FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Simulátor deskových her
Student:	Adam Španěl
Vedoucí:	Ing. Radek Richtr, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Počítačová grafika
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Simulátor deskových her je určen k obecné simulaci her především ve fázi vývoje. Systém bude umět simulovat nehráčské tahy (např. údržba propriet na konci kola). Systém bude zaměřený na rychlé iterace herního designu, co nejpohodlnější a bezbariérové hraní z uživatelského hlediska, a s pohodlnou možností vyhodnocování a prohlížení již odehraných partií. Při vývoji simulátoru spolupracujte s experty v odvětví.

1) Proveďte rešerši existujících systémů pro hraní deskových her online, zaměřte se na vhodnost systému pro použití při vývoji deskové hry.
2) Navrhněte systém pro vývoj a testování deskových her.
3) Vytvořte prototyp simulátoru.
4) Vytvořte simulaci alespoň jedné netriviální deskové hry.
5) Systém vhodně otestujte.

Elektronicky schválil/a Ing. Radek Richtr, Ph.D. dne 21. prosince 2022 v Praze.

Bakalářská práce

SIMULÁTOR DESKOVÝCH HER

Adam Španěl

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Radek Richtr, Ph.D.
1. ledna 2023

Odkaz na tuto práci: Španěl Adam. *Simulátor deskových her.* Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Obsah

Seznam obrázků

Seznam výpisů kódu

Chtěl bych poděkovat Ing. Radku Richtrovi, Ph.D. za vedení a pomoc při psaní bakalářské práce. Díky patří pak i především Vlaadovi Chvátilovi, Petru Murmakovi a potažmo firmě CGE, že mi poskytli prostor pro vytvoření tohoto systému.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na jejímž základě se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 1. ledna 2023 .....

Abstrakt

V práci jsou zanalyzovány systémy pro hraní deskových her online. Závěr je, že žádný z nich není vhodný pro vývoj deskových her. Byl tedy navrhnut a implementován systém specificky určený pro vývoj a testování deskových her online, kde je kladen důraz na pohodlný vývoj adaptace deskové hry, rychlé a pohodlné hraní a data připravená k pohodlné analýze. V systému může herní designer vytvořit implementaci deskové hry, ve které může hru manuálně odehrát bez nutnosti tisku fyzického prototypu. Implementátor v další fázi vývoje může do této implementace přidat automatizaci, která výrazně urychlí a zpohodlní hraní hry. Kontrola pravidel navíc zajistí, že hráči nebudou hrát proti pravidlům. Výsledkem pak budou data odehraných her, která bude možné použít k další analýze a vývoji deskové hry.

Klíčová slova Deskové hry, Vývoj deskových her, Simulace deskových her, Online, Testování

Abstract

Analysis of existing systems for playing board games online was done. It was concluded none of them are suitable for development of board games. A system was designed and implemented, specifically for development and playing board games online. Strong emphasis is put on easy and frictionless implementation of board games, fast and convenient playing and data prepared for analysis. Game designer can create implementation of a board game without need of coding, and manually play a full game. In the next phase, a game implementator will add automatization to the implementation. That will greatly reduce the game time and ease of play. The result will be a dataset of finished games, ready for further analysis and development of the board game.

Keywords Board games, Board game development, Board game simulation, Online, Testing

Shrnutí

Motivace

Odbornou motivací bylo dodat systém, který usnadní vývoj deskových her, a nabídne nové pracovní postupy v dynamickém a mladém světě vývoje deskových her.

Cíl práce

Cílem práce je vytvořit webovou aplikaci podle požadavků zadavatele. Mezi výzvy specifické pro systém patří:

- Reprezentace herní situace
- Možnost vytvoření kostry implementace bez nutnosti kódování
- Předání této kostry implementace kodérovi, který pak může implementaci dodělat. Způsob, kterým bude kodér implementaci automatizovat.

Postup

Nejprve jsem zanalyzoval existující systémy pro hraní deskových her, a některé vyzkoušel při vývoji komerčních deskových her. Při návrhu jsem se soustředil na funkcionalitu, která mi při vývoji chyběla, a naopak jsem se nechal inspirovat funkcionalitou, která se osvědčila. Postup řešení jsem popsal v této práci.

Výsledky práce

Výsledek práce odpovídá zadání. Konkrétně tedy byla provedena rešerše a analýza existujících systémů pro vývoj a hraní deskových her online.

