jMonkeyEngine ... 11
3.4.9 jPCT ... 12
3.4.10 ogre4j ... 12
3.4.11 Xith3D ... 13
3.5 Ostatní ... 13
3.5.1 Jzy3D ... 13
3.5.2 Processing ... 13
4 Carcassonne On-line ... 14
4.1 Java RMI ... 14
4.2 Napojení 3D aplikace na existující hru ... 14
5 Modely pro hru ... 15
5.1 Software pro tvorbu modelů ... 15
5.1.1 Wings 3D ... 15

5.1.2 Blender 15
5.1.3 Zhodnocení 16
6 Implementace 17
6.1 Příprava 17
6.1.1 Volba 3D knihovny 17
6.1.2 Volba vývojového prostředí 17
6.2 Základ tvorby 3D programu 17
6.3 Modely, jejich nahrávání a vykreslování 18
6.3.1 Základ je trojúhelník 18
6.3.2 Display List 19
6.3.3 Vertex Arrays 19
6.3.4 Model - obj soubor 20
6.3.5 Třída OBJLoader 20
6.3.6 Vykreslování 20
6.4 Model 3D kartiček 21
6.5 Přikládání kartiček a figurek 22
6.5.1 Systém souřadnic v LWJGL 22
6.5.2 Pokládání figurek 22
6.6 Animace a pohyb figurek 23
6.6.1 Systém pohybu figurek 24
6.6.2 Seznam bodů 24
6.6.3 Konec seznamu 24
6.6.4 Obilí na loukách 25
6.7 Popis a ovládání programu 25
6.8 Návaznost na již existující bakalářskou práci 26
7 Závěr 27
8 Seznam použité literatury 28
9 Přílohy 29

Seznam obrázků

Obrázek 1: OpenGL logo 2
Obrázek 2: Zátiší se skleničkami vytvořené počítačem 3
Obrázek 3: Nové logo knihovny LWJGL 7
Obrázek 4: Úvodní obrazovka programu Blender 16
Obrázek 5: Trojúhelník vykreslený v okně LWJGL 19
Obrázek 6: Model 3D kartiček a regionů 21
Obrázek 7: Systém souřadnic knihovny LWJGL 22
Obrázek 8: Náhled na rozehranou hru 26

1 Úvod

3D grafika a počítačové hry dnes již k sobě nerozlučně patří. Grafika sice není všechno, hra musí mít například dobrou hratelnost. Výborná grafická stránka může hratelnost trochu zastínit, ale nebude to ono. Na druhou stranu, hra se špatnou grafikou asi nikoho nepřiláká. I když takový Minecraft je asi výjimka potvrzující pravidlo.

Rozhodl jsem se tuhle hru implementovat, protože rád hraju deskové hry, a tuhle hru Carcassonne znám. Desková hra sice s 3D grafikou nemá nic společného, ale aspoň jedno téma mi není cizí.

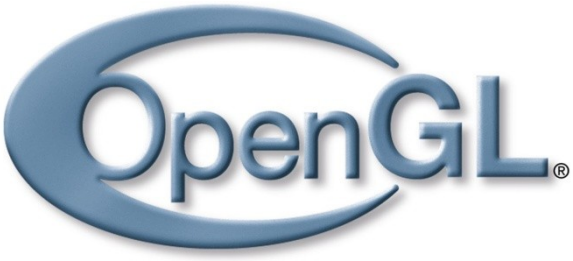
Carcassonne je moderní společenská hra německého typu, neboli typu EuroGames. Jejím autorem je Klaus-Jürgen Wrede. Jde o stolní hru pro 2 až 6 hráčů založenou na postupném přikládání jednotlivých kartiček. Tím vzniká krajina s cestami, městy, kláštery a loukami, kterou si hráči obsazují svými figurkami. Hra získala v roce 2001 první místo v anketě Spiel des Jahres (Hra roku) a Deutscher Spiele Preis (Hra německé veřejnosti).

V samotné práci napřed uvedu přehled jednotlivých knihoven pro tvorbu 3D grafiky v jazyce Java. Popíšu jejich vlastnosti, popíšu výhody a nevýhody a jednu knihovnu nebo modul si vyberu pro tvorbu programu. Uvedu základní přehled vytváření 3D modelů pro hry, a popíšu dva vybrané programy, které jsem používal.

Závěr práce patří popisu samotné implementace programu. Popíšu problémy, které jsem řešil. Uvedu postup řešení na reálných příkladech. Součástí je také programátorská dokumentace, obsahující UML diagramy vytvářeného programu.

2 OpenGL

OpenGL (Open Graphics Library) je průmyslový standard specifikující multiplatformní rozhraní (API) pro tvorbu aplikací počítačové grafiky. Používá se při tvorbě počítačových her, CAD programů, aplikací virtuální reality či vědeckotechnické vizualizace apod. Informace jsem čerpal z [1].



Obrázek 1: OpenGL logo

2.1 Implementace

Implementace OpenGL existují pro prakticky všechny počítačové platformy, na kterých je možno vykreslovat grafiku. Kromě implementací vestavěných v grafickém hardware (na grafické kartě) existují také softwarové implementace, které umožňují používat OpenGL i na hardwaru, který ho sám o sobě nepodporuje (ale obvykle nabízejí nižší výkon). Příkladem takové implementace je knihovna Mesa 3D s otevřeným zdrojovým kódem, která ovšem z licenčních důvodů nemůže být označena jako implementace OpenGL, ale pouze jako implementace API, které je „velmi blízké“ OpenGL.

2.2 Struktura OpenGL

Základní funkcí OpenGL je vykreslování do framebufferu. Umožňuje vykreslování různých základních primitiv (bodů, úseček, mnohoúhelníků a obdélníků pixelů) v několika různých režimech. Veškerá činnost OpenGL se řídí vydáváním příkazů pomocí volání funkcí a procedur (kterých OpenGL definuje cca 250). V OpenGL se nepoužívá objektově orientované programování. Jednotlivá primitiva jsou definována pomocí vrcholů – každý z nich definuje bod, koncový bod hrany nebo vrchol mnohoúhelníku. Každý vrchol má přiřazena data (obsahující souřadnice umístění bodu, barvy, normály a texturovací souřadnice).

Rozhraní OpenGL je založeno na architektuře klient-server. Program (klient) vydává příkazy, které grafický adaptér (server) vykonává. Díky této architektuře je možné, aby program fyzicky běžel na jiném počítači než na tom, na kterém se příkazy vykonávají, a příkazy se předávaly prostřednictvím počítačové sítě.

2.3 Historie

První verze OpenGL, verze 1.0 byla vydána v roce 1992. Od té doby bylo OpenGL postupně rozšiřováno vydáváním nových verzí specifikace.

3 Počítačová 3D grafika

Díky stále většímu zájmu o 3D grafiku, hlavně kvůli video a počítačovým hrám, začali vznikat rozhraní (API) pro ulehčení práce. Tato API se ukázaly jako zásadní pro výrobce grafických karet, protože poskytují jednotný přístupu pro programátory, a přitom využívají hardware konkrétní grafické karty.

První grafický 3D rámec (anglicky framework) byl Core (Jádro), který byl vyvinut v ACM (Association for Computing Machinery) a publikován v roce 1977.

3D grafika je označení pro speciální část počítačové grafiky, která pracuje s trojrozměrnými objekty. Převod 3D objektů do 2D zobrazení se nazývá vykreslování (renderování). Nejznámější využití počítačové 3D grafiky je vytváření animací, pro tvorbu filmů nebo počítačových her. Ale také se používá ve vědě i průmyslu, například pro počítačové simulace nebo trojrozměrné zobrazení orgánů.

Dnešní podoba počítačové grafiky je už velmi pokročilá. Hlavně díky počítačovým hrám, ve kterých se pořád vylepšují různé detaily, a hry už vypadají jako film. A při pohledu na nějaký dnešní animovaný film, jdou rozdíly od skutečných na první pohled jen těžko rozeznat. Následující obrázek to může ilustrovat. Modely na něm byly vytvořeny v profesionálních placených nástrojích. Vykreslen byl pomocí algoritmu známého jako Radiozita (metoda globálního světlení scény). Při prvním pohledu a řekl bych, že i při mnoha dalších by se jen těžko hledali nějaké stopy počítačové animace.

Obrázek 2: Zátiší se skleničkami vytvořené počítačem

3.1 Knihovny pro tvorbu 3D grafiky v jazyce Java

Pro jazyk Java dnes existuje řada různých podpůrných knihoven a herních modulů pro tvorbu 3D grafiky. Některé jsou už dnes zastaralé a jejich projekty byly zrušeny, ale i přes to se dají stáhnout a používat. Jiné zase nejsou úplně určené pro vývoj, spíš jen jako jednorázový počin jejich autorů s určitým cílem. Nakonec jsem vybral dva příklady, které nejsou vůbec určeny k tvorbě her, ale pro vědecké účely.

3.2 Základní přehled

Nízká úroveň přístupu

Sem patří všechny knihovny, které pouze poskytují přístup k OpenGL, případně k jinému grafickému rozhraní. Říká se jím obalové třídy (anglicky wrapper library), které jakoby obalují funkce daného rozhraní, a poskytují k nim přímí přístup prostřednictvím svého rozhraní.

