Mendelova univerzita v Brně
Provozně ekonomická fakulta

Tvorba volně dostupné naučné 3D aplikace

Diplomová práce

Vedoucí práce:

Ing. Jaromír Landa, Ph.D.

Bc. Michaela Portlová

Brno 2017

Děkuji Ing. Jaromíru Landovi Ph.D. za vedení mé práce. Také bych ráda poděkovala Lence Chalupové a Tomáši Šimíčkovi, za cenné připomínky a psychickou podporu, a celé mé rodině za veškerou poskytnutou oporu.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Tvorba volně dostupné naučné 3D aplikace** vypracoval/a samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom/a, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 2. února 2017 _____

Abstract

Portlová, M. Creation of freely available educational 3D application. Diploma thesis. Brno: Mendel University, 2017.
This thesis deals with the creation of education 3D application, which have an educational and entertaining character for the user. The thesis including the analysis and selection game engine, edit models and scenes, create necessary functions in the chosen game engine, used storage and implementation of the website.

Keywords

3D computer game, web application, architecture, database, database system, game engine, jMonkeyEngine, JME3, JME SDK, Java, XML, 3D scene, license.

Abstrakt

Portlová, M. Tvorba volně dostupné naučné 3D aplikace. Diplomová práce. Brno: Mendelova univerzita v Brně, 2017.
Tato práce se zabývá tvorbou naučné 3D aplikace, která má mít naučný a zábavným charakter pro uživatele. Popisuje analýzu a výběr game enginu, úpravu modelů a scény, tvorbu potřebné funkcionality ve vybraném game enginu, použitá uložiště a implementaci na web.

Klíčová slova

3D počítačové hry, webová aplikace, architektura, databáze, databázový systém, game engine, jMonkeyEngine, JME3, JME SDK, Java, XML, 3D scéna, licence.

Obsah

1 Úvod a cíl práce

1.1 Úvod

Technologie, aplikace, internet, hry, virtuální realita a mnoho dalších jevů poukazuje na to, jak se svět neustále rychle posouvá kupředu, modernizuje a umožňuje vývoj nových možností. Ale nezapomíná přitom na staré? Je pravdou, že vývoj moderních technologií napomáhá v odhalování historických faktů. Počítačové scanery umožňují uchovávat staré dokumenty. Vylepšené ultrazvuky pomáhají nalézt staré tunely a pohřebiště a pomocí nejmodernějších přístrojů provádět analýzu nalezených objektů a pozůstatků. Ale neubývá lidí, které historie zajímá?

Sjednocení těchto dvou elementů může znít bláznivě, ale současná doba právě toto dělá. Ať už v muzeích, kde je historie a věda ukazována a osvětlována na moderních přístrojích nebo i v herním průmyslu, kde se hráči rádi ponořují do historických, i když často také vymyšlených příběhu.

Snahou této práce bude spojit reálnou historii jednoho městečka v Českomoravské vrchovině s moderní technologií, pokud možno co nejzábavnější formou. Vytvořením hry pro všechny generace tak, aby si mohli zjistit či připomenout, jaká je historie jejich města. Úsilím je také dát návod ostatním, jak takové hry vytvářet.

K vytvoření takovéto "naučné aplikace" bude vybrán game engine vybraných kvalit v průběhu práce. Aby ji mohl vyzkoušet každý, bude umístěna ve webovém prohlížeči.

Předností dnešních game engine je schopnost využívání real-time, tedy možnost zobrazování scény a její změny v reálném čase. Umožní tak hráči si hru náležitě užít, ať již u počítače (do míry, kterou dovoluje) nebo na vyšší úrovni přes virtuální realitu ve 3D brýlích. Pro lepší tvorbu i hratelnost je tedy zapotřebí modernější hardware (3D brýle, Xbox a ovládání nebo výkonný počítač). Pro účely této práce postačí základní počítačové rozhraní, softwarové vybavení i grafické zpracování.

1.2 Cíl práce

Cílem diplomové práce je vytvoření volně dostupné naučné a propagační 3D hry a to formou webové aplikace, pro snadnější přístup uživatelů. Hra bude existovat také v desktopovém provedení, pro uživatele bez přístupu k internetu, který je již v dnešní době téměř samozřejmostí. Nicméně existují případy, kde to samozřejmostí není, tak ať zájemce není ochuzen.

Dílčím cílem práce je provést analýzu vybraných game engine a jejich porovnání využitelnosti v různých situacích, návrh architektury internetové aplikace a databáze.

Výsledek této práce by měl představit a přiblížit osobám bez rozdílu věku historii města Poličky v záživnější podobě. A také tím mít částečný přínos i pro muze-

um v Poličce, kde by mohlo být očekáváno zvednutí zájmu o historii města v reálné formě. Snahou aplikace je také ukázat zajímavosti města, a tedy vybídnout mladší generaci, že učit se a zkoumat historii není nudná záležitost.

2 Vývoj her

Část Vývoj her se zaměřuje na teoretické informace o 3D grafice, game enginech, jejich architektuře a základních pojmech využívaných ve vývoji her.

Spojení „počítačová hra“ obvykle vyvolává představy o virtuálním světě, kde je hlavní „postavou“ osoba, zvíře nebo vozidlo pod kontrolou hráče. Existují různé typy her, jako jsou 1st person, 3rd person, taktické, 2D a 3D adventury, textové adventury, strategie, simulátory a další. U některý těchto typů her mohou mít formy online a multiplayer. V současné době existuje tolik typů her, že kombinacemi různých typů bude pravděpodobně jen jiný typ. Každá s her je postavena na jiné počítačové grafice. Obvykle se jedná o dvou nebo tří dimenzionální.

2.1 3D grafika pro hry

Základní pojmy ohledně 3D počítačové grafiky byly obecně probrány v mé bakalářské práci „Tvorba interaktivních průchodů v programu Blender„. Práce obsahovala informace o základních objektech (křivky, plochy a tělesa), světle, jeho zdrojích a odrazech, texturách, interaktivitě, kolizích a jejích detekcí. Bakalářská práce obsahovala základní náhled na problematiku v 3D grafice. V diplomové práci budou rozebrána některá z daných témat trochu podrobněji, ale hlavně vhledem k počítačovým hrám.

Hra je matematický model virtuálního světa simulovaného v reálném čase na některém zařízení (PC, mobilní zařízení, PS apod.). Proto matematika prakticky ve všech odvětvích (trigonometrie, algebra, statistika atd.) prostupuje vším v herním průmyslu.

Viditelná scéna ve hře se obvykle skládá ze dvou druhů dat. Základním jsou geometrická tělesa (brush geometry) definována jako kolekce konvexních obalů, z nichž je každá definována pomocí několika rovin. Obvykle jsou vytvářeny přímo v editoru herního enginu. Rychle se vytvářejí a jejich vykreslování je snadné. Nevýhodou je jejich nízké rozlišení a nemožnost vytváření složitých tvarů.

Dalším druhem jsou 3D modely (meshe). Jsou vhodné pro složitější objekty a detailní scény. Mesh objekt je komplexní tvar reprezentovaný trojúhelníky a vrcholy. Pro využívání na grafickém hardwaru současné doby je každý model, který není typu mesh, nutné převést. 3D model je možné získávat z různých grafických nástrojů (Maya, 3ds Max, atd.) a exportovat ho v široké škále možných formátů. Obvyklé formáty často nejsou dokonale vhodné pro vývoj her, a proto si mnohdy tvůrci game enginu vytvářejí vlastní formáty vhodné přímo pro jejich engine. (Gregory, 2009)

Mezi nejvyužívanější geometrické metody používané ve hrách patří posunutí (translation), otáčení (rotation) a změna měřítka (scale). Všechny tyto funkce jsou v matematice řešeny tzv. transformačními maticemi typu 4 x 4. Ve 3D je posunutí určeno vektorem posunutí a transformováno transformační maticí posunutí a inverzní maticí. Při otáčení kolem jednotlivých souřadných os se využívá pro reprezentaci otočení daná osa, úhel a inverzní matice. Při změně měřítka

v trojrozměrném prostoru se využívá transformační matice, v níž koeficienty na diagonále (z leva doprava) určují změnu ve směru dané souřadnicové osy.

Kvaterniony (Quaternions) je alternativní matematický subjekt, který je ve vývoji 3D her používám k reprezentování rotace. Použití kvaternionů má v mnoha situacích velkou výhodu, oproti použití rotačních matic. Ty obsahují nadbytečné údaje a jsou obtížně interpolovatelné. Otáčení do více směrů zároveň je za využití interpolovaného otáčení velmi složité. Zřetězení kvaternionů vyžaduje méně aritmetických operací, čímž snadněji vytvářejí hladkou animaci rotace.

Většina současných game enginů má vnitřní nebo externí možnost knihoven pro práci s vektory, maticemi nebo kvaterinony. Ty lze využít i jinak než na posun, rotaci a změnu velikosti. Vektory lze zužitkovat například na mapování textur (texturovací souřadnice), určení pozice ve scéně, určení barvy (tři složky), případně s alfa kanálem (čtyři složky). (Žára, 2004)

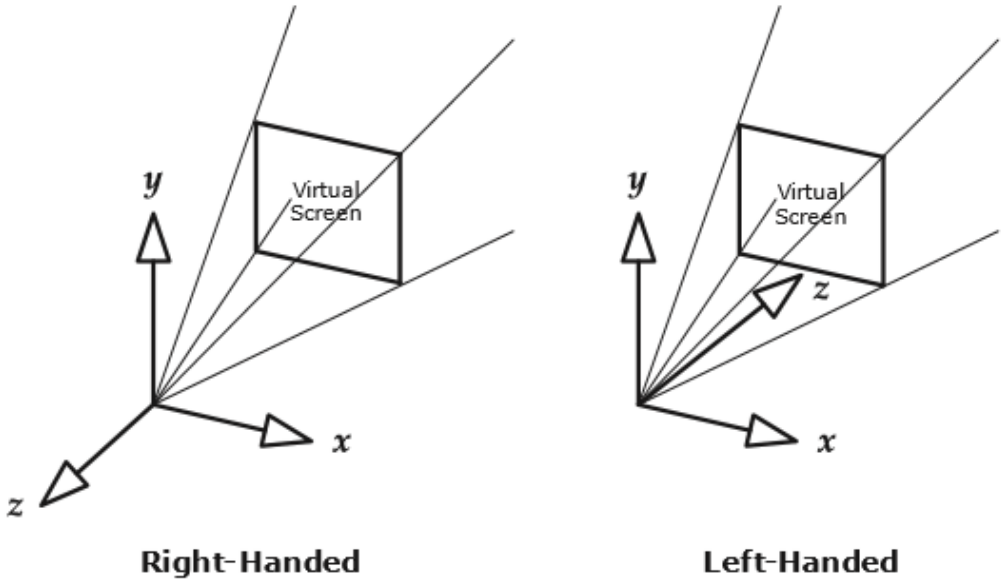
2.1.1 Prostor

Poloha všech bodů ve scéně je vždy vyjádřena relativně vzhledem k sadě souřadných os. Při vývoji her se používá různé sady souřadnicových prostorů (tzv. space). Mezi nejznámější patří model space, world space a view space.

Model space je také nazývám jako prostor objektu nebo lokální prostor. Je specifikovaný kartérským souřadným systémem. Obvykle je umístěn na centrálním místě uvnitř objektu.

World space reprezentuje souřadnicový prostor, který určuje pozice, orientace a měřítka všech objektů v herním světě. Umístění world space je libovolné, ale převážně se lokalizuje v blízkosti středu. Libovolná je i orientace os (*x*, *y*, *z*). V enginech se lze setkat s tzv. *y-up* nebo *z-up*. V herním světě je častější y-up, který má osu y znázorněnou vzhůru. *Z-up* umožnuje pravoúhlý pohled na herní svět a tak vytvořit klasický dvourozměrný pohled na *xy*-plochu.

View space je souřadnicový rámec přichycený ke kameře (camera space). Jeho centrální bod je umístěn v centru kamery. Stejně jako u předešlého je prostoru je i u view space možné nastavení libovolné orientace os. V herním světě se převážně využívá *y-up* konvence s tzv. systémem levé ruky (left-handed), která umožnuje ose *z* reprezentovat hloubku obrazovky. Jsou enginy (OpenGL) a aplikace, které využívají opačný systém, tedy pravé ruky (right-handed), který představuje zápornou hloubku osy *z*. Oba případy lze vidět na následujícím obrázku (Obr. 1). (Gregory, 2009)



Obr. 1 Left-handed a right-handed systém zobrazení prostoru[1]

2.2 Game engine

V základu je vhodné uvést rozdíl mezi hrou a herním enginem (GE). Hra je kompletní aplikací skládající se z částí jako modely, animace, zvuk, fyzika a specifický kód určující funkcionalitu, tedy například ovládací prvky a umělou inteligenci. Zatímco game engine je softwarový framework, který vývojáři, ale také umělci využívají k vytvoření a rozvoji interaktivních aplikací, jako jsou hry, průchody a interaktivní animace. Jeho podstatným přínosem je, že poskytuje sadu vizuálních vývojových nástrojů. Tyto nástroje jsou obvykle poskytovány v integrovaném vývojovém prostředí jiné interaktivní aplikace s real-time grafikou k tomu, aby zjednodušili a zrychlili vývoj her. Nástroje umožnují vytvořit interakci s uživatelem a vykreslovat scény v reálném čase. (Mitra, 2009)

Vývojáři jen používají k vytváření her pro konzole, osobní počítače a mobilní zařízení. Herní enginu typicky obsahuje vykreslovací jádro („render“) pro 2D nebo 3D grafiku, detekce kolizí nebo fyzikální engine, zvuk, skriptování, animace, umělou inteligenci, síťové prvky, streamovaní, podpora lokalizace, graf scény a jiné. Cílem game engine je být univerzální a přizpůsobivý, aby bylo možné na jednom enginu vytvořit více různých her pro různé platformy. (CTI Reviews, 2016)

Jednotlivé enginy jsou často přizpůsobeny k určitému žánru a poté se liší i jejich specifikace. Rozdílné požadavky jsou kladeny například při masivně multiplayerové online hry (MMOG), střílečky z prvního pohledu (FPS) a nebo real-time strategie (RTS). Důležité pro členění je také forma uživatelského vstupu a výstupu.

[1] Zdroj: (Gregory, 2009)

V základním rozdělení se dělí žánrů her na FPS, hry z pohledu třetí osoby, bojové hry, závodní hry, real-time strategie, masivní multiplayerové online hry, ostatní žánry (Tetris). (Gregory, 2009)

Jednotlivé enginy se liší v kvalitě a nabízených funkcích, čemuž se práce více věnuje níže v kapitole „Analýza game engine“, kde jsou i blíže rozebrány některé vybrané game enginy

2.3 Architektura hry

Architekturu hry případně game enginu lze pojmout z různých pohledů. Jeden z nich je popsán v knize (Gregory, 2009). V této práci je k těmto principům částečně nahlíženo a využíváno, proto je uvedena krátká charakteristika architektury a prvků ze zmíněné knihy.

Game engine se obvykle skládá ze sady nástrojů a runtime komponenty. Obr. 28 ukazuje všechny runtime komponenty, které tvoří typický 3D game engine. Stejně jako softwarové systémy je i architektura game engine postavěna ve vrstvách. Vždy je třeba, aby horní vrstvy byly závislé na spodních a ne naopak. Pokud by tomu tak bylo jednalo by se o tzv. kruhovou závislost, která by vedla k nežádoucím spojením mezi systémy, netestovatelnosti softwaru a omezila opětovné použití kódu. Proto je třeba se kruhové závislosti vyhnout.

Pro tvorbu her ve velkém je důležité dobře a do hloubky pochopit jednotlivé složky architektury a naučit se, jak tyto součásti integrovat do funkčního celku.

Dále je uveden základní přehled složek schématu neboli komponenty, na které by se měl vývojář zaměřit při tvorbě hry. Při popisu jsou také vysvětleny některé základní pojmy důležité pro tvorbu her pro lepší orientaci v oblasti.

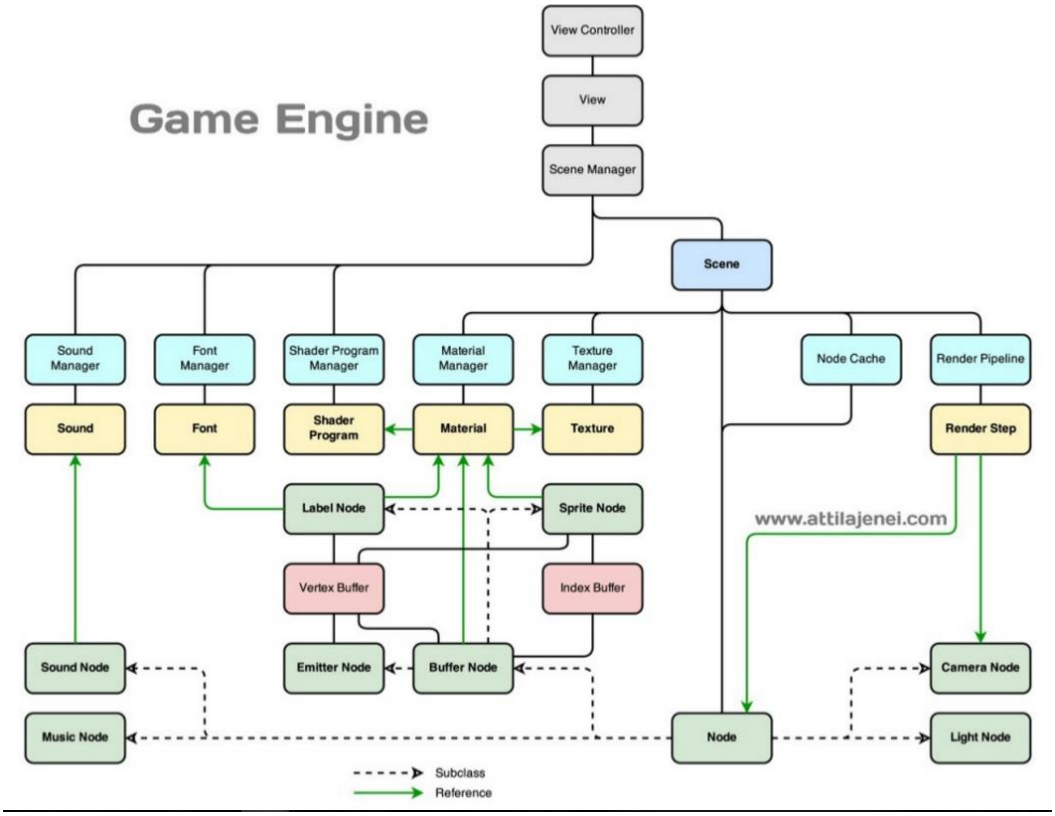
- **Hardware** – vrstva cílového hardwaru (PC, XBOX, PS, mobilní zařízení atd.), určuje, na jaká zařízení a platformu (Linux Microsoft Windows, Apple Xbox atd.) by měl být výsledný produkt určen.
- **Ovladače** – jedná se o komponenty nízké úrovně softwaru poskytované operačním systémem nebo dodavatelem hardwaru.
- **Operační systém** na počítači instrumentuje přístup více spuštěných programů ke sdílení hardwaru na jednom počítači (tzv. preemptivní multitasking). Program tak nikdy nemá plnou kontrolu nad hardwarem. Dříve byl operační systém na konzolích pouze nepatrnou částí v knihovně a hra tak mohla ovládat celý přístroj. V současné době se operační systém na konzolích rozšiřuje a může například přerušit hru kvůli zobrazení online zprávy apod.
- **Softwary třetích stran** (3rd Party SDKs) jsou produkty vytvořené jinými výrobci než tvůrcem samotného game engine. Často se jedná o různé vývojová prostředí (SDK) a knihovny, které poskytující například datové struktury (STL), grafický renderovací engine (DirectX, OpenGL), detekci kolizí a fyziku těles (Havok, PhysX, ODE), animaci (Granny, Havok animation, Edge), umělou inteligenci (Kynapse).

- **Platformy** – čím větší má game engine nezávislost na platformách, tím má větší konzistentnost chování napříč hardwarovými i softwarovými platformami a tedy i využitelnost na trhu. Většina dobrých game enginu se snaží zacílit na co největší množství game engine.
- **Jádro systému** každého game engine by mělo poskytovat velké množství užitečných softwarových nástrojů. Jde například o matematickou knihovnu, video přehrávač, správu paměti, alokaci paměti, generátor náhodných čísel a mnoho dalších.
- **Zdroje** (Game assets) – zahrnují potřebné soubory a data v enginu. Některé game enginy obsahují přímo správce zdrojů, který je rozhraním pro přístup k jakýmkoliv typům souborů a dalším datům enginu. Jedná se o 3D modely, textury, materiály, fonty, kolize, fyziku a jiné.
- **Render engine** – je jedním z největších a nejsložitějších složek herního enginu. Game enginy využívají pro renderování grafických nástrojů, jako jsou DirectX a OpenGL. Vykreslování se u jednotlivých enginu mírně liší, ale často mají podobný základ. Základní a efektivní vykreslovací způsob je nízko úrovňové vykreslování (*low-level renderer*), který pracuje se základními geometrickými primitivi a umožnuje pracovat co nejrychleji bez kontroly toho, co se vykresluje do scény. Low-level renderer vykresluje veškerou geometrii na scéně, i když není potřebná nebo vidět. Pro optimalizaci se využívají různé prostředky. U malých herních světů se využívá odstranění objektů mimo záběr kamery. U velkých je pro zlepšení efektivnosti vykreslování používán systém vyspělejších datových struktur pro prostorové dělní. Mezi podoby prostorového dělení patří například BSP Tree, kd-tree, QuadTree nebo prostorová hierarchie. Prostorové dělení je někdy nazýváno jako *graf scény*. Jedná se druh datové struktury, která pomáhá stanovit viditelnost objektů na scéně.
- **Vizuální efekty** – nabízí většina moderních game enginů. Efekty se využívají v podobě jako částicové systémy (kouř, oheň, kapky vody apod.), systém otisků (díry po kulkách, stopy apod.), dynamické stíny, mapování prostředí, mapování světel a mnohých dalších.
- **Front end** využívá 2D grafiky překrývající 3D scénu pro různé účely. Zahrnují například *heads-up display* (HUD), který slouží pro zobrazování informací a údajů o hráči (často získané body, mapa, odehraný čas, životy, počet nábojů, používaná zbraň nebo tachometr). Dále *graphical user interface* (GUI), které dovoluje hráči provádět složité úkony ve hře (konfigurace jednotky k boji aj.), a *in-game menus*, konzole a další vývojové nástroje. Front end zahrnuje také přidaná videa jako *full-motion-video* (FMV) nebo *in-game cinematics* (IGC).
- **Profilování a ladící nástroje** se využívají pro optimalizaci výkonu a paměťových zdrojů při real-time hrách. Zahrnují nástroje jako in-game ladění zařízení, ladění vykreslování a možnost nahrávání a přehrávání hry pro testování a ladění.