Dále byl navrhnut systém pro vývoj a testování deskových her online. Oproti existujícím systémům má především následující funkcionalitu:

- Rychlá automatizace deskové hry
- Možnost upravovat herní situaci během hry bez ohledu na automatizaci
- Pohodlné přehrávání odehraných partií

Dalším výsledkem je implementovaná aplikace.

Systém byl otestován implementací několika deskových her a jejich hraním různými hráči. První projekt, při jehož vývoji byl systém použit, je dnes již vydaná desková hra s desítkami tisíc prodaných kusů. Další projekt je vývoj rozšíření úspěšné deskové hry. Další dva projekty jsou osobního rázu, a jedná se o deskové hry v ranné fázi herního designu. Pátým projektem je klasická desková hra.

Úvod

Deskové hry v posledních třech desetiletích jdou jako průmysl nahoru. Moderní deskové hry podobné těm dnešním se ukazovaly již začátkem 20. století, ale éra moderních deskových her přišla paralelně s nástupem internetu, který usnadnil získávání informací a spoluhráčů, společně s rostoucím množstvím volného času lidí a rostoucí chutí do volného času investovat peníze.

Kolem roku 2000 začal rychlý růst průmyslu deskových her. Rostoucí poptávka vedla ke vzniku nových herních vydavatelů a studií [1]. Mnoho lidí navíc deskové hry vnímá jako alternativu k videohrám, protože volný čas nechtějí trávit u obrazovek počítačů bez fyzického kontaktu s ostatními hráči. Mnoho lidí u obrazovek tráví většinu svého pracovního života, což vysvětluje, že dávají přednost deskovým hrám před videohrami. Všechny tyto aspekty vedly k renesanci deskových her, která právě teď probíhá.

Design a vývoj moderních deskových her je tedy vcelku mladý a dynamicky se měnící průmysl. Na vývoj deskových her nejsou standardizované postupy, a je zde tedy mnoho prostoru pro inovace, experimenty i nové pracovní postupy.

Tématem, které hýbe deskoherním světem a polarizuje hráče, je hraní deskových her online. Je to velmi kontroverzní téma, a spousta hráčů hraní deskových her online kategoricky odmítá. Častým argumentem je právě to, že nechtějí trávit čas u obrazovek. Je to vlastně docela zajímavé ve smyslu, že deskové hry svého času nabraly na popularitě právě díky videohrám, a v dnešní době (i pod významným vlivem pandemie Covid-19) obešly celý kruh, a vracejí se do digitálního světa ze druhé strany.

Vývoj deskové hry je fascinující proces plný překvapení. K vydání moderní deskové hry v dnešní době vede mnoho netriviálních kroků. Herní design je disciplína na rozhraní matematických disciplín jako je statistika a pravděpodobnost, ale na druhé straně i psychologie a teorie her (i když paradoxně právě teorie her se v herním designu aplikuje spíše zřídka, a ve velmi specifických případech). Kromě herního designu je však potřeba vyřešit i komponentové řešení a samozřejmě veškerou grafiku hry. Oproti videohrám je potřeba věnovat péči i výrobě a distribuci, ale i fyzickým aspektům hry, jako třeba jak příjemné jsou komponenty na dotek a manipulaci. Omezení fyzického světa bez elektroniky přinášení při vývoji mnoho výzev.

Velkou součástí vývoje deskové hry je hraní deskové hry jako takové. K tomu existuje mnoho různých přístupů: buď výroba fyzických prototypů, a nebo právě použití digitálních technologií k hraní her bez nutnosti fyzického prototypu. Součástí vývoje deskové hry je téměř vždy iterativní design: tedy postup, kdy designer navrhne změny, implementuje je, vyzkouší je, a na základě vyzkoušení celý cyklus opakuje. Často právě implementace změn znamená úpravu fyzického prototypu, což znamená buď přepsání komponent tužkou nebo nálepkami, nebo celý nový výtisk. Vyzkoušení změn často vyžaduje fyzickou přítomnost dalších lidí, což může také znamenat komplikace.

Dnes již legendární herní designer a programátor Vlaada Chvátil své složitější hry designuje a vyvíjí pomocí svých vlastních neveřejných softwarových nástrojů, které kroky implementace a

vyzkoušení změn hodně zjednodušují.