Vysoká úroveň přístupu

Jde o takzvané herní moduly (v angličtině "game engine"). Moduly poskytují větší míru abstrakce, při vytváření modelů nebo herní scény. Pro přístup ke grafickému rozhraní používají převážně některou ze zde uvedených knihoven.

Knihovny - nízká úroveň přístupu

- GL4Java
- Java 3D
- Java OpenGL (JOGL)
- libGDX
- LWJGL

Moduly - vysoká úroveň přístupu

- 3DzzD
- Ardor3D
- Auriga3D
- Bindenlicht
- Env3D
- Jake2
- Java is Doomed
- jMonkey Engine
- jPCT
- ogre4j
- Xith3D

Ostatní - nejsou určeny pro tvorbu her, ale pouze pro zobrazování 3D grafiky

- Jzy3D
- Processing

3.3 Knihovny

3.3.1 GL4Java

GL4Java neboli OpenGL for Java je starší knihovna pro podporu OpenGL v jazyce Java. Poslední verze byla vydána na konci roku 2001. Vychází z původního OpenGL, které známe z C.

Podporuje OpenGL verze 1.3 a GLU verze 1.2. OpenGL funkce jsou volány přes Java Native Interface, proto je výkon aplikace limitován rychlostí hostitelského systému. Je možné ji použít ve spojení s knihovnami AWT a Swing.

Ve své době byla populární, ale byla vytlačena modernějším JOGL. [2]

3.3.2 Java 3D

Java 3D je doplněk ke standardním knihovnám jazyka Java a slouží k zobrazování trojrozměrné grafiky. Není to tedy jen obyčejná vazba mezi Java a OpenGL. Programy napsané v Java 3D, lze spustit na různých typech počítačů, v různých typech pracovního prostředí a mohou běžet i přes internet.

Knihovna tříd Java 3D poskytuje rozhraní které je jednodušší než rozhraní mnoha jiných grafických knihoven, ale zároveň je dostačující k tvorbě dobrých her a animací. Java 3D staví na existujících technologiích jako je DirectX a OpenGL takže programy neběží tak pomalu jak bychom mohli čekat. Java 3D také umožňuje začlenit objekty vytvořené 3D modelovacími nástroji jako je TrueSpace nebo dokonce VRML modely. [3]

Java 3D software je zdarma k dispozici od Sun Microsystems (nyní Oracle). Poslední vydaná verze je 1.5.2. Podle dokumentace vyšla v roce 2008, tehdy ještě pod původní Sun Microsystems.

3.3.3 Java OpenGL (JOGL)

Java OpenGL je knihovna která umožňuje používat funkce z OpenGL v jazyce Java. Jinými slovy také vazba jazyka Java na OpenGL. JOGL byl původně vytvořen a vyvíjen skupinou Game Technology Group od firmy Sun Microsystems. Od roku 2010 se stal nezávislým projektem s otevřeným zdrojovým kódem, šířený pod licencí BSD. Nyní je JOGL vyvíjen ve společnosti JogAmp, kde má dobrou podporu a dokumentaci.

JOGL umožňuje přístup k většině OpenGL funkcí, dostupné pro programovací jazyk C, pomocí použití Java Native Interface (JNI). Nabízí přístup ke standardním funkcím OpenGL a také ke knihovně nástrojů. Nástroje pro volání na standardní uživatelské prvky systému nejsou k dispozici, JOGL je totiž integrovaný do knihoven, které používá Java jako je AWT nebo Swing. [4]

3.3.3.1 Návrh

Základní OpenGL rozhraní v jazyce C, jsou přístupné v JOGL pomocí JNI volání. Pro fungování JOGL musí tedy operační systém podporovat OpenGL jako takové.

JOGL se liší od ostatních knihoven a rámců tím, že pouze poskytuje přístup k OpenGL rozhraní pomocí několika tříd a metod, než se snažit mapovat OpenGL funkce do objektově-orientovaných struktur. Ve skutečnosti se velká část zdrojového kódu JOGL tvoří automaticky z hlavičkových souborů OpenGL jazyka C. O to se stará nástroj zvaný GlueGen, který byl vytvořen s cílem usnadnit vytváření JOGL knihovny.

Tento návrh má však svoje výhody i nevýhody. Nevýhodou je, že metody a jejich struktura, které souvisejí s JOGL je jiná, než ty které jsou typické pro jazyk Java. Což může být nepříjemné pro mnoho programátorů. Na druhou stranu, díky jednoduchému mapování metod z OpenGL rozhraní jazyka C do jazyku Java, je jednoduší naučit se pro toho, kdo zná OpenGL z jazyka C. Navíc umožňuje lehčí přeměnu již existujících aplikací v jazyce C, a používání již hotových příkladů a cvičení. Tenká vrstva abstrakce, kterou JOGL poskytuje, přináší efektivnost při běhu programu. To má ale za následek obtížnější tvoření kódu, v porovnání s vyšší úrovní abstrakce třeba u knihovny Java 3D. Protože se většina kódu tvoří automaticky, mohou být změny provedené v OpenGL rychle přidány také do JOGL.

3.3.3.2 Stav a Standardizace

Verze 1.1.0 je referenční implementace pro projekt JSR-231.
Ve verzi 1.1.1 se objevila základní podpora pro vykreslování křivek a ploch přes tradiční GLU rozhraní.
Ve verzi 2.0.2 přibyla podpora OpenGL verze 4.3 a OpenGL ES verze 1,2 a 3.
Od roku 2014 (verze 2.1.4) JOGL poskytuje úplný přístup ke specifikaci OpenGL 4.3 a stejně tak ke všem rozšířením (OpenCL, OpenAL).
Poslední verze JOGL 2.3.1 vyšla 27. března 2015.

3.3.4 libGDX

LibGDX je rámec určený pro vytváření her v jazyce Java. Rámec je převážně napsaný v jazyce Java, ale obsahuje části jazyka C a C++ pro zvýšení výkonu. Poskytuje jednotné rozhraní, které funguje na všech podporovaných platformách.

Podporované platformy

- Windows
- Linux
- Mac OS X, iOS
- Android (2.2+)
- BlackBerry
- webový prohlížeč - JavaScript/WebGL

Rámec poskytuje prostředí pro rychlý návrh a iteraci. Místo testování na Androidu/iOS/Javascript po každé změně kódu, stačí hru spustit ladění hry pouze jednou na počítači. Vlastnosti virtuálního stroje na počítači, jako je výměna kódu za běhu programu, práci značně urychlí.

LibGDX se snaží nebýt alfou a omegou ("end all, be all"). Nabízí spoustu funkcí, ze kterých si lze jednoduše vybrat. [5]

3.3.5 LWJGL

LWJGL (Lightweight Java Game Library), v překladu odlehčená knihovna pro tvorbu her v jazyce Java. Je knihovna určena pro profesionální i amatérské programátory a k tomu aby mohly vznikat kvalitní komerční hry. LWJGL poskytuje vývojářům přístup k vysoce výkonným multiplatformním knihovnám OpenGL, OpenCL a OpenAL umožňující vytvářet nejmodernější 3D hry a 3D zvuk. Podporuje přístup ke standardním příslušenstvím počítače jako je klávesnice a myš, navíc také podporuje zvláštní typy, například gamepad, volant, joystick. Vše přes jednoduché rozhraní.

Obrázek 3: Nové logo knihovny LWJGL

Cílem LWJGL není to, aby se hry vyvíjeli snadno, knihovna hlavně umožňuje vývojáři přístup k prostředkům, které jsou v jazyce Java špatně dostupné, nebo vůbec nedostupné. Autoři předpokládají, že se LWJGL díky svému vývoji a rozšířenosti, stane základem pro další herní knihovny a moduly, a zjednoduší další vývoj, díky rozhraní které knihovna přináší.

LWJGL je dostupný pod BSD licencí, software s otevřeným zdrojovým kódem a volně k dispozici bez poplatků. [6]

Informaci o stáři knihovny se mi nepodařilo vyhledat. Na oficiálním fóru jsou první zmínky z poloviny roku 2003. Aktuální a poslední vydaná verze 2.9.3 je z 15. ledna 2015. Verze 2 už se pravděpodobně nadále vyvíjet nebude, protože 13. listopadu 2014 byla ohlášena nová verze LWJGL 3.0, ve které půjde o kompletní předělání dosavadního LWJGL.

Knihovna verze 2 byla použita při tvorbě desítek her. Na stránkách knihovny je k dispozici obchod, ve kterém se dají některé hry zakoupit. V LWJGL byla například vytvořena hra Minecraft, která se hodně rozšířila, nebo také méně známá Revenge of the Titans.

Cíle, kterých se autoři drželi a drží, mají za následek to, že se projekt dostal, na úroveň na které je dnes, a také to kterým směrem se bude ubírat jeho tvorba nadále.

Cíle jsou:

- rychlost
- velká rozšířenost
- jednoduchost
- být malý
- bezpečnost
- robustnost
- minimalismu

Rychlost:

Smyslem LWJGL bylo přinést rychlost vykreslování 3D grafiky v jazyce Java do 21. století. Třeba zrušit metody které jsou efektivní v jazyce C, ale v Java nedávají smysl, například *glColor3fv*. Nebo vyhodit výjimku, když není v systému Windows k dispozici hardwarová akcelerace. Nemá přece cenu hrát hru, která se bude vykreslovat rychlostí 5 snímku za sekundu.

