Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Systém pro sběr dat o podobnosti trojúhelníkových sítí formou webové hry

Plzeň 2018 Marek Zábran

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 25. června 2018

Marek Zábran

Abstract

Visual information is subjective and therefore difficult to evaluate on the basis of technical criteria. Many different metrics with different results concerning this topic may be found, but the assessment of their accuracy is possible only by comparing with the subjective opinions of human observers.

This thesis presents a system which collects data on human visual perception based on the similarity of objects. This data can be used to compare the sources of distortions applied to 3D triangle meshes. The created system is a browser game.

Abstrakt

Vizuální informace je subjektivní vjem a z toho důvodu je obtížné ji hodnotit na základě technických kritérií. Existují na to různé metriky, které mají různé výsledky, zhodnocení jejich správnosti je ovšem možné jen prostřednictvím porovnání se subjektivním vjemem lidských pozorovatelů.

Cílem této práce je vytvořit systém, v rámci kterého bude možné sbírat data o lidském vizuálním vjemu na základě podobnosti objektů. Tato data bude možné použít pro porovnání zdrojů zkreslení aplikovaných na trojrozměrné trojúhelníkové sítě. Vytvořený systém má podobu webové hry.

Obsah

1 Úvod

Trojúhelníková síť (triangle mesh) [16] je jednou z digitálních reprezentací třírozměrných objektů používaných v počítačové grafice, ve které, oproti obecnější mnohoúhelníkové síti (polygon mesh) [15], jsou všechny stěny v této síti tvořeny trojúhelníky. Běžné trojúhelníkové sítě jsou uloženy jako množina vrcholů daná geometrickými souřadnicemi a množina trojúhelníků, které tvoří stěnu a spojují tak vždy právě tři vrcholy – tak je tomu také například u souborů typu .OBJ [9], který byl v našem případě použit pro reprezentaci trojúhelníkové sítě. Kromě množiny bodů a množiny stěn může tento soubor obsahovat také doplňkové informace, kterými můžou být souřadnice textury, která je na objekt aplikována, a normálové vektory důležité zvláště pro potřeby stínování.

Různé trojúhelníkové sítě se můžou lišit z hlediska jak velikosti, úrovně detailu, tak i efektivnosti, jakou je dokáže různý software zpracovat, případně přítomností skryté informace (vodoznak). Pro zlepšení kvalit trojúhelníkové sítě v těchto oblastech je potřeba ji příslušně zpracovat a toto zpracování je pak až na nejjednodušší možné úpravy ztrátové. Z tohoto důvodu je důležité použít ke zpracování trojúhelníkové sítě takový algoritmus, při jehož použití se z vizuálního hlediska trojúhelníková síť změní co možná nejméně.

Tyto algoritmy ovšem ovlivňují každou trojúhelníkovou síť odlišným způsobem a tedy je jími každá trojúhelníková síť odlišným způsobem a do odlišné míry poškozena. Tato míra poškození je velmi obtížná na změření, neboť se jedná převážně o vizuální informaci. Nejčastěji se k tomu používají metriky založené na strukturální podobnosti (SSIM - Structural similarity [11]) dvou objektů (meshů), případně střední kvadratická odchylka (MSE - Mean squared error) nebo poměr vrcholového signálu k šumu (PSNR - Peak signal-to-noise ratio [10]).

I když jsou ovšem vizuální informace pro stroj jen obtížně interpretovatelné, živé organismy, tedy i lidé, je dokáží interpretovat velmi jednoduše. Tento fakt byl využit v rámci projektu MeshTest [28], pro testování nové metriky odchylka úhlu mezi dvojicí stěn sítě (DAME - Dihedral angle mesh error [29]). V rámci tohoto projektu množina dobrovolníků volila vždy z dvojice trojúhelníkových sítí takovou, která více připomínala originál. Množina těchto osob byla ovšem poměrně nízká a vzhledem k povaze činnosti pro ně také nebyl projekt příliš zábavný, což mohlo ovlivnit jejich efektivitu a hlavně pak přesnost.

Účelem tohoto projektu je vytvořit webovou aplikaci na motivy zmíně-

ného MeshTestu, která by stejným nebo obdobným způsobem získávala data o podobnosti trojúhelníkových sítí při zachování stejného formátu sbíraných dat za minimalizace veškerých systémových chyb, které by mohly ovlivnit data. Vzhledem k povaze úkolu byla aplikace zvolena jako webová hra, která by měla šanci lépe nalákat případné zájemce a více je motivovat ke splnění úkolu. Předpokládá se, že uživatele, jakožto hráče, by neměl vědecký výzkum příliš zajímat, což by tak mělo vést ke snížení případné závislosti výsledků na záměru dobrovolníků a jiných vnějších vlivů.

Aby bylo možné správně pochopit účel této práce, je potřeba nejprve pochopit problematiku trojúhelníkových sítí, neboť právě zkoumání vlastností trojúhelníkových sítí - zvláště to, které zdroje zkreslení nejméně vizuálně ovlivňují které trojúhelníkové sítě - je předmětem této práce. Problematika trojúhelníkových sítí je rozebrána v kapitole Trojúhelníkové sítě, kde se nejprve obecně vysvětluje jejich obecná podoba a jejich nejjednodušší reprezentace v rámci objektů formátu .OBJ [9], který byl použit pro reprezentaci v rámci této práce. Následně kapitola stručně vysvětluje problematiku zdrojů zkreslení a metrik používaných k jejich hodnocení z hlediska podobnosti trojúhelníkových sítí vzhledem k originálu, které by díky této práci mělo být možné porovnat s daty od lidských pozorovatelů. A také vysvětluje způsob, jakým funguje MeshTest, ze kterého tato práce vychází nejvíce a do značné míry jej imituje.

Kapitola Webové aplikace vysvětluje obecně problematiku vývoje webových aplikací, kterou je také aplikace, která je předmětem této práce. Při vývoji webových aplikací se dnes běžně používá MVC architektura z důvodů velké modulárnosti a logického rozdělení, proto byla užita i v našem případě a je v této kapitole vysvětlena. Dále se kapitola zabývá problematiku útoků na webu a hlavně možností ochrany před těmito útoky, které byly v rámci aplikace použity. Rovněž jsou popsány technologie potřebné pro vývoj aplikací používající trojúhelníkové sítě na webu OpenGL a Three.js, které byly k tomuto účelu použity. Na konci kapitoly je vysvětleno, co to je webová hra, kterou je také vypracovaná aplikace.

O problematice designu videoher, zvláště pak o herních žánrech a o herních odměnách, které motivují hráče hrát hry, informuje kapitola Herní design. Herní design byl použit při vypracování herního návrhu hry a výrazně napomohl při vymýšlení motivačních herních mechanik a pro pochopení rozdílů mezi hrou a pracovní aplikací jako byl MeshTest.

Tento rozdíl je následně vysvětlen v kapitole Herní návrh, která popisuje základ herního návrhu hry představované v tomto článku, včetně herních mechanik, herního světa a ochranných mechanik. Z důvodu špatného časového managementu se nepodařilo včas implementovat zcela všechny základní

mechaniky a je na ně proto v textu upozorněno. Další rozšiřující mechaniky včetně původního návrhu je možné najít v souboru *Design dokument.odt*.

Kapitola Realizace popisuje, jakým způsobem byl tento herní návrh realizován jako webová aplikace po stránce webové, stránce databáze a zprovoznění zobrazení trojúhelníkových sítí.

2 Trojúhelníkové sítě

Trojúhelníková síť (triangle mesh) [16] je jednou z digitálních reprezentací třírozměrných objektů používaných v počítačové grafice, ve které, oproti obecnější mnohoúhelníkové síti (polygon mesh) [2], jsou všechny stěny v této síti tvořeny trojúhelníky. Běžné trojúhelníkové sítě jsou uloženy jako množina vrcholových bodů daná geometrickými souřadnicemi a množina trojúhelníků, které tvoří stěnu a spojují tak vždy právě tři vrcholy – tak je tomu také například u souborů typu .OBJ [9], který byl v našem případě použit pro reprezentaci trojúhelníkové sítě. Kromě množiny vrcholů a množiny stěn může tento soubor obsahovat také doplňkové informace, kterými můžou být souřadnice textury, která je na objekt aplikována, a normálové vektory důležité zvláště pro potřeby stínování.

2.1 Soubory OBJ

Formát OBJ [9] je jedním z nejčastěji používaných formátů, ve kterých jsou uloženy mnohoúhelníkové sítě.

Nejjednodušší verze se skládá pouze z množiny vrcholů, které jsou dány svými souřadnicemi v trojrozměrném prostoru (na Obrázku 2.1 je každý vrchol označen na začátku písmenem v a následně trojicí těchto souřadnic) a množinou stěn, které jsou tvořeny indexy vrcholů, které je tvoří (na Obrázku 2.1 je každá stěna označena na začátku písmenem f a následně trojicí těchto indexů vrcholů).

Jak vypadá jednoduchá mnohoúhelníková síť můžeme vidět na Obrázku 2.2. Kromě množiny vrcholů a množiny stěn může soubor OBJ obsahovat také další dodatečné informace, kterými můžou být:

- souřadnice textury – ty se udávají ve tvaru [vt u v], kde u a v odpovídají procentuálním souřadnicím X, Y bodu na textuře (tedy nabývají hodnot 0 až 1).

- normály vrcholů – ty se udávají ve tvaru [vn x, y, z], kde x, y a z odpovídá X, Y, Z souřadnicím vektorů normály a dle [9] tyto nemusí být normované.

V případně přítomnosti nějaké z těchto dodatečných informací se zápis množiny stěn příslušně upraví tak, aby u každého indexu vrcholu obsahoval

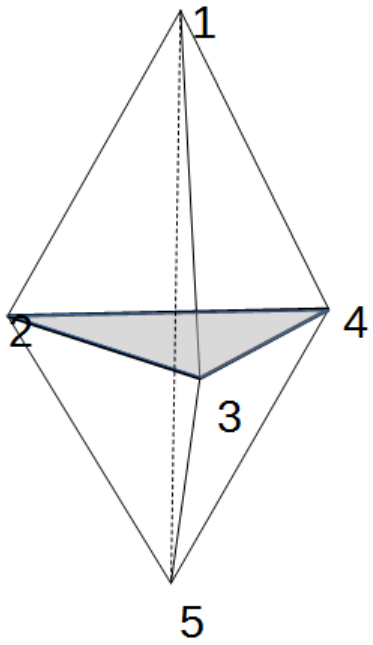
```
# object DoubleTetrahedron
#

v  0.000000000000 0.000000000000 1.000000000000
v  -1.000000000000 0.000000000000 0.000000000000
v  0.000000000000 1.000000000000 0.000000000000
v  1.000000000000 0.000000000000 0.000000000000
v  0.000000000000 0.000000000000 -1.000000000000
# 5 vertices

g GeoSphere001
f 1 2 3
f 1 3 4
f 1 4 2
f 2 3 5
f 3 4 5
f 2 5 4

# 6 faces
```

Obrázek 2.1: Jedna z nejjednodušších možných manifoldních trojúhelníkových sítí ve formátu OBJ.



Obrázek 2.2: Grafické znázornění trojúhelníkové sítě z Obrázku 2.1.

také příslušný index souřadnice textury nebo index dané normály. Množina stěn tak nabývá těchto podob:

- "[f $v1/t1$ $v2/t2$ $v3/t3$] – pro obsažení souřadnic textury, kde vi je index vrcholu a ti index souřadnice textury.
- "[f $v1//n1$ $v2//n2$ $v3//$n3] – pro obsažení normál vrcholů, kde vi je index vrcholu a ni je index normály.
- "[f $v1/t1/n1$ $v2/t2/n2$ $v3/t3/n3$] – pro obsažení souřadnic vrcholů i jejich normál, kde vi je index vrcholu, ti je index souřadnice textury a ni je index normály.

2.2 Zdroje zkreslení

Různé trojúhelníkové sítě se můžou lišit z hlediska jak velikosti, úrovně detailu, tak i efektivnosti, jakou je dokáže různý software zpracovat, případně přítomností skryté informace (vodoznak). Pro zlepšení kvalit trojúhelníkové sítě v těchto oblastech je potřeba ji příslušně zpracovat a toto zpracování je pak až na nejjednodušší možné úpravy ztrátové. Z tohoto důvodu je důležité použít ke zpracování trojúhelníkové sítě takový algoritmus, při jehož použití se z vizuálního hlediska trojúhelníková síť změní co možná nejméně.

Nejčastějšími typy zdrojů zkreslení jsou například:

- Vodoznak
- Komprimace
- Zjednodušení
- Šum
- Vyhlazení

Kromě zjednodušení by měla všechna z nich zachovávat konektivitu tělesa a ideálně by měly změnit těleso po vizuální stránce jen minimálně. Nejmenší změnu na trojúhelníkové síti lze zajistit citlivou změnou souřadnic vrcholů z množiny vrcholů. K měření odlišnosti takto upravené trojúhelníkové sítě od trojúhelníkové sítě původní se používají různé metriky, viz níže.

2.3 Metriky pro měření zkreslení

Zdroje zkreslení ovlivňují každou trojúhelníkovou síť odlišným způsobem a tedy je jimi každá trojúhelníková síť odlišným způsobem a do odlišné míry poškozena. Tato míra poškození je velmi obtížná na změření, neboť se jedná převážně o vizuální informaci. Nejčastěji se k tomu používají metriky založené na strukturální podobnosti (SSIM - Structural similarity [11]) dvou objektů (meshů). Tyto metriky můžou buď porovnávat přímo trojúhelníkové sítě, nebo jen jejich vyrenderované snímky [23].