Existuje množství systémů pro hraní deskových her online. Často jsou určeny buď na hraní již hotových her, nebo na rychlé prototypování bez nutnosti psaní kódu. Žádný z těchto systémů však není navrhnut primárně pro vývoj deskové hry, a chybí jim nástroje právě pro rychlý iterativní proces.

Tato práce popisuje systém, který duševně jde v krocích Vlaadových systémů, a který dále rozvíjí myšlenku návrhu a vývoje her v digitálním prostředí.

Kapitola 1
Rešerše

Na úvod definuji pojmy, které mají mnoho různých definic a interpretací, aby nemohlo dojít k záměně, a aby bylo jasné, o čem text pojednává. Pro hraní deskových her online je již v provozu několik existujících systémů. V této kapitole se zaměřím na vybrané příklady takových systémů a popíšu jejich možnosti a omezení. Dále popíšu příklady existujících populárních systémů pro hraní a testování deskových her online. Půjde konkrétně o Tabletop Simulator, Tabletopia a Board Game Arena. Dále ve stručnosti zmíním nativní implementaci deskové hry, a popíšu i jednotlivé fáze vývoje deskové hry.

1.1 Definice pojmů

Ještě než začneme, je potřeba přesně vymezit a definovat pojmy týkající se deskových her. V této práci se budeme ve skutečnosti zabývat pouze celkem malou částí deskových her, pro které je vhodné testování digitální formou. Nejedná se o standardní sadu definic, ale pro kontext práce jsou definice vhodné.

▶ **Definice 1.1.** *Herní komponenta je část deskové hry, která se používá při hře, a není dělitelná. Např. karta, kostka, figurka apod.*

▶ **Definice 1.2.** *Stav hry je konečné množství proměnných, k jejichž deklaraci nejsou použity herní komponenty, a hráči si tyto proměnné musí během hry typicky pamatovat*

▶ **Příklad 1.3.** Součástí stavu hry je například který hráč je na tahu, kolikáté je kolo, kterou akci hráč zvolil, množina efektů, které ještě musí hráč vyhodnotit apod., ale někdy třeba i počet hráčů, kteří hru hrají, a proměnné týkající se jednotlivých hráčů.

▶ **Definice 1.4.** *Herní situace je dvojice:*

- *Rozložení herních komponent na stole*
- *Stav hry*

▶ **Definice 1.5.** *Konec hry je herní situace, ve které žádný hráč nemůže provést žádný herní tah.*

▶ **Definice 1.6.** *Výsledek hry je funkce $R(s, p) \to a$, která každému konci hry a každému hráči přiřadí počet bodů.*

▶ **Definice 1.7.** *Pravidla hry je trojice:*

- *Vyhodnocovací funkce $E(s, d) \to s$. Tato funkce na základě herní situace s a rozhodnutí hráče d vrátí množinu herních situací společně s číslem p určujícím pravděpodobnost dané herní situace.*
- *Funkce $V(p, s)$, která každé herní situaci a každému hráči přiřadí část herní situace s, která je viditelná hráčem p (této části herní situace budeme říkat herní informace hráče p).*
- *Výsledek hry*

▶ Poznámka 1.8. Jsou hry, kde výsledek není určen počtem bodů, ale vítěznou podmínku mají definovanou jinak. V takovém případě výsledek hry je pro vítěze 1, a pro poraženého 0.

▶ **Definice 1.9.** *Herní tah (někdy též jen "tah") je změna herní situace na základě pravidel způsobená rozhodnutím hráče.*

▶ Poznámka 1.10. Součástí herního tahu může být nějaký náhodný krok (typicky hod kostkou a zamíchání balíku karet, ale nikoli lízání karty z balíku, protože to, jak je balík zamíchaný, je součástí skryté informace). Z tohoto důvodu vyhodnocovací funkce vrací množinu výsledných herních situací společně s jejich pravděpodobnostmi.

▶ **Definice 1.11.** *Veřejná informace je ta část herní situace, která je viditelná všem hráčům.*

▶ **Definice 1.12.** *Skrytá informace je ta část herní situace, která je viditelná alespoň jednomu hráči, ale alespoň jednomu hráči není viditelná.*

▶ **Definice 1.13.** *Neznámá informace je ta část herní situace, která není viditelná žádnému hráči.*

▶ **Příklad 1.14.** Ve hře Osadníci z Katanu jsou suroviny v rukou hráčů skrytá informace. Rozestavění komponent na plánu je veřejná informace. Pořadí akčních karet v balíku akčních karet je neznámá informace.