- **Kolize a fyzika** jsou obvykle velmi pevně propojeny. Při detekci kolize je téměř vždy řešena i fyzika objektů. Detekce kolizí je nezbytnou součástí každé hry. Bez jejího využití by docházelo k průchodům tělesy navzájem. Fyzika systému, spíše nazývána jako dynamika tuhých těles, slouží pro realistickou nebo polo-realistickou dynamiku simulace. Většinou se jedná o pohyb tuhého tělesa a síly (kinematika) a momenty (dynamiky), které způsobují nastání pohybu.
- **Animace a Audio** se používají pro reálnost scény. Animace by měl osahovat každá hra zahrnující organické nebo polo-organické prvky (zvířata, lidi, roboty atd.). Specifikuje se pět základní typů animací ve hrách, mezi které patří sprite nebo texturová animace, animace hierarchie tuhého tělesa, morph, vertex animace nebo kosterní animace. Audio je často přehlížený, ale ne méně důležitý, komponent hry. Mezi zvukové sady nástrojů (audio engine) patří XACT, SoundR!OT nebo Scream (3D audio engine).
- **Vstupní a výstupní** (I/O komponenty) **zařízení** jsou nutnou součástí každé hry pro komunikaci s uživatelem (**HID**). Základní vstupní zařízení jsou klávesnice, myš, joypad. V rozšířené verzi se může jedna o volanty, taneční podložky WiiMote a mnoho dalších. Některé z nich jsou přímo I/O komponentami, kde například joypad vrací vibrace nebo WiiMote zvuk. Někdy také zahrnuje detekci akordů (více tlačítek najednou), sekvence (postupná kombinace tlačítek) a gest.
- **Multiplayer (online)** umožnují více hráčům (herním postavám) obývat jeden virtuální svět. Existují čtyři základní přístupy. *Single-screen multiplayer*, který zahrnuje více vstupních zařízení připojených k jednomu přístroji (jedna kamera a jedna obrazovka). *Split-screen multiplayer* zahrnuje více vstupních zařízení (více kamer a rozdělení obrazovky do sekcí). *Networked multiplayer* obsahuje více výstupních zařízení propojených k sobě, kde každý přístroj ovládá jeden hráč. *Massively multiplayer* online game (MMOG) je on-line virtuální svět na výkonném centrálním serveru s nekonečným množstvím hráčů.
- **Herní základy** – poskytují sadu základních zařízení, které specifikují hru a její logiku. Akce, které jsou prováděny ve hře, pravidla virtuálního světa, mechanika hráče, záměr a cíl hráče a další postavy a objekty na scéně. Stanovuje, zda je hra vytvářena v základním jazyce enginu, ve kterém je i napsán i game engine, nebo v nadstaveném skriptovacím jazyce. Nejvhodnější je pokud má game engine jazyk implementovaný v rozhraní. Zahrnuje herní svět a objekty v něm obsažené (statické geometrie – silnice, terén, budovy atd., dynamická tuhá tělesa – nábytek, kamení atd., herní postava, automobily, světla, kamery aj.), události systému (komunikace objektů mezi sebou – grafické herní menu), umělá inteligence.
- **Herní podsystémy** – jsou vrstva, která se většinou příliš neliší. Některé herní specifické znalosti trvale prosakují přes vrstvu herních základů. Někdy až k hernímu jádru. Herní systémy jsou obvykle početné a rozmanité. Tyto systémy zahrnují různé kamerové systémy, umělou inteligenci, zbraňové systémy, vozidla, mechaniku hráče a mnoho jiných.

(Gregory, 2009)

Architekturu hry lze pojmout také z hlediska velmi odlišného, a to podobně jako při pohledu na základní diagram tříd (Obr. 2). V toto případě se architektura skládá z hlavních tříd, pod kterou spadá scéna i funkční prvky, a podtříd. View zastává funkci herních i vykreslovacích smyček a základních událostí. Scene obsahuje prvky scény a uzly pro renderování. Další zdroje zahrnují grafická data (materiál a textury, font) a zvukové zdroje. Graf zobrazuje také uzly pro jednotlivé prostředky (kamera, světlo, zvuk) včetně jejich logického propojení mezi jednotlivými složkami. V neposlední řadě zobrazuje také renderovací oddíl. (Jenei, 2014)



Obr. 2 Architektura game enginu[2]

2.4 Pojmy spojené s vývojem her

Některé pojmy byly uvedeny ve výčtu komponent architektury game engine. V této části je doplněno několik dalších pojmů.

- **Physic engine** (fyzikální engine) – je počítačový program, který simuluje Newtonskou fyziku modelů, použitím proměnných, jako je hmotnost, rychlost,

[2] Zdroj: (JENEI, 2014)

tření a odolnost proti větru. Je možné simulovat a předvídat účinky fyzikálních veličin za různých podmínek, tak aby napodobovali reálný svět. Fyzikální engine mají dvě základní součásti a to systém detekce kolizí a fyzikální simulace (dynamika tuhých těles) komponentu zodpovědná za řešení sil působících na simulovaný objet. (Mitra, 2009, str. 35)

- **Graphics pipeline** – nebo též grafický řetězec, je pojem v oblasti počítačové grafiky, kterým označujeme sekvenci procesů, jejichž aplikací na data popisující scénu získáme dvourozměrný obraz této scény. O vlastní provedení těchto procesů se stará grafický procesor (GPU), který je součástí grafické karty počítače. Získaný obraz lze následně zobrazit vhodným výstupním zařízením (např. monitor). Konkrétní podoba a chování vykreslovacího řetězce je dána některým ze standardů pro 3D grafiku. Mezi nejvýznamnější standardy v této oblasti patří OpenGL a Direct3D(součástí DirectX). (Paris, 2015)
- **Artificial intelligence** – umělá inteligence. Dříve byla umělá inteligence vytvářena specifickým způsobem pro každou hru. První univerzálním řešením byla middleware SDK Kynapse od firmy Kynogon, která poskytuje nízkoúrovňové UI stavební kameny. Nabízí například statické a dynamické vyhýbání objektu nebo rozhraní mezi umělou inteligencí a animací. (Gregory, 2009)
- **Shader** – program pro řízení jednotlivých částí grafické karty. Umožnuje ovlivňovat vykreslování scény pomocí základních vykreslovacích algoritmů. Tyto shadery se využívají v grafických engine a pokročilých renderovacích programech například pro změnu geometrie vykreslovaných ploch (pro tzv. displacement mapping), modifikaci nanášené textury, změny způsobu osvětlení těles nebo modifikaci samotného způsobu vykreslování. Dá se mluvit o 3 typech, kterými jsou vertex shader, geometry shader a pixel shader. (Tišnovský, 2005)
- **OpenGL** – knihovna, představující multiplatformní standart pro tvorbu 2D a 3D grafických aplikací. Příkazy knihovny OpenGL lze použít v na různých platformách při různých programovacích jazycích. Byl vytvořen jako API k akcelerovaným grafickým kartám. Je velmi rychlý a flexibilní a lze pomocí něj vytvářet 3D grafiku s vysokou vizuální kvalitou. (Tišnovský, 2003)
- **DirectX** – grafika od firmy Microsoft. Obsahuje podporu pro vysoce výkonnou 2D a 3D grafiku. Poskytuje sadu rozhraní API, která může být použita k vytváření her a jiných výkonných multimediálních aplikací. (Microsoft, 2017)
- **LWJGL** (Lightweight Java Game Library) – knihovna napsaná v jazyce Java, která umožnuje multiplatformní přístup k nativnímu API. Je vystavěna na OpenGL. Používá se při vývoji grafických (OpenGL), audio (OpenAL) nebo paralelních výpočetních (OpenCL) aplikacích. Obsahuje vše potřebné pro vytvoření 2D i 3D grafiky s využívá se v různých game enginech. (LWJGL, 2012)
- **Real-time** je možnost vykreslovat scénu v reálném čase na výstupním zařízení uživatele. Při pohybu hráče ve 3D scéně se celý obsah obrazovky neustále

mění. Při vykreslování v reálném čase se využívá stejný princip jako u filmů, kde v rychlém sledu za sebou předloží divákovi sérii statických snímků. Pro nestálé opakování využívají hry tzv. rendrovací smyčku (rendering loop).

- **Loop** (neboli smyčka) zajišťuje obnovu subsystémů.. Jedná se například o renderování, detekce kolizí, fyzika těles, zvuk, multiplayer atd. Obvykle se rychlost aktualizace pohybuje mezi 30 až 60 Hz. Dynamika simulace se může měnit s každým subsystémem. Některým subsystémům stačí obnovovat méně často a některé naopak častěji. Herních smyček může být v jedné hře i více (i pro každý subsystém).

3 Webové aplikace a databáze

Tato kapitola popisuje teoretické základy webových aplikací, jejich architekturu a bezpečnost. Také se věnuje databázím a jejím druhům.

3.1 Webové aplikace

Webová aplikace je aplikace, která nemusí být instalována na cílovém zařízení uživatele (telefon, tablet, PC, apod.), ale lze ji spustit z jakéhokoliv zařízení pomocí webového prohlížeče (Chrome, FireFox, Internet Explorer, aj.).

Tato aplikace je spouštěna na straně serveru a je nazývána jako „lehký klient". Webová aplikace může na první pohled běžnému uživateli připadat jako webová stránka, ale většinou se jedná o složitější aplikaci, provádějící obtížnější úkony s možným napojením na databázové a další systémy. V dnešní době jsou webové aplikace hojně rozšířené a napojené i na aplikace v organizacích. Webové aplikace v této době dokáží plně nahradit i software nainstalovaný na počítači.

Výhody webových aplikací:

- není nutná instalace ani aktualizace,
- jsou nezávislé na operačním systému,
- data jsou uchovávána na serveru a dostupná odkudkoliv.

Nevýhody webových aplikací:

- nutné připojení na internet,
- rychlost aplikace je závislá na připojení k internetu,
- možná bezpečnostní rizika.

(ManagementMania, 2016)

3.1.1 Architektura webové aplikace

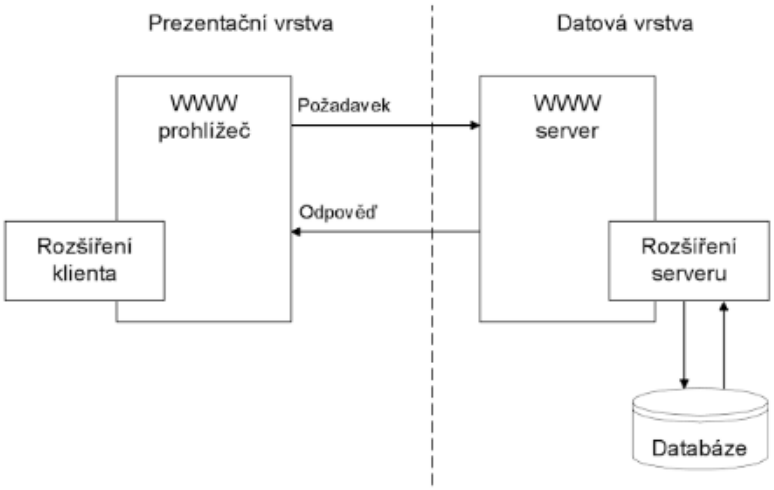
Základní architektura webových aplikací je dvouvrstvá. Tvoří ji webový server (vrstva datová) a webový prohlížeč (vrstva prezentační). Prvky, zlepšující dojem uživatele, vytvořené pomocí JavaScriptu nebo Java appletu neoddělují aplikační ani prezentační vrstvu aplikace, ale pouze zvyšují propracovanost prezentační vrstvy.

V dnešní době jsou však webové prohlížeče rozšířené o různé pluginy, applety nebo COM komponenty, díky čemuž se prezentační vrstva rozděluje na prezentační a aplikační logiku. Serverová rozšíření mohou také do webové aplikace přidat další vrstvy.

Aplikace lze rozdělit dle typu na:

- **Jednovrstvé architektury** - nerozdělují od sebe různé části aplikace. Tato architektura je vhodná pro malé utility spouštěné z příkazové řádky.
- **Dvouvrstvé architektury** - rozdělují aplikační a prezentační logiku (vrstva prezentační) od přístupu k datům (vrstva datová). Tato architektura se využívá pro databázové aplikace, jednoduché webové aplikace a aplikace typu klient-server. Nabízejí sdílení dat mezi více aplikacemi. Základní myšlenkou je zpracovávat data lokálně, ale ukládat by se měli na sdíleném centrálním serveru. Dvouvrstvé aplikace musí mít logiku pro práci s více databázemi v aplikační vrstvě. A přesune se tak logika pro přístup k datům z databázového serveru přímo do aplikace.
- **Třívrstvé architektury** - odděluje prezentační logiku, aplikační logiku a logiku pro přístup k datům do tří různých oddělených částí. Výhodou je možnost přesunout doménovou vrstvu z klientské vrstvy na jeden nebo více serverů. Umožňuje přidělit různým částem aplikace hardwarové prostředky, čímž lze zvýšit výkon a škálovatelnost aplikace. Třívrstvá architektura vyžaduje přídavnou síťovou vrstvu, která může snížit výkon a zkomplikovat údržbu aplikace.

(Bollinger, 2003)



Obr. 3 Základní architektura[3]

[3] Zdroj: (Bollinger, 2003)

3.1.2 Bezpečnost internetových aplikací

Bezpečnost je nutná na mnoha úrovních. Cílem útoku se může stát operační systém serveru, ale i prohlížeč uživatele. Proto je nutné proti útokům zabezpečit aplikaci již při její tvorbě. (Ferschmann, 2008)

Data na webu je nutné zabezpečit šifrováním, jelikož v dnešní době jsou webové aplikace hojně využívány a obsahují tak i citlivá data (např. internetové bankovnictví, vzdálený přístup do podnikového informačního systému apod.).

Základním krokem zabezpečení webové aplikace při jejím tvoření je síťový protokol HTTPS, který zabezpečí spojení mezi klientem a serverem před odposlouchávám. Šifrování je tvořeno pomocí protokolu SSL (Secure Sockets Layer) nebo novějšího TLS (Transport Layer Security). U webových aplikací, vyžadující zabezpečení, je nutností mít nastavené automatické přesměrování z nezabezpečeného HTTP na šifrovanou verzi. Novější prohlížeče podporují technologii HSTS (HTTP Strict Transport Security). Pro dosažení větší bezpečnosti slouží certifikáty, které využívají protokolu SSL či TLS a spojují zařízení s danou osobou. Klientské certifikáty jsou využívány především v internetovém bankovnictví.

Nejčastější slabinou webových aplikací je chybějící kontrola vstupu od uživatele. Data získané od uživatele je tak potřeba validovat na straně serveru. Uživatel může měnit obsah formulářových polí, URL adresu, HTTP hlavičku a cookies. Špatná kontrola vstupu může vést k tzv. SQL injection. Tudíž je nutné kontrolovat ve vstupu od uživatele, aby chybný vstup nezpůsobil spuštění SQL dotazu. V případě, že se vyskytne další chyba nezabezpečené stránky, může dojít k Cross site scripting (XSS) útoku, kde může útočník do stránky vložit například javascript a získat tak citlivé údaje jako session id. Veškeré uživatelské data tak musí být nejprve ověřena nebo zašifrována. (Kosek, 2014)

Existuje však i možnost externí ochrany webové aplikace tzv. WAF (Web application firewall) systémy. WAF je možné mít přímo jako software na serveru, nebo také jako samotné hardwarové zařízení mezi klientem a serverem. Tento systém dokáže na základě definovaných pravidel analyzovat veškerá data, které server přijímá. WAF zachytí dotaz a dle pravidel rozhodne, zda jej pošle dál na server či ne. Výhodou WAF je nezávislost na stávající architektuře a aplikacích. (Pomazal, 2010)

Základy bezpečnosti aplikací

Při tvorbě aplikací se bere v úvahu sedm základních bezpečnostních funkcí:

- autentikace - potvrzení identity,
- integrita - potvrzení, že nebyl požadavek při přenosu modifikován,
- utajení - ochrana informací,
- autorizace - řízení přístupu,
- zaručení - zaručení autenticity,
- dostupnost - zaručení, že aplikace dokáže reagovat na požadavky,

- audit - sledování činnosti uživatele.

(Bollinger, 2003)

3.1.3 Technologie a služby

- **Cloud computing** – je jakýkoliv program nebo služba, kterou máme dostupnou odkudkoliv a z jakéhokoliv zařízení s přístupem na internet. Uživatelská data a aplikační logika jsou ukládány na serveru na internetu. Pod pojmem cloud computing je možné si představit například email nebo v podstatě jakékoliv další služby od Google (Kalendář, Disk, Fotky a další služby). (Čížek, 2008)
- **Applet** – je program, který běží ve webovém prohlížeči a je psaný v programovacím jazyce JAVA. Applet se vkládá do HTML kódu. K zobrazení appletu je nutný Java Virtual Machine, který se do počítače instaluje spolu s prohlížečem. Applety mají přísná bezpečnostní pravidla. (TutorialsPoint, 2017)
- **HTML** (HyperText Markup Language) – je značkovací jazyk určený k vytváření dokumentů, které obsahují hypertextové odkazy a pokročilejší formátování. HTML poskytuje elementy, díky kterým lze vytvářet, upravovat a formátovat webové stránky. (Písek, 2014)
- **HTML5** – je nadstavba HTML. Za specifikaci tohoto jazyka jsou zodpovědné tři organizace - WHATWG (Web Hypertext Application Technology Working Group), W3C (WorldWide Web Consortium) a IETF (Internet Engineering-TaskForce). Při vývoji jazyka byly pomocí analýzy milionu stránek odhaleny typické identifikátory elementů DIV a zjištěno obrovské množství shod, na jejichž základě byly vytvořeny nové značky. Díky HTML5 vznilo XHTML5, aby mohly nástroje pro práci s XML generovat platný kód HTML5. Již od počátku vzniku je u HTML5 kladen důraz na zabezpečení. Všude, kde je to možné, se v HTML5 odděluje forma a obsah pomocí kaskádových stylů. Většina metod pro prezentaci obsahu tak není v nové verzi dostupná, ale díky kompatibilitě však bude fungovat nadále. HTML5 také nabízí podporu funkcí, které byly dříve využívané pomocí zásuvných modulů. (Lubbers, 2011)
- **CSS** (Cascading Style Sheets) – je samostatný, ale doplňující jazyk k programovacímu jazyku HTML. S jazykem CSS jsou aplikovány styly pro danou webovou stránku. Vzhled všech elementů je možné změnit pomocí vložených stylů. (Lazaris, 2014)
- **AJAX** – je zkratkou pro asynchronní JavaScript a XML. Díky možnosti měnit obsah svých stránek bez nutnosti opakovaného načítání, Ajax nabízí tvorbu webových aplikací podobající se desktopovým aplikacím. Pomocí Ajaxu jsou tvořeny například tzv. našeptávače formulářů nebo je využíván u kontroly dat na serveru ještě před odesláním stránky uživatelem serveru, například při registraci. (Láslo, 2008)

3.2 Databáze

Databáze je systém převádějící sady dat na abstraktní nástroje. Nabízí možnost uživateli pohodlně vyhledávat a prohlížet požadované informace. (Brookshear, 2016)

Databázi je možné považovat za systém sloužící k modelování objektů a vztahů reálného světa prostřednictvím uspořádaných dat pro efektivní manipulaci, tj. vyhledávání a provádění potřebných operací s daty, jako například zobrazení, přidání nových údajů nebo úprava stávajících údajů.

Databázové systémy

Databázové systémy již neslouží jen pro správu firemních dat, jak tomu bývalo dříve, ale staly se technologií, která zajišťuje fungování mnoha populárních webů. Webové stránky tak poskytují rozhraní mezi klientem a databází. Zákazník pošle požadavek, server se zeptá databáze a uspořádá výsledky dotazu do webové stránky, kterou odešle zpět klientovi.

Webové aplikace, informační systémy nebo aplikační software jsou v dnešní době velmi často používány spolu s databázovým systémem. Ve většině společností tvoří základ informačního systému. (Brookshear, 2016)

Pojem databázový systém zapouzdřuje systém řízení báze dat a databáze. Data jsou organizována v databázi a jsou řízena systémem řízení báze dat.