2.3.1 MSE – Mean Squared Error

Mean Squared Error (střední kvadratická odchylka) je metrika obecně používaná v pravděpodobnosti a statistice – je druhým centrálním momentem náhodné veličiny. Při použití v počítačové grafice pro porovnávání snímků nebo trojúhelníkových sítí je porovnávaný rozdíl absolutní, tedy bez jakéhokoliv ohledu na vizuální informaci, bere se v úvahu jen rozdíl pixelů mezi snímky, případně pozice vrcholů mezi sítěmi. Dle [22] není ovšem tato metrika vhodná pro porovnávání vizuální informace.

2.3.2 PSNR – Peak signal-to-noise ratio

Peak signal-to-noise ratio [10] (poměr vrcholového signálu k šumu) je metrika používaná pro určení míry zkreslení signálu vzhledem k poměru maxima signálu k šumu. Tato metrika je podílem kvadrátu maximální hodnoty (v případě snímku maximálnímu jasu) a MSE. Vzhledem k výrazně rozdílným hodnotám, které může tento poměr nabývat, se u tohoto poměru používá logaritmické měřítko a hodnoty PSNR se tak běžně udávají v decibelech. Používá se zpravidla pro porovnávání kvality v rámci stejného typu kodeku, pro porovnávání čehokoliv jiného není vhodný [19]. Dle [20] PSNR není vhodnou metrikou pro měření kvality obrazu, neboť oproti ostatním metrikám výrazně zaostává.

2.3.3 SSIM – Structural similarity

Structural similarity [11] (strukturální podobnost) je metrika používaná pouze pro porovnávání digitálních snímků, ať už snímků videa, obrázků a nebo renderované scény. Porovnává přitom nový obrázek, který je nějakým způsobem změněný, s původním nezměněným obrázkem. Vychází z metod PSNR a MSE, která ale na rozdíl od SSIM využívají absolutní chybu, zatím co SSIM je založena na způsobu, jakým je vnímán rozdíl ve struktuře

objektu. Myšlenkově vychází z předpokladu, že blízké a hlavně pak sousední pixely mají mezi sebou podstatně větší závislost než pixely vzdálené. Také bere v úvahu, že rozdíl mezi snímky je výraznější ve světlých částech než v tmavých.

2.3.4 MSDM – Mesh Structural distortion Measure

Mesh Structural distortion Measure [25] (míra strukturálního zkreslení sítě) je metrika vycházející z SSIM, která je upravená tak, aby mohla být použita pro třírozměrné mnohoúhelníkové sítě. Zaměřuje se na lokální křivost v okolí vrcholů a dosahuje poměrně vysoké korelace se subjektivním vizuálním vjemem [29].

2.3.5 MSDM2 – Mesh Structural distortion Measure 2

Mesh Structural distortion Measure 2 [24] (míra strukturálního zkreslení sítě 2) je vylepšená verze MSDM, která na rozdíl od MSDM používá více vrstev, které se liší velikostí okolí vrcholů, pro které se počítá velikost lokální křivosti. Tyto vrstvy jsou následně normalizovány a složeny do jedné za použití Minkowskiho prostorového sdružování (Minkowski spatial pooling). Díky tomu má MSDM2 podstatně blíže k metrice lidského vjemu, ovšem je také podstatně pomalejší.

2.3.6 DAME – Dihedral angle mesh error

Dihedral angle mesh error [29] (odchylka úhlu mezi dvojicí stěn sítě) je metrika založená na porovnávání orientovaných úhlů mezi dvojicemi stěn dvou trojúhelníkových sítí. V porovnání s MSDM a zvláště pak MSDM2 je tato metrika podstatně rychlejší a dosahuje výrazně větší podobnosti vzhledem k lidskému vjemu i v porovnání s MSDM2.

2.4 MeshTest

Program MeshTest [28] napsal roku 2010 doc. Ing. Libor Váša, Ph.D. v C# ve v té době velmi populárním frameworku XNA [14]. Tento framework bohužel již není podporován, což zesložiťuje snahu o replikaci MeshTestu v původním stavu. MeshTest samotný je velmi jednoduchý program, který umožňuje zobrazení mnohoúhelníkové sítě ve vizuálně příjemné podobě s použitím dvou opačně orientovaných směrových světel a dává uživateli možnost

tyto modely posouvat, rotovat a škálovat (zvětšovat nebo zmenšovat). Uživatel vždy vidí tři modely (mnohoúhelníkové sítě), přičemž jsou vždy všechny stejně natočené, posunuté a stejně velké. Prvním z těchto modelů je originální nezměněný model, zbylé dva jsou úmyslně poškozené verze originálního modelu jedním z běžných algorithmů popsaných v [29]. Úkolem uživatele je vybrat z těchto dvou modelů ten, který je více podobný originálu. Takto uživatel v rámci MeshTestu provádí čtyřicet různých výběrů a následně se od uživatelů sesbíraná data automaticky odesílala mailem vývojáři.

Pečlivost uživatelů při plnění MeshTestu je sporná, dle [29] trvalo některým uživatelům vyplnění MeshTestu pouze čtyři minuty, v průměru pak patnáct minut. Uživatelé byli předem srozuměni, že plněním MeshTestu se podílí na vědeckém výzkumu a z toho také plynula jejich motivace. Velký počet výběrů za sebou měl navíc pravděpodobně za následek postupný pokles pozornosti. Je zjevné, že pokud by se podařilo MeshTest předělat tak, aby jej uživatelé mohli plnit opakovaně, ovšem v menším množství a byli nějak motivování snažit se dosáhnout co nejpřesnějších výběrů, byly by výsledky takové aplikace podstatně přesnější.

Takové předělání znamená využít videoherního designu a vytvořit na základě MeshTestu plnohodnotnou webovou hru.

3 Webové aplikace

Webové aplikace se ze základu rozdělují na dvě části – frontend, prezenční část, která obsahuje vše, co uživatel může vidět, a backend, kam patří všechno ostatní. Podstatně důležitější je ovšem rozdělení podle MVC.

3.1 MVC

MVC architektura [8], celým názvem Model-View-Controller (Model-pohled-řadič) je návrhový vzor používaný hlavně u webových aplikací, jehož hlavní úlohou je oddělení kódu aplikace do třech různých částí, kde každá má oddělenou funkci, což zlepšuje modulovatelnost, zajišťuje jednodušší kooperaci více vývojářů a větší znovupoužitelnost jednotlivých modulů (částí aplikace). Jedná se konkrétně o dělení na tyto základní části:

3.1.1 View (pohled)

Toto je část kódu, kterou uživatel může přímo vidět a přímo s ním komunikuje, tedy prezentační grafická část, která typicky nijak nepracuje s nestatickými daty. Jedná se zvláště o kód HTML s CSS, který je případně doprovázen Javascriptem. V pohledu by se neměl nacházet kód žádných vnitřních výpočtů, které přímo nesouvisí s grafickou stránkou aplikace a nesmí se v něm nacházet žádná komunikace s databází. View typicky obsahuje malou část kódu v PHP, který volá obsluhu řadiče. Pokud potřebuje informace z databáze, požádá o ně řadič a nikdy by neměl přímo žádat model.

3.1.2 Controller (řadič)

Tuto část kódu už uživatel typicky nevidí. Probíhá zde část výpočtů a aplikace zde hlavně rozhoduje o činnosti aplikace. Typicky se jedná o kód v PHP nebo nějakém jiném programovacím jazyce. V řadiči by se neměl nacházet HTML kód stránky a nemělo by v něm docházet k přímému kontaktu s databází. Řadič typicky jedná na základě žádosti od uživatele (povětšinou získané z pohledu). Pokud potřebuje informace z databáze, požádá o ně model. Odpověď na žádost buď vrací přímo uživateli (pohledu) nebo se vrací přes model.

3.1.3 Model (model)

Tuto část kódu uživatel nesmí vidět. Dochází zde ke komunikaci s databází provozovanou přes libovolný programovací jazyk. Typicky k této komunikaci dochází na základě podnětu řadiče a model informace buď z databáze rovnou předává pohledu (například přes session), nebo jej předá nejprve řadiči pro další zpracování. Může zde docházet k výpočtům, které přímo souvisejí s daty v databázi. Model a řadič společně zajišťují, že data v databázi nebudou nijak poškozena na základě požadavku uživatele.

3.2 Útoky na webu

Každý systém může být určitým způsobem napaden. Situace v případě webových aplikací je o to horší, že potřebuje zajistit propojení několika oddělených vnitřních prvků (HTML, PHP, Javascript, databáze) a navíc komunikovat s klientem, což zvyšuje možnost vzniku neohlídaných bezpečnostních chyb. Nejznámější možné útoky na webové aplikace jsou následující [26]:

3.2.1 SQL injection

Útok spočívající v tom, že je uživatelem dodaná informace (typicky metodou POST) v nečekaném tvaru, který ovšem nezpůsobí chybu. Není tak nalezen a může vést k nečekanému chování – typicky k tomu, že uživatel dostane od serveru data, ke kterým by neměl mít přístup. Například pokud má obsluha databáze tvar: *[“select * from hrac where id = “.*$id]*, tak databáze za běžných okolností vydá informace entity hrac s id $*id*. Jenže pokud je *$id = “0 or 1=1“*, pak by v tomto případě databáze vydala informace o zcela všech entitách hrac. Řešení tohoto problému jsou následující:

- Testování uživatelem dodané informace – například pokud by se mělo jednat pouze o číslo, nesmí proměnná obsahovat řetězec znaků. Lze řešit velmi jednoduše (přes *“$id+=0;“* se z $*id* stane 0, pokud je $*id* řetězec) nebo i složitě přes regulární výrazy.

- Minimalizace možných informací zaslaných od uživatele – například uživatel vybírá z předem připravené nabídky a přes POST pouze vybírá jeden z prvků. Toto ovšem nelze použít vždy.

- Použití ochranných parametrů – například *[“select * from hrac where id = :id“]* a dodání *[array(’:id’ => $id)]*. Toto je nejúčinější řešení SQL injection.

3.2.2 Načtení nepovolených souborů

Útok spočívá v použití GET k získání souborů, ke kterým by uživatel neměl mít přístup. Typicky to lze řešit kontrolou práv uživatele a kontrola parametrů GET.

3.2.3 Krádež session

Session soubory jsou uloženy na serveru a uživatelé k nim sami nemají přístup. Do session souborů server ukládá dočasné informace o uživateli a zpravidla se na ně vážou i uživatelská práva a identifikátor uživatele. Aby bylo jasné, komu patří jaká session, generuje server každému uživateli unikátní identifikátor, kterým se serveru hlásí. Pokud se uživatel hlásí s cizím identifikátorem, server jej také považuje za jiného uživatele s jinou session, což může tento útočník použít k získání informací, ke kterým nemá přístup. Řešení jsou následující:

- Krátká platnost session – jen snižuje šanci krádeže.
- Nepředvídatelný identifikátor – zabraňuje jen úmyslné krádeži, stále je možná náhodná krádež nebo krádež získáním cookie.
- Kontrola IP adresy uživatele – lze obejít, pokud se útočník přihlašuje ze stejného (veřejného) počítače. Výrazně ovšem pomáhá.

3.2.4 Krádež cookies

Cookies jsou dočasné soubory uložené na straně uživatele. Uživatel k nim má přístup, ale typicky o nich vůbec neví. Jedná se o obdobu session na straně uživatele a může nabývat stejné informace. Nebezpečné jsou zvláště cookies obsahující session identifikátory. Pokud cookies nejsou na veřejných počítačích mazány, může lehce dojít ke zneužití uložených informací. Pro bezpečnost jak serveru, tak uživatele je vhodné použití cookies minimalizovat, zvláště pak u důležitých a lehce zneužitelných informací.

3.2.5 XSS

Cross Site Scripting je způsob napadnutí uživatele s využitím škodlivého kódu umístěného v URL, které se tváří jako URL vašeho webu. Útočník takto úpravou URL poskytuje ostatním uživatelům, ze kterých se tak stávají potencionální oběti. Tento kód zpravidla načte kód v Javascriptu z jiného webu a aktivuje jej bez znalosti uživatele. Tomuto problému lze zamezit,

pokud se na každé stránce zakáže Javascript, nebo se předem škodlivý vstup identifikuje a dojde k bezpečnostnímu přesměrování URL. Zvláštní typ XSS je takový, při kterém se škodlivý kód uloží přímo do databáze a odtud se trvale načítá (například jako příspěvek v internetové diskuzi). Tomu je třeba zabránit ošetřením vstupu do databáze podobně jako u SQL injection.

3.2.6 XSRF

Cross-site request forgery je praktika podobná XSS, kdy uživatel omylem použije příkaz GET (nebo i POST) na váš web, ovšem aniž by si toho byl vědom. Podobně jako u XSS mu URL adresu poskytne útočník a za předpokladu, že má oběť na webu platnou session, příkaz se vykoná včetně parametru (tzn. včetně přijmutí vstupu od uživatele). Například v případě banky by mohl útočník v rámci parametru poslat přes oběť žádost o přesun peněžní částky na jiný (útočníkův) účet. Jediným efektivním řešením tohoto útoku je u každé důležité operace požádat o náhodné potvrzení (například opsání náhodně vygenerovaného kódu).

3.3 WebGL

WebGL (Web Graphics Library) [13] je javascriptové rozhraní pro vykreslování počítačové grafiky způsobem vycházejícím z OpenGL. Nepotřebuje k tomu ze základu žádné přídavné pluginy, ani knihovny, používá pouze HTML elementy a není třeba jej nijak instalovat, neboť je přímo podporováno většinou běžných webových prohlížečů:

WebGL ve verzi 2.0 podporují Chrome 56+, Firefox 51+, Opera 43+.