Velká rozšířenost:

Knihovna je určena pro práci na malých zařízeních, jako jsou mobilní telefony, ale také k víceprocesorovému vykreslování na serverech. Při vývoji sice ještě neexistovali mobilní telefony s tak rychlým virtuálním strojem a podporou 3D grafiky, ale do budoucna se to předpokládalo. Proto byla knihovna přizpůsobena, aby v budoucnu podporovala OpenGL ES. To kladlo na knihovnu omezení velikosti, jinak by se nikdy neuchytila na platformě J2ME. Binární podoba knihovny tak dosahuje velikosti půl megabajtu, to stačí na 3D zvuk, grafiku a vstupy / výstupy.

Jednoduchost:

LWJGL musí být jednoduché, aby ho používala široká škála vývojářů. Mělo by usnadnit začátky nováčkům, ale taky aby ho bylo možné používat profesionálně. Proto bylo zapotřebí zvolit princip, který přinese OpenGL jak na mobilní zařízení tak také na počítače. Bylo odstraněno spoustu funkcí, které nemusí znát 99% programátorů her.

Být malý:

- malý = jednoduchý Čím méně způsobů jak udělat určitou věc, tím snazší je naučit se je.
- malý = náš kód má velmi málo chyb Asi byste nechtěli hledat chyby v naší knihovně, místo hledání chyb ve vašem kódu.
- malý = jednoduše stažitelný LWJGL je dost malý, aby mohl být stažen zvlášť pro každou aplikaci, která ho používá.
- malý = J2ME

Bezpečnost:

Pokud by nebyla zaručena bezpečnost knihovny, asi by jí moc lidí používat nechtělo. Vývojáři se rozhodli přestat používat ukazatele a začít používat vyrovnávací paměť. Přidáváním dalších kontrol, limitů zabránit útokům na vyrovnávací paměť.

Robustnost:

Spolehlivý systém je mnohem užitečnější než rychlý systém. Při testování kódu po částech nevyšli najevo problémy, které se ukázaly při testech na reálných aplikacích. Autoři se rozhodli ustoupit od vlastních návrhů a používat kontroly typu "if(…)" a vyhazování výjimek. Dále došli k závěru, přesunout kontrolu chyb v OpenGL z nativního kódu až do kódu jazyka Java. Odpadá tak nutnost další knihovny pro režim ladění.

Minimalismus:

To co v knihovně opravdu být nemusí, to tam není. Původním cílem bylo vytvořit knihovnu, která poskytne pouze ty nejdůležitější funkce pro přístup k hardware, které samotný jazyk Java nepodporuje. Toho cíle se autoři také drží. Autoři z projektu odstranili GLU knihovnu, protože pro vývoj her není až tak důležitá, maximálně pár jejich funkcí.

3.4 Moduly

3.4.1 3DzzD

3DzzD byl 3D modul obsahující 3D sceno graf a podporoval základní fyziku. Byl napsaný v jazyce Java, a obsahoval všechny nástroje pro vykreslování 3D scény v prostředí HTML. Pro vykreslování mohl používat jak podporu software, tak i hardware. Vývoj projektu byl ukončen, a místo něho lze použít WebGL. WebGL je nyní podporován všemi prohlížeči a dokonce i na platformě Android. [7]

3.4.2 Ardor3D

Ardor3D je 3D herní modul založený na scéno grafu a celý je napsaný v jazyce Java. Používá OpenGL pro vysoce výkonné vizualizace ve hrách. Modul začali vytvářet Joshua Slack a Rikard Herlitz 23. září 2008 jako větev projektu jMonkeEngine kvůli určitým problémům v původním projektu. Problémy se týkaly jména, původu, systému udělování licencí a struktury společnosti. Chtěli totiž vytvořit silný Java modul s otevřeným zdrojovým kódem s organizovanou firemní podporou. První veřejná verze vyšla 2. ledna roku 2009, a další verze pak vycházely vždy po několika měsících.

V roce 2011 Ardor3D využila NASA ve svém projektu Mars Curiosity mission, pro vizualizaci terénu a pohybu vozítka na Marsu. Poslední stabilní verze vyšla 23. července 2013. V březnu 2014 původní autor Joshua Slack prohlásil, že se od projektu upouští. Software zůstal poté pod zlib licencí, a je nyní volně k dispozici. Část projektu je stále aktivní pod jménem JogAmp's Ardor3D Continuation a udržuje ho Julien Gouesse. [8]

3.4.3 Auriga3D

Auriga3D je taky jeden ze starších enginů napsaných v jazyce Java. Podporuje JOGL i LWJGL OpenGL vazby na hardwarově akcelerované funkce. Je komponentově orientovaná.

Auriga3D vychází ze hry Quake3, pracuje s mapami a osvětlením. Původní stránka projektu dnes už neexistuje a projekt byl ukončen. [9]

3.4.4 Bindenlicht

Bindenlicht je vazba jazyku Java na Irrlicht Engine, která na platformu Javy přináší vysoce kvalitní 3D modul s otevřeným zdrojovým kódem.

Irrlicht Engine

Irrlicht Engine je vysoce výkonný 3D realtime modul napsaný v jazyce C++. Je multi-platformní a používá D3D, OpenGL a svůj vlastní software pro vykreslování grafiky. Obsahuje všechny nejmodernější funkce, které lze najít v komerčních projektech. [10]

3.4.5 Env3D

Motto: "od nuly po 3D za pět minut"

Env3D je inovativní 3D modul napsaný v jazyce Java. Byl vytvořen převážně pro výuku programování a 3D programování. Vytváření 3D video her je zábavné a jednoduché i při učení programování v jazyce Java. Modul má pár unikátních vlastností, kterými se chlubí. [11]

Unikátních vlastnosti:

- Úzká integrace s BlueJ IDE - vizualizace 3D objektů během několika minut učení jazyku Java
- Snadné použití - stačí importovat jen jednu třídu pro vytvoření 3D video hry
- Výkonný - i přes jednoduché rozhraní, se dá hodně upravit,
- uživatelé mohou importovat 3D objekty a animace pomocí formátu OBJ
- osvědčený v praxi - studenti se raději učí programování

3.4.6 Jake2

Jake2 je herní modul napsaný v jazyce Java, do kterého autoři přepsali celý modul hry Quake 2. Jake2 používá JOGL na OpenGL grafiku a JOAL pro 3D zvuk. Od verze 0.9.4 jsou k dispozici také ovladače pro LWJGL, jako alternativa k JOGL/JOAL. Poslední verze 0.9.5 vyšla v roce 2007. Není to modul pro tvorbu dalšího software, jen ho uvádím jako zajímavost. [12]

3.4.7 Java is Doomed

Java is Doomed je další herní modul naimplementovaný v jazyce Java. Má otevřený zdrojový kód a pro přístup k OpenGL používá JOGL. Cíl projektu bylo vytvořit modul s otevřeným zdrojovým kódem, který by mohli používat vývojáři, kteří mají programování jako hobby. První hra vytvořena v tomhle modulu se jmenovala Escape. Hra se inspirovala v legendární hře Doom a dá se stáhnout v balíku společně s knihovnou. Poslední update vyšel v dubnu roku 2013. [13]

3.4.8 jMonkeyEngine

Motto: "Multi platformní herní modul pro dobrodružné vývojáře v jazyce Java"

jMonkeyEngine je herní modul vytvořený speciálně pro moderní 3D vývoj, protože značně používá technologii shaderů. jMonkeyEngine je napsaný v jazyce Java a používá knihovnu LWJGL jako výchozí pro vykreslování grafiky. Podle oficiálního fóra je k dispozici vykreslování založené na knihovně JOGL. Modul plně podporuje verzi OpenGL 2 až OpenGL 4.

jMonkeyEngine je projekt s otevřeným zdrojovým kódem, šířený pod licencí BSD. Modul používá několik komerčních herních studií, a vzdělávací instituce. Po stažení aktuální verze jMonkeyEngine 3, je k dispozici pokročilé SDK s vlastním vývojovým prostředím (IDE). [14]

3.4.8.1 jMonkeyEngine 3 SDK

Sám o sobě je modul jen sbírka knihoven, což z něho dělá nástroj pro vývoj her na nízké úrovni. Integrované vývojové prostředí dělá z jMonkeyEnginu nástroj na vysoké úrovni s mnoha grafickými komponenty. SDK je založeno na platformě NetBeans, což umožňuje přístup ke grafickým editorům a zásuvným modulům (plugin). Vedle aktualizací NetBeans, jsou k dispozici také vlastní aktualizace a zásuvné moduly. Lze vybrat stabilní verze, nebo takzvané noční aktualizace (nightly updates).

3.4.8.2 Historie

jMonkeyEngine byl vytvořen s cílem zaplnit mezeru v nedostatku plnohodnotných grafických modulů napsaných v jazyce Java. Historie by se dala rozdělit do dvou částí, protože současném jádře vývojového týmu není už nikdo s původních tvůrců.

jMonkeyEngine 0.1 - 2.0

Období od vydání první verze v roce 2003 až po poslední verzi, která vyšla v roce 2008. Když hlavní vývojáři koncem roku 2007, postupně přestávali na projektu pracovat, verze 2.0 ještě nebyla dokončena. Přesto byla tato verze přijata a využívána pro komerční účely a podporována komunitou.