NoSQL databáze

Počátkem roku 2009 se v souvislosti s databázemi určenými pro web a cloud computing začali objevovat tzv. NoSQL databáze. Do NoSQL databází se zahrnují XML databáze, grafové databáze, databáze dokumentů nebo objektové databáze. Účelem těchto databází je dosažení horizontální škálovatelnosti databázového zpracování v dynamickém prostředí databází. (Tyrychtr, 2015)

Objektový model dat

Objektový model dat je určený objektům modelovaným v databázových aplikacích. Objekt představuje základní entitu v konkrétní aplikaci. Rozdíl mezi klasickým a objektovým modelem dat je, že při tvorbě datového modelu klasickým způsobem zobrazuje prvky reálného světa do předem připravených struktur. U objektů pro prvky reálného světa vytváříme nové objekty, které se jim podobají. Příklady využití objektových databázových systémů jsou například geografické informační systémy, počítačové navrhování a počítačová podpora výroby.

V relačním modelu se nachází atomické atributy. Objektově relační databázové systémy podporují komplexní objekty. (Tyrychtr, 2015)

U objektové databáze není třeba data převádět do tabulek jako u objektově relačního mapování databáze.

Objektově relační mapování

U objektově relačního mapování místo pole hodnot získáváme z databáze přímo objekty. Uživatel zde nepoužívá SQL a tabulky v databázi jsou zobrazeny jako ko-

lekce objektů. SQL dotazy jsou zde vytvářeny automaticky a mohou být i neefektivní. (Čápka, 2014)

4 Analýza game engine

Existuje velké množství různých enginu s rozdílnými vlastnostmi. Mezi nejzákladnější rozdíly se řadí například tyto:

- placené, částečně placené a neplacené game enginy, dle jejich licencí,
- rozdílné skriptovací jazyky,
- funkcionalita a kvalita, kterou engine poskytuje,
- uživatelské rozhraní.

Na začátek je uvedeno a vysvětleno několik licencí, které jsou game enginy využívány. Poté je vybráno několik game enginů, které jsou základně rozebrány. Protože je nepřeberné množství game enginů a některé jsou až na drobné rozdíly, jako je uživatelské rozhraní, téměř totožná, vybráno bylo několik příkladů pro ukázku.

4.1 Softwarové licence

Licence udávají podmínky využitelnosti jednotlivých softwarových produktů. V tomto případě se zaměřujeme především na volně dostupné licence, které využívají některé game enginy. (Malý, 2011)

- **Open source** – obvykle znamená, že zdrojový kód je volně k dispozici a tedy každý může přispět rozvojem softwaru. Pokud má stanovenou licenci jedná se z pravidla o GNU GPL nebo LGPL.
- **GNU GPL** (General Public License) – licence plně odpovídající Debian Free Software Guidelines a umožňuje volné používání, modifikování a šíření softwaru. Takto vzniklý software může být dále šířen bezplatně a umožňuje také získat zdrojové kódy. Licence se týká samotného software i softwaru od něj odvozené-ho.
- **BSD** (free software) – licence umožňující volné šíření obsahu (softwaru) za podmínek uvedení autora a informace o licenci, spolu s uvedením na zřeknutí se odpovědnosti za dílo.
- **MIT** – licence umožňující libovolně používat, kopírovat, modifikovat, slučovat, publikovat, distribuovat či prodávat software. Pod podmínkou uvedení textu li-cence do všech kopií odvozeného softwaru.
- **Apache 2.0** – (free software, open source) licence vyžadující zřeknutí se odpovědnosti. Umožnuje svobodné užívání software k distribuci, upravování a následné šíření softwaru.
- **Eula** (End User License Agreement) – licenční ujednání nastavené podle požadavků klienta.

(GitHub, Inc.)

4.2 Rozbor game engine

Pro základní výběr bylo určeno kritérium, aby byl game engine zaměřen na 3D grafiku a tedy určen pro 3D hry. Proto se ve výběru nacházejí pouze enginy, které toto kritérium splňují. Specifikace a využitelnost většiny enginů je velmi rozsáhlá, a proto některé původně zamýšlené specifikace, jako kvalita grafiky aj. byly vynechány.

Jako varianty byli vybrány enginy s licencemi uzpůsobenými tak, že umožňují neplacené využívání. U game enginu Unity, který je v základní formě zdarma, je uvedeno pár informací i o game enginu Unreal, který je při nasazení zpoplatněn. Mezi základní požadavky práce patří možnost webového nasazení, které bylo zjištěno u všech vybraných enginů, ale dopředu nemohlo být předpokládáno.

Specifikace a případně využitelnost jsou uvedeny přímo u daných game enginů. Výběr vhodné varianty dle parametrů dané aplikace (webová aplikace volně dostupná, možnost ukládání a znovu obnovení, rychlost, přesnost, přenosnost)

4.2.1 Unity 3D

Unity 3D je program pro tvorbu 2D a 3D her využívající licenci Eula, která zahrnuje 4 varianty. Nejzákladnější a často i nejvyužívanější je verze „Personal“. Ta je zdarma a zahrnuje základní nabídku, která poskytuje všechny funkce enginu, možnost uložení do všech poskytovaných platforem, cloud uložiště apod. Tutu verzi je možné využívat pouze do 100 tisíc dolarů hrubého výnosů za rok. U firem se jedná o celkové hrubé příjmy firmy za rok. Při překročení tohoto příjmu je třeba zakoupit vyšší verzi. Dle výše zisku se rozdělují další dva druhy licencí. Základní verze „Personal“ je vhodná také pro studenty. (Unity, 2017a) Podobnou licenci poskytuje engine Unreal, který umožnuje učit se v tomto programu ve studentské verzi zdarma. Pokud je z výsledného produktu utržen zisk, pak je dáno, že Unreal si účtuje 5 % z hrubého výnosu při použití volné licence. (Epic Game, 2017)

Pomocí obou game enginů je možné vytvořit plnohodnotné real-time hry s různým zaměřením a jejich využitelnost je na více než 20 používaných platformách, jako jsou iOS, Android, Windows, Linux, PS4, Facebook Gameroom a jiné. Oba programy mají grafické rozhraní, ve kterém je možné vytvářet a upravovat scénu. Unreal i Unity využívají pro snadnější nastavení grafické stránky i funkcionality uzly, které jsou často rychlejší než psaní celé funkcionality v kódu. Uzly jsou v takovém případě kódem pouze doplněny.

Jako skriptovací jazyk Unity 3D Engine využívá C# a JavaScript. V obou jazycích je možné vytvořit stejné funkce, takže výběr jazyka je čistě na preferencích programátora.

Unity 3D podporuje importování modelů ve velkém množství formátů. Importování mesh objektů do Unity může být provedeno pomocí exportovaných otevřených souborů (.FBX, .3DS, .obj, Collada atd.) nebo souborů proprietárních (.Max, .MB, .Blender atd.)

Unity 3D podporuje vlastní databáze v podobě „AssetDatabase“. Jedná se o API, která umožňuje přístup k objektům obsažených v projektu. Rozhraní „AssetDatabase“ je k dispozici pouze v editoru. (Unity, 2017b)

Od roku 2013 je dle licence „Web Player License Agreement“ možné vytvářet pomocí Unity pluginu „Unity Web Player“. Vytvoření webové aplikace lze uskutečnit pouhým uložením do potřebné verze. (Unity, 2017c)

4.2.2 JMonkeyEngine 3.0

Daný engine funguje na více platformách, mezi které patří i základní Windows, Linux, Mac OS i Android. Hlavním programovacím jazykem pro jMonkeyEngine je Java a je možné v tomto jazyce, ve spojení s jazykem XML, vytvořit celou plnohodnotnou hru v real-time.

Pro tvorbu databázového systému je možné přímo v programu stáhnout a doinstalovat plugin. Je možné využít objektovou databázi Hibernate pro NetBeans. Mezi pluginy je také modul Database poskytující vizuální a textový editor SQL a prohlížeč databázových připojení, tabulek a jejich data. Ke zhotovení hráčského menu engine využívá knihovnu NiftyGui, přes kterou je možné vytvářet uživatelské rozhraní pomocí jazyka Java nebo XML.

Pomocí sestavení projektu („Build Project“) lze lehce vytvořit webovou aplikaci. Tu je možné následně vložit do internetové stránky pomocí kombinace HTML, .jnlp, .jar souborů a knihoven.

V programu je možné vytvořit 3D scénu za pomoci více modelů. U modelů jsou možné drobné úpravy přímo v programu. Jedná se například o nastavení textur a materiálů, velikost, rotace a posun modelu ve scéně. Podporuje velké množství formátů. Je tedy možné využít modely z více 3D grafických programů ve formátu například .blender, .obj atd. Nejlepší je využít export z programu Blender do formátu Ogre3D, který vyexportuje soubor formátu .scene a .mesh, které jsou v programu jMonkeyEngine snadno využitelné a upravitelné.

Umožňuje vkládání a úpravu animací a animovaných modelů, které leze využít k živější scéně nebo ve spojení s kódem k reálnému pohybu postav. Animaci je třeba vytvořit již v 3D grafickém programu (např. Blender, 3Ds Max a Maya).

Velkou výhodou daného enginu je, že existuje velké množství návodů, rad a fór zabývajících se daným game enginem a poskytují plnohodnotné a užitečné informace pro vytvoření celkové hry.

JMonkeyEngine je poskytován jako open source pod licencí BSD a produkty pomocí něj vytvořené je možné dále šířit. Daný game engine je používán i několika komerčními herními studii.

Informace o daném game enginu jsou z internetové stránky (jME, 2016) a (jME) a z vlastního vyzkoušení daného programu. Program má velký potenciál a Java je přívětivý programovací jazyk, který umožní tvorbu hry.

4.2.3 Panda3D

Jedná se multiplatformní game engine, který poskytuje pomocí programovacího jazyka C++ nebo Python funkce ke tvorbě plnohodnotné real-time hry. Game engine potřebuje k tvorbě editor na psaní kódu Greany, který je vhodný pro psaní v programovacím jazyku Python. Tento editor umožňuje přehlednost kódu a zajišťuje propojení s game enginem Panda3D. Po překladu a spuštění základního kódu spustí okno, kde je možné upravovat scénu a získat parametry, jež je potřeba přepsat do kódu.

Engine podporuje pouze jeden formát pro importování objektů. A to formát EGG, což značně omezuje jeho využitelnost, i když program Blender export do daného formátu podporuje.

Soubor P3D lze jednoduše vložit na webové stránky, pomocí nastavení objektu v HTML nebo pomocí souboru JavaScript RunPanda3D.js. Ve webovém prohlížeči lze danou aplikaci přímo spustit. Game engine Panda3D využívá vlastní databázový systém, pro jehož pochopení je potřeba důkladná studie.

Panda3D využívá nepatrně modifikovanou licenci BSD. Umožňuje tedy při splnění daných podmínek redistribuci a použití ve zdrojové i binární formě úplně zdarma. Pro daný game engine je na oficiálních stránkách produktu sepsán manuál pro oba skriptovací jazyky. Jsou také uvedeny video návody, ukázky prací a fórum pro komunikaci s ostatními tvůrci. (Panda3D, 2016)

4.2.4 Torque 3D

Torque 3D pro svoji funkčnost vyžaduje nejnovější verzi DirectX a editory „World Editor“ a „GUI Editor“. World Editor je nástroj pro sestavení herních úrovní tzn. pro přidávání a umisťování terénu, herních předmětů, modelů, účinky prostředí, osvětlení atd. GUI Editor slouží pro grafické uživatelské rozhraní, jako je úvodní obrazovka, hlavní menu, dialogy apod. Pak také pro navržení a vytvoření menu, množství životů, nahrávání scén, přehrávač inventarizačního systému.

V základu Torque 3D využívá „Toolbox“, který poskytuje prázdné i hotové projekty. Projekty je možné dále upravovat pomocí již zmíněných editorů.

Torque 3D povoluje import objektů pouze v daném formátu, a to DTS (statické modely), DSQ (animace) a Collada (XML), která dovoluje úpravy. Tyto formáty je možné získat například z programu Blender, Maya nebo 3Ds Max.

Engine využívá pro naprogramování herní logiky a ovládání kamer skriptovací jazyk TorqueScript, který je dle popisu výkonný a flexibilní. Jeho syntaxe se podobá programovacímu jazyku C++. Jako uložiště využívá multiplatformní databázový systém MySQL. Dále je možné integrovat Havok SDK sloužící pro rozšíření hry.

Vytvořenou real-time hru lze aplikovat například na platformách Windows nebo Linux.

Torque 3D využívá licence MIT a je tedy v základu bezplatná. Po bližším zkoumání daného game enignu byl zjištěn problém, že omezení využitelnosti funkcí mají editory. Jedná se například o „Debug“ v programu, který je přístupný pouze

v placené nebo trial verzi. V neplacené verzi při delším používání jsou opravy chyb problémovější.

Dokumentace Torque 3D obsahuje informace o všem, co daný program umožňuje, komentáře a datové struktury pro lepší pochopení nastavení celého systému. (Torque), (GarageGames, 2016)

4.2.5 Blender

Tento 3D grafický program, stejně jako game engine v něm obsažený, je dostatečně popsán v již zmiňované bakalářské práci (Portlová, 2013), která také udává pozitiva, ale také komplikace s programem spojenými. V bakalářské práci je rozebírána tehdy nejaktuálnější verze 2.64, v současné době je vydaná verze 2.78a. V období mezi těmito typy bylo změněno uživatelské rozhraní, opraveno velké množství chyb a některé nově vznikly, byly rozšířeny některé funkce (např. u animace, ovládání kostí atd.) a vývojové možnosti (např. senzory) a přidány úplně nové funkcionality, jako například nové světelné zdroje, efekty, modifikátory a podobně.

Pro shrnutí je uvedeno několik základních informací o programu. Blender je open source 3D program s licencí GNU GPL, který funguje na platformách Windows, Linux a Mac OS. Jeho struktura je vytvořena programovacím jazykem C++ a je možném v něm vybudovat real-time hru od úplného základu. V programu Blender je možné celou scénu vymodelovat nebo nahrát 3D modely z jiných zdrojů. Podporuje významný počet formátů. Pokud není daný formát v základní nabídce, Blender umožnuje rozšíření nabídky doinstalováním určitých souborů. Skriptovacím jazykem je Python spojený s vývojovými uzly. Dílo je možné publikovat jako samospustitelnou desktopovou aplikaci nebo ji distribuovat na web, záleží na výběru uložení/exportu.

Komunita zabývající se programem Blender je velmi rozmanitá a zahrnuje velké množství vývojářů, programátorů a uživatelů, kteří vytvářejí četné návody s informacemi o programu a možných funkcích. Blender je stále ve vývoji a v častých intervalech vycházejí opravované verze s úpravami funkcí a opravami chyb. (Blender, 2017)

4.2.6 Vyhodnocení

V současné době jsou game enginy na podobné úrovni. Umožnují tedy v základu vytvořit jakýkoliv styl hry s podobnou funkcionalitou. Často záleží na bližším seznámení se s enginem, a tedy důsledným prozkoumáním všech možností a naopak negativ do velkých podrobností, aby se mohly stanovit veškeré dílčí rozdíly. Tak, jak byly enginy prozkoumány pro dílčí cíle této práce, nebylo možné určení jejich přesného zařazení.

Každý tvůrce interaktivní aplikace (hry, průchody scénami apod.) musí sám rozhodnout, jaký programovací či scriptovací jazyk mu vyhovuje, jakým stylem bude chtít pracovat s rozmístěním scény, zda mu bude více vyhovovat kód nebo grafický náhled, a podobně. Pokud bude uživatel nezkušený většina game enginů uvádí mnoho informačních návodů v psané, obrazové nebo video formě. Poté už je

jen na tvůrci, s jakým game enginem se bude chtít seznámit a naučit se s ním zacházet.

Někdy se rozlišují spíše enginy placené a volně dostupné, kde placené verze jsou na vyšší úrovni a zpravidla umožnují uživateli vyšší míru funkcionalit, než enginy s volnou licencí využití. I když u četného množství game enginů se jedná o placené verze až při dosažení určitého zisku nebo pokud chce uživatel více funkcí, než nabízí základní verze programu. V dnešní době se dá tedy říct, že game enginy s volnou licencí jsou na velmi vysoké úrovni a ve funkcionalitě se podobají enginům placeným. A lze tedy i je využívat pro obsáhlejší a kvalitnější projekty a komerční účely.

4.2.7 Výběr

Vzhledem k pokračování v bakalářské práci by se mohlo zdát jako řešení pokračovat i v programu Blender. Ale i když byly některé negativní důvody pro možné nevyužití uvedené v bakalářské práci odstraněny, tento program nebyl pokládán za vhodný, vzhledem k jiným nedostatkům a skriptovacímu jazyku Python.

Dle více zdrojů, mezi obecně nejznámější a nejlépe hodnocené patří Unity 3D a Unreal Engine, možná právě kvůli intuitivnějšímu uživatelskému rozhraní a nabízené kvalitě. Jedná se také o nejčastěji využívané enginy pro komerční využití.

Vzhledem k funkcionalitě, rozložení a hlavně vlastním schopnostem (programovací jazyk Java, znalost programu NetBeans, významnějšímu zájmu při rozboru atd.), byl po zhodnocení všech game enginů, zvolen pro tvorbu webové aplikace program jMonkeyEngine, a to i přes klady Unity 3D.

4.3 JMonkeyEngine

Základní informace o tomto game enginu byly zveřejněny již výše (3.1.3 jMonkeyEngine 3.0). Podrobněji jeho historie, rozhraní, prostředí a základní funkce budou probrány nyní.

4.3.1 Vznik a vývoj

jMonkeyEngine je plně vytvořený v programovacím jazyku Java. Počátky vzniku jMonkeyEngine jsou v roce 2003, kdy autorem projektu byl Mark Powell. (Pan, 2011) Ve stejném roce se k projektu připojil Joshua Slack. První stabilní verze (2.0) byla vydána roku 2008. O rok později byla vydána verze 3.0, ke které následující rok přibylo vývojové prostředí JME SDK, které je hlavním nástrojem pro práci s daným enginem. (Skála, 2016)

4.3.2 Nejnovější verze

Nejnovější stabilní verzí v současné době je jMonkeyEngine 3.0.10. Vzhledem k obsahu, kvalitě a funkcionalitě se prozatím v nejbližší době nepředpokládá vydání novější verze. Využívá technologií Java, NetBeans Platform, Gradle a mnoho knihoven jako NiftyGUI, Blender, Bullet a jiné. Program se dá stáhnout ve formátu .exe

pro Windows, .sh pro Linux nebo jako .zip pro Mac OS. Tento game engine je používán několika komerčními herními studii a kurzy počítačových věd. V této verzi byly vytvořeny hry jako Boardtastic 2[4] (mobilní pro android) Mythruna[5] (desktopová). (GitHub, 2017)

4.3.3 Prostředí a významné funkce

Nejvyužívanějším vývojovým prostředím pro daný engine je jMonkeyEngine SDK 3.0. Jeho rozhraní je velmi podobné Netbeans. Liší se především v možném zobrazení modelů a vytváření a úpravy scén. TerrainEditor slouží pro vytvoření a úpravu terénu scény, VehicleCreator pro úpravu vozidla a jeho vlastností a SceneComposer, který umožnuje sestavit celou scénu a následně ji použít při načítání namísto jednotlivých objektů. Obr. 4 zobrazuje rozložení JME SDK a scénu v SceneComposer.

V programu je možné vytvářet nové prázdné soubory například typu NiftyGui, Material nebo Scene nebo základy celých projektů, jako BasicGame nebo Java Application.

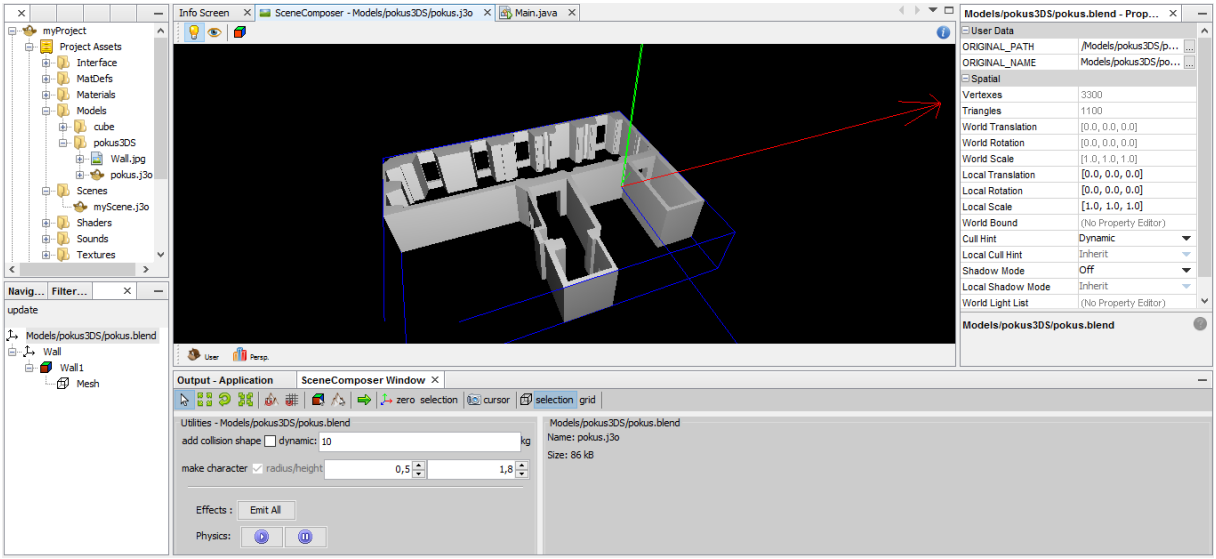
Do programu je možné importovat různé typy formátů. Mezi nejvhodnější patří OGRE (.scene + .mesh), Blender (.blend), Obj (.obj). Všechny tyto formáty je možné konvertovat do formátu .j3o Binary, který je potřebný pro další zpracování. Je možné soubory konvertovat až při kompilaci, ale vzhledem k výpočetnímu času je nejefektivnější využívat již konvertované soubory. Při importování přes danou funkci se soubor kompiluje automaticky. Vybrané objekty lze při využití SceneComposeru upravovat pomocí palety vlastností nebo jim vlastnosti (polohu, rotaci, materiál, kolize atd.) přiřazovat přímo pomocí kódu. Pokud uživatel využívá soubory .blend a nestačí mu využití možností scény, je možné přímo z programu model otevřít v programu Blender a upravit ho tam.