WebGL ve verzi 1.0 plně podporují Chrome 9+, Firefox 4+, Safari 6+, Opera 12+, Edge 10240+.

V případě mobilních zařízení verzi 1.0 podporují BlackBerry 10+, Firefox for mobile 4+, Chrome 25+, Internet Explorer 11+ na Windows Phone 8.11+, Edge na Windows 10 Mobile, Opera Mobile 12 na Androidu, Safari 6+ na iOS 8.

WebGL 1.0 částečně podporuje Internet Explorer 11+ na desktopu a na mobilních zařízeních Windows Phone 8.11+.

WebGL není podporováno v Android Browseru.

V prvních verzích prohlížečů, které WebGL podporují, často z převážně bezpečnostních důvodů platí, že je potřeba podporu WebGL nejprve manuálně aktivovat a až pozdější verze mají podporu WebGL aktivovanou v základním nastavení.

WebGL ze základu, podobně jako základní OpenGL, nabízí jen poměrně základní funkce a rozhodně samo o sobě nepodporuje načtení a zobrazení mnohoúhelníkových sítí. Pro tento účel je potřeba dodat kód obsluhy v Javascriptu. Napsat takový kód je časově velmi náročná práce, není to ovšem nezbytně nutné, protože již existuje velké množství knihoven, které podporu těchto nejčastěji potřebných funkcí nabízejí. Jednou z takových knihoven je knihovna Three.js.

3.4 Three.js

Three.js [12] [3] je javascriptová knihovna pro zobrazování 3D grafiky používající WebGL. Vzhledem k tomu, že funkce WebGL jsou podporovány přímo prohlížečem a Three.js je jinak tvořeno pouze javascriptovými soubory, nepotřebuje Three.js pro svou funkčnost na straně uživatele žádné doplňky. Celý zdrojový kód Three.js je volně dostupný na GitHubu. Three.js podporuje pokročilejší prvky počítačové grafiky jako jsou scéna, kamera, světla, mnohoúhelníkové sítě a jejich načítání, textury, antialiasing a mnohé další. Načítání mnohoúhelníkových sítí pak provádí zejména ze souborů formátu OBJ.

3.5 Webové (prohlížečové) hry

V polovině nultých let 21. století došlo k nárůstu popularity takzvaně webových her (browser game), které bylo možné hrát (a velkou část z nich stále jde hrát) pouze ve webovém prohlížeči bez potřeby přídavných doplňků. Typickým příkladem takové hry jsou například Divoké kmeny, Travian, Gladiatus, The West nebo Ikariam. Většina z nich funguje na velmi jednoduchém základě, využívá pouze HTML a PHP s databází a není proto nijak zvlášť akční. Aby se zajistilo, že aktivnější hráči nezískají výhodu nad těmi pasivnějšími, omezují tyto hry hráče tahy nebo akčními body, které se přidávají/aktivují systematicky v závislosti na reálném čase, ovšem hráč je může do určitého maxima použít naráz. Například tak může hráč vykonat nějakou činnost jednou za den, nebo může získat každých šest hodin deset akčních bodů do maxima dvaceti bodů, které může kdykoliv použít.

Typické pro tyto hry je jednoduchá grafika zajištěná CSS a rastrovými obrázky, RPG prvky a multiplayerovost s důrazem na kompetetivnost, případně kooperativnost s omezenou skupinou hráčů. Největším nepřítelem zpravidla není herní systém, ale ostatní hráči, a cílem hráče je být ten nejlepší ze všech hráčů.

4 Videoherní design

Videoherní průmysl se neustále vyvíjí, zvláště pak v posledních letech se vzestupem nezávislé (*Indie*) produkce a fenomény *Kickstarter* a *Early Access*, existuje však několik zavedených žánrů, které mají své vlastnosti a pravidla, zároveň existuje mnoho jejich kříženců. První věcí, kterou si herní vývojář musí při tvorbě videohry rozmyslet, je, o jaký se bude jednat žánr, či prvky jakých žánrů bude obsahovat. Stejně jako u filmu nebo u knih zaujme každý žánr jiný typ člověka. Obecně základním rozdílem videohry od filmu nebo knihy je překážka – knihu stačí přečíst, film stačí pustit a sledovat, ale u hry musí hráč činit rozhodnutí, kterými vyřeší herní problémy. Takovým rozhodnutím u videohry se rozumí např: Jít doprava, nebo doleva. Vyslat jednotky na cestu, nebo ještě počkat na posily. Koupit si luk, nebo šetřit peníze na lepší meč. Tato rozhodnutí mohou být jednoduchá jako rozhodnutí zmíněná výše, můžou být velmi složitá, pokud je správným rozhodnutím jen některé nebo některá jejich kombinace (logické hry, strategie) a v některých případech mohou být i kontroverzní a s morálním základem. Po těchto rozhodnutích přichází výsledek, buď špatný (Game Over), který přímo či nepřímo nutí hráče volbu zopakovat a rozhodnutí zlepšit, nebo dobrý, který hráči přináší pozitivní pocit úspěchu, který je doprovázen nějakou herní odměnou.

4.1 Herní odměny

Určité herní odměny jsou vyhledávány určitým typem hráče a jsou spjaty s určitým typem hry. O fungování těchto odměn hovořil například v rámci [5] Tom Chatfield, autor knihy [6], která se také dotýká tohoto tématu. O těchto odměnách jako o předmětu psychologického výzkumu dále zde [30] nebo zde [21]. Odměny můžeme rozdělit mezi tyto typy:

4.1.1 Objevování

Touha poznat, co je za horizontem, jaké to je použít novou schopnost, jestli se ve hře lze opít. Odměnou je právě možnost naplnit tuto hráčovu zvědavost. Objevování bylo původně devízou RPG a jiných příběhových her, dnes je hlavním nosným prvkem sandboxových her, jejichž hlavní výhodou oproti ostatním je, že se tato odměna vlivem náhodného generování jen velmi pomalu opotřebovává ve srovnání s prací k tomu využitou.

4.1.2 Zlepšení herní

Z počátku hry je hráčská postava slabá a porazit ji dokáže i obyčejná krysa. Hráčovou touhou je to změnit, naučit postavu lépe máchat mečem, ten meč naostřit, či rovnou koupit nový. Odměnou je právě pocit úspěchu, zlepšení, kterého hráč se svou postavou dosáhne, zvláště když se o něm může osobně přesvědčit tím, že kdysi silného protivníka porazí jedinou ranou. Tento typ odměny je nosným prvkem RPG, dnes se dokonce považuje za synonymum pro RPG prvky.

4.1.3 Zlepšení hráče

Zlepšovat se nemusí pouze hráčova postava, ale též hráč jako takový. I to s sebou nese onen potřebný pocit naplnění a o to více, že hráč neuspěl jen díky nadpřirozené síle své postavy, nýbrž proto, že dokola opakoval tu samou věc, než vlastními silami dokázal dosáhnout zlepšení. Nosný prvek všech kompetetivních a těžkých her. Důležitá je u hráče zpravidla jak rychlost, tak bystrost. Ze všech současných žánrů je tento typ odměny nejvíce zastoupen u žánrů kompetetivních online her jako jsou hry typu MOBA (Multiplayer Online Battle Arena) nebo multiplayerové střílečky.

4.1.4 Příběh

Tato odměna částečně souvisí s touhou objevovat, je ale velmi specifická. Známý videoherní vývojář John Carmack v minulém století dost nešetrně řekl [7]: *„Story in a game is like a story in a porn movie. It's expected to be there, but it's not that important.“* Nutno říct, že v žádné z tehdejších střílečeček, které tvořil, nebyl příběh, kterého by si většina hráčů vůbec všimla. V tomto století se však tento trend obrátil a prakticky žádná velká hra dnes nevyjde bez poutavého příběhu. Přesto se ve hrách příběh zpravidla podrobuje hratelnosti a dosahuje hodnoty spíš béčkových filmů. Příběh je nosnou částí všech adventur a jejich odrůd (interaktivní filmy, vizuální novely) a často také akčních her a RPG.

4.1.5 Humor

Levnou alternativou k poutavému příběhu je zábava spojená s herními vtipy, stylizovanou bombastičností, Easter eggy, či jinou formou, která slouží k čistému pobavení. Tato odměna se považuje obecně za pokleslou a u her s velkým rozpočtem se objevují vyloženě raritně. Problematická je paradoxně

náročnost a subjektivnost vtipu na rozdíl od příběhu, který, je-li jednoduchý, působí na většinu lidí velmi podobně, přičemž vtip i pochopený nemusí pobavit. Vytvořit dobrý vtip tak bývá paradoxně obtížnější, než vytvořit použitelný příběh. Přesto se humor ještě dnes objevuje u mnoha adventur a akcí a paradoxně též humorných simulátorů (například *Goat Simulator*).

4.1.6 Edukace

Nejedná-li se o edukativní software, pak rozhodně není edukace hlavní stimulující herní odměnou pro hráče ani jejím klíčovým prvkem. Mnoho realistických (simulátory) a historických her (pokud není historie nadměrně upravena) však přináší hráči možnost pasivně získat spoustu bližších vědomostí, o kterých by jinak neměli ani tušení, či případně je nadchne k tomu si o dané tématice zjistit víc. Mnoho hráčů navíc vypovědělo, že jim tyto netušené pasivně získané vědomosti výrazně pomohly při studiu, zvláště například anglického jazyka.

4.1.7 Socializace

Veřejností neprávem přehlížen bývá sociální faktor her, jakožto nejen možnost najít nové lidi, ale i je blíže poznat, případně svůj vztah s nimi utužit, ať už kompetetivně, či spoluprací. To pochopitelně nelze v běžném smyslu aplikovat u her, které nejsou multiplayerové, výrazné využití socializace lze tak najít jen u MMO her. Těmi jsou dnes do počtu hráčů nejhranější multiplayerové střílečky, MOBA hry a takzvané survival hry, socializace je však podstatně typičtější pro MMO RPG, která jsou mezi hráči pověstná jako místa, kde hráč může najít životního partnera s podobnými zálibami a dokonce jsou i dohledatelné případy, kdy se v prostoru hry dohrála svatba reálných lidí (dokonce i jeden případ pohřbu [27].)

4.1.8 Atmosféra

Grafické zpracování, soundtrack, stylizace, příběh a celkově herní design jsou tím, co vytváří nejsubjektivnější součást hry – atmosféru. Rozhodně to není jen audiovizuální záležitost [18]. Atmosféra je něco, co nejde vyjádřit, nejde změřit a nejde jednoduše říct, jak se to dělá dobře a jak špatně. Není to jenom soubor prvků, ale jejich ucelený žijící celek. Jednoduše, herní atmosféra je tím jediným, v čem se pozná, jestli byla hra prostě jen vytvářena, nebo do ní bylo dáno srdce. Zkušení hráči právě podle atmosféry rozlišují ty nejlepší hry, které podle nich mají duši a jsou blízko jejich srdci. Podle

atmosféry také rozlišují, které hry jsou umění, a které hry jen dojným mainstreamem. Zatímco geniální gameplay hráče pohltí, geniální herní atmosféra donutí hráče hru milovat. Protože hrát hru s dobrou atmosférou je samo o sobě odměnou. V současné době vzniká velké množství nezávislých uměleckých her, které se ve hře snaží předat něco víc, výjimečnou herní atmosféru, herní duši. Některé jsou i úspěšné, ale jen u určitého typu hráče. Geniální herní atmosféra je podle soudobých herních veteránů záležitostí starých her, zatímco ty nové musí teprve dozrát.

4.1.9 Nostalgie

Zdánlivě jen specifický poddruh atmosféry, který ale svou specifičností vybočuje. Podkreslíte-li herní scénu truchlivou hudbou a vizuální stránku stylizujete do pohřebního hávu, neboť právě zemřela jedna z klíčových postav, pak vytvoříte smutnou atmosféru a možná se vám některé hráče podaří donutit k pláči. Když to uděláte špatně, budou se hráči nudit, budou naštvaní, či budou rádi, že jejich neoblíbená postava konečně zemřela. Nostalgie je pocit specifický, neboť ji dokáže pocítit jen hráč, který hru nehraje poprvé. Vzpomíná si na své zážitky z předchozích her a ty mu pasivně zlepšují současný zážitek a nutí zažívat identické události znovu a znovu, aby si je připomněl, stejně jako čtenář, když si chce znova přečíst knihu. Hráči pak může být jedno, že vlastně nenachází nic nového, jen opakuje staré, protože hraje čistě pro pocit nostalgie, který je zpravidla vyvolán právě výjimečnou atmosférou.

4.1.10 Prokázání síly

Ne každý hráč je dobromyslný, někteří mají i temnou, škodolibou část. Ta je nutí se vyžívat v nesmyslném herním násilí, nejčastěji na jiných hráčích, kteří se však stávají herní obětí ne pro nějaké zlepšení hráče nebo herní postavy, ale nejčastěji zcela bezdůvodně. Například když hráč s velmi silnou postavou úmyslně zabíjí jiné postavy, které teprve začaly hrát. Dělá tak ne kvůli nějakým herním pravidlům, ale aby pro sebe vytvořil tvůrci hry většinou nezamýšlenou odměnu, kterou je škodolibá radost z utrpení jiného. Tato odměna je odvrácenou stranou socializace a temnou stránkou her, nejčastěji pochopitelně MMO RPG. Někdy ale nemusí být brána pouze negativně, získá-li hráč takovou odměnu za pomoci svých vlastních schopností (dnes typické pro online střílečky a MOBA), pak je rozhodně odměnou zaslouženou.