2003: Práci začal Mark Powell (známý jako MojoMonkey), jako vedlejší projekt, aby zjistil, jestli může být plnohodnotné grafické rozhraní napsané v jazyce Java. V začátcích se inspiroval knihou C++ book 3D Game Engine Design od autora David Eberly.

Leden 2004: Mark se spojil s Joshua Slack (známý jako Renanse) a spolu v průběhu následujících dvou let, s pomocí dalších přispěvatelů projek dokončili.

15. srpna 2008: Joshua Slack oznamuje, že odchází od aktivního rozvoje projektu jMonkeyEngine.

jMonkeyEngine 3.0

Po odchodu hlavních vývojářů jME na konci roku 2008 projekt stagnuje několik měsíců, občas jsou komunitou vydány nějaké opravy. Verze 3.0 začíná jako experiment. První přehled verze 3.0 vzbudil velký rozruch a většina se shodla, že tohle bude oficiální nástupce jME 2.0. Po vyřešení všech formalit, mohlo vzniknout nové jádro vývojářů, které se dnes skládá s devíti angažovaných jednotlivců.

1. dubna 2009: Kirill Vainer začíná pracovat na nové verzi modulu a zveřejňuje první veřejně dostupný kód.

24. června 2009: Projekt se dočkal přepracování, k projektu se přidává Erlend Sogge Heggen a krátce potom Skye Book.

17. května 2010: Vydaná první alfa verze jMonkeyEngine 3. Ve stejnou dobu vychází také jMonkeyEngine SDK, jen několik měsíců po první fázi plánování. jMonkeyEngine SDK se tak stává hlavním projektem.

7. září 2010: Vznikají nové stránky projektu, kompletní vzhled je předělán.

22. října 2011: Byla vydaná beta verze jMonkeyEngine SDK. Lze také stáhnout pravidelně aktualizované noční vydání.

15. Února 2014: Oficiální jMonkeyEngine 3.0 SDK je vydán. I když stabilní verze byli už dříve, s vydáním se čekalo.

3.4.9 jPCT

jPCT je volně dostupný modul pro jazyk Java a Android. Běží na platformách Windows, Linux, Max OS, Solaris x86 a na mobilních telefonech a tablech s operačním systémem Android. Nepotřebuje žádné speciální knihovny pro detekci kolizí a používá standardní gui Javy Swing / AWT. Poslední verze 1.29 byla vydána 3. 11. 2014. [15]

Základní vlastnosti

- načítání 3DS, OBJ, MD2, ASC a XML souborů
- podpora pro octrees a portal rendering
- animace snímků (převzaté z MD2 souboru, nebo vlastní nastavení)
- animace kostry prostřednictvím Bones API

3.4.10 ogre4j

ogre4j je projekt, který umožňuje používat OGRE knihovnu v jazyce Java. OGRE (Object-Oriented Graphics Rendering Engine) je scénově orientovaný, flexibilní 3D modul napsaný v jazyce C++. Používá systémové knihovny, jako jsou Direct3D a OpenGL.

Poslední verze 1.6.2 byla vydaná 5. června 2009 a je označována jako beta. Samotný projekt vypadá jako zapomenutý, ale jeho verze, včetně poslední, se dají volně stáhnout na stránkách projektu. [16]

3.4.11 Xith3D

Aplikační rozhraní Xith3D je velice podobné Java 3D. Není vázáno na konkrétní rozhraní, podle volby programátora může pracovat s libovolným rozhraním. Lze volit mezi JOGL nebo LWJGL. Pro fungování stačí implementovat pár základních věcí. Poslední verze vyšla 6. září 2010, která má být dále vydávána na repositáři. [17]

Základní vlastnost

- umí nahrávat scény a modely ve formátech např.
  AC3D, OBJ, 3DS, MD2, MD3, MD5, COLLADA
- obsahuje systém pro řešení kolizí

3.5 Ostatní

3.5.1 Jzy3D

Knihovna pro jazyk Java s otevřeným zdrojovým kódem, která umožňuje snadno vykreslovat vědecké 3D grafy. Například povrchy, rozptylové grafy, sloupcové grafy a mnoho dalších 3D vizualizací. Je postaven na API JOGL. Díky tomu lze použít na platformách Windows, Unix, MaxOS, a začlenit do grafického rozhraní Javy, Swing, AWT nebo SWT. [18]

3.5.2 Processing

Nejde o knihovnu pro jazyk Java. Processing je programovací jazyk s otevřeným zdrojovým kódem, který staví na jazyku Java. Používá zjednodušené syntaxe, je určený pro neprofesionály a v základu připomíná jazyk C. Je velice podobný Wiringu, který se používá při programování Arduina. Není určený pro tvorbu her, ale spíš pro grafické vizualizace. [19]

4 Carcassonne On-line

Hra Carcassonne, jak serverová tak i klientská část, je naimplementována v jazyce Java. Komunikaci klient-server zajišťuje technologie Java RMI (remote method invocation).

4.1 Java RMI

Java remote method invocation (Java RMI) je technologie programovacího jazyka Java umožňující z jednoho virtuálního stroje (JVM) volat metody objektů na jiném virtuálním stroji, který obvykle běží na jiném počítači (SERVER). Výhodou RMI je, že programátor zachází se vzdáleným objektem, jako by byl místní. Rozhraní a třídy, které jsou zodpovědné za funkčnost RMI jsou nadefinovány v balíčku *java.rmi*. [20]

4.2 Napojení 3D aplikace na existující hru

Klienti připojení k dané hře si mezi sebou vyměňují informace o hře pomocí serveru a RMI. Klient posílá na server Stav hry. StavHry je proměnná typu List<List> a obsahuje celkem 4 další listy.

- packOfCards - balíček karet
- usedCards - použité karty
- players - informace o všech hráčích, kteří hrají danou hru
- allRegions - všechny regiony ve hře, cesty, města…

Pro zobrazení náhledu ve 3D mi stačí vytáhnout si z celého listu listů pouze jeden list, a tím je usedCards. Tenhle list, obsahuje všechny dosud položené kartičky na hrací plochu. Každá kartička nese informaci o regionech, které se na ní nacházejí. V těchto regionech je dále informace, jestli byla na kartičku položena figurka. Po provedení tahu je stav hry aktualizován a zaslán na server. Server pošle stav hry všem hráčům, kteří jsou ke hře připojeni. O to se starají metody posliNext a prijetiNext. A právě tady přichází příležitost poslat list usedCards do 3D aplikace.

3D aplikace běží v původní třídě *Gui*, ale v samostatném vlákně. Při přijetí nebo odesílání stavu hry, pošlu také stav hry vláknu, ve kterém běží 3D aplikace. Poslání listu do druhého vlákna provádím pomocí rozhraní *BlockingQueue*, a konkrétní implementace *ArrayBlockingQueue*. Třída *Gui* vloží metodou *put* do fronty seznam. Druhé vlákno si tyto data odebere metodou *take*.

Problémy

Vlákno, ve kterém běží 3D aplikace, musí neustále ve smyčce vykreslovat scénu. Nemá čas čekat na zámku nad objektem daného listu. Proto v každé smyčce pouze testuji, jestli ve frontě něco je. Když aplikace zjistí, že fronta není prázdná, zavolá metodu na získání dat. Problém nastal při procházení získaného listu cyklem foreach. Program občas vyhodil výjimku *ConcurrentModificationException*. Ta je vyhozena v případě, když se prochází seznam a dojde k jeho modifikaci. Problém jsem vyřešil zkopírováním seznamu karet do nového seznamu, a ten pak dál posílám do fronty.

5 Modely pro hru

Pro hru bylo potřeba vytvořit modely. Zvolil jsem způsob, vytvoření základního modelu pro každou kartičku ve hře. Základní hra má celkem 72 kartiček., které se ovšem opakují. Takže na základní hru bylo potřeba vytvořit pouze 19 modelů. Aktuální verze, která obsahuje jak základní hru, tak dvě rozšíření, obsahuje 114 kartiček, ale díky opakování stačilo pouze 45 různých modelů.

Před modelování je potřeba zvolit vhodný nástroj, ve kterém půjdou vytvořit všechny modely potřebné pro hru.

5.1 Software pro tvorbu modelů

Programů pro tvorbu 3D modelů existuje celá řada. Profesionální programy jsou placené, poskytují velké množství funkcí. Spousta programů je volně dostupná, a některé svými funkcemi nijak nezaostávají. Pro moji tvorbu jsem zvolil dva programy, které jsem porovnal a jeden vybral.

5.1.1 Wings 3D

Wings 3D je moderní modelářský nástroj, který pracuje na principu subdivision surface, neboli dělení povrchu. Byl inspirovaný programy Nendo a Mirai od Izware.

3D Wings nabízí širokou škálu nástrojů pro modelování, přizpůsobitelné rozhraní, podporu světel a materiálů, a má nástroj pro mapování textur. Program je freeware, dostupný pro Windows i Linux, jak v 32bitové tak v 64bitové verzi. Má otevřený zdrojový kód a je zcela zdarma pro použití v osobních i komerčních projektech.