Program umožňuje v „Project explorer“ členit soubory na Project Assets, kde jsou ukládány důležité podklady grafické stránky (materiály, modely, nifty, textury atd.), a „Source Package“ pro kódy určujíc funkcionalitu aplikace.

Veškeré funkce, které poskytují editory v JME SDK 3.0, lze vytvořit také kódem, který umožnuje zhotovit i více a rozšířit hru po grafické i funkční stránce.

[4] Zdroj: https://play.google.com/store/apps/details?id=com.boardtastic.skateboarding

[5] Zdroj: https://mythruna.com/



Obr. 4 JME SDK 3.0 při spuštění SceneComposer

JME SDK 3.0 využívá knihovny LWJGL 2.9.0, OpenGL 4.0.0 a OpenAL pro lepší grafické a zvukové rozhraní. Například OpenAL umožnuje 3D zvukový kanál. Jak již bylo zmíněno, tak JME SDK 3.0 využívá programovací jazyk Java, který je při tvorbě her méně častý. Častější a využívanější variantou bývá C#, JavaScript nebo Python. I přesto jMonkeyEngine poskytuje širokou škálu funkcí.

Asi se dá říci, že jako každý jiný game engine v základu obsahuje funkce, jako více druhů světelných zdrojů a stínů pro vytvoření reálné nebo méně náročné scény. Dále se jedná o možné použití reálné fyziky, detekce kolizí, nastavení kamer, grafu scén a optimalizace, které poskytuje. Tyto funkce budou uvedeny při jejich využití v kapitole Implementace.

Funkce, která využita nebude je multiplayer, který game engine také umožňuje vytvořit. Lze jej vybudovat přes webovou síť pomocí aplikace SpiderMonkey s využitím protokolů UDP nebo TCP. Pomocí SpiderMonkey je možné vytvořit klienta, server, identifikaci, notifikace a možnosti posílání a přijímání zpráv mezi klienty a serverem.

4.3.4 Distribuce

Defaultní nasazení pro Javu je Java archive (JAR). Soubor JAR obsahuje spustitelnou multiplatformní desktopovou aplikaci. Spuštění souboru JAR je možné z příkazové řádky, nebo je potřeba předem systém nakonfigurovat pro jeho spuštění. Protože ne každý uživatel je tak zkušený, poskytuje Java i uživatelsky přívětivější možnosti nasazení.

Hry vytvořené programovacím jazykem Java lze publikovat jako desktopové aplikace, Android mobilní hry, Web Start, nebo applet prohlížeče. U každé varianty je dobré zvážit jejich pro a proti.

V případě desktopová aplikace lze zhotovit typické dvou-klik aplikaci. Nevýhodou je právě používání na více platformách. Je tedy nutné vyhotovit oddělené

formáty závislé na platformě (pro Windows (.exe), pro Linux (.jar, .sh), pro Mac (.app)).

Mobilní aplikace Android lze naistalovat přímo z obchodu Play Google. Alternativě může být stažen soubor APK z internetu a přes USB kabel a souborový průzkumník Android naistalována do mobilního zařízení. Je třeba vytvořit verzi, která nebude k funkčnosti vyžadovat vstup z klávesnice. U většiny mobilních zařízení je problém v nepodporování post-procesoru, takže je třeba nabídnout jiný způsob, buď jej zakázat, nebo jej nahradit.

Applet prohlížeč (.html + .jar) slouží pro hraní hry přímo na webové stránce v prohlížeči. Výhodou je, že není nutná instalace. Nevýhodou, že není možné hrát offline. Applety poskytují rychlé a uživatelky přívětivé řešení pro webové hry. Oproti desktopové verzi může být pomalejší, záleží na rychlosti připojení internetu a výkonu webhostingu

Pokud nevyhovuje varianta spouštění hry přímo na webové stránce v prohlížeči, a tedy stálého potřebného připojení na internet, ani instalování programu na lokální disk, je alternativou *Web Start* (.jnlp). Tato varianta po vstoupení na adresu URL stáhne aplikaci (hru) do mezipaměti a spustí ji mimo webový prohlížeč. Není zde tedy nutná instalace a hru takto uloženou v mezipaměti je možné hrát i v offline režimu. Přínosem pro uživatele je, že vždy spouští (stahuje) nejaktuálnější verzi. Nevýhoda nastává při prvním stahování, které je celkem pomalé. Omezení nepodepsané Web Start aplikace je i v nemožnosti uložení na disk a opětovného načtení z disku. (KUSTERER, 2013)

5 Metodika práce

V návrhu práce je v základu proveden průzkum stávající problematiky vzhledem k propagačním kanálům města, příčině tvorby práce a historii samotné. Je tedy rozebráno, jak se město a muzeum propaguje, jaké jsou důvody vzniku webové aplikace a jaká je historie daného města.

V další části je vytvořena architektura webové aplikace, architektura vývoje hry a popis využitých technologií.

5.1 Analýza stávající problematiky

Podle zkoumání v současné době není vytvořena žádná hra zahrnující historii města Polička. O městu a jeho historii existují záznamy na několika různých webových stránkách[6], papírových nebo elektronických[7] průvodcích a jsou také videa[8] propagující krásy města.

Vytvořením hry se nechce dosáhnout pouze propagace muzea, ale propagace města a jeho historie jako takové. Možná by se také dalo říci, že nejde tak o propagaci, jako o přiblížení historie mladším generacím.

Město momentálně pro propagaci historie, muzea a svých akcí využívá internetové stránky, Facebook, plakáty, tištené průvodce a videa.

V následujících sekcích jsou uvedeny specifikace požadavků na aplikaci a stručně základní fakta v dějinách toho historického města.

5.1.1 Požadavky na aplikaci

Požadavky na aplikaci nebyli příliš rozsáhlé, jejich úprava byla z větší části ponechána na tvůrci (tedy mě) s případnými průběžnými komentáři. Mezi zmíněné zadané požadavky patřily:

- volně dostupná hra, ve které by se každý mohl sekat s historií města Polička (otázky, úkoly, obrazy historie, atd.),
- intuitivní a snadno ovladatelná (klávesové ovládání, snadný pohyb po scéně, návod se základními informacemi),
- zajímavá (3D scéna s úkoly, grafika),
- dostupná odkudkoliv bez nutnosti instalace (online),
- menší náročnost na výpočetní techniku (binární ukládání, snížené grafické nároky).

---

[6] Oficiální stránky města – http://www.policka.org/

[7] Mobilní Polička – http://www.policka.org/detail/9069/Turismus/Polickou-s-novym-mobilnim-pruvodcem/

[8] Město Polička – YouTube - https://www.youtube.com/user/mestopolicka

Příběh i grafické zpracování bylo ponecháno na tvůrci s přáním na důstojné zpracování, ale spíše dětskou formou.

5.1.2 Polička a její historie

Polička je historické město nacházející se ve výšce 555 metrů nad mořem. Byla vystavěna na loukách zvanými „Napolickach" nacházející se na české straně pohraničního hvozdu. V současné době je unikátní zachováním 1220 m gotických hradeb a všech 19 bašt. Nabízí různorodou nabídku festivalů a akcí. Každoročně pořádá například festival 555, Martinů Fest a Colour Meeting Polička. Nejvýznamnější postavou narozenou v tomto městě je skladatel hudební moderny 20. století Bohuslav Martinů.

Mezi významné budovy města patří rodná světnička Bohuslava Martinů v kostele sv. Jakuba, Poličská radnice a hradby, Centrum Bohuslava Martinů v Poličce a hrad Svojanov, který městu Poličce patří a je jedním z nejstarších českých hradů (Město Polička, 2017)

Historické milníky

- 1265 – založení města českým králem, vévodou rakouským a štýrským a markrabětem moravským Přemyslem Otakarem II. jako součást soustavy měst při obchodní stezce spojující Prahu s moravskými centry (posílení královské moci).
- 1307 – Polička se stala věnným městem královským, když správu města český král Rudolf I. Habsburský postoupil své manželce Alžbětě (Eliška Rejčka).
- 1360 – Karel IV. dal městu privilegium vydláždit město, které mělo za následek značné zlepšení životního prostředí města. Byla také provedena přestavba dosud dřevěných domů na kamenné a město začalo bohatnout.
- 15. století – období husitských válek, ve kterých byla Polička z počátku na straně královny, později se ale přiklonila k husitům (přistoupila k Pražskému městskému svazu). Po husitských válkách vzdala v Jihlavě hold Zikmundu Lucemburskému. Nakonec se díky husitské revoluci stala Polička ryze českým městem.
- 1523 – jeden z nejničivějších požárů, který strávil značnou část města.
- 1547 – odepření moci Ferdinandu I. Habsburskému. Tvrdé potrestání města přineslo pozastavení práv, zkonfiskování majetku a uložení pokuty. Statky mohlo město odkoupit zpět až za 12 let.

Město mělo přiděleno velké množství privilegií, mezi které patřilo například privilegium hradební (obehnat město hradbami), mílové (v okruhu jedné míle od města nesměla být prováděna výdělečná činnost na úkor obyvatel), várečné (povolení vařit pivo), hrdelní (konání soudů a poprav) a jiné. Polička je významná také tím, že nikdy nebyla dobyta. V 18. století Polička patřila mezi nejvýstavnější česká měs-

ta. Jednalo se o zlatý věk pro měšťany spojeným s barokní přestavbou. (Město Polička 2017)

5.2 Architektura webové aplikace

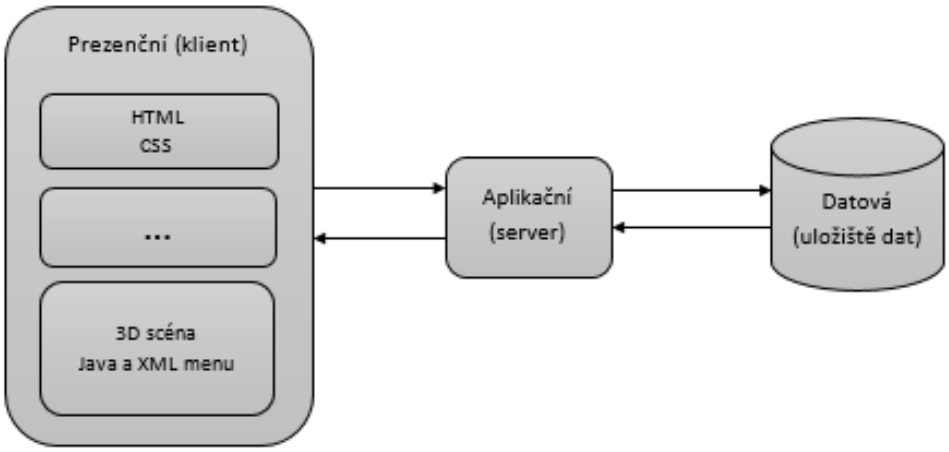
Architektura webových stránek je obvykle vystavěna jako vícevrstvá architektura. Typicky se skládá ze tří vrstev, stejně jako v této práci. Jedná se o vrstvu prezentační, aplikační a datovou, kde jednotlivé úrovně plní následné funkce:

- **Prezenční** – uživatelské rozhraní.
  Jedná se o vizuální vzhled stránky nebo aplikace. Obvykle je vytvářen jazyky HTML, CSS atd. v této práci ho tvoří také 3D scéna a menu vytvořené programovacím jazykem Java a XML.
- **Aplikační** - funkcionalita dané aplikace.

  Aplikační vrstva zahrnuje veškeré funkce ve hře, tedy od činností herního menu, přes načítání scény až k ovládání herní postavy.
- **Datová** – ukládané informace a data.

  V této vrstvě se převážně využívají databázové systémy v různých verzích. V této práci se daná vrstva skládá z lokálního uložiště na disku a databázového systému pro online uložení dat.

V tomto projektu je schéma vrstev a jejich komunikace znázorněna na Obr. 5. Aplikační vrstva zastává hlavní funkci a zajišťuje možnou interaktivitu. Z datové vrstvy získává informace, které dále zpracovává nebo je předává uživateli v prezenční vrstvě. Opačným způsobem umožnuje data získaná od uživatele zpracovat, upravit, opětovně zobrazit uživateli nebo uložit do datové vrstvy.



Obr. 5 Třívrstvá architektura aplikace.

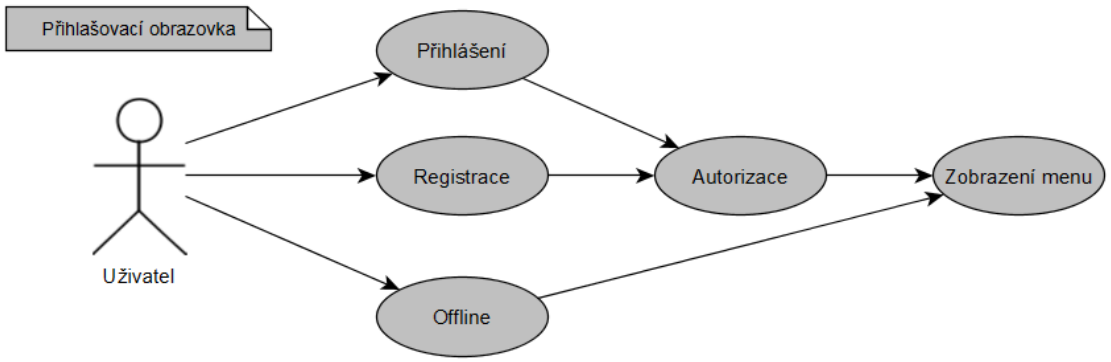
5.3 Části aplikace

Na problém je nutné pohlédnout i z hlediska game enginu, kde architektura určuje strukturu samotné hry. Diagram základní architektury hry je k nahlédnutí v příloze B (Obr. 29).

5.3.1 Menu

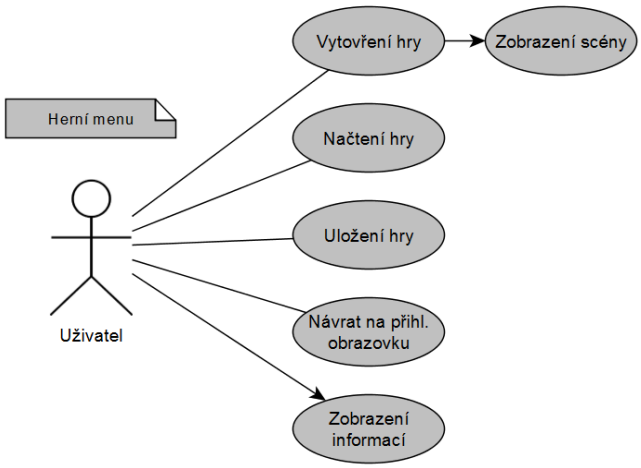
V této sekci je popsána jak přihlašovací obrazovka, herní menu, tak herní ukazatelé ve scéně. Jedná se především o inventář se získanými informacemi a objekty, které v postupu hry bude hráč získávat.

Úvodní obrazovka by měla být přihlašovací. Obsahovat by měla textové pole (pro zadání jména a hesla uživatele) a poté možnosti přihlášení (již existující uživatel), registrace (přidá nového uživatele do databáze) a možnost hraní offline bez databázové podpory s ukládáním na lokální disk. Při přihlášení a registraci proběhne autorizace a zobrazí se úvodní herní menu, jak je znázorněno na Obr. 6 Diagram Use Case – Přihlašovací obrazovka.



Obr. 6 Diagram Use Case – Přihlašovací obrazovka

Úvodní herní menu by mělo zahrnovat funkce (tlačítka) pro spuštění nové hry (vytvoření a načtení nové scény), možnost uložení, opětovného načtení a ukončení hry. Dále bude uvedeno tlačítko pro zjištění informací o hře. Funkce a návaznost herního menu je zobrazena na následujícím diagramu (Obr. 7).



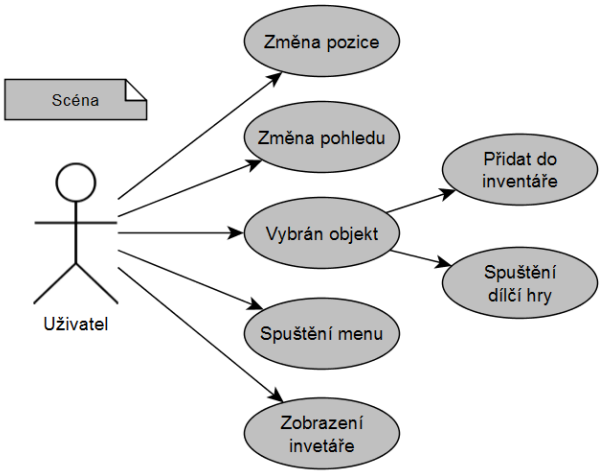
Obr. 7 Diagram Use Case – Herní menu

5.3.2 Scéna

Prvním krokem přípravy pro tvorbu scény v aplikaci je rozčlenění scén, modelů na ní a určení funkcí, které bude na scéně možné provádět.

První, a v úvodu základní scénu, tvoří místnost v prvním patře, která je propojena se dvěma chodbami a schodištěm. Scény tvoří stěny (zeď, výplně, oblouky), podlaha, strop, dveře a okna. První scénu tvoří také schodiště a klenbové oblouky nad ním. Každá scéna bude obsahovat modely pro doplnění, jako jsou obrazy, lustry a objekty potřebné ke hře. Všechny scény budou postaveny na podobných principech.

Následující diagram Obr. 7 znázorňuje, že na scéně získá uživatel funkce ve formě pohybu, rotace kamery, možného návratu do hlavního menu, zobrazení inventáře a reakce na kolize s objekty. Aktivní prvky mohou být buď přiřazeny do inventáře, nebo se při jejich aktivaci spustí dílčí hra.



Obr. 8 Diagram Use Case – Scéna

Hra, kromě scény a průchodu v ní, obsahuje také hledání tajemství, věcí a dílčí hry, díky kterým jsou získávány body a jsou odhalovány informace o historii města. Hledání je vytvořeno pomocí pohybu hráče a pohybem kamery neboli zaměřováním ve scéně pomocí výpočtu protnutí elementu s přímkou danou pohledem kamery hráče.

5.4 Distribuce

Pro publikaci webové aplikace je nutné provést sestavení hry v režimu Applet s následným provedením podpisu jednotlivých souborů ve formátu .jar a pro zajištění funkčnosti bez vytváření výjimek běhu pro Java Security. Takto vytvořené a upravené souboru stačí pouze přenést na daný webhosting, jelikož již obsahují všechny nezbytně nutné soubory pro běh hry ve webovém prohlížeči.

Hra bude také zveřejněna v desktopové aplikaci pro Windows, tedy uložena do formátu .exe.

5.5 Využité technologie

Pro práci se předpokládá využití následujících technologií:

- **Blender** – 3D grafický program pro rozčlenění a přetexturování modelů.
- **jMonekyEngine** – program obsahující game engine, který umožní vytvoření 3D scény a funkcí potřebných pro chod aplikace.
- **Java** – programovací jazyk, pro vytvoření potřebných funkcionalit aplikace (pohyb postavy, nastavení světla, detekce kolizí, fyzikálních vlastností, kamery, řízení úkolů, doladění textur, ovládání a funkce herního menu, ukládání a načítání hry atd.).
- **XML** – programovací jazyk pro vytvoření grafického náhledu herního menu (tlačítka základního menu, herní menu zobrazující inventář a body, grafické zobrazení dílčích her atd.).
- **Applet** – jedná se o třídu (java.applet.Applet) v programovacím jazyku Java zajišťující spuštění (init(), start()), překreslení (paint()), přerušení (stop()) a ukončení (destroy()) aplikace. Pro účely této práce byl využit k načtení aplikace, vytvořené v jMonkeyEngine, na internetovou stránku.
- **HTML** – programovací jazyk určený k vytvoření obsahu internetové stránky. V tomto případě se jedná o navržení základní strukturu stránky.
- **CSS** – programovací jazyk využitý pro zhotovení úhlednějšího vzhledu stránek.