4.2 Herní žánry

Hry samotné pak bez ohledu na herní odměny můžeme rozdělit do těchto žánrů [1] nebo do množiny těchto prvků:

4.2.1 Akční hry

Hry vyžadující u hráče hlavně rychlost, tedy orientaci v prostoru, rychlé rozhodování, multitasking, všímavost, přesnost. Vyhledávané největší částí hráčstva, zvláště pak hráči, kteří hledají adrenalin. Do této skupiny patří skoro všechny největší (mainstreamové) tituly. Žánr jednoduchý z hlediska tvorby pohlcujícího gameplaye, který má ale tendenci fungovat na úkor ostatních prvků. Patří sem zejména střílečky (FPS), akční prvky ale obsahuje většina her vyjma tahových strategií a akce je považována za nutnou součást úspěšného titulu.

4.2.2 RPG

Celým jménem *Role Playing Game*, v české podobě *hra na hrdiny*. Jak už bylo zmíněno v Kapitole 4.1.2, RPG prvky se rozumí prvky rozvíjející herní alter ego. Žánr původně vzniklý podle deskové hry Dragons and Dungeons, proto neoznačuje každou hru s možností činit za virtuální postavu důležité volby, nýbrž se RPG prvky rozumí systém odměnového zlepšování herního alter ega a to typicky systémem levelování, díky kterému navíc hráč často může ovlivnit to, jakou odměnu jeho virtuální já získá. Nejčastěji tedy hráčova postava za herní úspěchy (u RPG se rozumí hlavně questy) získává zkušenostní body (*XP*), které při překročení předem určené hranice vedou ke zvýšení úrovně hráčské postavy (*level-up*), což znamená pro hráče zesílení jeho postavy a případně možnost si vybrat jaké schopnosti a vlastnosti (tedy i perky) získá nebo zlepší. Dnes je tendence přidávat byť nepatrné možnosti zlepšování herní postavy do téměř každé hry pro zaujmutí hráče hledajícího právě tento typ odměny.

4.2.3 Strategie

Strategickou hru můžeme teoreticky popsat jako RPG nebo akci zvětšenou takovým způsobem, že místo jednoho hrdiny nebo jedné skupiny hrdinů ovládáte celé společenství postav. Strategie může obsahovat RPG prvky a velmi často v ní dochází k taktickému boji mezi hráčovými jednotkami (už se nejedná o postavy, ale nejčastěji o jednotky bez jakkoliv popsaného charakteru) a nepřítelem, typickým prvkem strategie je ale stavění (jednotek,

budov pro jednotky, budov pro budovy, budov pro výzkumy...), které zpravidla značí hlavní vývoj ve hře. Hrát strategii je jako hrát šachy, hráč musí pochopit složitá herní pravidla a najít svojí vlastní cestu k vítězství. Potřebuje být bystrý, trpělivý a důmyslný, což se navíc ztěžuje, jedná-li se o hru s akčními prvky (RTS, MOBA) anebo složitým mikromanagementem. Čistou strategií se pak myslí strategie tahové a budovatelské. Výraz strategie se používá i pro hry, které jsou zaměřením na taktické úrovni. Slovo taktické se buď zaměňuje se slovem strategické, nebo používá jen velmi výjimečně.

4.2.4 Logické hry

Logické hry nebo také puzzly jsou hry lišící se od ostatních svojí krátkostí a povětšinou také vysokou obtížností (akční prvky ji ještě zvyšují), která požaduje logické myšlení stejně jako může požadovat všechny vlastnosti strategie a akce. Sestávají se zpravidla z většího množství kratších puzzlů využívajících stejné mechanismy. Své místo mají pro svou krátkost hlavně na mobilních zařízeních (na kterých je to pozitivem) a jsou oblíbené zejména u hráčů, kteří si chtějí procvičit logické myšlení.

4.2.5 Adventury

Adeventury jsou hry, které potlačují prvky výše zmíněných žánrů a zaměřují se hlavně na příběh. Občas se objevují rébusy, v současných adventurách často nalézáme akční prvky prostřednictvím *Quick Time Events* a rovněž některé adventury nabízejí složité volby, které ovlivňují příběh, či častěji vedou k neúspěšnému konci hry. Adventurními prvky se většinou myslí nenásilná interakce. Adventurou jsou dnes myšleny především staré a nezměněné *point-and-click* adventury, jejichž hlavní aktivitou je hledání interaktivních předmětů (*pixelhunting*) a následně hledání správné kombinace předmětů pro vyřešení rébusu, který nemusí mít vždy logický základ (například když musíte použít opici místo hasáku). Dále moderní akční adventury (například od *Telltale*), které jsou často hanlivě označované jako interaktivní filmy pro jejich ve hrách nenáviděnou tunelovitost, a v poslední řadě také do popředí vstupující vizuální novely, typicky japonského původu.

4.2.6 Sandboxy

Jak z názvu vyplývá, jde o hry zpravidla bez příběhu a jejichž hlavní prvkem je jejich otevřenost [17]. Ve hře prakticky vždy chybí hlavní cíl, hráč si sám musí rozhodnout, co chce dělat, přičemž by jeho největším omezením měla

být jen herní obtížnost. Sandboxová hra správně není nikdy stejná, naopak by měla zaručit, že (a z tohoto důvodu se sandboxy většinou spoléhají na náhodné generování) je pokaždé úplně jiná. Sandboxovými prvky se většinou myslí prvky herní nelinearity (otevřený svět apod.)

4.2.7 Simulátory

V počátcích videoherního průmyslu byly prakticky všechny videohry tendenčně označovány jako simulátory. Postupně se zavedly jednotlivé žánry a jako simulátory se začali označovat pouze hry z těchto žánrů vybočující, které nějak kopírují realitu, tedy přibližují člověku virtuálně nějakou praktickou zkušenost a to od létání v letadle přes práci farmáře až po život "gumové kozy". Jako simulátory (prvky simulátoru se nepoužívá) se také označují hry jiných žánrů, které jsou nezvykle realistické.

5 Herní návrh

Původní MeshTest ve svém původním stavu lze jen těžko označit za hru v pravém slova smyslu. Dle [4] by kromě jiného totiž měla mít hra následující vlastnosti:

- Zábavnost – Hráči hrají hry proto, že je baví. Ovšem vyplnit MeshTest nebylo zábavné, naopak to uživatele výrazně nudilo.
- Pravidla – Hra má pravidla, kterými se hráči musí řídit a tato pravidla nesmí být možné porušit. MeshTest měl také pravidlo – vybrat objekt, který nejblíže odpovídá originálu, ovšem toto pravidlo nešlo nijak kontrolovat, neboť šlo o subjektivní vjem.
- Neužitečnost – Účelem hry není dosáhnout ničeho užitečného. Oproti tomu MeshTest sloužil k vědeckým účelům a uživatelé MeshTestu jej právě proto vyplňovali.
- Fiktivnost – Hra není součástí reality, vytváří se kolem ní imaginární svět s vnitřními pravidly. MeshTest nepředstavoval žádný od reality oddělený svět.
- Nejistota – Hráč předem neví, jak hra dopadne. V případě MeshTestu uživatel vykonává činnost – výběr ze dvou sítí tu, kterou považoval za podobnější originálu – ovšem výsledek této činnosti se nijak nelišil dle toho, jak vybíral.

MeshTest tedy nebyl hrou, ale pouze nástrojem pro získání statistických dat. Aby se MeshTest mohl stát plnohodnotnou hrou, je nutně třeba výše zmíněné body vyřešit. Vyřešit je lze patrně pouze přidáním herních mechanik. Konkrétně tedy:

- Zábavnost – Hra by měla obsahovat zábavné herní mechaniky, které se navzájem doplňují. V našem případě tak kromě kradení obsahuje také mechaniky okrádání a boje s jinými hráči, RPG prvky a nebo kolo štěstí.
- Pravidla – Hráč musí být odměněn za správné rozhodnutí a potrestán za špatné rozhodnutí. V našem případě o správnosti rozhodujeme názorem většiny hráčů.

- Neužitečnost – Hra by měla být hrána jen proto, že ji hráče baví hrát.
- Fiktivnost - Aby hra dávala nějaký vnitřní smysl, byl vytvořen fiktivní herní svět, ve kterém hráč zajímá úlohu překupníka, potažmo gangstera.
- Nejistota – Hráč musí mít pocit, že záleží na jeho herních rozhodnutích, ač nedokáže předem odhadnout výsledek. Proto náš odměnový systém využívá prvky náhody pro velikost odměny, ovšem ne pro hodnocení správnosti řešení.

5.1 Zábavnost

Hrát hru by mělo hráče bavit, ať už je tomu tak z libovolného důvodu. Zajistit, že hráče bude hra bavit, není možné. Ovšem je možné do hry zapracovat prvky, které by hráče mohli bavit. Čím více, čím lépe budou propojeny a čím lépe budou vymyšleny, tím větší je šance, že hráče bude hra bavit a tedy ji bude hrát. Navrhované herní prvky jsou popsány v dokumentu přílohy *Design dokument.odt*, základ z nich tvoří:

5.1.1 Pošta

Pošta se dělí na tři podkupiny: Systémová pošta a Hráčská pošta a Odeslaná pošta.

- V Hráčské poště si hráči můžou navzájem psát zprávy a tak se domlouvat na spolupráci proti ostatním hráčům.
- V Systémové poště najde hráč vše, co se stalo s jeho postavou bez jeho vědomí. Kdy a kdo (pokud není protivník příliš dobrý) jej napadl nebo okradl, jestli se mu to podařilo a jak moc byl úspěšný, případně jak skončil. Hráči se také pošle zpráva, když se hráčova postava vrátí z vězení nebo nemocnice.
- V Odeslané poště najde hráč všechny zprávy, které odeslal ostatním hráčům přes poštu.

5.1.2 Překupnictví

Hlavní mechanika hry. Hráči je nejprve vygenerován set voleb. Hráč jej na základě průměrné obtížnosti a možného zisku může přijmout nebo odmítnout. Pokud jej odmítne, je mu vygenerován nový set. Pokud jej hráč přijme,

musí jej dokončit. V rámci každé volby má k dispozici originální objekt (trojrozměrnou trojúhelníkovou síť) a dva objekty pozměněné, ze kterých vybírá. Všechny objekty může posouvat, rotovat i zvětšovat a zmenšovat, změny se provádějí na všech naráz. Úkolem hráče je vybrat takový objekt, který je nejpodobnější originálnímu. Po dokončení celého setu se hráčovy volby vyhodnotí a hráč se dozví jak dopadl. Pokud neuspěl (prodělal na poplatcích více, než získal na zisku), jde do vězení.

5.1.3 Kradení

Hráč si vybere postavu jiného hráče a pošle svou postavu, aby se ji pokusila okrást. Výsledek je náhodný. Pokud uspěje, podaří se postavě získat část peněz protivníka. Pokud neuspěje, nezíská nic. Pokud kriticky neuspěje, je postava zatčena a jde do vězení na dobu odpovídající úrovni postavy. Počítá se dle následujících vzorců:

$X = Rand(1, 100) + SK_z - SB_o$

$Z_z = (X - 50) * P_o$ pro $X > 50$

$Z_z = 0$ pro $X \leq 50$

$t_v = L_z$ pro $X < 20$

$t_v = 0$ pro $X \geq 20$

$Rand(x, y)$ - rovnoměrně náhodné přirozené číslo mezi x a y včetně, SK_z - atribut schopnosti kradení zloděje, SB_o - atribut schopnosti bezpečnost okradeného, Z_z - zisk zloděje, P_o - peníze okradeného, t_v - počet hodin, které zloděj stráví ve vězení, L_z - level zloděje.

5.1.4 Vězení

Pokud je postava hráče přistižena při nějakém zločinu (nepovedené překupnictví, krádež nebo loupež), je postava zatčena a jde do vězení. Doba pobytu ve vězení přibližně odpovídá škodám, které svými činy způsobila nebo mohla způsobit (viz Kapitoly 5.1.3, 5.5.4 a 5.1.5). Během pobytu ve vězení postava nemůže krást, ani loupit. O skončení pobytu ve vězení je hráč informován systémovou zprávou v poště.

5.1.5 Loupení

Hráč si vybere postavu jiného hráče a pošle svou postavu, aby se ji pokusila oloupit. Výsledek je náhodný. Pokud uspěje, postava protivníka skončí v nemocnici. Pokud neuspěje, nic se nestane. Pokud kriticky neuspěje, je postava zatčena a jde do vězení na dobu odpovídající úrovni postavy. Počítá se dle následujících vzorců:

$X = Rand(1, 100) + SU_z - SO_o$
$Z_u = (X - 50) * P_o * 0.1$ pro $X > 50$
$Z_u = 0$ pro $X \leq 50$
$t_v = L_u$ pro $X < 20$
$t_v = 0$ pro $X \geq 20$
$t_n = (X - 50) * SU_z$ pro $X > 50$
$t_n = 0$ pro $X \leq 50$

$Rand(x, y)$ - rovnoměrně náhodné přirozené číslo mezi x a y včetně, SU_z - atribut schopnosti útok útočníka, SB_o - atribut schopnosti obrana oběti, Z_z - zisk útočníka, P_o - peníze oběti, t_v - počet hodin, které útočník stráví ve vězení, L_u - level útočníka, t_n - počet minut, které oběť stráví v nemocnici.

5.1.6 Nemocnice

Pokud je postava hráče napadena jinou postavou a útočník je úspěšný, postava oběti skončí v nemocnici. Doba pobytu v nemocnici odpovídá úspěchu útočníka (viz Kapitola 5.1.5). Během pobytu ve vězení postava nemůže krást, ani loupit. O skončení pobytu v nemocnici je hráč informován systémovou zprávou v poště.