▶ Poznámka 1.15. Může nastat situace, že hráč získá skrytou nebo dokonce neznámou informaci pomocí dedukce ze své herní informace a předchozích herních tahů. V takovém případě se však stále jedná o skrytou, respektive neznámou informaci.

▶ **Definice 1.16.** *Moderní desková hra (dále jen desková hra) je dvojice:*

- *Množina herních komponent*
- *Pravidla hry*

▶ Poznámka 1.17. Naše definice deskové hry nezahrnuje část her, které se jinak typicky mezi deskové hry řadí. Patří mezi ně například postřehové, zručnostní, kreslící či předváděcí hry jako třeba Activity, Pictomania nebo Ubongo. U těchto her ale není vhodný digitální vývoj, protože u nich je věrné převedení do digitální formy výrazně náročnější. Za zmínku však stojí, že definice zahrnuje slovní hry, jako třeba Krycí Jména.

▶ Poznámka 1.18. Na rozdíl od klasických her, u kterých pravidla bývají upravována na základě místních tradic, jsou pravidla moderních deskových her vytvořená vývojářem hry, a modifikace pravidel nebo komponent samotnými hráči připadá v úvahu jen v krajních případech (např. když oficiální pravidla nepodchycují nějaké krajní herní situace).

▶ **Definice 1.19.** *Herní design je proces návrhu pravidel hry a herních komponent.*

▶ **Definice 1.20.** *Vývoj deskové hry je proces, během kterého se z návrhu pravidel hry a herních komponent vytvoří data připravená k výrobě. Součástí vývoje je grafický design, ilustrace, textace a sazba herních komponent a pravidel hry, balancování, testování, úprava pravidel a herních komponent na základě zpětné vazby testerů, případně návrh 3D modelů.*

▶ **Definice 1.21.** *Digitální adaptace deskové hry (dále jen "implementace hry") je počítačový program nebo soubor otevřený ve specializovaném počítačovém programu, který simuluje hraní deskové hry. Hráči mohou hrát střídavě na jednom zařízení, jeden hráč může hrát proti virtuální hráčům ovládaných AI, nebo mohou hráči hrát každý na svém zařízení.*

▶ **Definice 1.22.** *Digitální hraní deskové hry je proces použití digitální adaptace deskové hry. Analogové hraní deskové hry je proces hraní fyzické deskové hry (tj. bez použití digitální adaptace.)*

▶ **Definice 1.23.** *Implementátor je člověk, který vytváří digitální adaptaci deskové hry.*

▶ Poznámka 1.24. Pro potřeby této práce budeme předpokládat, že každá digitální adaptace má jednoho implementátora, i když v praxi za digitální adaptací často stojí týmy několika lidí.

▶ **Definice 1.25.** *Automatizace je součást implementace hry, která zajišťuje, že části herního tahu, na které nemá hráč vliv, a v kterých neprovádí žádná rozhodnutí, budou provedeny bez interakce uživatele (hráče).*

Nedílnou součástí automatizace je kontrola pravidel, která znemožní hráči provést nepovolený herní tah.

▶ **Definice 1.26.** *Plně automatizovaná implementace je implementace, ve které nelze provést herní tah proti pravidlům, a která dělá všechny části herního tahu, na které hráč nemá vliv, automaticky.*

▶ **Definice 1.27.** *Implementace typu sandbox (dále jen "sandbox") je implementace bez automatizace.*

1.2 Tabletop Simulator

Tabletop Simulator (zkr. TTS) [2] je v dnešní době nejpoužívanější systém pro hraní deskových her online, a dal by se dokonce označit prakticky za industry standard. U vznikajících her je TTS mod už téměř samozřejmostí. Systém má obchodní model jednorázového poplatku.

Systém se snaží o co nejvěrnější napodobení zážitku z analogového hraní deskové hry. Typické je, že digitální adaptace v TTS nekontrolují herní pravidla, ani nemají automatizaci.

1.2.1 Uživatelské rozhraní

Hra v TTS se odehrává vždy ve 3D scéně, ve které se jednotliví hráči mohou pohybovat, přesunovat herní komponenty, a dělat operace běžné při hraní deskové hry, jako je míchání balíku karet, hod kostkou apod.

K hernímu stolu se můžou připojit libovolní hráči. Tvůrce stolu může nastavit heslo, které je k připojení třeba zadat.