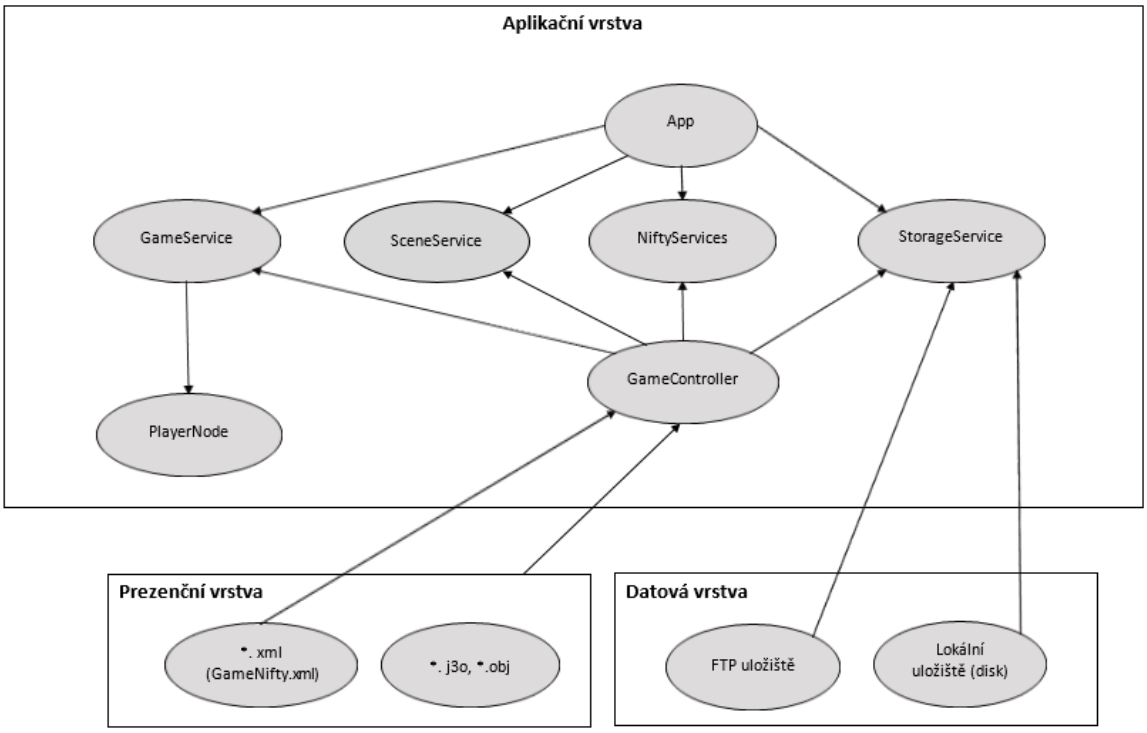
6 Implementace

Po teoretické části, kde je rozebrána problematika 3D scény, game enginu, webových aplikací a je provedena analýza game enginů a návrhové části, kde jsou uvedeny využité technologie a architektura, je přistoupeno k samotné implementaci.

Diplomová práce je z části postavena na bakalářské práci „Tvorba interaktivních průchodů v programu Blender“. 3D modely jsou zpracovávány v programu Blender a zbytek projektu pomocí game engine jMonkeyEngine 3.0.10, jeho knihoven a programu JME SDK 3.0.

Při implementaci byly využity knižní publikace, dokumentace a návody na stránkách jMonkeyEngine. Využívána byla také informace z internetových diskuzích.

Na následujícím obrázku (Obr. 9) lze vidět architekturu hry v rámci tříd a využívaných souborů.



Obr. 9 Architektura webové aplikace dle hlavních tříd

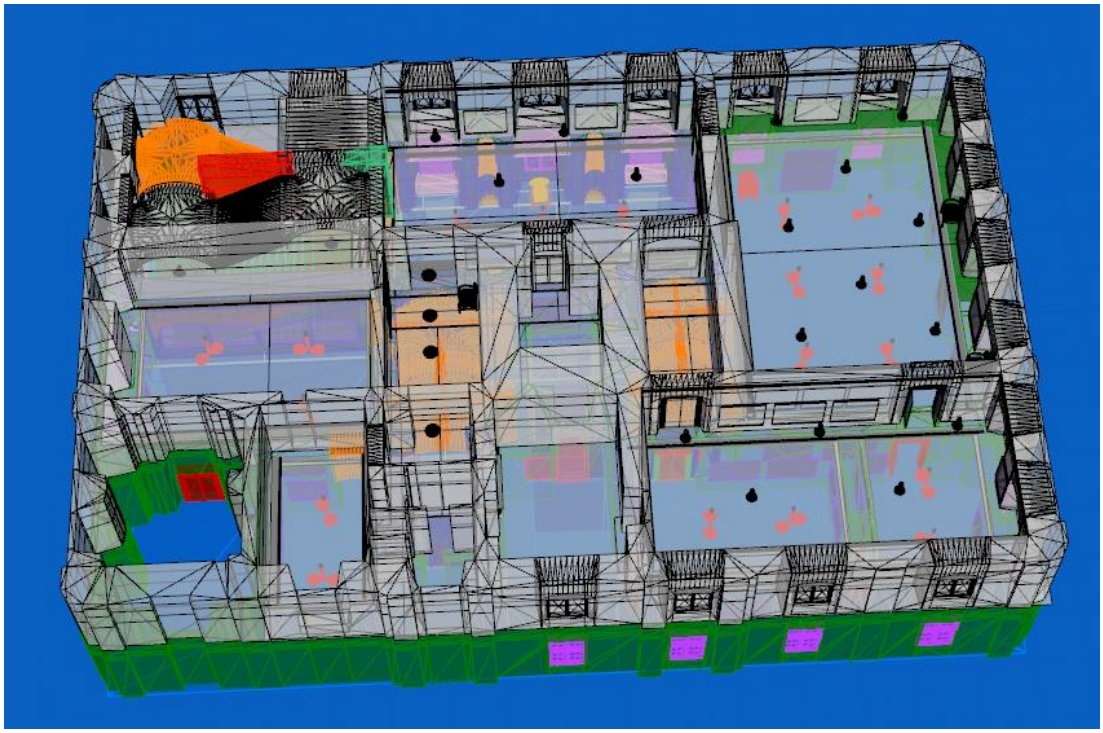
6.1 Modely

Z bakalářské práce je převzat základní model budovy, který byl následně upraven. Při jeho tvorbě byla udělána velká chyba nerozčlenění scén. Celkový model byl tedy hodně velký a graficky náročný. Pro účely této práce bylo potřeba model více

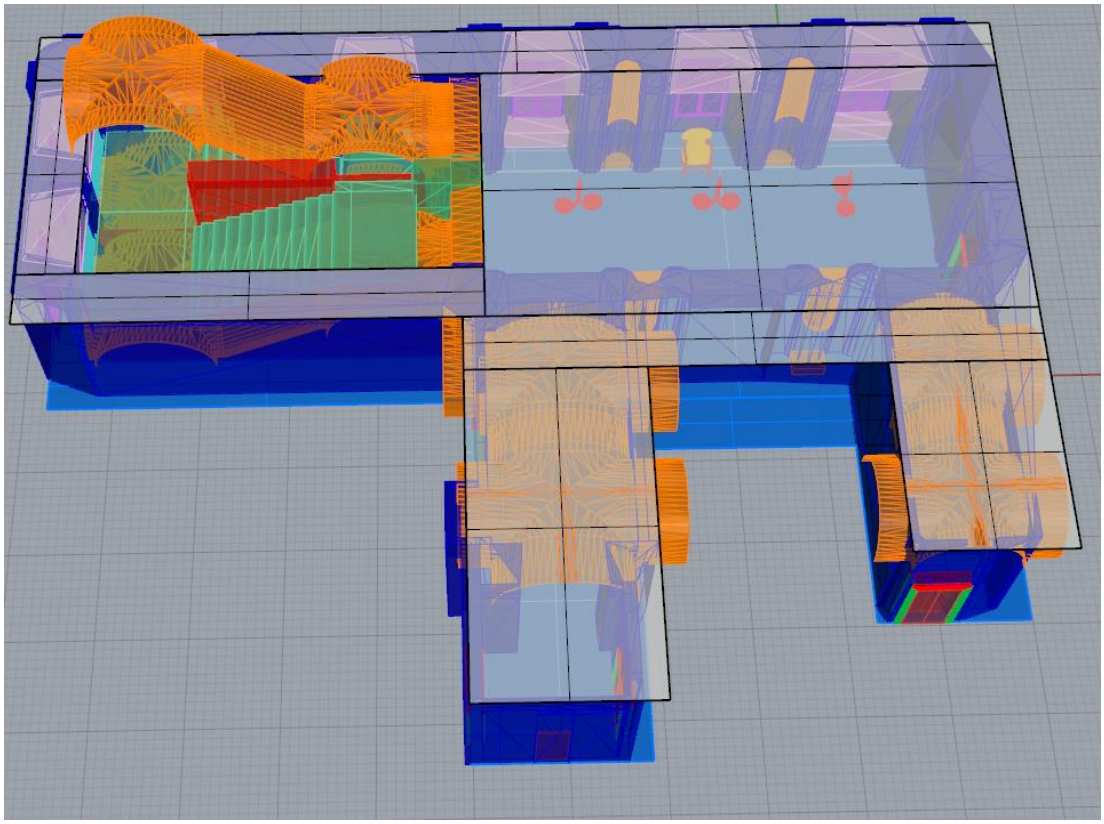
rozčlenit a utvořit jednotlivé scény. Budova proto byla rozdělena na jednotlivé scény, nejčastěji se jednalo o samostatné pokoje.

Pro úpravu byly využity grafické programy Blender a Rhinoceros. Postupně byly vyzkoušeny funkce pro rozčlenění objektu, mezi které patří střih, boolean - rozdíl, apod. Tyto nástroje postupně kvůli struktuře modelu selhali. Jedinými vhodnými, ale zdlouhavými, možnostmi se ukázalo „rozbít“ mesh síť a odstranit nepotřebné části, nebo vybudovat potřebné části znovu. Rozdělit bylo potřeba především zdi místností a schody v patrech.

V některých případech bylo možné využít již zmiňované základní nástroje. Například rozčlenit schody nebyl tak zásadní problém jako vytvoření jednotlivých místností. Pro nejlepší výsledek bylo otestováno vytvoření nových stěn, stejně jako vymazání částí sítě. V konečném výsledku se jedná o kombinaci obou akcí. Místnosti byly odděleny vymazáním mesh sítí ostatních místností. V chybějících neboli v nedokonalých částech byla síť doplněna novými plochami. Nakonec byl celý model sjednocen a opět doplněn o součásti místnosti.



Obr. 10 Celý původní model radnice (BP)

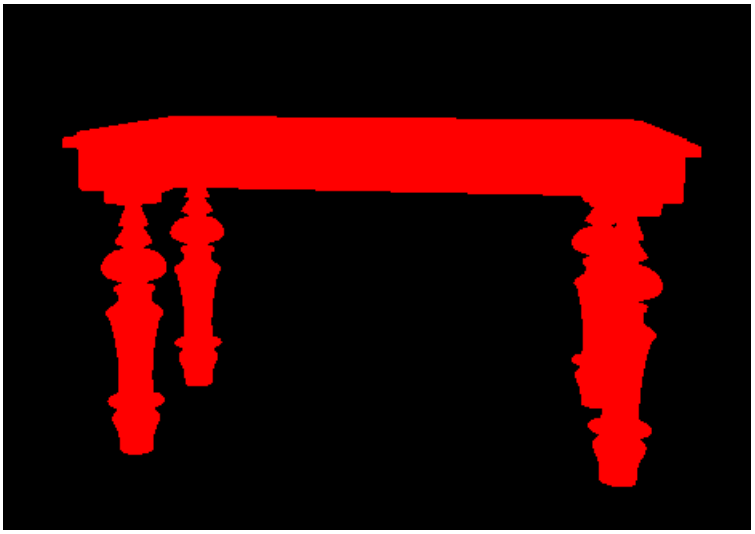


Obr. 11 První scéna hry - ořez původní scény

6.1.1 Importování modelů

Pro účely práce byly objekty exportovány v podporovaném formátu .obj. Jedná se o textový formát, který exportuje souřadnice vrcholů a textur. Se souborem OBJ se automaticky z programu Blender exportuje soubor .mtl, který udržuje více informací o materiálech a texturách.

I přes soubor .mtl se u projektu vyskytly problémy v zobrazování textur. Některé byli zobrazeny ve špatném rozlišení, transparentnosti nebo zcela scházely. Objekty s problémovou nebo chybějící texturou byly opětovně prozkoumány a případně přetexturovány v grafických programech. Bylo zjištěno několik špatných nastavení materiálu a textur. V několika případech byli překážky opraveny. Bohužel se ale u některých objektů nepodařilo přijít na základní příčinu chybovosti, a proto bylo využito otexturování objektů přímo v programu JME SDK. Jak je vidět na Obr. 12, objekt byl nahrát bez funkční textury, proto bylo využito texturování přímo v JME SDK, který poskytuje dvě možnosti. Celé otexturování provést pouze kódem, který musí zahrnovat vytvoření nového materiálu a načtení a nasazení textury. Materiál je poté nutné aplikovat na objekt. Druhou variantou je přímo v programu vytvořit materiál, tedy soubor formátu .j3m, který bude obsahovat zadané informace i načítání textury. Tento materiál je následně na objekt pouze načten a nasazen.



Obr. 12 Model bez nahraný bez textury

```
Material matT = new Material(am,
      ModelConsts.MATERIAL_UNSHADED);
matT.setTexture("ColorMap", am.loadTexture(new TextureKey(
      "Models/wood/wood.jpg")));
table.setMaterial(matT);

table.setMaterial(am.loadMaterial(
      "Materials/Generated/mTable.j3m");
```

Při základním nastavení parametrů je výsledek totožný (viz Obr. 13). Parametry lez stanovit například natočení textury, její opakování, rozlišení apod.

Obr. 13 Výsledek texturování modelu v programu JME SDK

jMonkeyEngine zvládá pracovat pouze se soubory formátu .j3o, .scene nebo .mesh. .mesh a .scene lze konvertovat například z formátu .ogre. Soubor .obj je možné konvertovat do formátu .j3o.

Pokud model exportovaný do souboru není v podobě mesh sítě, je automaticky převeden na daný typ. Což je výhodně pro efektivnější zpracování.

K dílčím hrám bylo vytvořeno několik objektů, které doplní scénu a umožní rozvíjet herní příběh a možnosti. Jedná se například o již zobrazený stůl na Obr. 13, na kterém se nachází dílčí hra v podobě kvízu (více v kapitole Děj a dílčí hry) nebo obraz s rámem, za kterým se skrývá hra „Trezor“, k jehož otevření je třeba vyřešit jiné dílčích úkolů.

Pro usnadnění načítání většího množství objektů a scén byla vytvořena metoda *createModel(),* která nejenže načítá samotný model, ale také nastavuje kolizní vlastnosti a fyziku objektu.

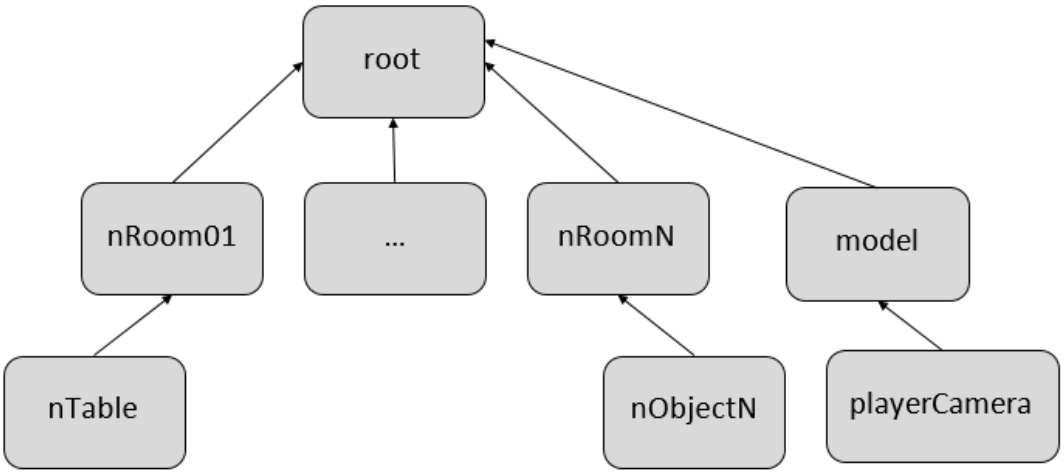
```
spatial.addControl(new RigidBodyControl(0));
App.getInstance().getPhysics().add(spatial);
```

6.2 Scéna

Viditelnost prvků stejně jako vykreslení scény a nastavení světelných zdrojů bylo zajištěno pomocí game engine jMonkeyEngine, který využívá grafický engine pro automatické výpočty a urychlení vykreslování realistické trojrozměrné scény v reálném čase. jMonkeyEngine pro vykreslování 3D scény využívá knihovnu LWJGL a základní funkce OpenGL. Pro účely práce nebylo třeba nastavení upravovat ani rozšiřovat.

Vykreslení scény je možné ovládat také přes již zmiňovaný graf scén, který umožnuje data v ní obsažená uspořádat do hierarchické stromové struktury, podle které je možné ovládat zobrazování jednotlivých prvků. V základu graf scén zobrazuje kořen (rootNode) a s ním i jeho potomky.

V základu jedné scény je možné jej využít k uspořádání objektů a jejich vykreslení na scéně. V této práci je graf scény v širším měřítku využíván pro jednotlivé místnosti. Například při změně místnosti (scény) se pomocí hierarchie lehce překrývají jednotlivé scény, a tak není nutné vykreslovat všechny scény najednou a využívat příliš výpočetního výkonu. V každé místnosti je tako možné regulovat i objekty v ní zahrnuty. V první místnosti se využívá například na stůl, do jehož uzlu se řadí věci na něm obsažené. Základní ukázkový diagram hierarchie je zobrazen na Obr. 14, který zobrazuje celkovou hru jako roodNode, místnosti jako jeho potomky, ve kterých roomNode obsahují další potomky zahrnující objekty potřebné pro danou scénu. Vykreslování poté záleží na nastavení jednotlivých uzlů v grafu scén.



Obr. 14 Graf scény – vzorová hierarchie uzlů hry

Scéna nezahrnuje pouze konstantní objekty dané místnosti (zdi, klenby, schody atd.). Mezi elementy scény se řadí také kamera, případně model hráče a aktivní prvky (stůl, obrazy a jiné).

Zorné pole je stanoveno jako pohled z první osoby a je udáván aktuálním zaměřením pohledu uživatele a základním nastavením a rozsahem kamery, která využívá raycasting, který je používán pro zaměřování objektů. Standardně se používána jedna kamera, která vytváří monokulární obraz. Tuto kameru je možné upravovat před metodu *render()*, ale pro účely této práce je plně dostačující základní nastavení. Velikost viditelné scény udává také velikost okna, která není standardně definováno a záleží na volbě uživatele a velkosti zobrazovacího zařízení.

Pohledu hráče byla nastavena rozteč možné rotace po scéně. Horizontálně je možné otáčet hráče o 360° a vertikální rozteč je nastavena na maximum 60 * FastMath.DEG_TO_RAD a minimum -3 * FastMath.DEG_TO_RAD. Zobrazení je kontrolováno metodou *verticalRotate()*.

Světelné zdroje jsou vytvořeny ve třídě SceneService. Dané metody inicializují světlo daného typu a barvy, a jsou nastaveny ve třídě GameService, kde jsou metody volány.

Původně byl zvolen standartní světelný zdroj Ambient light, který ovlivňuje jas a barvu scény po celém světě nezávisle na směru ani umístění. Svítí všude. Toto světlo má velký nedostatek, protože nevrhá žádné stíny, objekty vypadají uměle a scéna je méně reálná.

Proto je zvolen jiný světelný zdroj. Vzhledem k využití ve vnitřních prostorách, je použito bodové světlo PointLight (*createPointLight()*).

```
public static PointLight createPointLight(
       Vector3f poz, float rad) {
```

```
  PointLight lamp = new PointLight();
  lamp.setPosition(poz);
  lamp.setRadius(rad);
  lamp.setColor(ColorRGBA.LightGray);
  return lamp;
}
```

Která umožnuje zadání pozice a síly záře v parametru metody. Pro první místnost je použito 5 těchto světelných zdrojů na různých pozicích ve scéně se stejně nastavenou silou záře na 25 a barvou světla světle šedou. Přidání vytvořeného světelného zdroje je přiřazeno do uzlu dané místnosti v tomto případě nRoom01.

```
nRoom01.addLight(SceneService.createPointLight(
       new Vector3f(-1, 0.5f, 25), 25));
```

Do venkovního okolí scény, vzhledem k oknům, je zasazeno světlo typu DirectionalLight (*createDirectLight()*)nahrazující slunce. K těmto světlům je také přidána metoda na vytvoření stínů (PointLightShadowRenderer) *createShadows()* pro docílení větší reálnosti scény.

```
public static void createShadows(PointLight light) {
  FilterPostProcessor fpp = new
       FilterPostProcessor(App.getInstance().
       getAssetManager());
  PointLightShadowRenderer plsr = new
       PointLightShadowRenderer(App.getInstance().
       getAssetManager(), 2);
  plsr.setLight(light);
  App.getInstance().getViewPort().addProcessor(fpp);
}
```

6.3 Menu

Celý kód k vzhledu a rozložení hernímu menu je k nahlédnutí na přiloženém CD v souboru GameNifty.xml. Tato třída obsahuje jak kód XML pro vzhled herního menu tak i pro vzhled dílčích her. Jednotlivé části by bylo možné pro přehlednost rozdělit do jednotlivých souborů. Toto řešení nebylo zvoleno z důvodu nutnosti další režie na řízení NiftyDisplay, který umožňuje použití jednoho zdroje Nifty.

6.3.1 Události a zobrazení

Samotnout obsluhu událostí zajišťuje třída GameControler a její metody. Například se jedná o přechody mezi jednotlivými obrazovkami pomocí metody *navigateTo()*, anebo o specializované metody pro správu hry jako jsou *loadGame()*, *saveGame()*, *quitGame()* a další. Tyto metody dále předávají řízení podle potřeby dalším třídám

zajišťující běh hry. Jedná se například o třídu StorageServices, která slouží pro ukládání a načítání.