5.1.7 Uplácení

Pokud je hráčova postava zatčena, může se pokusit se vyplatit. Hráč vybírá částku, kterou nabízí jako úplatek. Pokud je tato částka dost velká vzhledem k levelu a atributu obchod jeho postavy, uplácení se povede, hráčova postava se vyhne vězení a od peněz hráčovy postavy se odečte vybraná částka. Pokud tato částka není dost velká, uplácení se nepovede a čas, který měla postava strávit ve vězení, se zvýší o 25%. Tedy dle vzorců:

$M = Snizeni(L * 1000)$
$t_v = 0$ pro $V \geq M$
$t_v = t_{v0} * 1.25$ pro $V < M$

$Snizeni(x)$ - mechanika snížení ceny (Kapitola 5.1.8), L - level postavy, t_v - čas, který postava stráví ve vězení, t_{v0} - původní čas, který postava měla strávit ve vězení.

5.1.8 Snížení ceny

Na část poplatků a cen (například mez při uplácení) se automaticky aplikuje mechanika snížení ceny. Mechanika spočívá v tom, že se tato cena sníží vzhledem k hodnotě atributu obchod dle následujícího vzorce:

$Snizeni(x) = x * (1 - SO * 0.005)$

$Snizeni(x)$ - snížení ceny aplikované na hodnotu x, SO - atribut schopnosti obchod.

5.1.9 Ukecávání

Pokud je hráčova postava zatčena, může se pokusit se vymluvit ze zatčení. Výsledek je náhodný. Pokud uspěje, vyhne se vězení. Pokud neuspěje, jde do vězení a doba ve vězení se zvyšuje. Pokud kriticky uspěje, postava se nejen vyhne vězení, ale navíc získá i odškodnění. Počítá se dle následujících vzorců:

$C = (3 * L)$ do maxima 50

$X = Rand(1, 100) + SR - C$

$t_v = t_{v0} * (1 + 0.01 * (50 - X))$ pro $X < 50$

$t_v = 0$ pro $X \geq 50$

$Z = 0$ pro $X < 80$

$Z = L * (80 - X) * 10$ pro $X \geq 80$

L - level postavy, $Rand(x, y)$ - rovnoměrně náhodné přirozené číslo mezi x a y včetně, SR - atribut schopnosti řečnictví, t_v - počet hodin, které útočník stráví ve vězení, t_{v0} - počet hodin, které útočník původně měl strávit ve vězení, Z - peněžní zisk postavy.

5.1.10 Banka

Hráč může peníze herní měny volně vkládat na herní bankovní účet. Peníze, které jsou na tomto účtu, nemůžou být zaměřeny v rámci kradení a jsou tedy v bezpečí. Banka má ovšem záporný úrok 2% za den a navíc si při výběru z účtu účtuje 10% poplatek, proto se nevyplatí mít v bance dlouhodobě uložené peníze. Jedná se o mechaniku, která má chránit hráče před přehnanými útoky jiných hráčů, například pokud se proti jednomu hráči domluví ostatní, nebo pokud hráč chce dočasně přestat hrát, ale nechce riskovat ztrátu svého herního kapitálu.

5.1.11 Kolo štěstí

Kolo štěstí je v současné verzi hry jediný způsob, jak využít peníze. Hráč si vybere částku, kterou chce vsadit (a o kterou tedy přijde) a na základě velikosti této částky a náhody je mu vybrána odměna, kterou získá. Výběr probíhá dle vzorců:

$X = Rand(1, 100)$

$Z_p = (C * (X - 15) * 0.04)$ pro $20 \leq X < 50$
$Z_p = 0$ jinak
$Z_{XP} = 0.5*(X-50)*10$ když $(50 \leq X \leq 90)$ a $(C \geq Snizeni(L*X*10))$
$Z_{XP} = 0$ jinak
$Z_{SP} = (X - 90)$ když $(X > 90)$ a $(C \geq Snizeni(L * X * 10))$
$Z_{SP} = 0$ jinak

$Rand(x, y)$ - rovnoměrně náhodné přirozené číslo mezi x a y včetně,Z_p - penězní zisk postavy, C - vsazená částka, Z_{XP} - zisk zkušenostních bodů, $Snizeni(x)$ - mechanika snížení ceny (Kapitola 5.1.8), L - level postavy, Z_{SP} - zisk schopnostních bodů SP.

5.1.12 Neimplementované základní mechaniky

Jedná se o mechaniky stojící na systému předmětů. Tento systém se ukázal jako podstatně komplexnější než se očekávalo. Hráčova postava měla mít možnost umísťovat předměty (například kalhoty, čepice, nůž) do některých svých políček (slotů) – například nohy, hlava, ruce. Přesun z a do těchto políček měl probíhat z inventáře a skladu postavy, které měly být omezené, což mělo hráče nutit zbavovat se přebytečných předmětů. Předměty umístěné v políčkách postavy měly postavě dodávat bonus k hodnotě jejích atributů schopností. Tyto předměty mělo být možné získat náhodně během překupnictví nebo kola štěstí, ukrást jinému hráči nebo zakoupit z některých herních obchodů. Tato mechanika pouze komplementuje ostatní herní mechaniky a není tak nezbytně nutná pro funkčnost hry. Její hlavní úlohou bylo využití peněz, které už ale využívá implementované kolo štěstí.

5.2 Pravidla

Hráč musí být penalizován, pokud volí špatně a odměněn, pokud volí dobře. Odměna by přitom měla být tím větší, čím těžší je úkol – nesmí to ovšem platit naopak. Aby bylo možné hráče penalizovat za špatnou volbu, je potřeba nejprve vědět, která volba je správná a která je špatná. Vzhledem k tomu, že úkolem MeshTestu bylo právě tuto informaci získat, to bohužel ze základu nevíme.

Na základě mnoha hráčských voleb je ovšem možné získat statistiku a tato statistika může být rekurentně použita pro získání informace o správnosti volby i obtížnosti volby. Jako správná se bere ta volba, pro kterou se rozhodla většina hráčů. Obtížnost volby je stanovena na základě poměru mezi častostí obou možností volby. Konkrétně to vyjadřují následující vzorce pro modely (trojúhelníkové sítě) i a j:

$H_i = (\sum KV_i)/(\sum VV_i)$
$SV_{ij} = |H_i - H_j|$

H_i - hodnocení i, KV_i - volby, ve kterých bylo správně zvoleno i, VV_i - všechny volby s i, SV_{ij} - složitost volby i, j. Minimum je 0.1 pro nejsložitější volbu. 0.1 je tedy nejtěžší obtížnost, 1 je nejnižší obtížnost. Pro lepší přehled hráče je pro něj obtížnost při prezentaci převrácena (1 je nejlehčí obtížnost, 10 nejtěžší obtížnost).

Spravedlivost systému by se tak měla postupem času jen zvyšovat, pokud se tedy někomu nepodaří systém zneužít, systém je tedy třeba nastavit tak, aby to nebylo možné (viz Kapitola 5.6). Navíc je potřeba, aby počáteční data byla vyplněna zodpovědnými hráči, neboť při tomto nastavení nebude při prvních volbách dostupná žádná statistika, odpovědi tak budou vždy správně a hráči tak budou mít na systém největší vliv. Pokud bude ovšem systém hráče motivovat k co největší přesnosti, měl by mít systém schopnost se postupně samoregulovat.

5.3 Neužitečnost

Hráč hraje hru proto, že ho to baví a není to pro něj práce. Nejen, že by ho to, že jeho hraní má i jiné účely (v našem případě generovat vědecká data) nemělo zajímat, ideální by bylo, pokud by tuto skutečnost ignoroval, aby tak tato informace nemohla ovlivnit jeho chování.

5.4 Fiktivnost

Pokud se má hráč do hry ponořit a nebrat ji jen jako zbytečnost, měla by mít hra svůj vlastní svět, ve kterém má činnost, kterou hráč vykonává, nějaký vnitřní smysl. Pokud má mít hra svůj svět, musí tento svět odpovídat činnosti, kterou hráč provádí. Vymyslet takový svět, respektive vymyslet důvod, proč by si někdo měl vybírat ze dvou objektů objekt podobnější originálu, není zcela jednoduché.

Různé návrhy jsou načrtnuty v dokumentu přílohy *herní svět.odt*, jako nejlepší z nich byl vybrán návrh překupníka padělků, který si vybírá padělek, který chce prodat, a musí si logicky vybrat takový padělek, který více připomíná originál. Okolo toho bylo pak také možné postavit zbytek herního světa zpracovaný jako gangsterské podsvětí, ve kterém se hráčské postavy můžou okrádat a napadat a můžou být zatčeny.

5.5 Nejistota

Hráč (uživatel) musí mít pocit, že činnost, kterou vykonává, má nějaký smysl. Tedy za provedení úkolu musí dostat nějakou odměnu nebo naopak trest. Nejjednodušeji implementovatelnou odměnou pro hráče za splnění malého úkolu je herní zlepšení (viz Kapitola 4.1.2). Tato odměna má ve svém nejobecnějším stavu charakter skóre, které můžou ostatní vidět a můžou tak mezi sebou soupeřit. Pro hráče podstatně zajímavější je ovšem, pokud má toto skóre nějaký vnitřní smysl, kterým jsou nejčastěji nějaké RPG prvky jako například herní peníze nebo herní zkušenosti.

Pro vyřešení tohoto problému byl vymyšlen celý systém herních odměn a trestů. Za každou volbu (volbou myšleno výběr ze dvou trojúhelníkových sítí tu, která je podle hráče podobnější originálu) získává hráč ohodnocení podle toho, jak složitá tato volba byla a jestli zvolil správně (viz Kapitola 5.2). Odměny a tresty jsou následující:

5.5.1 Peníze

Slouží prozatím jako ukazatel skóre a dle původního plánu by měly sloužit i pro obchodování a jiné účely. Při správné volbě získá hráč zisk odpovídající složitosti volby. Při špatné volbě nezíská žádné peníze. Množství peněz, které hráč může získat z jedné volby se počítá takto:

$ZV_i = L * 100 * (1 + Rand(1, 10) * RandF(0.1, 1)) * (3/SV_i) + Pov * Rand(1, 10)$

ZV_i - peněžní zisk z volby i, L - level postavy, SV_i - složitost volby i, Pov - pověst, $Rand(x, y)$ - rovnoměrně náhodné přirozené číslo mezi x a y včetně, $RandF(x, y)$ - rovnoměrně náhodné reálné číslo mezi x a y včetně.

Bez ohledu na správnost je při každé volbě nucen hráč zaplatit v herní měně poplatek odpovídající přibližně polovině možného zisku z dané volby. To motivuje hráče dávat si větší pozor, aby vybral správně (viz Kapitola 5.2) Poplatky se počítají takto:

$Pop = 1.1 * (\sum_{i=1}^{n} ZV_i)/2$

Pop - poplatky, ZV_i - zisk z volby i.

5.5.2 Pověst

Pověst slouží jako ukazatel toho, jak dobře dokáže hráč objekty vybírat a zároveň nepatrně zvyšuje hráčův potencionální zisk. Pověst se získává každou správnou volbou a ztrácí každou špatnou volbou, přičemž platí, že čím složitější volba byla, tím více pověsti získá v případě správné volby nebo v opačném případě tím méně pověsti ztratí. Konkrétně:

$Z_P = 1/SV$
$Zt_P = 20 * SV$
Z_P - zisk pověsti, Zt_P - ztráta pověsti, SV - složitost volby

5.5.3 Zkušenosti

Zkušenosti představují vstupní bránu k RPG prvkům. Zkušenosti hráč získává za každou volbu bez ohledu na správnost, čím složitější volba, tím více zkušeností může získat. Dle vzorce:

$Z_{XP} = L * (1 + rand(1)) * (1/SV)$

Z_{XP} - zisk zkušenostních bodů, L - level postavy, SV - složitost volby, $Rand(x, y)$ - rovnoměrně náhodné přirozené číslo mezi x a y včetně.

Vždy při dosažení určité mezní hodnoty zkušeností se zkušenosti využijí pro zvýšení úrovně hráče a hráč také získá body (SP), které může použít pro zlepšení atributů své postavy do maxima 100 bodů. Vzorce jsou následující:

$Max_{XP} = L * 100$
$Z_{SP} = 10$
$C_{SP,i} = roundUP(A_i/10)$

Max_{XP} - počet zkušenostních bodů potřebných ke zvýšení levelu postavy, Z_{SP} - zisk schopnostních bodů SP při zvýšení levelu, $C_{SP,i}$ cena SP pro zvýšení jednoho bodu atributu i, A_i - body atributu i, $roundUP(x)$ - zaokrouhlení čísla x nahoru na celá čísla.

5.5.4 Zatčení

Pokud jsou poplatky, které má hráč v rámci mechaniky překupnictví zaplatit, větší než zisky a hráčova postava nemůže zbytek doplatit, je hráčova postava zatčena. Postava je také zatčena v případě, že se jí nepovede krádež nebo loupež. Hráč může buď poslat svou postavu rovnou do vězení, nebo se pokusit vězení vyhnout použitím mechanik uplácení a ukecávání (Kapitoly 5.1.7 a 5.1.9). V případě vězení kvůli nedostatku peněz na zaplacení ztráty v překupnictví se doba hráčovy postavy ve vězení v sekundách vypočítá jako:

$DV = Pop - Z$

DV - doba, kterou postava stráví ve vězení v sekundách, Pop - poplatky, Z - zisk

5.6 Ochranné mechaniky proti herním podvodům

Aby mohla hra správně fungovat, hráči se nesnažili nepovoleným způsobem obohacovat, ať už na účet jiných hráčů nebo na účet systému, a aby se zajistilo, že shromážděná vědecká data budou minimálním způsobem ovlivněna, je třeba zajistit takové obranné herní mechanizmy, které budou tuto možnost zcela vylučovat.