Jednou nevýhodou programu je, že nepodporuje animace. Druhá, která je dost zásadní, program při práci občas spadne. Tím nechci tvrdit, že je program špatný, nezkoušel jsem ho spouštět na jiném počítači, tudíž to nemusí být chyba programu.

Výhody:

- jednoduché ovládání
- volně dostupný, i pro komerční použití
- obsahuje českou lokalizaci

Nevýhody:

- nepodporuje animace
- při práci občas spadne

5.1.2 Blender

Blender je také otevřený software pro modelování a vykreslování 3D počítačové grafiky a animací s využitím různých technik (např. sledování paprsku, radiosita, scanline rendering, GI). Vlastní interface je vykreslován pomocí knihovny OpenGL. OpenGL umožňuje nejen hardwarovou

akceleraci vykreslování 2D a 3D objektů, ale především snadnou přenositelnost na všechny podporované typy pracovního prostředí.

Bleder narozdíl od Wings nabízí mnohem větší řadu nástrojů a nastavení. Blender se ovládá v naprosté většině případů pomocí klávesových zkratek. To pro nové uživatele může představovat určitou překážku, ale jde o efektivní a posléze intuitivní způsob rychlé tvorby modelů a animací bez nutnosti hledat funkce v několikanásobném menu.

Po vydání verze programu 2.44 si ho za pouhý měsíc stáhlo více než 800 000 uživatelů. Poslední a aktuální verze 2.47 vyšla 31. března 2015. Já jsem na svou práci používal Blender ve verzi 2.73a.

Výhody:

- volně dostupný, i pro komerční použití
- obsahuje českou lokalizaci
- podporuje animace

Nevýhody:

- složitější ovládání pro nové uživatelé

Obrázek 4: Úvodní obrazovka programu Blender

5.1.3 Zhodnocení

První modely jsem vytvářel v programu Wings 3D. Jak už jsem, program sem tam z ničeho nic přestal odpovídat a spadl. Po čtvrt hodině práce přidáváním, odebíráním a posouváním hran v modelu to není nic příjemného. Na ovládání jsem si taky dlouho zvykal, to u Blendru sice taky, ale tam mi přišlo rozumnější. Co se týče podpory a kvality, je na tom Blender určitě lépe. Proto jsem pro modelování kartiček a modelů do hry zvolil nakonec Blender. I přes to že má složitější ovládání. Nicméně podporuje animace, které jsou potřeba pro animování pohybujících se figurek.

6 Implementace

6.1 Příprava

6.1.1 Volba 3D knihovny

Před začátkem práce jsem se musel rozhodnout, kterou knihovnu, případně modul budu používat. Neměl jsem ještě moc zkušeností v téhle oblasti, a při prvním hledání se mi zalíbila knihovna LWJGL. Podobný JOGL nebyl tehdy ještě pod společností JogAmp a z mého pohledu neměl tak kvalitní dokumentaci. Zvažoval jsem i možnost použít modul jMonkeyEngine, který by mohl usnadnit práci. Modul má však svoje specifické vývojové prostředí, do kterého by mohl být problém dostat stávající hru Carcassonne on-line.

Rozhodl jsem se tedy pro knihovnu LWJGL. Zalíbily se mi stránky projektu, kde mají řadu návodů pro začátečníky ale i pokročilé, které jsem rád využíval. Dále pak odkazy na různé řešené problémy, a taky fórum, které je aktivně využívané.

Když jsem začínal tvořit program k bakalářské práci, byla k dispozici verze LWJGL 2.9.1, která vyla vydána 2. prosince 2013. Během práce na programu vyšly další 2 verze a dokonce beta nové verze LWJGL 3.0. Já jsem však zůstal u původní verze 2.9.1.

6.1.2 Volba vývojového prostředí

Pro vývoj programu jsem zvolil vývojové prostředí Eclipse. Používám ho od začátku programování v jazyce Java a také pro platformu Android. I když jsem myslel, že znám jeho funkce, tak jsem se v průběhu práce na programu přesvědčil, že ne až tak dobře.

Hru Carcassonne on-line vytvářel její autor ve vývojovém prostředí NetBeans, které podporuje tvorbu formulářů a uživatelského rozhraní pomocí knihovny Swing.

Vývojové prostředí Eclipse neumí přímo importovat projekty z NetBeans. Podle jednoduchého návodu je to však možné. Stačí vytvořit v Eclipse projekt se stejným jménem a poté překopírovat všechny zdrojové soubory (Název.java) z projektu v NetBeans. Případně zkopírovat další zdrojové soubory, jako jsou obrázky, textové či binární soubory. Poté stačí projekt sestavit, a pokud vše proběhne v pořádku tak je "import" dokončen.

6.2 Základ tvorby 3D programu

Základem pro vykreslování grafiky je nějaké okno v operačním systému. Knihovna LWJGL používá svoje vlastní okno, displej (Display). Třída Display má 3 hlavní metody, jsou to: create(), update(), destroy(). Na začátku programu je potřeba displej nastavit. To zajistí metoda Display.setDisplayMode(new DisplayMode(800, 600)). Dále se zavolá metoda pro vytvoření okna Display.create() a program zobrazí okno o velikosti 800×600.

Pak se v nekonečné smyčce provádí Display.update(). LWJGL používá dvojitou vyrovnávací paměť. Všechno co se za jeden průchod vykresluje, se ukládá do paměti, které není viditelná. Při zavolání funkce update(), se tyhle dvě paměti vymění, a na displeji se zobrazí všechno, co jsme vykreslili. Vykreslování celé scény se prování v každé smyčce. Obsah paměti se vymaže, a všechny objekty ve hře musí být vykresleny. Podle počtu provedených cyklů za vteřinu se poté dá vypočítat počet snímků za sekundu.

Snímková frekvence

Snímková frekvence se obvykle udává v jednotkách **fps** (z anglického frames per second). Podle počtu snímků za sekundu se dá hodnotit plynulost hry. Jako minimální hodnota pro plynulou grafiku se bere 30 fps. Zhruba 60 fps se bere jako naprosto dostatečné pro bezproblémový chod hry.

Na začátku každé smyčky se také testuje, pokud okno nebylo zavřené uživatelem. K tomu slouží metoda isCloseRequested(). Když program zjistí, že uživatel zavře okno, vyskočí s aktuální smyčky a ukončí program. Před koncem je potřeba zavolat metodu destroy(), která ukončí displej a uvolní všechny zdroje.

Ještě před vytvořením displeje je potřeba nastavit samotný displej. K dispozici je několik možností včetně zobrazení na celou obrazovku. Poté je potřeba nastavit a povolit různé funkce OpenGL, jako je nastavení pozadí okna, povolení mapování textur atd.

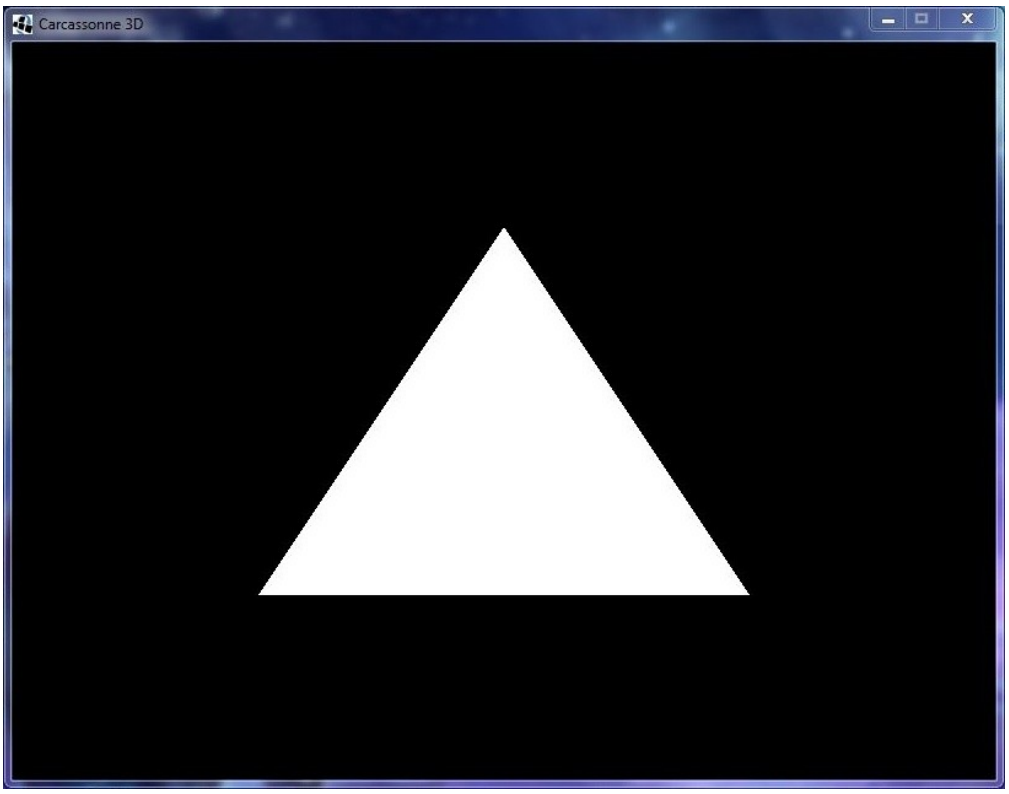
6.3 Modely, jejich nahrávání a vykreslování

Rozhodl jsem se pro program používat modely ve formátu *.obj. Blender umí tyhle soubory vytvořit, a přidat k nim vše potřebné pro správné zobrazení.