Propojení akcí komponent s aplikací je pro každou vytvořenou obrazovku *<screen>* v XML definováno pomocí atributu cotroller, jehož hodnotou je název třídy včetně její „class path“ zajišťující obsluhu.

```
<screen id="start" controller="radnice.controllers.
       nifty.GameController">
```

Nicméně povolení uživatelské interakce myší je dále povolen až pomocí atributu visibleToMouse na komponentě panel.

```
<panel childLayout="vertical" visibleToMouse="true">
```

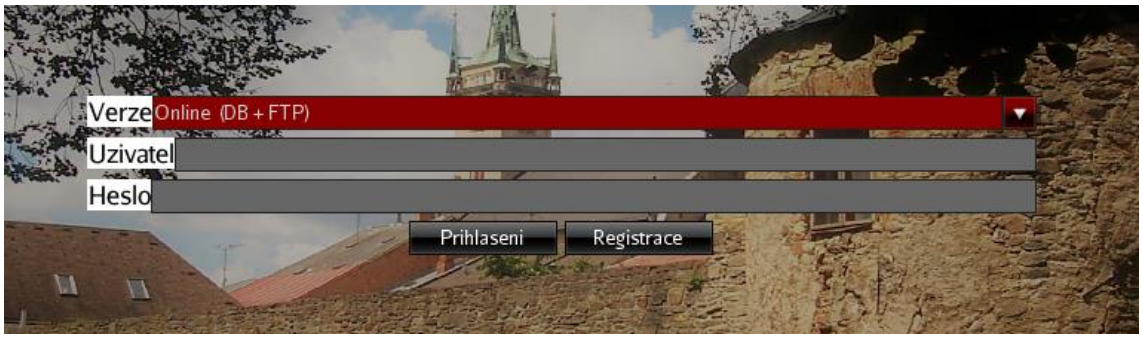
Všechny vytvořené obrazovky využívájí defaultní styly a komponenty definové pomocí následujícího kódu.

```
<useStyles filename="nifty-default-styles.xml" />
<useControls filename="nifty-default-controls.xml" />
```

Pro finální rozložení grafický prvků *<control>* (tlačítka, textová pole, apod.) obsahují obrazovky elementy vrstev *<layer>* a panelů *<panel>*.

6.3.2 Přihlášení a registrace

Přihlašovací obrazovka (Obr. 15) je první, co uživatel uvidí při spuštění hry. Nabízí možnost volby, zda bude chtít hráč hrát offline (lokálně) nebo online (s připojením k databázi a FTP uložišti). Obrazovka tedy obsahuje komponenty pro výběr způsobu připojení (dropDown) a textová pole (textField) pro zadání přihlašovacích údajů včetně tlačítek pro potvrzení zvolené akce. Samotné přihlašování a registrace je zajištěno metodami ve třídě DatabaseService, která je rozebrána v kapitole 6.4 Jádro.



Obr. 15 Úvodní obrazovka – přihlášení a registrace

6.3.3 Herní menu

Po přihlášení jedním ze dvou způsobů je zobrazena hlavní obrazovka, v níž je začleněno herní menu, které obsahuje nabídku funkcí (nová hra, uložení, načítání, ukončení a informace).

Obr. 16 Herní menu

Některé z těchto funkcí zobrazí specifické obrazovky, meny a nebo provedou jednoduchou akci. Například tlačítko „Ukončit“ volá metodu *quitGame()*, která pouze skončí zobrazení obrazovky a ukončí běh aplikace.

```
public void quitGame() {
  NiftyService.showScreen(NiftyConsts.SCREEN_END);
  App.exit();
}.
```

Dále po stisknutí tlačítka „Info“ se vyvolána metoda *aboutScreen()*, která překreslí menu a zobrazí nabídku nápovědy (podobná jako při nové hře) a informace o tvůrci a využité technice. Samozřejmostí je existunce také tlačítka „Zpět“ pro návrat.

Do tohoto menu se lze také vrátit ze spuštěné hry a ostatních obrazovek pomocí klávesy „esc“. Registrace této události obstarává metoda *basicActions()* a o provedení samotné akce se stará metoda *actionGame()* ze třídy GameService.

```
public static void basicActions() {

  //nacteni zdroju
```

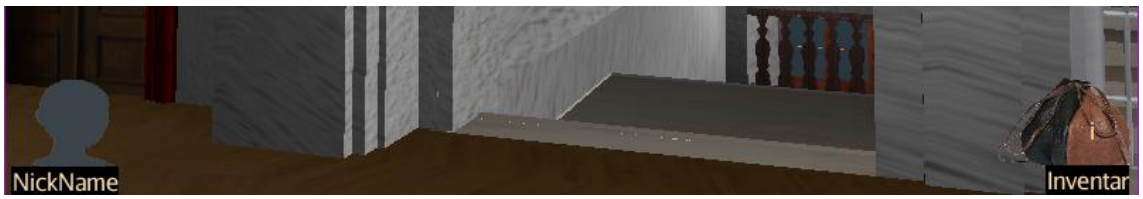
```
  App app = App.getInstance();
InputManager inputManager = app.getInputManager();

  //zmena mapovani
  inputManager.deleteMapping(
       SimpleApplication.INPUT_MAPPING_EXIT);
  inputManager.addMapping("Menu", new KeyTrigger(
       KeyInput.KEY_ESCAPE));

  //vytvoreni odposlechu
  inputManager.addListener(app, "Menu");
}
```

Mezi základní vlastnosti patří také nápověda a informace o aplikaci. Obrazovka pro stisku tlačítka Info je rozdělena na tři následné možnosti. Tlačítko zpět, jehož funkce je zřejmá, a tlačítka Informace a O aplikaci. Informace spouští podobnou obrazovku jako při novém spuštění hry (intro), kde je zobrazen scrollbar s obrázkem kláves pro ovládání. Tlačítko O aplikaci spouští základní informace o aplikaci. Jedná se o copyright, důvod vzniku a práva užití.

6.3.4 Nová hra

Tlačítko „Nová hra“ volá metodu *introScreen()*, která v následující obrazovce načte obrazovku intro, která zahrnuje především komponentu scrollPanel, ve kterém se nacházejí základní informace pro hráče (klávesy potřebné pro pohyb ve scéně, cíl hry a uvítání). Pod panelem jsou možnosti na pokračování („Pokračovat“) nebo návratu („Zpět“).

Po potvrzení se zobrazí scéna, obsahující elementy popsané v kapitole 6.2 Scéna (Obr. 17) a základní grafické prvky Nifty. Text pro zobrazování jména hráče a ikonka inventáře, pro jehož otevření je registrováno tlačítko „i“. Více o inventáři v kapitole Inventář.



Obr. 17 Menu na scéně

6.3.5 Uložit a Načíst

Při volbě tlačítka „Uložit“ (*saveScreen()*) nebo „Načíst“ (*loadScreen()*) se zobrazí podobné obrazovky, které obsahují kontroler *listBox*, ve kterém se hierarchicky, podle data uložení, znázorňují již uložené hry. Dále zahrnují *textfield* pro zapsání

názvu uložení. V případě načítání zobrazí název vybraného souboru. Následně je možné soubor uložit/načíst nebo se vrátit na úvodní herní menu tlačítkem „Zpět“.

Podrobný popis ukládání a načítání dat je popsán v kapitole 6.7 Uložiště.

Obr. 18 Prázdná obrazovka pro načtení

6.4 Jádro

Třída App je hlavní třídou, která spouští samotnou aplikaci. Dědící z třídy SimpleApplication, která obsahuje základní instance pro běh aplikace. Aplikace z ní především používá instance inputManager (myš), audioManager (zvuk), assetManager (Nifty displej), roodNode (hlavní uzel), stateManager (fyzika) a přetěžuje její metody:

- *simpleUpdate()* - zajišťuje hlavní smyčku aplikace, díky které dochází ke změnám ve hře (posun pozice, změna kamery),
- *simpleInitApp()* - volaná při vytváření hry a slouží pro inicializaci instancí, nastavení defaultních hodnot a vytvoření nifty.

Třída dále implementuje rozhraní ActionListener → metodu *onAction()* (předání informace o události při zmáčknutí registrovaného tlačítka) a AnalogListener → *onAnalog()* (předání informace o události při registrovaném pohybu myši).

Vlastní potřebné instance využívá pro snadnější běh, mezi které patří nifty (vzhled xml), state (třídy BulletAppState z jME pro fyziku), player (hráč), status (příznak pro ovládání menu), version (příznak verze připojení) a connection (konektor databáze).

Pomocné služby, zpracování dat a komunikaci mezi třídami zajišťují statické metody těchto tříd (jejich vzájemné propojení je zobrazeno na Obr. 9):

- GameService – nastavení hry a její příprava (hráč a scéna),
- SceneService – inicializace instancí modelů, světel a dalších prvků pro scénu,
- NiftyService – načítání a úprava dat kompont GameNifty.xml,
- DatabaseService – správa dat databáze (uživetelé a uložené hry),

- StorageService – specifické ukládání a načítání.

Všechny akce (action()) jsou základně odposlouchávány v App a dále jsou zasílány do GameService, kde se akce dělí na základní a hráčské akce. Základní jsou mapovány a realizovány přímo v GameService, na rozdíl od hráčských, které jsou implementovány v PlayerNode. Stejný princip platí pro analog() a update().

6.5 Hráč

Třída PlayerData slouží k vytvoření hráče. Hlavním cílem bylo dosáhnout pohledu tzv. první osoby, pro průchod scénou. Tohoto pohledu lze dosáhnout jednou kamerou, při monokulární nastavení. Charakteru (hráči) je umožněno pohybovat se po scéně a reagovat na jiné objekty (detekce kolizí a zaměřování objektů). Na scéně se nacházejí pouze objekty aktivní. Hráč tedy žádným z objektů nemůže projít. Aktivní objekty lze dále rozdělit na ty, které provedou reakci a spustí nějakou viditelnou akci, a na ty které pouze při reakci nepropustí hráče skrz. Viz níže.

6.5.1 Inicializace

Při inicializaci hráče mohou nastat dvě možné skutečnosti. Buď je hráč vytvářen nově (Nová hra) nebo je načten již existující z uloženého souboru.

Pokud je hráč načítán, je volána prázdná instance a ze souboru jsou jí metodou *initLoad()* přiřazena data. V obou variantách je hráči nastavena kamera a pozice podle dat. V případě nového hráče je třeba nastavit jméno (přezdívku) a následně je zavolána metoda *initNew()* a *setup()* z třídy PlayerData. Metoda *initNew()* inicializuje také charakter (BetterCharacterControl), inventář (InventoryList) a kvíz (QuizList) a případně model (Spatial) přestavující postavu. Dále jsou uvedeny důležité body metody *initNew()*.

```
//data
  quiz = new QuizList();
  inventory = new InventoryList();
//kamera
  playerCamera = new CameraNode("playerCamera", gameCamera);
//hrac
  character = new BetterCharacterControl(1.2f, 4f, 550f);
```

Při tvorbě byla vytvořena „postava“, tedy model přestavující charakter pro snadnější testování scény, vlastností, pozicování a nastavení. V této chvíli byl využit pohled „third person view“. Tato „postava“ byla při konci práce odstraněna a ponechán byl pouze kamerový pohled a zastávající charakter, aby bylo docíleno pocitu průchodu hráčem („first person view“), jak je vidět na Obr. 19.

Obr. 19 First-person view

6.5.2 Akce a události

Veškeré schopnosti hráče byli naimplementovány ve třídě PlayerData. Obsahuje metodu *update()*, která je závislá na hlavní smyčce aplikace a má za úkol obnovu zobrazení kamery při posunu nebo otočení.

Nejdůležitějšími metodami jsou *setUpKeys()*, *action()* (onAction volá GameService a ten volá tuto metodu) a *analog()*.

Metoda *setUpKeys()* využívá instanci aplikace třídy InputManager, která je schopná vytvořit mapování kláves a pohyby myší. Toto mapování lze dále nastavit pro odposlech (onAction a onAnalog).

```
inputManager.addMapping("Left", new KeyTrigger(
     KeyInput.KEY_A));
inputManager.addListener(app, "Left");
```

Metoda action() je provolána z hlavní třídy aplikace a má za úkol nastavit příznaky pohybu na základě mapování a odposlechu klávesnice.

```
if (binding.equals("Left"))
  {
     left = value;
  } …
```

Metoda *analog()* je obdoba metody *action()*. Na rozdíl od ní jde o odposlech pohybu myši, která se využívá pro rotaci pohledu. Pro změnu směru pohybu je využito hyperkomplexní číslo Quaternion, které rozšiřuje třídimenzionální rotaci na čtyřdimenzionální.

```
public void analog(String binding, float value, float tpf)
```

```
{
if (binding.equals("TurnLeft"))
{
   Quaternion turn = new Quaternion();
   turn.fromAngleAxis(
        mouselookSpeed*value, Vector3f.UNIT_Y);
   character.setViewDirection(turn.mult(
        character.getViewDirection()));
} …
```

Po zjištění zaměření aktivního objektu, který má na sebe navázanou dílčí činnost, je možné pomocí levého tlačítka myši spustit akci objektu. Jedná se o uložení informaci o objektu do inventáře nebo spuštění dílčí hry. Kontrola zmáčknutí tlačítka je prováděna stejně jako u klávesových tlačítek, jak je možné vidět na následujícím kódu.

```
inputManager.addMapping("Action", new MouseButtonTrigger(
   MouseInput.BUTTON_LEFT));
```

Důležitou událostí hráče je výběr objektů pro další možné zpracování. Samotné nalezení vybraného objektu funguje na principu hledání objektů (třída CollisionResults) protínajicí se s přímkou (třída Ray) vytvořenou na základě pohledu kamery hráče. Z takto nalezených výsledků je vybrán nejbližší objekt, který je dále ověřen vůči globální kolekci obsahující aktivní objekty scény s přiřazenými akcemi. Tuto funkcionalitu zajištuje metoda *checkoutAim()* ve třídě PlayerData.

```
private void checkoutAim() {
  CollisionResults results = new CollisionResults();
  Ray ray = new Ray(gameCamera.getLocation(),
       gameCamera.getDirection());
  App.getInstance().getRootNode().collideWith(ray, results);
  if (results.size() > 0) {
      Spatial target = results.getClosestCollision().
            getGeometry();
       try {
            UseableObject useable = target.getParent().
                 getUserData("useable");
            Action event = App.getInstance().
                 getCollisions().get(useable);
            if (event != null) {
                 event.actionPerformed(null);
            }
       } catch (Exception e) {
       }
}
```

```
}
```

Definované akce jsou využity pro spouštění dílčích her, používání nebo sbírání předmětů do inventáře. Dílčí hry a struktura inventáře jsou blíže popsány v kapitole 6.6 Děj a dílčí hry.

6.5.3 Detekce kolizí a fyzikálních vlastností

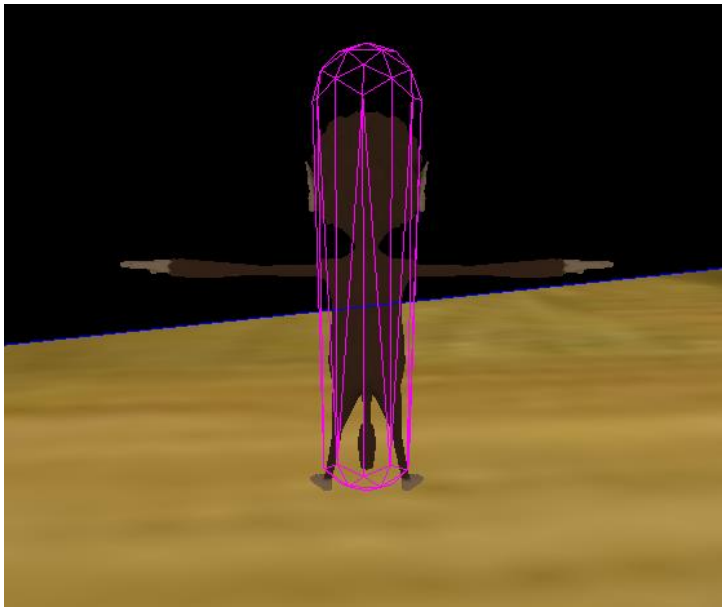
Veškeré propočty detekcí kolizí a fyzikálních simulací provádí jMonkeyEngine automaticky na pozadí aplikace. Ve specifických situacích se dají vlastnosti upravit. Pro záměry projektu zcela postačují přednastavené funkce a hodnoty.

Jak bylo řečeno objekt hráče v sobě díky třídě BetterCharacterControl obsahuje již při vytvoření základní fyzikální vlastnosti (přirozená gravitace) a detekci kolizí s objekty obsahujícími kolizi. Základní nastavení pro game engine (BetterCharacterPlayer) lze vidět na následujícím kódu.

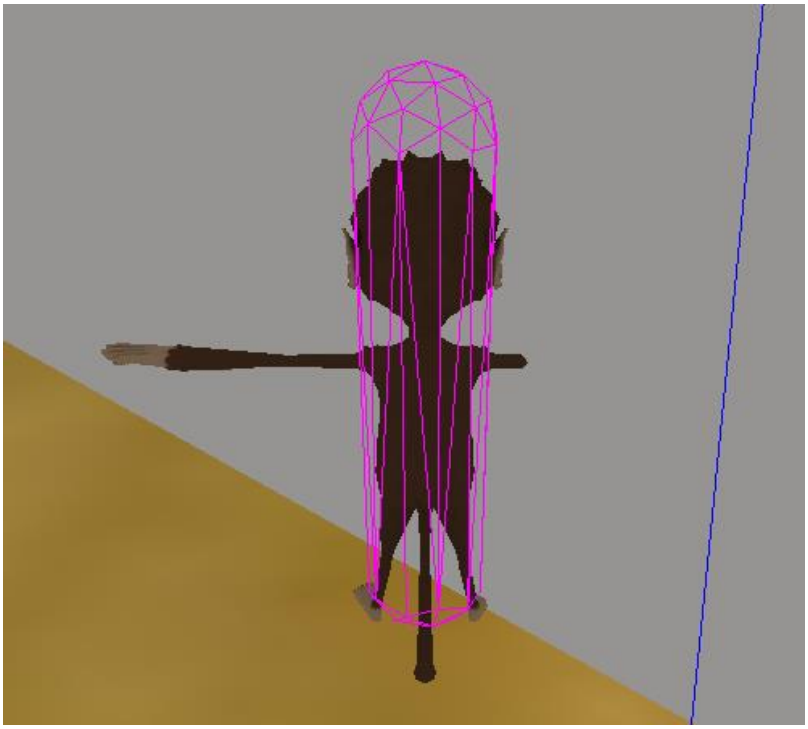
```
protected CollisionShape getShape() {
  CapsuleCollisionShape capsuleCollisionShape = new
       CapsuleCollisionShape(getFinalRadius(), (
            getFinalHeight() - (2 * getFinalRadius())));
  CompoundCollisionShape compoundCollisionShape = new
       CompoundCollisionShape();
  Vector3f addLocation = new Vector3f(0, (
       getFinalHeight() / 2.0f), 0);
  compoundCollisionShape.addChildShape(
            capsuleCollisionShape, addLocation);
  return compoundCollisionShape;
}
```

Tyto vlastnosti jsou pro hru dostačující a nijak se neliší od standardního.

Na Obr. 20 je možné pozorovat původní model využitý jako postavu „zabalenou“ do obálky. Detekce kolizí tak velmi snižuje náročnost na výpočetní výkon, ale právě v tomto případě je nevhodné rozpažení postavy, příčinou nereálného vykreslení. Jak je vidět na druhém obrázku (Obr. 21) dochází k průniku modelu postavy do modelu zdi. Ne v každém případu je tedy vhodné využití automatického nastavení a pro realistický výsledek je třeba nastavení obálek věnovat více času.



Obr. 20 Obálka využita na modelu postavy



Obr. 21 Nedokonalost využité obálky

Ostatní objekty scény nemají automatické vytvoření detekce kolizí ani jiných fyzikálních vlastností, a proto je třeba přidat každému modelu ovladač typu RigidBodyControl a následně jej zaregistrovat do fyziky aplikace (BulletAppState).

```
public static Spatial createModel(
  String model, boolean collision) {
        Spatial spatial = App.getInstance().getAssetManager()
              .loadModel(model);
        if(collision) {
              spatial.addControl(new RigidBodyControl(0));
              App.getInstance().getPhysics().add(spatial);
        }
  return spatial;
}
```

Při volání konstruktoru třídy RidigBodyControl je automaticky definována obálka modelu (CollisonShape) a lze ji argumenty konstruktoru upravit. V tomto případě je argumentem mass typu float. Z kódu je patrné, že do obálky je „zabalen" model „spatial", tedy každý nově načtený model.

6.6 Děj a dílčí hry

Velkou část hratelnosti tvoří právě dílčí hry, ve kterých je utvářeno povědomí o historii. Veškeré tyto hry byli vytvořeny za využití knihovny NiftyGUI (XML – vzhled) a programovacího jazyka Java (funkce). A jejich specifikace podoby je zaznamenána stejně jako herní menu v GameNifty.xml. Hry se zobrazují při nalezení bodu ve scéně pomocí metody *checkoutAim()*, ale samy o sobě jsou zobrazeny mimo 3D scénu. Metoda *checkoutAim()* je také využívána také při kolizi s objekty, které uživatel může sebrat a přidat do inventáře.

Interakci v dílčích hrách, jako např. ve hře Kvíz obstarává třída GameController, která v tomto případě obstarává načítání otázek, kontrolu odpovědí, zobrazení kódu k trezoru atd.

S postupem v místnostech (scénách) se postupuje také po časové linii. První scéna založena na první místnosti odkrývá založení města a první historické „krůčky" města. Mezi posledními místnostmi by se měla vyskytovat moderní historie (Bohuslav Martinů, Tylův dům apod.)

6.6.1 Inventář

Hráč pro úschovu nalezených nebo získaných objektů z dílčích her využívá kolekci InventoryList, která obsahuje položky typu UseableObject inicializované z konstant. Tyto získané položky lze dále zobrazovat stisknutím tlačítka „I", používat k plnění dalších úkolů nebo k provedení určitých akcí (např. otevření dvěří do další scény). Přístup k inventáři je vytvořen pomocí metody *getInventory()* a přidání položky pomocí *addToInventory()* ve třídě PlayerData.