Konkrétně je třeba chránit hlavně před těmito problémy:

1. Náhodná volba – Hráč během volby volí náhodně.
2. Domluva – Hráči se domluví, pro který model budou volit a tím ovlivní získaná data i systém hodnocení správnosti a obtížnosti ve svůj prospěch.
3. Malá přesnost – Hráči se nesnaží poctivě najít rozdíl mezi objekty a soustředí se jen na zřetelné rozdíly.
4. Neomezené kradení – Hráči místo činění voleb (která generují vědecká data, která chceme) jen okrádají ostatní hráče.
5. Neomezené překupnictví – Hráči neustálým činěním voleb (překupnictvím) zcela předeženou ostatní hráče a ti tak nemají motivaci hrát, protože hráče ve vedoucí pozici už nemůžou dostihnout.

5.6.1 Náhodná volba

Velkým problémem by bylo, pokud by hráč mohl vydělávat už jen tím, že při volbě vybírá náhodně. Ovšem vzhledem k tomu, že poplatky při volbě představují v průměru nepatrně více než padesát procent zisků, měla by být tato činnost v průměru ztrátová. Hráč tedy nemá motivaci tak činit úmyslně. Navíc by tato praktika měla mít výrazně negativní vliv na atribut pověst, který lze monitorovat a lze tak lehce poznat, jestli někdo tuto činnost praktikuje. Vzhledem k tomu, že jsou volby ukládány do databáze s identifikátorem hráče, lze tyto špatné volby všechny zpětně dohledat a případně eliminovat.

5.6.2 Domluva

Největším nebezpečím z hlediska sbírání validních dat je možnost domluvy mezi hráči. Vzhledem k tomu, že hráči můžou používat interní poštu, mají

tak možnost se přímo ve hře domluvit i na tom, že budou vždy ve volbě vybírat jeden určitý objekt, čímž můžou potencionálně devalvovat všechna nasbíraná data a lze jen obtížně zjistit, že k tomu dochází. Na rozdíl od náhodné volby v tomto případě nemusí pověst hráčů, kteří tento postup praktikují, nabývat záporné hodnoty, protože tento objekt může být správný a nebo se může stát správným následkem hráčských voleb – tento efekt tedy je nejvýraznější u prvních voleb s tímto objektem a prvními hráči by tak ideálně měly být hlavně zodpovědné osoby, u kterých je jistota, že budou volby činit dle svého nejlepšího vědomí.

Systém před tímto problémem jinak chrání hlavně fakt, že jednotlivých objektů je mnoho a lze je od sebe obtížně rozpoznat. Objekty totiž nejsou mimo trojrozměrné reprezentace nijak označeny a dokonce i výpočet zisků se provádí až po dokončení sekvence – hráč tedy neví, jakou měla která volba obtížnost, ani kdy zvolil správně a kdy špatně. I pokud by se tedy hráči chtěli mezi sebou domluvit, museli by si pamatovat velké množství modelů a ani tak by je od sebe nejspíše nerozpoznali. Vzhledem k obtížnosti takového úkolu je tak podstatně jednodušší plnit původní úkol a snažit se vybrat objekt bližší originálu.

Jediný způsob domluvy, při kterém by tak mohlo lehce dojít k tomuto problému, je výběr vždy pravého objektu, případně výběr vždy levého objektu. Ovšem vzhledem k tomu, že stejná volba může probíhat s objekty na obou stranách a strana objektu se ukládá do databáze, je toto možné jednak jednoduše rozpoznat a jednak by to nevedlo k vychýlení správnosti objektu, neboť se objekt může nacházet na obou stranách a výsledek by tak spíše konvergoval k remíze, což by mělo mít stejný efekt jako náhodný výběr.

5.6.3 Malá přesnost

Je také dost možné, že se hráčům nebude chtít trávit čas hledáním rozdílů mezi objekty a že pokud se jim je nepodaří jednoduše najít, vyberou náhodně. Tomu nelze jednoduše zabránit a hráči na to mají právo, nejedná se tak o podvod v pravém slova smyslu a spíše se jedná o hledání, trestání a jinou snahu o zabránění vzniku tohoto problému nejlépe na základě motivace:

Obtížnější volby představují větší zisky, ale také větší poplatky – tedy větší penalizace. Hráči mají možnost odmítnout set, přičemž vědí, jaká je jeho průměrná obtížnost, jak vysoké jsou poplatky a jak velké můžou být zisky. Hráči jsou tak motivováni se snažit o co nejpřesnější odpověď a pokud se jim nechce snažit dlouho ji hledat, můžou si nechat vygenerovat jednodušší set voleb.

5.6.4 Neomezené kradení

Pokud by nějaký hráč mohl krást bez omezení, mohl by tak postupně okrást všechny ostatní hráče a snaha hráčů získat co nejvíce peněz přes překupnictví by tak byla zcela zbytečná a marná. Toto je ošetřeno přímo v rámci mechaniky kradení – výsledek je totiž náhodný a kradení může skončit kritickým neúspěchem, při kterém hráčova postava skončí na určitou dobu ve vězení a nemůže tak krást.

5.6.5 Neomezené překupnictví

Pokud by nějaký hráč hrál velmi aktivně, pochopitelně by také pro nás generoval velké množství dat a tak nám byl velmi prospěšný, jenže je možné, že by zároveň způsobil nechuť ostatních hráčů hrát, protože ti by neměli šanci ho dostihnout, protože buď nemají tolik času, nebo začali hrát až později. To by mohlo způsobil snížení aktivity ostatních hráčů, případně i jejich úplné přestání s hraním. Předtím chrání mechanika kradení, která umožňuje hráčům, pokud se jim to podaří, okrást jiné hráče o procentuální část jejich herní měny. Hráči ovšem neví, kolik má kdo peněz, pouze jak dlouho někdo už nebyl okraden, proto hráči nemusí vždy zaměřovat jen nejbohatší hráče. Hráč se navíc těmto útokům může bránit tím, že dá peníze do banky (Kapitola 5.1.10).

6 Realizace

6.1 Realizace webové části

Webová aplikace používá rozdělení dle architektury *Model-View-Controller*, navíc jsou dále odděleny javascriptové soubory, CSS soubory a konstanty, které sice spadají pro view (pohled), ale v rámci souborů se je sluší oddělit.

6.1.1 Pohled

Všechny stránky webu jsou uloženy ve složce *view*. Název každého souboru ve složce odpovídá názvu pohledu (bez koncovky .phtml). Každý tento soubor volá metody ze souboru *zaklad.php*, který pomocí příkazu *echo* vrací základní komponenty webu, které jsou shodné pro všechny stránky webu jako jsou hlavička, horní menu, levé menu a zápatí, je tedy jednoduchou náhradou za *Twig*. V rámci hlavičky pak také načítá používané nastavení CSS.

Pokud tato stránka potřebuje Javascript, volají se jeho metody ze souborů ve složce *js*. Javascript se ovšem používá pouze na jediném místě – na stránce pohledu *prekupnictvi*.

V rámci view souborů neprobíhají žádné výpočty, pouze probíhá kontrola hráčských vstupů a akcí. Pro výpočty a použití databáze se volají metody kontrolerů (řadičů).

6.1.2 Kontroler

Soubory kontrolerů mají název, který odpovídá názvu pohledu +“Kontroler.php“ a jsou uloženy ve složce *kontrolery*. Tímto způsobem jsou také dynamicky načítány. Všechny kontrolery dědí od abstraktního kontroleru s názvem Kontroler.php a dědí tak od něj hlavně metodu vypisPohled(), která na základě pohledu nastaví header (hezké URL), načte správný view a ošetří data proti HTML znakům (ochrana proti *Cross Site Scripting*).

O určení tohoto pohledu se stará třída *SmerovacKontroler.php*, který na základě URL nastaví pohled a ostatní základní parametry (titulek, popis, klíčová slova). Tak činí jednak v rámci kontroly, když je zavolána v rámci view souboru, jednak v rámci souboru *index.php*, na který by měl být vždy uživatel přesměrován díky souboru *.htaccess*. Soubor *index.php* také zajišťuje periodické výpočty pro všechny uživatele (přepočítání hodnocení modelů) a načítá základní soubory (konstanty a případně potřebné kontrolery).

Ostatní kontrolery odpovídají pohledům se stejným názvem a obsahuje metody specifické pouze pro svůj pohled. Ty také konají většinu výpočtů aplikace, komunikují s databází přes modely a případně poskytují view informace buď jako návratovou hodnotu funkce nebo přes session.

6.1.3 Model

Třídy modelů zajišťují kontakt s databází. Získané informace buď předávají přímo kontroleru, nebo je rovnou zapisují do session. Složka modely obsahuje tyto třídy:

- Třída *Pripoj* obsahuje obsluhu samotného kontaktu s databází. Umí se k databázi připojit, odpojit a posílat různé typy bezpečných (zabezpečené proti *SQL injection*) žádostí (například vrátit počet řádků, vrátit jeden řádek, všechny řádky apod.)
- Třída *Models* obsahuje obsluhu databáze pro práci s modely, sekvencemi, volbami a záznamy.
- Třída *Users* obsahuje obsluhu databáze pro práci s hráči.
- Třída *Zpravy* obsahuje obsluhu databáze pro práci se zprávami v poště.
- Třída *Help* nepracuje s databází, ale je knihovní třídou, která obsahuje metody, které používá více kontrolerů.

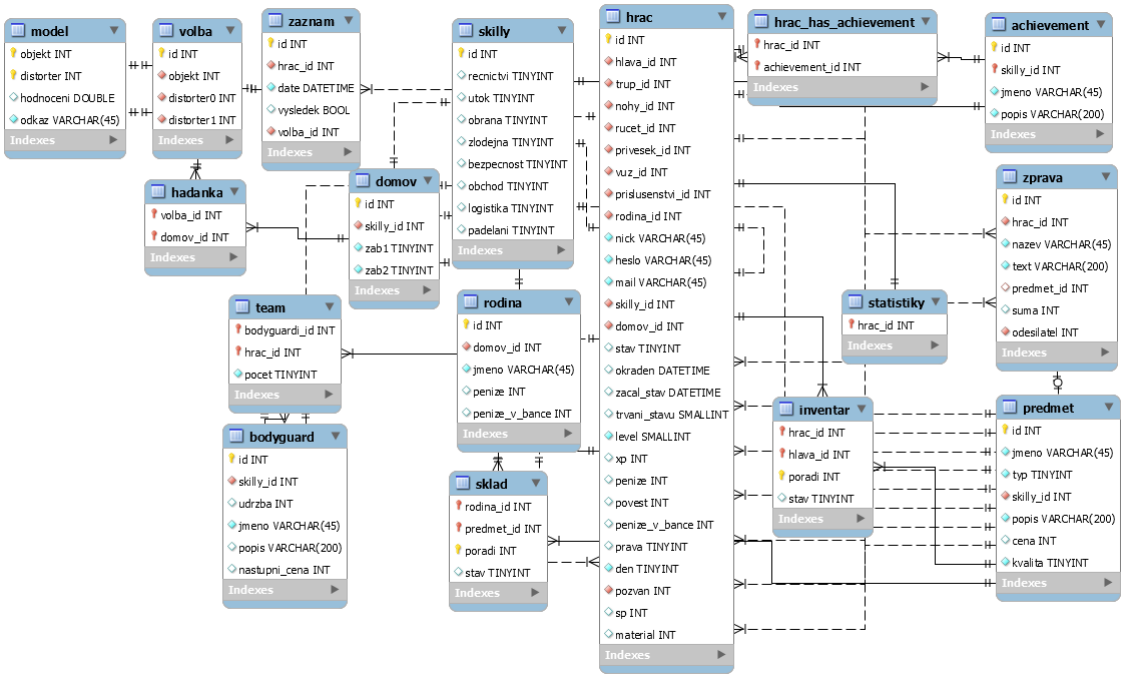
6.2 Databáze

Databáze byla navrhnuta s cílem minimalizovat systémovou chybu a možné hráčské zneužití (vysvětluje kapitola 5.6). Pokud by měla být v budoucnu databáze upravována, mohlo by vlivem manipulace s databází dojít ke ztrátě dat a rozhodně by během tohoto předělání nemohla aplikace fungovat v rámci běžného provozu. Abychom se takovým problémům vyhnuli, byla databáze rovnou sestavena tak, aby pokrývala všechny navrhnuté mechanizmy aplikace, a obsahuje tak i entity, které prozatím nejsou používané. Celá databáze na Obrázku 6.1.

Nejdůležitější částí databáze jsou tyto entity:

6.2.1 model

Tato entita představuje trojúhelníkové sítě.



Obrázek 6.1: Použitý model databáze. Oproti realizaci je atribut *vysledek* entity *zaznam* typu bool - změněno pro případnou volbu s více modely.

- Primárními klíči jsou:
  - *objekt* – číslo daného objektu – původní trojúhelníkové sítě.
  - *distorter* – číslo distorteru – algoritmu, kterým byla původní trojúhelníková síť pozměněna.
- Atribut *hodnoceni* ukazuje, jak si daná trojúhelníková síť stojí oproti ostatním trojúhelníkovým sítím stejného objektu v rámci hráčských voleb. Nabývá hodnot od 0 (nejhorší) po 1 (nejlepší).
- Atribut *odkaz* obsahuje název trojúhelníkové sítě, pod kterým je možné trojúhelníkovou síť nalézt a načíst.