6.3.1 Základ je trojúhelník

Kreslení trojúhelníků je na většině grafických karet o hodně rychlejší než kreslení čtverců. Většina karet stejně čtverce převádí na trojúhelníky. Pro vykreslení trojúhelníku zavoláme třikrát funkci glVertex3f(). Funkce má tři parametry, které představují souřadnice na osách x, y, z. Příklad vykreslení základního trojúhelníku:

```
glBegin(GL_TRIANGLES);
    glVertex3f( 0.0f, 1.0f, 0.0f);
    glVertex3f(-1.0f,-1.0f, 0.0f);
    glVertex3f( 1.0f,-1.0f, 0.0f);
glEnd();
```



Obrázek 5: Trojúhelník vykreslený v okně LWJGL

6.3.2 Display List

Nikdy nebudeme vykreslovat pouze jeden trojúhelník. Každý objekt se skládá s velkého množství takových trojúhelníků. Vykreslovat jeden trojúhelník za druhým by bylo značně neefektivní a velmi zdlouhavé. Proto jsem pro vykreslování objektů začal používat Display List. Všechny vektory z modelu se místo vykreslení uloží do seznamu. Při uložení jsou všechny zkompilovány a uloženy do operační paměti grafické karty. Seznam se vytvoří metodou glGenLists(). Poté se mezi metody glNewList() a glEndList() jakoby vykreslí všechny vektory. Při zavolání tohoto listu metodou glCallList(), se už nemusí nic počítat a posílat a data se pouze vykreslí na obrazovku. Za nedlouho jsem zjistil, že tenhle způsob je zastaralý začal jsem používat Vertex Arrays.

6.3.3 Vertex Arrays

Vertex Arrays umožňují uložit všechny vektory do jednoho pole. Místo vykreslování všech vektorů postupně mezi příkazy glBegin() a glEnd() můžeme vytvořit instance třídy FloatBuffer do které se vektory uloží. Poté pomocí funkce glDrawArrays() vykreslit celé pole najednou. Tím dojde k omezení počtu volaných funkcí a sníží se počet nadbytečných vektorů. Třeba když jeden trojúhelník navazuje na druhý.

Tahle metoda podporuje samozřejmě souřadnice pro normály a pro textury. Informace stačí taky uložit do zvláštních polí. Když je všechno uloženo v polích, stačí dát programu vědět, kde v paměti se ony pole nacházejí. K tomu slouží funkce glVertexPointer(), glNormalPointer(), glTexCoordPointer().

6.3.4 Model - obj soubor

OBJ je klasický textový soubor. Obsahuje seznam všech vektorů, normálových vektorů a souřadnic pro texturu pro daný model. Dále obsahuje informace o tom, z čeho se skládá každá plocha (anglicky face), z kterého vektoru, normály a souřadnic pro texturu.

K souboru obj patří soubor mtl. Ve kterém jsou uloženy informace o materiálech. Model může mít jeden nebo více materiálů. U každého materiálu je uvedená například barva, míra průhlednosti a hlavně adresa k souboru s texturou. Textura může být libovolný obrázek. Program podporuje několik základních formátů. Jedinou podmínkou pro OpenGL je, aby obrázek měl rozměry mocniny čísla 2. Například 512×512, 1024×1024 ale také 256×512.

6.3.5 Třída OBJLoader

Pro nahrávání modelů do hry, jsem vytvořil třídu OBJLoader. Při načtení souboru obj, napřed načtu odkazovaný soubor mtl a uložím si do seznamu všechny materiály, které v něm jsou. Ke každému materiálu načtu texturu ze souboru. Každá plocha v objektu má pak dané, ke kterému materiálu patří, tudíž jakou bude používat texturu.

Potom uložím do seznamů všechny vektory, normály, souřadnice pro textury a popis všech ploch modelu.

Každá plocha, v mém případě trojúhelník, je popsána třemi vektory. Formát je následující: vektor / textura / normála. Každý vektor má tři souřadnice, jelikož se bavíme o trojrozměrném modelu. Plocha potom vypadá takto: v1/t1/n1 v2/t2/n2 v3/t3/n3. Získaná čísla nejsou přímo souřadnice, jsou to jakoby indexy do dříve načtených polí. Důvod je asi jasný, pro vytvoření krychle nám tak stačí 8 vektorů. Kdybychom popisovali každý trojúhelník zvlášť, potřebujeme 36 vektorů.

Třída uloží ke každému materiálu seznam ploch, které s ním souvisí. Aby bylo možné později při vykreslování určit, která část modelu používá jakou texturu.

6.3.6 Vykreslování

Aby bylo možné se ve 3D světě pohybovat a vytvářet animace, musí se vše neustále dokola vykreslovat. Každý objekt musí být vykreslen v každé smyčce.

Pro popis jednotlivých objektů, ať už samotných kartiček, panáčků, stromů a ostatních jsem se rozhodl použít jednu třídu, která se jmenuje Unit. Třída se stará o popis jednotlivých modelů, o jejich posun, rotaci a hlavně vykreslení. Vytvořil jsem seznam typu java.util.List, který je typově bezpečný. Při vytvoření List<Unit> do daného seznamu nepůjde vložit nic jiného. Do listu pak vložím všechny objekty modelů. Ve vykreslovací smyčce se tenhle seznam prochází, a vykreslí se každý model v něm uložený.

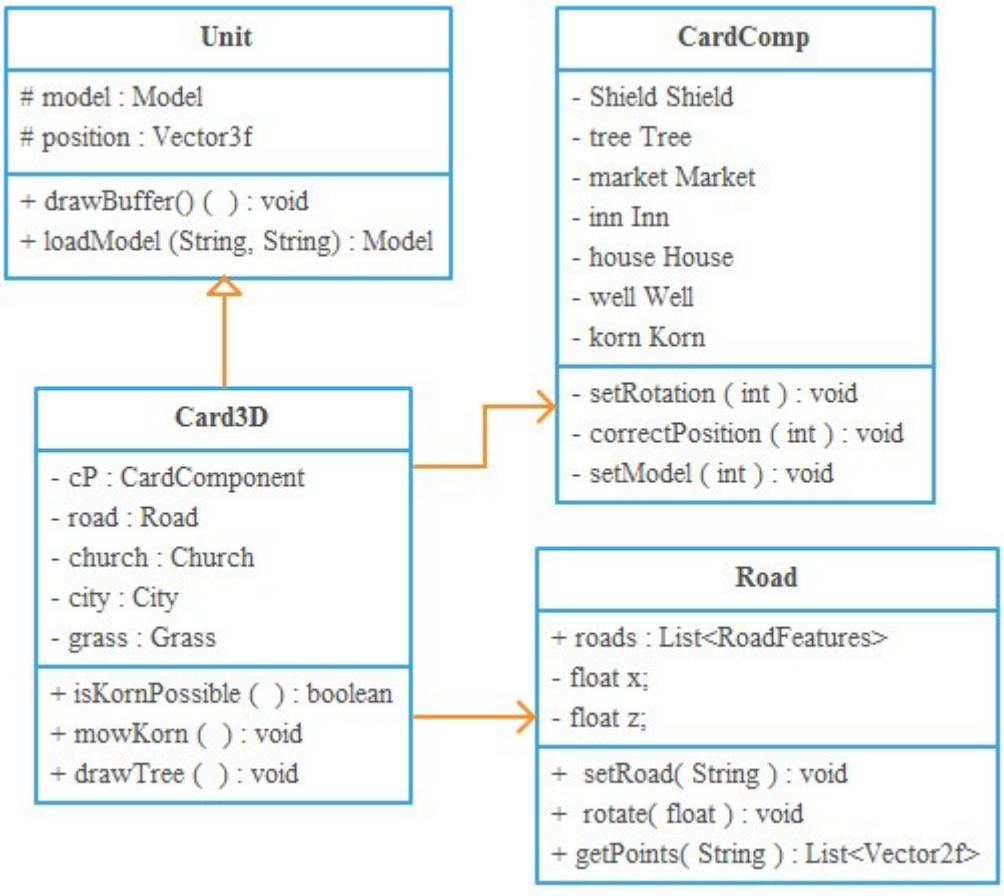
Pokud nějaký model potřebuje speciální funkce, stačí vytvořit novou třídu, která bude dědit ze třídy Unit. To zaručí, že půjde uložit do seznamu typu List<Unit>.

6.4 Model 3D kartiček

Pro modely kartiček jsem vytvořil třídu Card3D. Protože se jedná o model, tak dědí s třídy Unit. Objekty této třídy v sobě ukládají model dané 3d kartičky, pozici na herním plánu a natočení kartičky. Jaký model se má načíst a jaké má daná kartička prvky, určuje třída CardComp. Podle čísla id a další informací převzaté z původní hry. Dále podle natočení kartičky a její pozice ve hře, je potřeba upravit polohu a natočení modelu a všech prvků na něm. O to se starají funkce setRotation, a correctPosition.

Každá kartička se skládá z různých regionů. Pro každý druh regionu (cesta, město, louka, kostel) jsem vytvořil jednu třídu. Pokud kartička daný region obsahuje, vytvoří se instance dané třídy. Jedna kartička může obsahovat i více regionů stejného druhu, například křižovatka na cestě. Kde každý úsek cesty od křižovatky k okraji kartičky, představuje zvláštní region. Proto třída Road obsahuje seznam vlastností cesty.