6.6.2 Kvíz

Pravděpodobně první dílčí hrou, ke které by měl hráč přistoupit je kvíz ukrytý na stole. Struktura daného kvízu je navržena ve třídě QuizList a je vrácena v metodě *getQuiz()* ve třídě PlayerData, pro další použití.

Jedná se v základu o 10 otázek vytvořených na základě konstant, díky kterým by měl hráč získat kód k další dílčí hře.

Principem kvízu je, že obsahuje 10 historických otázek o založení města, ale hráč potřebuje správně zodpovědět pouze 4. Pokud nedokáže z 10 otázek zodpovědět 4 správně, začnou se špatně zodpovězené otázky opakovat. Při každé správné odpovědi získá jednu ze čtyř číslic kódu k trezoru s další nápovědou. Trezor je ale schován a hráč jej musí nalézt. Důležitou součástí pro správné fungování kvízu je třída QuizConsts, která obsahuje předdefinovaný výsledek (kód k trezoru)

```
public static final char[] ANSWER = new char[]{'1', '2',
  '6', '5'},
```

otázky, jednotlivé odpovědi a správnou odpověď.

```
QUESTIONS.add(new QuizQuestion(
  "Kdo založil město Polička?",
  "Zikmund Lucemburský",
  "Přemysl Otakar II.",
  " Rudolf I. Habsburský",
  AnswerOption.B //správná odpověď
  )
);
```

Datová struktura otázek je definována ve třídě QuizQuestion.

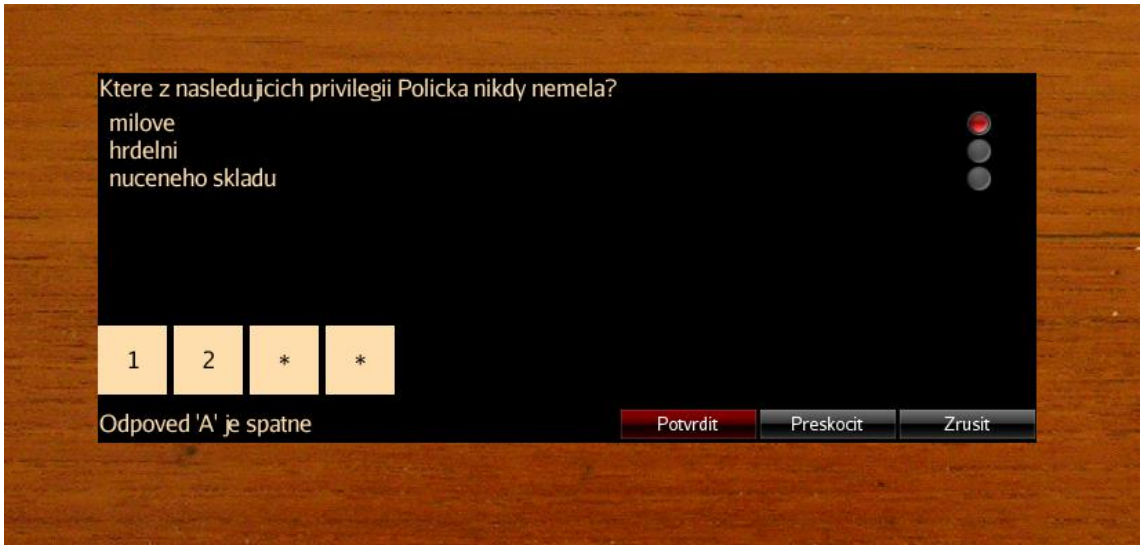
```
public QuizQuestion(String query, String optionA, String
  optionB, String optionC, AnswerOption answer) {
        this.query = query;
        this.optionA = optionA;
        this.optionB = optionB;
        this.optionC = optionC;
        this.answer = answer;
}
```

Na Obr. 22 je uvedena úvodní obrazovka, vytvořena v gameNifty.xml za pomoci obrázku a tlačítka. Na následujícím obrázku je zobrazen kvíz, ve kterém bylo dvakrát odpovězeno správně (dvě odhalená čísla kódu) a momentálně odpovězeno špatně (informace na dolním řádku obrazovky). Dále obrazovka znázorňuje novou otázku, zaškrtnutou položku a možnosti Potvrdit (potvrzení otázky), Přeskočit

(přeskočení na další) a Zrušit (vypnutí kvízu). Po splnění úkolu se získaný kód hráči uloží do inventáře, aby jej nezapomněl, než nalezne trezor k otevření.

Instance třídy QuizStatus dohlíží na status kvízu. Pokud hráč opustí ještě nedokončený kvíz, tak při návratu bude ve stejném stavu, jako při odchodu. V případě že kvíz dokončí, není možné v něm nadále pokračovat. Při dokončení se kvíz s kódem přesune hráči do inventáře a nadále již není aktivní.

Obr. 22 Úvodní obrazovka dílčí hry „Kvíz“



Obr. 23 Herní pole v kvízu

6.6.3 Doplňky a vizuální efekty

Pro doplnění scény byli použity doplňky, které využívají fyzikální vlastnosti, jako je míč, se kterým si hráč může kopat, nebo vizuální efekt hořících svíček. Plamen byl vytvořen přímo za pomocí game enginu funkcí Emitter s úpravou vlastností, aby připomínal hořící plamen. Tuto funkci je možné přenastavit na padající vločky sněhu, deště nebo zářící palety. Jiné přenesené ani vnořené animace použity nebyly, ale do budoucna se počítá s rozšířením. Stejně tak by bylo vhodné zužitkovat fyzikální vlastnosti pro jiné prvky.

Obr. 24 Vizuální efekt (částicový systém) – animace hořící svíčky

6.6.4 Nová scéna

Při splnění všech úkolů v dané scéně se uvolní průchod, obvykle dveře, do jiné scény. Například při přechodu z první scény do následující musí hráč nalézt dveře, ke kterým patří získaný klíč. Po nalezení dveří a využití klíče je hráč „přenesen“ do scény nové. Při přechodu je díky metodě *prepareGame()* z třídy GameService stará scéna (nRoom01) odstraněna a následně nastavena po inicializaci nová scéna (nRoom02), která již má přednastavený vzhled, osvětlení i nové dílčí úkoly.

6.7 Uložiště

Třída StorageService obsahuje statické metody pro ukládání a načítání dat rozehraných her uživatele za pomoci volání knihovních metod třídy SaveGame implementující základní uložení/načtení binárních souborů .j3o.

Hra díky online přístupu umožnuje také profilování a tedy i ukládání a načítání svých her z více počítačů připojených k internetu. V kombinaci s webovým spouštění hry, lze tedy ke svým hrám přistupovat v podstatě odkudkoliv. Není-li

internet k dispozici, je alespoň možné ukládání her bez profilování na danou stanici.

6.7.1 Zpracovávané data

Knihovní ukládání/načítání pracuje pouze s objekty tříd implementující rozhraní Savable a tedy bylo nutné pro potřeby uložení kompletního stavu této hry rozšířit následující třídy:

- PlayerData – data hráče (pozice, kamera, aktivní scéna, inventář, kvíz),
- QuizList – data kvízu (otázky, stavy, klíč),
- QuizQuestion – data otázky (otázka, možnosti, správná odpověd, stav),
- InventoryList – data inventáře (položky),
- UseableObject – data položky (jméno, popis, model).

6.7.2 Zpracovávání dat

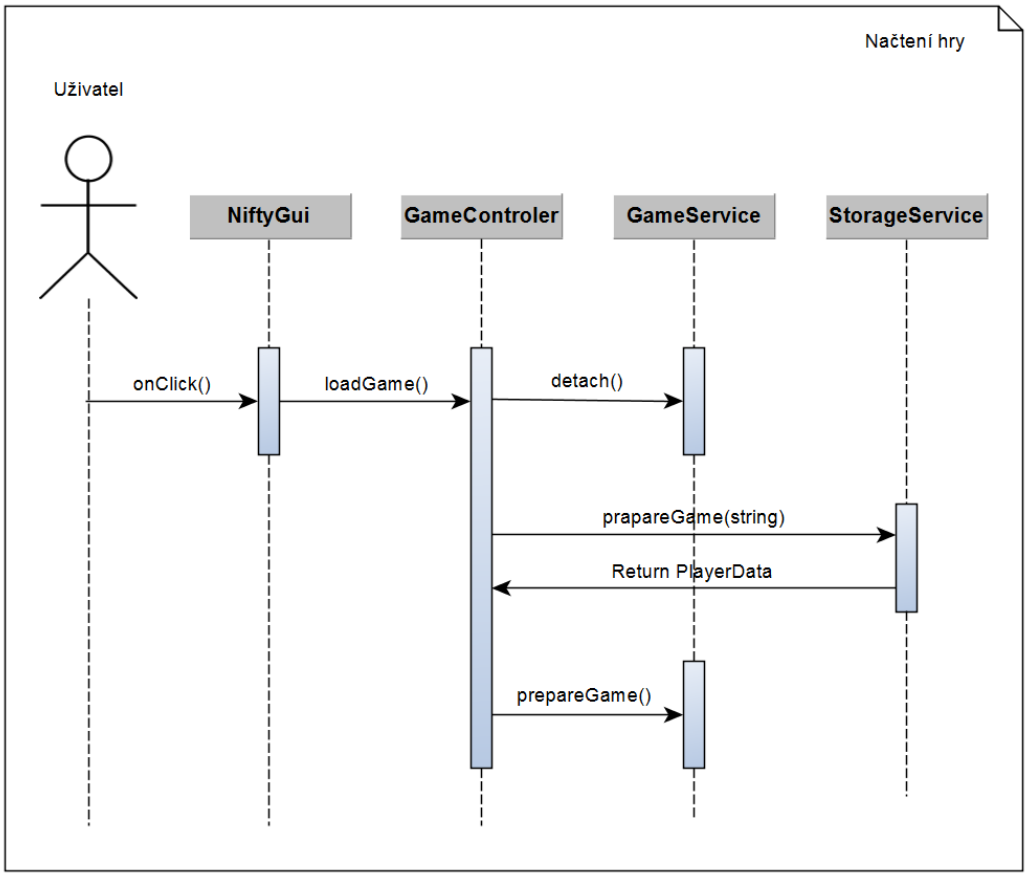
Samotné zpracování obaluje knihovní metody o možnost rozlišení použité verze připojení k definování automatických akcí a dělí se následovně:

- Online: Ukládání *saveToFtpl()* probíhající v tomto režimu po úspěšném uložení do binárního formátu provede FTP upload na uložiště webové stránky hry (webhosting) a uložení záznamu o hře do databáze pro daného hráče, v případě načítání (viz. Obr. 25) *loadFromFtp()* naopak před samotným převodem z binárního formátu provede FTP download na základě informací z databáze načtených do seznamu her, které jsou k dispozici,
- Offline: Ukládání *saveToLocal()* i načítání *loadFromLocal()* je zde pouze o knihovní funkcionalitu rozšířen systémem ověřování úspešného provedení.

Fyzické knihovní zpracování je založeno na využití rozhraní Savable používající metodu write při zápisu dat a metodu read během čtení binárních dat. Metody na následující ukázce zapisují/čtou data pro inventář, kameru a charakter. Každý takový použitý objekt může vlastnit a používat další Savable instance (rodičovská hierachie).

```
@Override
public void write(JmeExporter e) throws IOException {
  OutputCapsule capsule = e.getCapsule(this);
  capsule.write(inventory, "inventory", new InventoryList());
  capsule.write(playerCamera, "playerCamera",
       new CameraNode());
  capsule.write(character, "character",
       new BetterCharacterControl());
}
```

```
@Override
public void read(JmeImporter e) throws IOException {
  InputCapsule capsule = e.getCapsule(this);
  inventory = (InventoryList) capsule.readSavable(
        "inventory", new InventoryList());
  playerCamera = (CameraNode) capsule.readSavable(
        "playerCamera", new CameraNode());
  character = (BetterCharacterControl) capsule.readSavable(
        "character", new BetterCharacterControl());
  initLoaded();
}
```

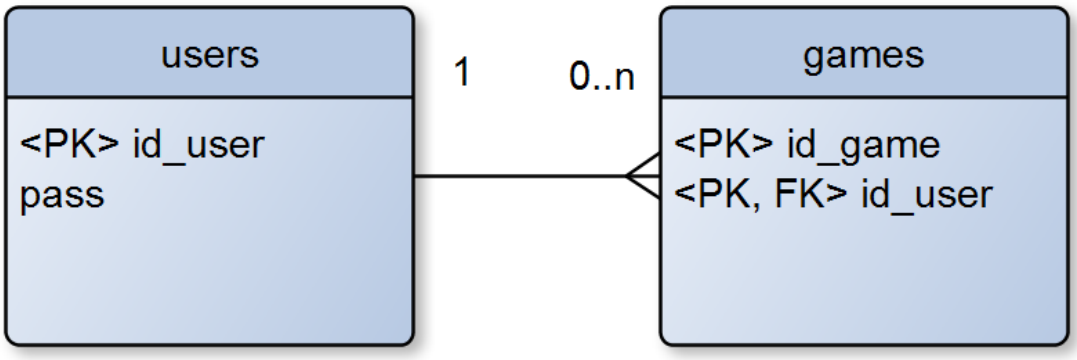


Obr. 25 Sekvenční diagram – Načítání hry

Pro online verzi se dále používá free MySQL databázi[9] pro profilování uživatelů a záznamů o uložených hrách a FTP přenos pro využití webhostingového uložiště. Realizace tabulek je zobrazena na Obr. 26 a pro její použití ve hře je využita zá-

[9] Zdroj databáze: https://www.freemysqlhosting.net/

kladní knihovna od MySQL[10] (JDBC ovladač) fungující na principu vytváření (PreparedStatement) dotazů vyhodnocených jako množina nalezených řádků (ResultSet).



Obr. 26 ERD – databázový systém

Finální přenos souboru na webhosting zajišťuje knihovna Apache Commons Net[11] pomocí metod *retrieveFile()* a *storeFile()* třídy FtpClient.

6.8 Webová aplikace a její implementace na webovou stránku

Aby bylo uskutečnitelné využití aplikace uživatelem, je třeba ji distribuovat. Existuje více možných variant publikace, mezi které patří již zmiňované (3.2.5) aplikace desktopové, typu Web Start, applet a mobilní.

Cílem této práce je umístění hry na internet, aby k ní měl každý přístup. Zde byl nakonec využit typ Web Start (.jar) namísto původně plánovaného appletu (.jar). Jak bylo stanoveno v cíli hra je také vytvořena jako desktopová aplikace pro Windows (.exe).

Pro spuštění aplikace v internetovém prohlížeči je potřeba mít v operačním systému nainstalovanou Java[12]. Tento požadavek je, ale dosti častý, a tak by s tím neměli být velké komplikace, nicméně je i zapotřebí nastavení adresy webových stránek do seznamu vyjímek zabezpečení (Java Security).

6.8.1 Distribuce

Před samotným vytvářením spustitelné verze byl upraven seznam využívaných souborů. Některým texturám bylo sníženo rozlišení, aby se zmenšila velikost na-

[10] Zdroj MySQL knihovny: https://dev.mysql.com/downloads/connector/j/

[11] Dokumentace FTP knihovny: https://commons.apache.org/proper/commons-net/apidocs/org/apache/commons/net/ftp/FTPClient.html

[12] Java – https://java.com/en/download/

hrávaného souboru a byly také odstraněny nepoužívané modely a modely nahrány v jiné scéně. Export aplikace do webové (applet) i desktopové varianty proběhl bez problému. Následně se začalo s testováním webové varianty na localhost umístění, při kterém se začaly vynořovat první problémy s aplikací nevyhovující Java Security. V případě nastavení výjimky došlo k dalšímu problému značící chybu „Invalid signature file digest for Manifest main attributes“ pro jejíž odstranění bylo zapotřebí úprava obsahu sestavených balíčků (.jar) a provedení jejich opětovného podepsání (viz Obr. 27), které sice vedlo k úspěšnému spuštění appletu, ale ve velmi nekonzistentním stavu (neinicializované objekty, problémy s přístupy).

```
Warning:
The signer certificate will expire within six months.
No -tsa or -tsacert is provided and this jar is not timestamped. Without a timestamp, users may not be able to validate
this jar after the signer certificate's expiration date (2017-04-29) or after any future revocation date.

C:\xampp\htdocs\radnice05\Applet>"C:\Program Files\Java\jdk1.8.0_73\bin\jarsigner.exe" lwjgl_util_applet.jar xportlov
Enter Passphrase for keystore:
jar signed.
```

Obr. 27 Soukromý podpis aplikace

V poslední době totiž nastává velký zlom v podpoře programu Java na webu, pravděpodobně zapříčiněného příchodem HTML5. Aplikace typu Java již plně nepodporuje Google Chrom a ostatní prohlížeče stanovily podporu pouze pro aplikace podepsány ověřeným nebo soukromým certifikátem s nastavenou výjimkou.

Protože prvotní umístění menší aplikace na webové stránky proběhlo v pořádku, nebyl problém na začátku odhalen a řešen. Při konečném nasazení bylo počítáno s podpisem aplikace a vložením na webové stránky jako při prvotním pokusu. Ani po dlouhém testování a pokusech o nápravu se nepodařilo opravit uvedené problémy, které neumožnily aplikaci spustit jako Java applet. Následně se vynořily také další problémy jMonkeyEngine ohledně nepodporování zobrazení kurzoru a přístupu na lokální disk z appletu.

Zmíněné problémy vedly k rozhodnutí o změně Appletu na Web Start umožňující potřebnou funckionalitu, kterému stačily pro správné spuštění soukromě podepsané zdroje s nastavenou výjimkou v Java Security. Jak již bylo řečeno výše Web Start aplikace je umístěna na webové stránce jako odkaz, při jehož „rozkliknutí“ se spustí v novém samostatném okně s hrou.

V případě desktopové aplikace se jednalo o jednoduchý test spuštění, který proběhnul v pořádku s možností využívat i soubory z Web Startu a naopak. Varianty byly nahrány na stránku dpxporlov.wz.cz, která byla založena pro distribuci testovací verze.

6.9 Optimalizace

Pro optimalizaci v oblasti scény je využito binární formy modelů (.j3o), které urychlují vykreslování, protože není nutné opětovně provádět konverzi modelu. Efektivnější je také lokálního uložiště, pro které jMonkeyEngine využívá binární

formát a urychluje tak ukládání a načítání souborů a tedy i vykreslení scény. Proto jej práce využívá.

Další optimalizaci je možné ohlídat při detekci kolizí, která může být na výkon velmi náročná. Je možné využívat základní tělesa nebo složitější objekty za jednoduché vydávat pomocí obálek těles. Pro reálnost scény je využití základních těles značně nepraktické. Z těchto důvodů byly některé objekty ohraničeny obálkou (CollisionShape).

I optimalizace kódu přispívá k lepšímu běhu aplikace, Mezi základní uspořádání patří vhodné provádění deklarací proměnných a vytváření základních objektů mimo smyčky a formování metod v efektivní, krátké a přehledné formě. Je vhodné také využívat menší počet smyček a „náročného“ kódu. Základním pravidlem, spíše kódu než optimalizace běhu, je pro lepší orientaci a efektivnosti při úpravách, dodržování jednoduchosti a přehlednosti názvů proměnných, metod a tříd.

7 Diskuze

Při importování základního modelu nastal problém v podobě špatně nebo vůbec nenahraných textur. Problém špatných textur byl částečně odstraněn přetexturováním modelu v grafickém programu a částečně otexturováním přímo v programu jMonkeyEngine. Další komplikací bylo nezobrazení některých ploch. Tato překážka měla být odstraněna úpravou sítě ploch, ale při bližším zkoumání bylo zjištěno, že problém je spíše optický než grafický. Stěny se ve scéně nacházely, pouze pod určitým úhlem ztrácely svoji viditelnost.

Další důležitý zádrhel pro českou hru vyvstal v textových polích, kde i při nastavení fontu s českou interpunkcí docházelo ke špatnému zobrazení. Proto byly důležité texty nahrazeny obrázky.

V současné době prohlížeč Google Chrome nepodporuje Java hry. (Oracle, 2015) Problémy s publikováním na webu má i Unity, který k zobrazení 3D obsahu přímo v prohlížeči potřebuje plugin „Unity Web Player“ a i po jeho stažení je na Windows kompatibilní pouze s prohlížeči, jako Internet Explorer, FireFox, Safari a Opera. (Unity, 2017c)

7.1 Využitelnost a přínos

Výstupem projektu je hra, která by měla mladší generaci přiblížit historii města Polička, a to formou hádanek a her. Celý děj se odehrává v Poličské radnici a v základu je určen pro všechny věkové kategorie, kteří umí číst a spustit počítače.

Vzhledem k vyvstalým překážkám byla testována pouze aplikace v podobě desktopové. Testování správné funkčnosti a grafického vzhledu aplikace odhalilo špatné nastavení některých položek v GameNifty.xml, které se objevovali při změně formátu velikosti zobrazovacího okna. Také byly objeveny objekty vysoké náročnosti, které zpomalovaly pohyb po scéně.

Pro testování aplikace bylo požádáno několik jedinců z různých věkových vrstev, kterým nebyly dány detailnější instrukce o ovládání a spuštění pro ověření intuitivnosti. Při testování a hodnocení uživatelské přívětivosti bylo nalezeno pár drobných detailů, jako je grafika scény, která má nižší úroveň a méně atraktivní menu. Hra sama o sobě je hratelná a intuitivní. Jak bylo zmíněno výše, pro méně obeznámené z obecnou funkcionalitou her je přidána úvodní informativní obrazovka o funkcích a ovládání hry. Tyto informace je možné nalézt také v nápovědě v hlavním menu.