6.2.2 volba

Tato entita představuje možné volby (výběr ze dvou trojúhelníkových sítí), které může hráč vyplnit.

- Primárním klíčem je vlastní identifikační číslo volby *id*.
- Sekundární klíče jsou následující:

  - *objekt* – číslo objektu, jehož trojúhelníkové sítě jsou předmětem volby
  - *distorter0* – číslo distorteru, který byl aplikován na trojúhelníkovou síť, kterou v rámci volby hráč vidí vlevo.
  - *distorter1* – číslo distorteru, který byl aplikován na trojúhelníkovou síť, kterou v rámci volby hráč vidí vpravo.

6.2.3 zaznam

Tato entita představuje záznam jedné hráčem uskutečněné volby.

- Primárním klíčem je vlastní identifikační číslo záznamu *id*.
- Sekundární klíče jsou následující:
  - *volba_id* – identifikační číslo volby, o které je proveden záznam.
  - *sekvence_id* – identifikační číslo sekvence, které je tento záznam součástí.
- Atribut *date* představuje datum a čas, ve kterém hráč volbu vyplnil.
- Atribut *vysledek* vyjadřuje, kterou z trojúhelníkových sítí si hráč zvolil. 0 pro trojúhelníkovou síť vlevo, 1 pro trojúhelníkovou síť vpravo.
- Atribut *poradi* vyjadřuje kolikátý je tento záznam v pořadí v rámci dané sekvence.

6.2.4 sekvence

Tato entita představuje množinu voleb, které si hráč může zvolit, jestli chce plnit. Pokud ji hráč odmítne, sekvence nemá žádné záznamy. Jakmile je sekvence přijata, nelze ji odmítnout.

- Primárním klíčem je vlastní identifikační číslo sekvence *id*.
- Sekundární klíč *hrac_id* představuje identifikační číslo hráče, kterému byla tato sekvence vygenerována.
- Atribut *poplatky* představuje poplatky v herní měně, které hráč zaplatí při dokončení sekvence.
- Atribut *zisk* představuje, kolik si hráč vyplněním celé sekvence vydělal. Pokud sekvence nebyla dokončena, nabývá atribut hodnotu 0.

- Atribut *obtiznost* představuje obtížnost dané sekvence vypočtenou jako průměr z rozdílů hodnocení zkoumaných modelů v rámci voleb. 0.1 pro nejtěžší obtížnost, 1 pro nejnižší obtížnost. Pro lepší přehled hráče je pro něj obtížnost převrácena (1 je nejlehčí obtížnost, 10 nejtěžší obtížnost).

6.2.5 hrac

Tato entita představuje hráčský účet a parametry jeho postavy.

- Primárním klíčem je vlastní identifikační číslo hráče *id*.
- Atribut *nick* představuje jméno hráče.
- Atribut *mail* představuje e-mail hráče.
- Atribut *heslo* představuje zahashované heslo hráče.
- Atribut *skilly_id* představuje identifikační číslo skillů hráčovy postavy.
- Atribut *stav* představuje současný stav hráčovy postavy. 0 je normální stav, 2 je vězení, 3 je nemocnice.
- Atribut *okraden* představuje čas, kdy byla hráčova postava naposledy okradena.
- Atribut *zacal_stav* představuje čas, kdy naposledy změněn stav hráčovy postavy.
- Atribut *trvani_stavu* představuje, jak dlouho by měl současný stav trvat (pokud stav není 0) v sekundách.
- Atribut *level* představuje level hráčovy postavy.
- Atribut *xp* představuje zkušenostní body hráčovy postavy.
- Atribut *penize* představuje peníze hráčovy postavy.
- Atribut *povest* představuje pověst hráčovy postavy.
- Atribut *penize_v_bance* představuje peníze hráčovy postavy, kterou jsou umístěné v bance.
- Atribut *sp* představuje schopnostní body SP hráčovy postavy.

6.3 Konverze trojúhelníkových sítí

Výrazným problémem pro implementaci trojúhelníkových sítí do systému byl jejich formát. *MeshTest* totiž obsahoval všechny trojúhelníkové sítě ve velmi specifickém formátu *.dat*. Tento formát se běžně nepoužívá a bylo tak nutné všechny trojúhelníkové sítě překonvertovat do formátu, který umí javascriptová knihovna *Three.js* načíst. Pro největší jednoduchost byl zvolen nejuniverzálnější formát OBJ, který umí *Three.js* načíst pomocí třídy *OBJLoader*. Na konverzi byl použit program *dat2obj* dodaný vedoucím bakalářské práce. Ten se ovšem ukázal jako ne zcela ideální a zkonvertované trojúhelníkové sítě neměly standardní formát OBJ (viz Kapitola 2.1), doprovázely je následující problémy:

- Normály z množiny normál byly místo prefixu "vn" označeny prefixem "n".
- Stěny byly u většiny trojúhelníkových sítí zkonvertovány s opačnou orientací.
- Stěny z množiny stěn neobsahovaly informaci o k nim se vztahujícím normálám. Respektive místo formátu "f 1//1 2//2 3//3" byly ve formátu "f 1 2 3".

Vzhledem k tomu, že formát zkonvertovaných souborů vypadal velmi podobně a navíc se očekávala bezchybnost této konverze, nebyla tato chyba odhalena až do pozdní části vývoje. Kvůli tomu způsobily tyto chyby ve formátu vážné problémy při implementaci zobrazení trojúhelníkových sítí a chyby byly hledány právě v této implementaci, což vedlo k řešení neexistujících problémů a přepisování fungujících knihoven, zvláště třídy *OBJLoader*, která při nalezení prefixu "n" spadla s chybou. Zároveň vzhledem k tomu, že nebyla obsažena informace o vztahu normál ke stěnám, aplikace dodané normály zcela ignorovala, používala zásadně ploché stínování získané z vlastních vygenerovaných normál. To bylo považováno za chybu vzniklou z chybného pochopení knihoven.

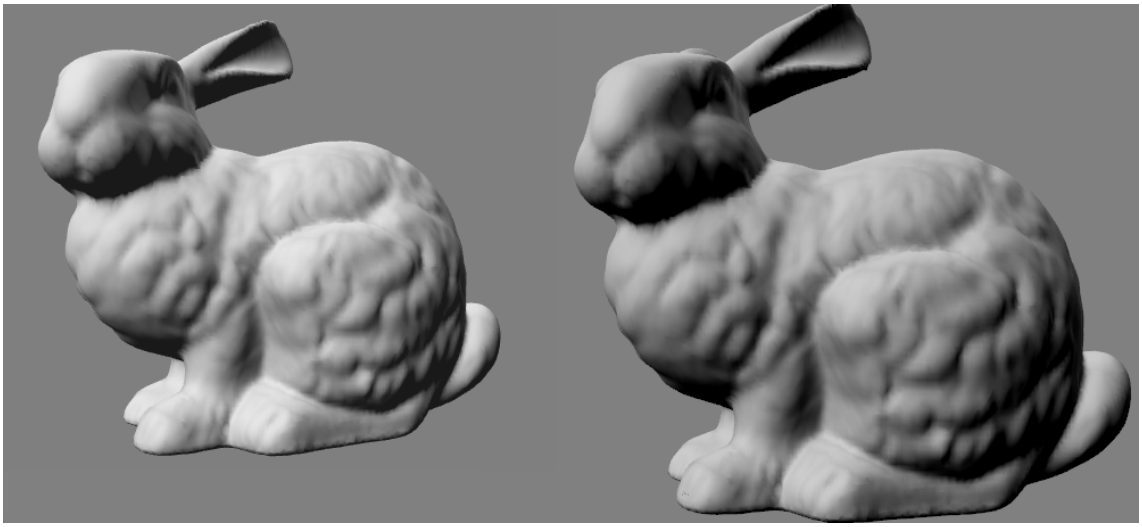
V případě prvního a třetího bodu (prefix normál a formát stěn) byl tento problém vyřešen jednoduchým přepsáním souborů na správný formát. Druhý bod (chybná orientace stěn) musel být kvůli obtížné detekci orientace stěn vyřešen oboustranným vykreslením stěn.

6.4 Zobrazení trojúhelníkových sítí

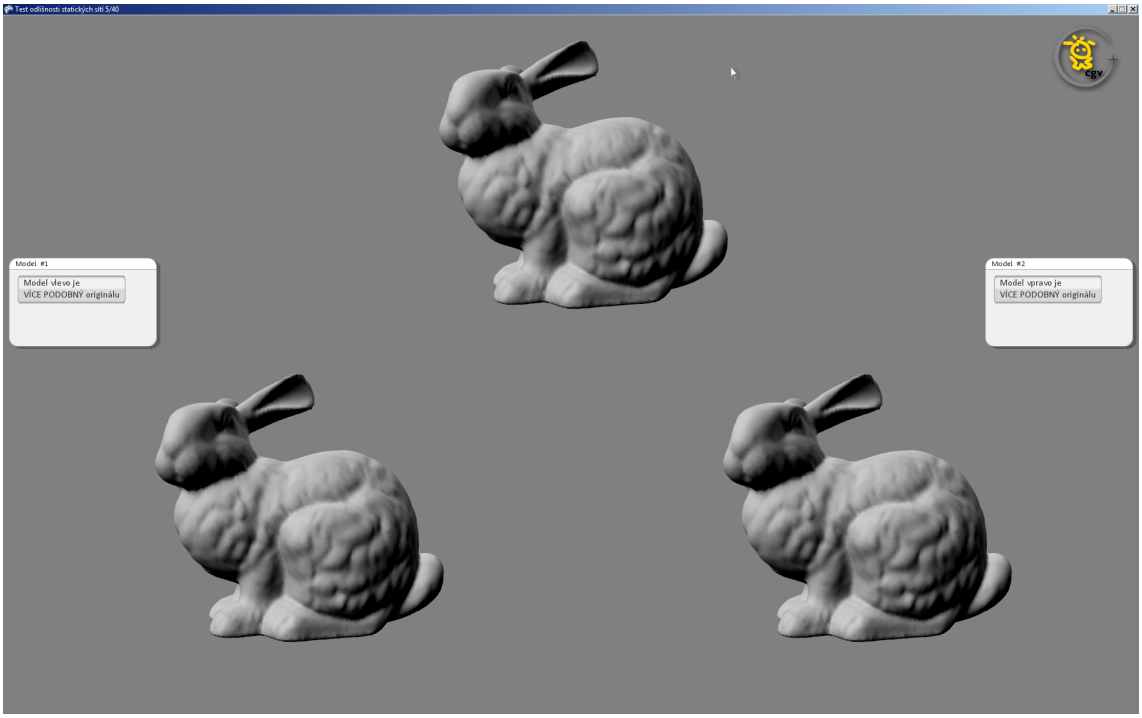
K realizaci zobrazení trojúhelníkových sítí pro potřeby aktivity překupnictví byla použita javascriptová knihovna *Three.js* fungující prostřednictvím *WebGL*, které je implementováno do každého běžného webového prohlížeče. Snahou bylo replikovat podmínky panující v MeshTestu. Použity byly tyto komponenty s následujícím nastavením:

- Komponentou pro vykreslování scény je *WebGLRenderer*. Tento *WebGLRenderer* má zapnutý antaliasing a velikostí odpovídající 1/3 šířky obrazovky na šířku a 1/2.5 výšky obrazovky na výšku. Při změně velikosti okna se velikost automaticky přenastaví a spolu s ní i nastavení kamery. Základní barvou pozadí je téměř plná černá.
- Kamera, třídy *PerspectiveCamera*, používá úhel zorného pole čtyřicet pět stupňů.
- Na objekt svítí dvě bílá opačně orientovaná přímá světla typu *DirectionalLight*, první s osmdesátiprocentní intenzitou z pozice vlevo nahoře vzadu, druhé s padesátiprocentní intenzitou z pozice vpravo dole vpředu.
- Uživatel může se scénou manipulovat díky *OrbitControls*, třídě využívající kameru a scénu. Tato komponenta dovoluje zoomovat a tím tak zvětšovat i zmenšovat objekt, posouvat kameru a otáčet kamerou.
- Trojúhelníková síť je načtena s pomocí třídy *OBJLoader*, která zajišťuje načtení mnohoúhelníkové sítě ve formátu OBJ včetně normálových vektorů vrcholů a případných textur (tedy za předpokladu, že je soubor uložený ve správném formátu). Načtení normálových vektorů vrcholů zajišťuje hladké stínování, díky kterému objekt nevypadá hranatě. Z důvodu špatného zkonvertování OBJ souborů jsou všechny stěny trojúhelníkových sítí vykreslovány oboustranně.