Rozložení tříd zobrazuje následující diagram. Třídy Church, City, Grass jsou prakticky stejné jako třída Road. Liší se akorát vlastnostmi krajiny, například cesty vedou středem, město je uzavřená plocha a louky jsou velké rozlehlé plochy.



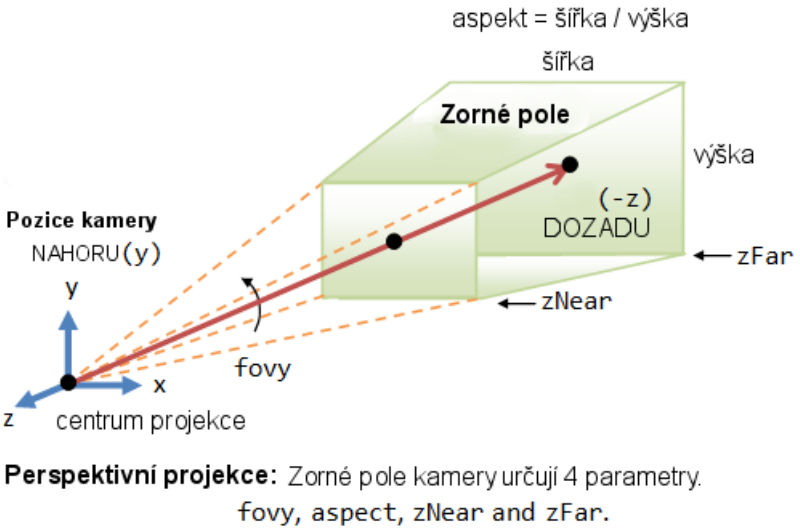
Obrázek 6: Model 3D kartiček a regionů

6.5 Přikládání kartiček a figurek

6.5.1 Systém souřadnic v LWJGL

Abych mohl umisťovat modely kartiček do 3D scény, potřebuji znát systém souřadnic. V původní hře jsou vykreslovány pouze obrázky na plochu. Střed je v bodě 1500, 1500. Ve 3D je používán ale jiný systém.

LWJGL, ale i OpenGL používá následující systém souřadnic. Předpokládejme střed scény uprostřed obrazovky. Osa x jde zleva doprava a uprostřed má souřadnici 0. Podobně osa Y jde zespodu nahoru. Osa Z je speciální, vede zpoza monitoru, a pokračuje směrem k divákovi. Osa Z určuje hloubku objektu ve scéně. Pro představu přikládám obrázek.



Obrázek 7: Systém souřadnic knihovny LWJGL

Modely kartiček jsem se rozhodl umisťovat od středu scény. Kartičky jsou pokládány na rovinnou desku. Deska má místo pro 20×20 kartiček. V podstatě je ale nekonečná, rozměry herní plochy jsou omezené původním programem, tam se vleze na plochu maximálně 48×48 kartiček.

6.5.2 Pokládání figurek

Když je do hry položená figurka, vytvoří se nová instance třídy Figure. Figurka se umístí na kartičku, do určitého regionu a zobrazí se na hracím plánu. Ve hře zůstává tak dlouho, dokud není obodovaná. Ve 3D programu přidávám figurky stejným způsobem. Když jsou aktivní, mají pod sebou barevné kolečko, podle barvy daného hráče. Když se oboduje a je odstraněna z herního plánu, tak já ji ve hře nechám, pouze odeberu barevné kolečko.

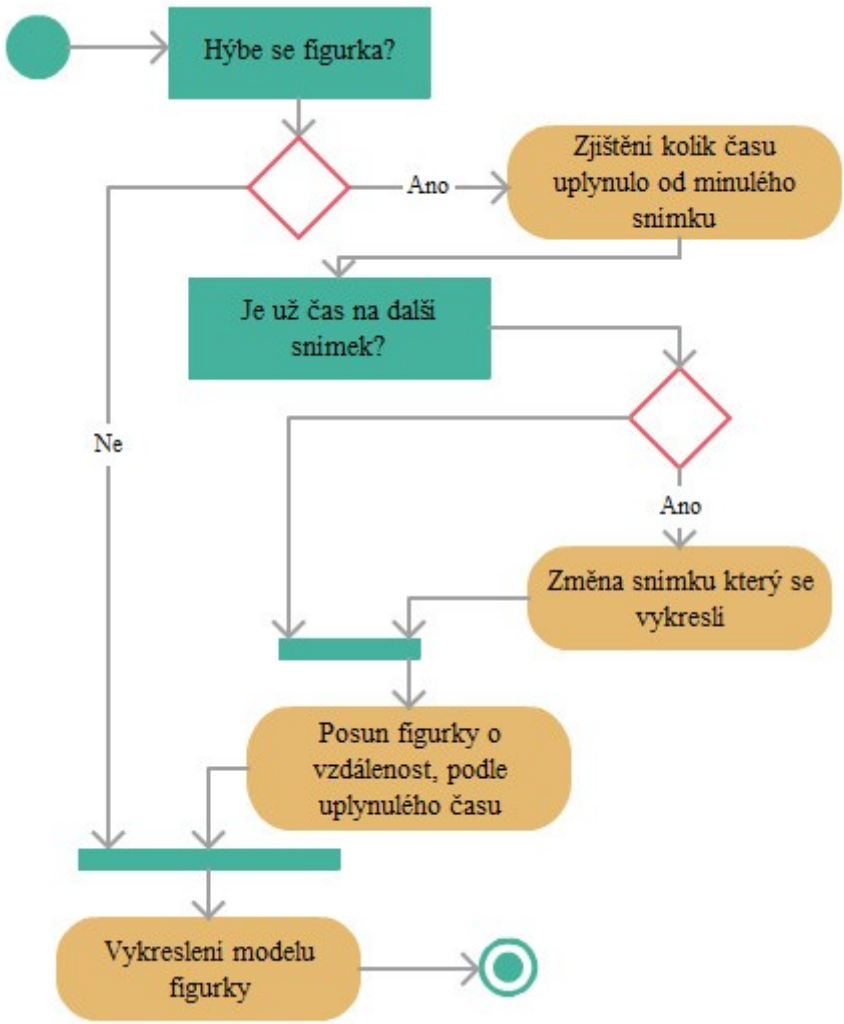
Při bodování figurky se vyskytnul jeden pro mě zásadní problém. Položená figurka je sice obodována a vrácena majiteli, ale ve skutečnosti zůstane na dané kartičce, a kartička je určena k překreslení, už bez figurky. Musel jsem trochu upravit původní hru. Do třídy jsem přidal proměnnou typu boolean která má hodnotu true pokud je figurka na herním plánu. Když se oboduje, nastavím tuhle proměnnou na false, abych věděl, že daná figurka už není aktivní.

Ve hře jsou celkem čtyři typy figurek Rytíř - do města, Zloděj – na cestu, Mnich – do kláštera, Sedlák – na louku. V klasické hře jsou všechny reprezentované stejnou dřevěnou figurkou. Ve 3D by sice bylo hezké, aby byli figurky rozdílné, ale tvorba a animace různých figurek by byla velmi časově náročná. Proto používám pouze jeden model figurek.

6.6 Animace a pohyb figurek

V klasické hře se figurky nepohybují, to asi není překvapivé. Figurka stojí na místě, kam byla postavena. Pouze označuje určitou oblast, za kterou dostane body. Ve 3D hře by statické figurky vypadali divně. Na základně pravidel lze tedy určit, kam figurka může a nechat jí chodit po určité oblasti. Pro tenhle účel jsem si stáhl z internetových stránek [21] model animované postavičky.

Musel jsem vyřešit, aby figurky neběhaly po plánu rychle, nebo aby nesmyslně kmitaly nohama. Pro každé vykreslení tak potřebuji znát čas, který uběhl od minulého snímku. Situaci každého vykreslení figurky popisuje následující diagram.



6.6.1 Systém pohybu figurek

Pro pohyb figurek, jsem se rozhodl použít systém bodů. Každý region na kartičce má definované body, do kterých se může figurka pohybovat. Třídě Mover předám objekt představující figurku na herní ploše. Podle typu regionu, na který byla figurka položena, se určí, který konkrétní Mover bude pro figurku použit. Pohyb po hrací ploše, pak zajišťuje konkrétní třída, pro cesty například RoadMover.

Na kartičce se může vyskytovat více stejných regionů. Na základě pozice umístěné figurky určím, na který region bude figurka umístěna.

6.6.2 Seznam bodů

Body na kartičce přestavují trasu pro figurku. Kdybychom je spojili, získáme lomenou čáru, vedoucí z jedné strany kartičky na jinou. Na začátku je figurka umístěna na první body dané cesty. Informace o pozici figurky, natočení, body trasy a to jestli daná figurka čeká, si ukládám do objektu třídy MovHelper. Jak už název napovídá, je to pomocník pro pohyb. První bod, si uložím do proměnné start. Další body cesty si uložím do proměnné end. Spojím li tyto dva body, dostanu vektor. Pro určení vektoru používám funkci arkus tangens. Znám souřadnice dvou bodů, které stačí odečíst a po zavolání funkce Math.atan() dostanu úhel vektoru. Funkce ale pracuje pouze v intervalu od -90° do 90, což vyplívá z její definice. Pro pokrytí celého kruhu potřebuji rozsah 360°. Když po odečtení souřadnic dostanu záporný výsledek, přičtu k výsledku funkce 180. To pokryje zbylých 180° (90°- 270°). Podle výsledku nastavím natočení dané figurky a nechám jí pohybovat.