Hra doposud nebyla nasazena na stránky muzea a její vystavení netrvalo ani dostatečně dlouho, na to aby bylo možné posoudit vyšší přínos. Stejně tak ekonomický přínos nemůže být zatím zhodnocen. Prospěšnost z pohledu zadavatele z hlediska ekonomických zisků je spíše skeptická, protože ač by rád nasadil aplikaci do oběhu, nepředpokládá rychlé navýšení zájmu o muzeum nebo město. Proto se zatím pouze předpokládá, že větší zvýraznění aplikace (například uvedení aplikace na více místech) by také mohlo zvýšit zájem o historii města, případně návštěvnost muzea a památek města, a tedy i nárůst jeho obratů.

7.2 Budoucí rozšíření

Existuje nepřeberné množství způsobů, jak by se hra dala v budoucí době rozšířit. Z mého pohledu je nejužitečnější nasazení multiplayeru, který by umožnoval uživatelům setkávat se, komunikovat a radit si ve hře. Neméně podstatné by bylo rozšíření hry o další scény, tedy místnosti a okolní prostředí. Obohatit je možné i uměleckou a zábavnou stránku o animace modelů ve scéně, a také intelektuální část o další úkoly v oblasti historie města ve stávajících ale i nově vytvořených scénách. Velkým nedostatkem by se dala nazvat chybějící optimalizace pro mobilní zařízení. Hře je také možné dodat ukládání do objektivního databázového systému a profilů pro umožnění hráči hrát odkudkoliv, i když by to zpomalilo aplikaci. V multiplatformní/databázové verzi by pak bylo možné vytvořit žebříček nejúspěšnějších hráčů.

Pro rozšíření využitelnosti aplikace je vhodné přeložit hru do více jazyků. Vícejazyčná forma hry by umožnila poznat historii města i cizincům.

7.3 Připomínky

Volba game enginu jMonkeyEngine by se dala nazvat spíše osobní nežli odbornou. jMonkeyEngine má velké možnosti, nabízí širokou škálu funkcí, grafických schopností, a protože je napsán i veden v programovacím jazyku Java, byl mi tak nejblíže. Při analýze byly mimo jiné otestovány funkce Nifty, 3D scéna i zkušební aplikace na webových stránkách, se kterou nebyl, kromě podpisu, žádný problém. Nepředpokládaly se tedy další komplikace.

Nakonec se ale ukázalo, že osobní volba nebyla nejlepší, protože nebyl prozkoumán důležitý aspekt, a to současný přístup Javy k zabezpečení (Java Security). Jednalo se o omezenou podporu Java appletů s nutnou ověřenou certifikací.

Ostatní aspekty game enginu byli plně dostačující a efektivní pro tvorbu dané aplikace. Jedná se tedy o dobrý engine, ale po prozkoumání se dá říci, že více určený pro desktopové aplikace.

Pokud by měla být scéna řádně využita na webu, bylo by třeba zakoupit ověřený certifikační podpis nebo vystavět aplikaci na jiném game enginu.

8 Závěr

Cílem diplomové práce, bylo vytvořit volně dostupnou naučnou 3D hru formou webové aplikace. Dá se říci, že tento cíl byl splněn, i přes vzniklé komplikace s certifikáty a funkcemi appletů. Nebyla vytvořena přímo appletová aplikace, jak bylo prvotním plánem, ale aplikace typu Web Start, která obsahuje odkaz na hru na novém okně. Je tedy splněn požadavek volné přístupnosti odkudkoliv bez nutnosti instalace.

Dílčím cílem práce bylo provést analýzu game engine. K velkému množství velmi dobrý, průměrných i úplně nepoužitelných enginu bylo vybráno několik z těch průměrných až výborných a u nich byl proveden rozbor. Následně byl zvolen engine jMonkeyEngine k realizaci hry. Bylo dosaženo první verze hry, která obsahuje scénu a dílčí hry k historii města Poličky.

V požadavcích na aplikaci bylo stanoveno mimo jiné intuitivní a snadná ovladatelnost, čehož bylo dosaženo základními vstupními zařízeními počítače (klávesnice a myš) a návodem se základními informacemi o fungování hry. Vytvoření 3D scény s dílčími úkoly poté splnila další z kritérií.

Samostatný dokument popisuje postup při tvorbě naučné 3D aplikace za pomoci vývojového prostředí JME SDK, game enginu jMonkeyEngine a programovacích jazyků Java a XML. Na začátku práce seznamuje se základními pojmy používaných při vývoji her a webových aplikací. Dále uvádí analýzu game engine a výběr jedné z variant. V části metodika práce je rozebrán základní princip připravované tvorby. Uvádí základní využité architektury, diagramy očekávaných funkcí a využívané technologie.

V oddílu implementace je přiblížena problematika samotného vývoje. Od úprav 3D modelu, tvorby scény a úskalími s tím spojenými, přes tvorbu herního menu pomocí knihovny NiftyGui, až k implementaci kamery, ovládání a funkcí dané hry.

9 Literatura

BLENDER Foundation. *Blender* [online]. 2017 [cit. 2017-01-03]. Dostupné z: https://www.blender.org/

BOLLINGER Gary a Bharathi NATARAJAN. *JSP Java Server Pages: podrobný průvodce začínajícího tvůrce.* Praha: Grada, 2003. Moderní programování. ISBN 80-247-0340-8.

BROOKSHEAR, J. Glenn. *Informatika.* Brno: Computer Press, 2016. ISBN 978-80-251-4266-0.

CTI Reviews. *Computer Science Illuminated* [online]. 5. Cram101 Textbook Reviews, 2016 [cit. 2017-01-16]. ISBN 978-1-4970-4010-6. Dostupné z: https://books.google.cz/books?id=sHi4DAAAQBAJ

ČÁPKA, David. ITnetwork: *3. díl - Přístupy pro práci s relačními databázemi v .NET* [online]. 2014 [cit. 2017-01-26]. Dostupné z: http://www.itnetwork.cz/csharp/databaze/c-sharp-tutorial-pristupy-pro-praci-s-relacni-databazi

ČÍZEK, Jakub. Živě: *Cloud computing: slibná budoucnost nebo marketing?* [online]. 2008 [cit. 2017-01-21]. Dostupné z: http://www.zive.cz/Clanky/Cloud-computing-slibna-budoucnost-nebo-marketing/sc-3-a-144443/default.aspx

EPIC GAME, Inc. *Unreal Engine* [online]. 2017 [cit. 2017-01-10]. Dostupné z: https://www.unrealengine.com/custom-licensing

FERSCHMANN, Petr. Zdroják: *Bezpečnost na webu – přehled útoků na webové aplikace* [online]. 2008 [cit. 2017-01-21]. Dostupné z: https://www.zdrojak.cz/clanky/prehled-utoku-na-webove-aplikace/

GARAGEGAMES Team. GarageGames: *Torque 3D* [online]. 2016 [cit. 2016-12-21]. Dostupné z: http://www.garagegames.com/

GITHUB, Inc. GitHub: *jMonkeyEngine* [online]. 2017 [cit. 2017-01-18]. Dostupné z: https://github.com/jMonkeyEngine/jmonkeyengine

GITHUB, Inc. *Choose an open source licens* [online]. [cit. 2016-12-22]. Dostupné z: https://choosealicense.com/

GREGORY, Jason. *Game Engine Architecture*. Wellesley, Mass.: A K Peters, 2009. ISBN 15-688-1413-5.

ORACLE. Java *and Google Chrome Browser: Java and Google Chrome Browser* [online]. 2015 [cit. 2016-11-25]. Dostupné z: https://java.com/en/download/faq/chrome.xml

JENEI, Attila. SliderShare: *Game Engine Architecture* [online]. 2014 [cit. 2017-01-16]. Dostupné z: http://www.slideshare.net/AttilaJenei/game-engine-architecture-34006558

JME Team. *JMonekyEngine* [online]. 2016 [cit. 2016-12-12]. Dostupné z: http://jmonkeyengine.org/

JME Team. *JMonkeyEngine: Documentation* [online]. [cit. 2016-12-23]. Dostupné z: https://docs.jmonkeyengine.org/

KOSEK, Jiří. Kosek: *Bezpečnost webových aplikací* [online]. 2014 [cit. 2017-01-21]. Dostupné z: http://www.kosek.cz/vyuka/4iz228/prednasky/bezpecnost/frames.html

KUSTERER, Ruth. *jMonkeyengine 3.0 beginner's guide* [online]. Online-Ausg. Birmingham: Packt Publishing Limited, 2013 [cit. 2017-01-19]. ISBN 978-184-9516-464.

LÁSLO, Petr a Zdeněk LEHOCKÝ. Programujte: *Ajax - Úvod* [online]. 2008 [cit. 2017-01-23]. Dostupné z: http://programujte.com/clanek/2008062101-ajax-uvod/

LAZARIS, Louis. *CSS okamžitě*. Brno: Computer Press, 2014. ISBN 978-80-251-4176-2.

LUBBERS, Peter, Brian ALBERS a Frank SALIM. *HTML5: programujeme moderní webové aplikace*. Brno: Computer Press, 2011. ISBN 978-80-251-3539-6.6.

LWJGL. *LWJGL: Lightweight Java Game Library 3* [online]. 2012 [cit. 2017-01-30]. Dostupné z: https://www.lwjgl.org/

MALÝ, Martin. Zdroják: *Softwarové licence: úvod pro obyčejné lidi* [online]. 2011 [cit. 2016-12-21]. Dostupné z: https://www.zdrojak.cz/clanky/softwarove-licence-uvod-pro-obycejne-lidi/

MANAGEMENTMANIA. *Webová aplikace* (Web Application) [online]. 2016 [cit. 2017-01-18]. Dostupné z: https://managementmania.com/cs/webova-aplikace-web-application

MĚSTO POLIČKA. *Polička: Oficiální stránky města* [online]. 2017 [cit. 2016-11-23]. Dostupné z: http://www.policka.org/

MICROSOFT. *Microsoft: Graphics and Gaming* [online]. 2017 [cit. 2017-01-30]. Dostupné z: https://msdn.microsoft.com/library/windows/desktop/ee663279.aspx

MITRA, Sharad. SliderShare: *Game Engine Overview* [online]. 2009 [cit. 2017-01-16]. Dostupné z: http://www.slideshare.net/sharadmitra/game-engine-overview?next_slideshow=1

OPENGL. *OpenGL* [online]. Khronos Group, 1997 [cit. 2017-01-30]. Dostupné z: https://www.opengl.org/

PAN, Zhigeng, Adrian D. CHEOK a Wolfgang MÜLLER. *Transactions on edutainment V*. Berlin: Springer, 2011. ISBN 978-364-2184-512.

PANDA3D development team. *Panda3D* [online]. 2016 [cit. 2016-12-17]. Dostupné z: https://www.panda3d.org/

PARIS, Clint. *Unit 66: 3D Modelling: HA7 Task 2* [online]. 2015 [cit. 2017-01-26]. Dostupné z: http://cparisunit66.blogspot.cz/2015/04/ha7-task-2.html

PÍSEK, Slavoj. *HTML: začínáme programovat.* 4. aktualiz. vyd. Praha: Grada, 2014. Průvodce (Grada). ISBN 978-80-247-5059-0.

POMAZAL, Jiří. SystemOnLine: *Hrozby pro bezpečnost webových aplikací a serverů* [online]. 2010 [cit. 2017-01-21]. Dostupné z: https://www.systemonline.cz/it-security/hrozby-pro-bezpecnost-webovych-aplikaci-a-serveru.htm

PORTLOVÁ, Michaela. *Tvorba interaktivních průchodů v programu Blender*: Brno, 2013. Bakalářská práce. Mendelova univerzita.

SKÁLA, Jakub. *Tvorba 3D her pomocí jMonkeyEngine* [online]. Praha, 2016 [cit. 2017-01-20]. Bakalářská práce. Dostupné z:

https://www.vse.cz/vskp/50615_tvorba_3d_her_pomoci_jmonkeyengine_30

TIŠNOVSKÝ, Pavel. *Root: Grafická knihovna OpenGL (1)* [online]. 2003 [cit. 2017-01-30]. Dostupné z: https://www.root.cz/clanky/graficka-knihovna-opengl-1/

TIŠNOVSKÝ, Pavel. *Root: Grafické karty a grafické akcelerátory (21)* [online]. 2005 [cit. 2017-01-30]. Dostupné z: https://www.root.cz/clanky/graficke-karty-a-graficke-akceleratory-21/

TORQUE Team. *Torque 3D: Documentation* [online]. [cit. 2016-12-21]. Dostupné z: http://docs.garagegames.com/torque-3d/official/

TUTORIALSPOINT. *Java - Applet Basics* [online]. 2017 [cit. 2017-01-23]. Dostupné z: https://www.tutorialspoint.com/java/java_applet_basics.htm

TYRYCHTR, Jan. *Provozní a analytické databáze*: *Teoretické základy*. Praha: ČSVIZ, 2015. ISBN 978-80-879-6802-4.

UNITY Technologies. *Unity Web Player* [online]. 2017c [cit. 2017-01-10]. Dostupné z: https://unity3d.com/webplayer

UNITY Technologies. *Unity: Documentation* [online]. 2017b [cit. 2017-01-10]. Dostupné z: https://docs.unity3d.com/ScriptReference/

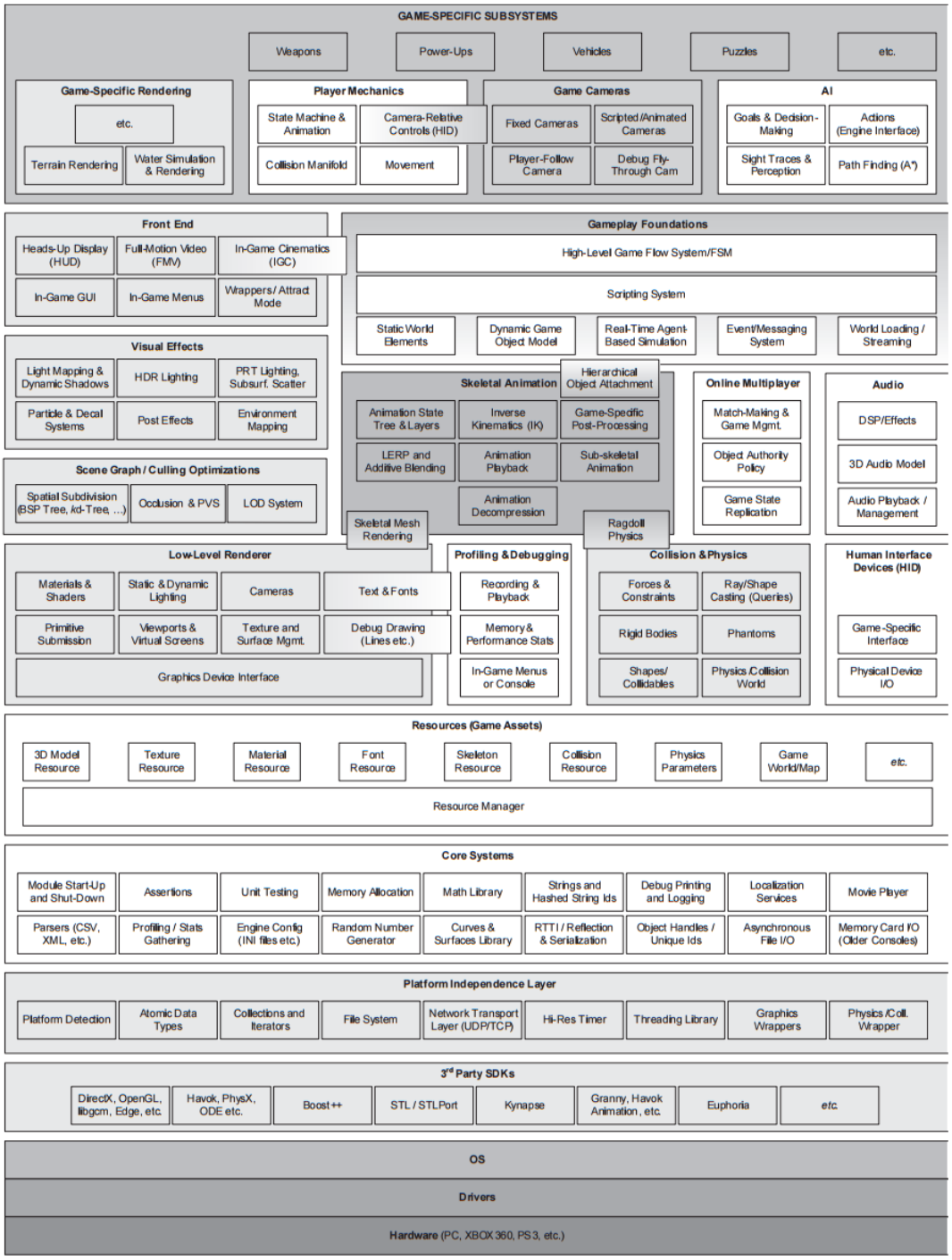
UNITY Technologies. *Unity: Store* [online]. 2017a [cit. 2017-01-10]. Dostupné z: https://store.unity.com/

ŽÁRA, Jiří, Bedřich BENEŠ, Jiří SOCHOR a Petr FELKEL. *Moderní počítačová grafika*. vyd 2. Brno: Computer Press, 2004. ISBN 80-251-0454-0.

10 Seznam obrázků

Přílohy

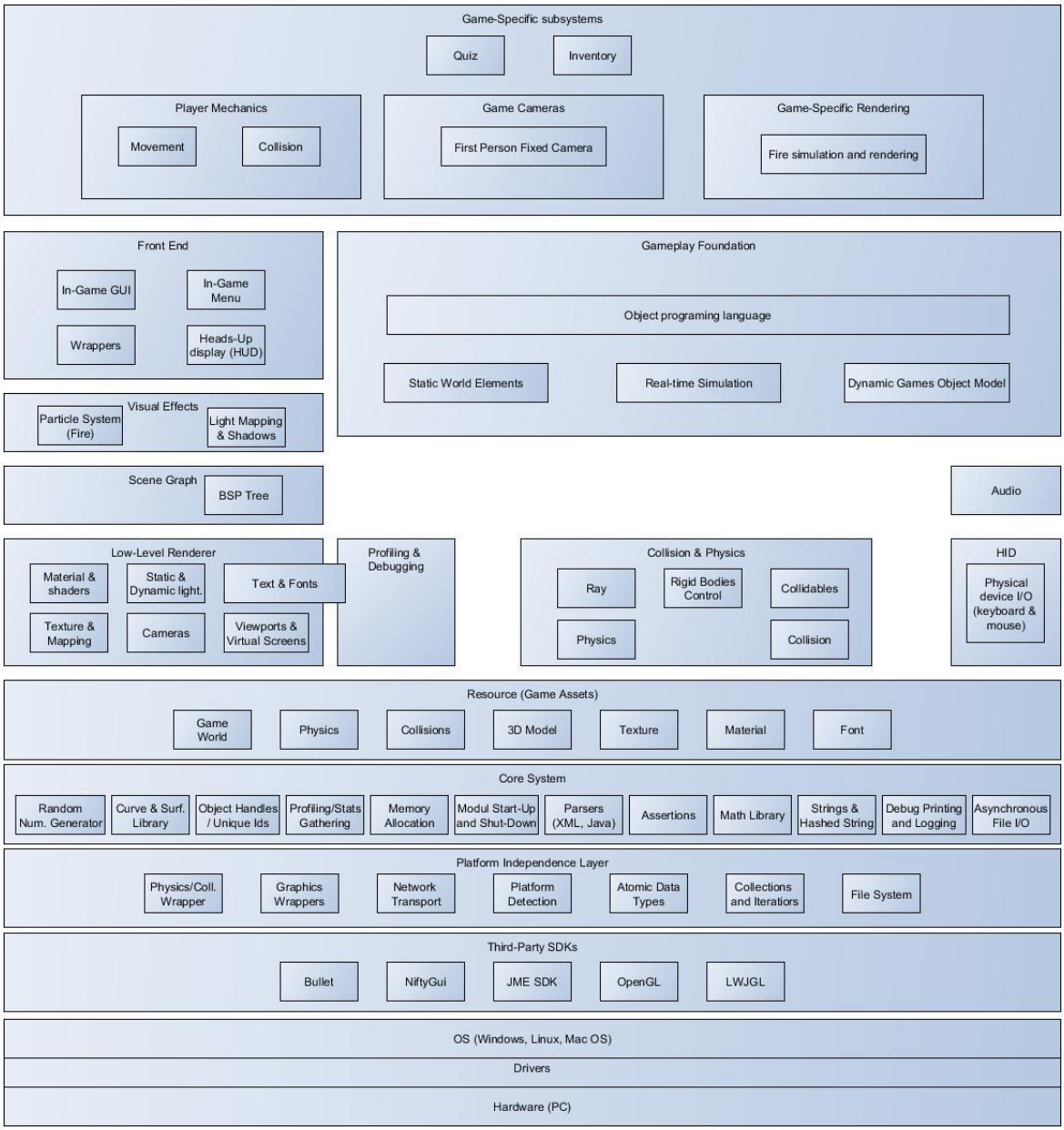
A Obecná architektura runtime game engine



Obr. 28 Runtime GEA - obecná[13]

[13] Zdroj: (Gregory, 2009)

B Použitá architektura game engine



Obr. 29 Runtime GEA – použita

C Obsah přiloženého CD

- Elektronická verze závěrečné práce (.pdf),
- Zdrojové kódy,
- Knihovny,
- 3D modely ve formátu .obj, .j3o,
- Zkompilovaná aplikace ve formátu .jnpl (web start), .exe (desktopová aplikace).