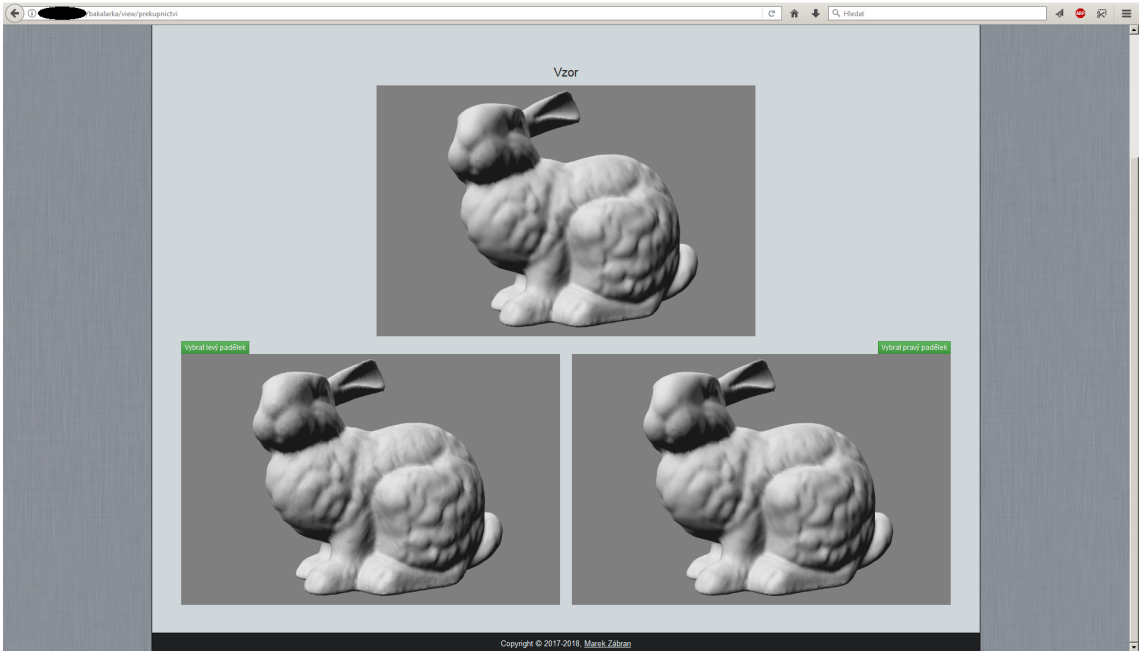
Vzhledem k problematickému chování komponent scény musí mít každá scéna své vlastní komponenty (včetně světel), kromě základních datových typů, kamery a OBJLoaderu (není součást scény). Aby bylo zajištěno, že hráč nebude zmaten kvůli rozdílným zobrazení u tří trojúhelníkových sítí, mají všechny scény nastavenou identickou kameru, čímž je zajištěno, že směr, zoom i pozice kamery jsou vždy ve všech třech komponentách stejné a komponenty se liší pouze trojúhelníkovými sítěmi. Porovnání mezi zobrazením trojúhelníkových sítí v MeshTestu a ve vlastní aplikaci na Obrázku 6.2, potažmo na Obrázcích 6.3 a 6.4.



Obrázek 6.2: Zobrazení trojúhelníkových sítí. Vpravo MeshTest, vlevo vlastní aplikace.



Obrázek 6.3: Zobrazení trojúhelníkových sítí v rámci MeshTestu.



Obrázek 6.4: Zobrazení trojúhelníkových sítí v rámci vlastní aplikace.

6.5 Testování

6.5.1 Testování konceptu

První testování tohoto programu proběhlo již ve stádiu konceptu. Vzhledem k jeho účelu bylo potřeba zajistit, aby nemohla být získaná data nijak poškozena uživateli nebo systematickou chybou. Metodou pokládání imaginárních otázek jsme došli k mechanikám, které sice nemůžou těmto potencionálním problémům úplně zabránit, ale můžou je minimalizovat, detekovat a umožnit případně jejich manuální odstranění. Tyto mechaniky byly představeny v Kapitole 5.6.

6.5.2 Prototypování

Následně bylo velmi obtížné testovat složité herní mechaniky už proto, že se vzájemně ovlivňují a text na papíře neříká nic o tom, jak bude aplikace v konečném důsledku vypadat. Z tohoto důvodu byla vytvořena maketa, která sice vypadala opticky dobře, ale bylo následně dle ní zjevné, že navrhovaná aplikace je příliš rozsáhlá, než aby ji bylo možné stihnout v požadovaném termínu, (což se potvrdilo).

Ze stejných důvodů byl vytvořen také prototyp databáze, který byl následně diskutován z hlediska modulárnosti a možných budoucích modifikací.

Verze *Alfa 0,1* pak fungovala jako prototyp, který ukazoval, že je jednoduše možné pomocí *Three.js* načíst trojúhelníkovou sít, zobrazit ji a dále s ní pracovat.

6.5.3 Nedostatky v testování

Tím ovšem testování konceptů z časových důvodů skončilo a dále nebyly testovány ani již vytvořené koncepty pro co nejrychlejší urychlení tvorby aplikace. Tím jsme si udělali medvědí službu, protože kvůli tomu byla přehlédnuta chyba popsaná v Kapitole 6.3 a následkem toho došlo k významné časové ztrátě. V rámci hledání této chyby sice byl testován javascriptový kód, ale až na výjimky ze zcela zbytečných důvodů.

6.5.4 Testování finální verze

Konečná verze aplikace tak byla testována pouze manuálně na *green day scenario* a na možnosti poškození dat v databázi v rámci chyby aplikace nebo nečekaných akcí hráče. Ostatní aspekty nebyly testovány.

7 Závěr

Práce na online aplikacích pro velké množství uživatelů, u kterých je potřeba brát v úvahu vliv uživatelů na data, uživatelskou interakci a navíc sbírat vědecká data je velmi náročná, neboť nutně potřebuje kombinovat více různých technologií, jejichž spojení často vede k naprosto nečekaným komplikacím. V našem případě nebyl výrazný problém v použitých technologiích, ale hlavně v nepřehledném vývoji, nekontrolování vstupních dat a zvláště pak špatném časovém managementu. Kvůli těmto problémům byla hra Překupník implementována s pouze základnějšími mechanikami, nebyla velmi důkladně testována a doposud si ji nemělo šanci zahrát větší počet hráčů.

Podmínky bakalářské práce byly i přesto v mezích splněny:

- Úspěšně jsem se seznámil s nástrojem pro uživatelské studie podobnosti trojúhelníkových sítí MeshTest a s tématem ztrátového zpracování trojúhelníkových sítí obecně.
- Byl vytvořen design dokument (*Design dokument.odt*), který navrhl velké množství herních mechaniky, které by spolu mohly fungovat, navzájem se komplementovat a splňují požadavky na zachování stejného formátu sbíraných dat s omezením všech možných zdrojů na zavedení systematické chyby. Mechaniky byly ve fázi konceptu z části otestovány a pravděpodobnost vzniku neodstranitelné systémové chyby, která zaměřuje sbíraná data, byla minimalizována.
- Část z těchto navržených mechanismů bylo implementováno a zbytek stále může být pro zlepšení hráčského zážitku implementován v budoucnu. Aplikace je webová a dokáže online shromažďovat data od velkého množství uživatelů včetně dodatečných atributů, jako jsou například pořadí voleb, čas vyplnění či informace o plniteli.

Aplikaci stále svazuje několik problémů, jako jsou:

- Neintuitivnost hry pro nehráče, která by se snad dala vyřešit nějakým rozsáhlým tutoriálem.
- Necitlivé ovládání pohybů s trojúhelníkovými sítěmi a jejich špatné základní velikosti, která by se dala vyřešit zvláštním nastavením pro každý typ sítě.

- Pomalé načítání trojúhelníkových sítí z důvodu jejich velké velikosti, což by šlo vyřešit jiným formátem souborů nebo případně trvalým ukládáním trojúhelníkových sítí na disku hráče.

Na aplikaci se ovšem stále pracuje a v budoucnu si snad najde oblibu většího množství hráčů, kteří ji budou pravidelně hrát.

Literatura

[1] Adams, E. – Rollings, A. *Fundamentals of Game Design (Game Design and Development Series)*. Prentice-Hall, Inc., 2006. ISBN 0131687476.

[2] Botsch, M. et al. *Polygon Mesh Processing*. AK Peters, 2010. ISBN 9781568814261.

[3] Cabello, R. et al. *Three.js* [online]. 2010-2018. [cit. 2018/06/20]. Dostupné z: `https://threejs.org/`.

[4] Caillois, R. *Les Jeux et les hommes; le masque et le vertige*. Gallimard, 1958.

[5] Chatfield, T. 7 ways games reward the brain, 2010. Dostupné z: `https://www.ted.com/talks/tom_chatfield_7_ways_games_reward_the_brain`.

[6] Chatfield, T. *Fun Inc.: why gaming will dominate the twenty-first century*. Pegasus Books, 2011.

[7] contributors, W. *John D. Carmack* [online]. 2018. [cit. 2018/06/20]. Wikipedia, The Free Encyclopedia. Dostupné z: `https://en.wikiquote.org/wiki/John_D._Carmack`.

[8] contributors, W. *Model–view–controller* [online]. 2018. [cit. 2018/05/25]. Wikipedia, The Free Encyclopedia. Dostupné z: `https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller`.

[9] contributors, W. *Wavefront .obj file* [online]. 2018. [cit. 2018/01/28]. Wikipedia, The Free Encyclopedia. Dostupné z: `https://en.wikipedia.org/wiki/Wavefront_.obj_file`.

[10] contributors, W. *Peak signal-to-noise ratio* [online]. 2018. [cit. 2018/06/06]. Wikipedia, The Free Encyclopedia. Dostupné z: `https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio`.

[11] contributors, W. *Structural similarity* [online]. 2018. [cit. 2018/06/06]. Wikipedia, The Free Encyclopedia. Dostupné z: `https://en.wikipedia.org/wiki/Structural_similarity`.

[12] contributors, W. *Three.js* [online]. 2018. [cit. 2018/05/25]. Wikipedia, The Free Encyclopedia. Dostupné z: `https://en.wikipedia.org/wiki/Three.js`.

[13] CONTRIBUTORS, W. *WebGL* [online]. 2018. [cit. 2018/06/20]. Wikipedia, The Free Encyclopedia. Dostupné z: https://en.wikipedia.org/wiki/WebGL.

[14] CONTRIBUTORS, W. *Microsoft XNA* [online]. 2018. [cit. 2018/06/20]. Wikipedia, The Free Encyclopedia. Dostupné z: https://en.wikipedia.org/wiki/Microsoft_XNA.

[15] CONTRIBUTORS, W. *Polygon mesh* [online]. Wikipedia, The Free Encyclopedia, 2011. [cit. 2018/02/16]. Wikipedia, The Free Encyclopedia. Dostupné z: https://en.wikipedia.org/wiki/Polygon_mesh.

[16] CONTRIBUTORS, W. *Trianle mesh* [online]. 2017. [cit. 2018/01/28]. Wikipedia, The Free Encyclopedia. Dostupné z: https://en.wikipedia.org/wiki/Triangle_mesh.

[17] EXPERTS, T. *Sandbox* [online]. [cit. 2018/06/20]. Technopedia. Dostupné z: https://www.techopedia.com/definition/3952/sandbox-gaming.

[18] HERBERS, B. *A Game's Atmosphere is Defined by its Mechanics, Not its Aesthetic* [online]. 2016. [cit. 2018/06/20]. Gamnesia. Dostupné z: https://www.gamnesia.com/articles/a-games-atmosphere-is-defined-by-your-interactions-and-not-just-its-world.

[19] HUYNH-THU, Q. – GHANBARI, M. Scope of validity of PSNR in image/video quality assessment. *Electronics Letters*. 2008, 44, 13, s. 800. doi: 10.1049/el:20080522.

[20] HUYNH-THU, Q. – GHANBARI, M. The accuracy of PSNR in predicting video quality for different video scenes and frame rates. *Telecommunication Systems*. Nov 2010, 49, 1, s. 35–48. doi: 10.1007/s11235-010-9351-x.

[21] K. PRZYBYLSKI, A. – SCOTT RIGBY, C. – RYAN, R. A Motivational Model of Video Game Engagement. 06 2010, 14, s. 154–166.

[22] KARNI, Z. – GOTSMAN, C. Spectral compression of mesh geometry. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH 00*. 2000, 27, s. 27–30. doi: 10.1145/344779.344924.

[23] LAVOUÉ, G. – LARABI, M. C. – VÁŠA, L. On the Efficiency of Image Metrics for Evaluating the Visual Quality of 3D Models. *IEEE Transactions on Visualization and Computer Graphics*. Aug 2016, 22, 8, s. 1987–1999. ISSN 1077-2626. doi: 10.1109/TVCG.2015.2480079.

[24] Lavoué, G. A Multiscale Metric for 3D Mesh Visual Quality Assessment. *Computer Graphics Forum*. 2011, 30, 5, s. 1427–1437. doi: 10.1111/j.1467-8659.2011.02017.x.

[25] Lavoué, G. et al. Perceptually driven 3D distance metrics with application to watermarking. *Applications of Digital Image Processing XXIX*. 2006. doi: 10.1117/12.686964.

[26] Meier, J. et al. *Improving Web Application Security: Threats and Countermeasures* [online]. Microsoft Corporation, 2010. [cit. 2018/06/20]. Microsoft Corporation. Dostupné z: `https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649874(v%3dpandp.10)`.

[27] mmoanthropology. *serenity now attack on virtual world of warcraft funeral* [online]. 2012. [cit. 2018/06/20]. Massive Online Anthropology. Dostupné z: `http://mmoanthropology.tumblr.com/post/14468677539/serenity-now-attack-on-virtual-world-of-warcraft-funeral`.

[28] Váša, L. Software for comparing models MeshTest, 2010. Dostupné z: `https://www.kiv.zcu.cz/en/research/downloads/product-detail-en.html?produkt_id=86`.

[29] Váša, L. – Rus, J. Dihedral Angle Mesh Error: a fast perception correlated distortion measure for fixed connectivity triangle meshes. *Computer Graphics Forum*. 2012, 31, 5, s. 1715–1724. doi: 10.1111/j.1467-8659.2012.03176.x.

[30] Yee, N. Motivations for Play in Online Games. *CyberPsychology & Behavior*. 2006, 9, 6, s. 772–775. doi: 10.1089/cpb.2006.9.772.