Vzdálenost figurky od cílového bodu nastavím na maximální hodnotu, kterou povoluje proměnná v programu. Pro typ double například Double.MAX_VALUE. Při každém vykreslení figurky, tj. při každém snímku, kontroluji vzdálenost mezi pozicí figurky a cílového bodu. Pokud se vzdálenost vzhledem k předchozí zmenšila, je všechno v pořádku. Novou vzdálenost si uložím a vše pokračuje normálně. V momentě, kdy je zjištěna větší vzdálenost než ta poslední figurka dosáhla cílového bodu. Tohle opatření používám, aby byl bod jakoby určitá oblast. Stačí, když figurka projde kolem, nemusí se trefit přesně na bod. To by bylo asi nereálné.

6.6.3 Konec seznamu

Když figurka dosáhne cílového bodu, nastavím ze seznamu další bod jako cílový. To pokračuje tak dlouho, dokud se nedostanu nakonec seznamu. Když figurka dojde na konec, musím zjistit, kde se nachází. Cesty vedou ve většině případů z okraje kartičky na jiné okraj. Pokud ne, tak alespoň jeden konec je na okraji. Výjimku tvoří některé louky, které jsou kolem obestavěné městy.

Pro zjištění kde se figurka nachází, postupně porovnávám její pozici s definovanými body okraje. Po nalezení pozice následuji zjištění, jestli s okrajem sousedí už nějaká další kartička. Pokud sousední kartička existuje, pak na ní musí být stejný region, jako je aktuální. Pravidla totiž nepovolují přiložit kartičku jinak, než aby okraje pasovaly. Pokud tedy sousední kartička existuje, najdu na ní příslušnou cestu, a figurce nastavím další bod z nové cesty. Když kartička neexistuje, figurka se zastaví, na chvíli počká a vydá se zpátky stejnou cestou. To se děje i v případě, kdy cesta končí uprostřed kartičky, například na křižovatce.

Na cestách je možný pouze jeden směr pohybu, respektive dva. V malých městech jsou pohyby taky omezené. Na větších kartičkách měst a na louce existuje více cest. Figurka si vždy zvolí náhodnou cestu, kterou se pak vydá.

6.6.4 Obilí na loukách

Na některých loukách může růst obilí. Model obilí jsem nevytvářel, místo toho jsem použil pouze žluté tyčinky, jako reprezentaci pole s obilím. Obilí nějaký čas roste, ve hře jsem rychlost hodně zrychlil, aby byly vidět nějaké změny. Když obilí vyroste, nastavím na dané kartičce informaci o tom, že je obilí k dispozici. Figurka, které chodí po louce, může k obolí přijít a posekat ho. Sekání jsem znázornil otočením figurky v místě obilí a následným zmizením. Vše se děje náhodně, takže figurka může delší dobu chodit kolem a obili pouze pozorovat.

6.7 Popis a ovládání programu

Pro ovládání programu jsem použil způsob známý z jiných počítačových her. Pohybovat se je možné pomocí šipek na klávesnici, nebo pomocí kláves W, S, A, D, které představují šipky, W = nahoru. Pomocí mezerníku se lze pohybovat nahoru, a pomocí klávesy X lze klesat. Natáčení kamery se provádí myší. Po stisknutí pravého tlačítka se lze volně rozhlížet po krajině.

Pro zrychlení pohybu kamery stačí přidržet klávesu Ctrl, a pro zpomalení naopak klávesu Shift.

Obrázek 8: Náhled na rozehranou hru

6.8 Návaznost na již existující bakalářskou práci

Má bakalářská práce navazuje na již existující bakalářskou práci. Ze začátku nebyla implementace do původní hry vůbec jednoduchá. Po pochopení všech mechanismů byla už implementace jednoduší.

7 Závěr

Cílem této bakalářské práce bylo seznámit se s možnostmi 3D grafiky v jazyce Java. Poznat jaké existují knihovny pro podporu tvorby her. Seznámit se s problematikou 3D počítačové grafiky, pochopit základy a na nich stavět.

Zvládnout základy počítačové 3g grafiky nebylo jednoduché. Myslím si, že jsem se v této oblasti posunul ze začátečnické úrovně minimálně na středně pokročilou. V některých případech jsem musel řešit složitější věci, které nebyli triviální a už vůbec ne na začátečnické úrovni.

Při programování jsem se snažil dodržet přehlednost kódu pro případné další rozšiřování hry. Dále jsem se snažil ovládání hry navrhnout tak, aby se příliš nelišilo od jiných her. Použil jsem získané zkušenosti z jiných počítačových her.

Hra má základní grafiku, nejedná se o nic výjimečného. Modely jsem vytvářel, aby měli pouze informativní charakter o herní krajině. Grafická stránka hry by se později upravit. Modely kartiček by šli přepracovat. A to buď jen samostatně, nebo potom následně upravit základ celé hry. Pro každý typ figurek by se dal vytvořit zvláštní model, kterému by se určilo zvlášť chování na různých typech krajiny a v různých situacích.

Výsledkem mojí práce je 3D náhled na herní plochu hry Carcassonne. Herním světem se dá létat jako divák a pozorovat chování figurek, které se v něm pohybují.

Hra je nyní pouze jako náhled, který se spustí současně s původní hrou. Hra by se dala osamostatnit použitím technologie RMI. Uživatel by se připojil jako divák na server a mohl by sledovat průběh hry.

Další variantou by bylo předělat celé ovládání do 3D programu. Kartičky by se přikládali po kliknutí myši na herní plochu a stejně tak by se mohli umisťovat figurky.

8 Seznam použité literatury

[1] OpenGL. *Wikipedia, The Free Encyclopedia.* [Online] 4. 4 2015. [Citace: 24. 04 2015.] http://en.wikipedia.org/wiki/OpenGL.

[2] GL4Java. [Online] 11. 12 2001. [Citace: 24. 4 2015.] http://gl4java.sourceforge.net/docs/.

[3] Java 3D. [Online] [Citace: 24. 04 2015.] http://www.java3d.org/.

[4] Java OpenGL. *Wikipedia, The Free Encyclopedia.* [Online] 13. 04 2015. [Citace: 24. 04 2015.] http://en.wikipedia.org/wiki/Java_OpenGL.

[5] LibGDX. [Online] 2013. [Citace: 24. 04 2015.] http://libgdx.badlogicgames.com/index.html.

[6] LWJGL. *Home of the Lightweight Java Game Library.* [Online] 18. 01 2015. [Citace: 24. 04 2015.] http://legacy.lwjgl.org/.

[7] 3DzzD. [Online] 08. 02 2014. [Citace: 24. 04 2015.] http://dzzd.net/.

[8] Ardor3D. *Wikipedia, The Free Encyclopedia.* [Online] 23. 12 2014. [Citace: 24. 04 2015.] http://en.wikipedia.org/wiki/Ardor3D.

[9] Autiga3D. [Online] 4. 8 2004. [Citace: 24. 4 2015.] http://auriga3d.tribe.net/.

[10] Irrlicht Engine. [Online] [Citace: 24. 04 2015.] http://irrlicht.sourceforge.net/.

[11] Env3D. [Online] 2011. [Citace: 24. 04 2015.] http://env3d.org/beta/.

[12] Jake2. [Online] 12. 05 2007. [Citace: 24. 04 2015.] http://bytonic.de/html/jake2.html.

[13] Java is Doomed. [Online] 08. 04 2013. [Citace: 24. 04 2015.] http://javaisdoomed.sourceforge.net/.

[14] JMonkeyEngine. *Wikipedia, The Free Encyclopedia.* [Online] 17. 04 2015. [Citace: 24. 04 2015.] http://en.wikipedia.org/wiki/JMonkeyEngine.

[15] jPCT. [Online] 03. 11 2014. [Citace: 24. 04 2015.] http://www.jpct.net/.

[16] ogre4j. [Online] 05. 06 2009. [Citace: 24. 04 2015.] http://ogre4j.sourceforge.net/.

[17] Xith3D. [Online] 06. 09 2010. [Citace: 24. 04 2015.] http://www.xith.org/.

[18] Jzy3D. *Jzy3d - Scientific 3d plotting.* [Online] [Citace: 24. 04 2015.] http://jzy3d.org/.

[19] Processing. [Online] [Citace: 24. 04 2015.] https://processing.org/.

[20] **Grosso, William.** *Java RMI.* Sebastopol : O'Reilly Media, Inc., 2001. ISBN 978-1-449-31535-1.

[21] 3D Models for Professionals. *TurboSquid.* [Online] http://www.turbosquid.com/.

9 Přílohy

- CD ROM

Obsah CD ROM

- Bakalářská práce
- Zdrojové kódy původní hry a herního serveru
- Zdrojové kódy 3D hry

Tento CD ROM je jako příloha k bakalářské práci Modul pro 3D herní plochu hry Carcassonne.