Ovládání TTS je založené na klávesových zkratkách, kde hráči používají myš k výběru herních komponent, a klávesové zkratky k transformaci či herním úkonům s danými komponenty.

TTS obsahuje fyzikální engine. Ten ale často spíš překáží v pohodlném hraní hry, a často je potřeba opravovat věci, které se kvůli fyzice rozbily. [3]

1.2.2 Postup vytváření implementace

TTS je navržen tak, aby bylo velmi jednoduché pro běžného uživatele vyrobit a sdílet hratelný prototyp hry. TTS funguje nad Steam Workshop, který používá pro uchovávání digitálních adaptací i všech assetů.

Definice komponent probíhá výhradně skrze grafické rozhraní TTS. TTS poskytuje množství předdefinovaných standardních komponent, jako jsou kostky, standardní hrací karty, figurky a

Obrázek 1.1 Hra Reversi v TTS. Hráči při hře musí sami kontrolovat pravidla hry, a pro otočení žetonu na druhou barvu musí daný žeton otočit pomocí klávesové zkratky F jako Flip

žetony. Kromě toho umožňuje nahrát .png obrázek, na základě kterého vytvoří žeton s odpovídajícím tvarem, kde průhlednost obrázku znamená absence materiálu.

Vytváření karet probíhá pomocí nahrání spritesheetu s kartami. Spritesheet je jeden grafický soubor, který obsahuje všechny karty v mřížce. Tento soubor je potřeba vytvořit třetí aplikací, nahrát do TTS, a nadefinovat počet karet na řádek a celkový počet karet.

Za zmínku stojí tzv. snap pointy: implementátor může na libovolné herní komponentě (typicky na herní desce) definovat body, na které se mají zarovnávat komponenty, které jsou na tuto komponentu přesunuty.

Celá digitální adaptace (které se říká *TTS mod*) se pak nahraje na Steam Workshop. Je však možné ji z TTS exportovat i jako .json a distribuovat libovolnými kanály.

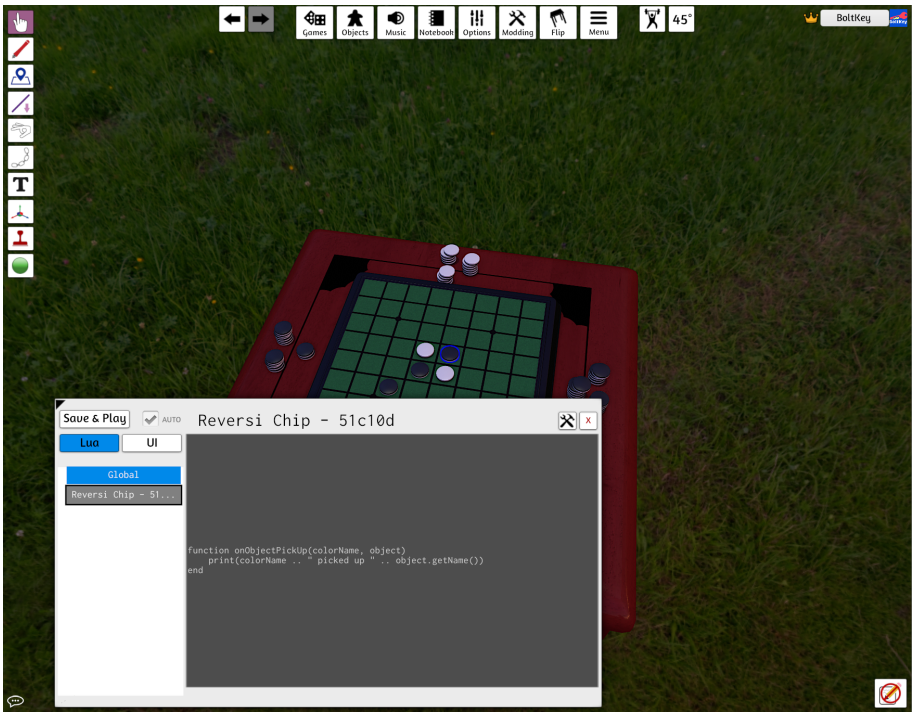
1.2.3 Skriptování pravidel a automatizace

TTS umožňuje skriptovat v jazyce Lua [4]. Každá herní komponenta má své GUID, na základě které se ze skriptu může transformovat či provádět herní operace. Navíc poskytuje rozhraní na zpracovávání uživatelských i jiných eventů. [5]

Herní situace je reprezentována standardně jen herními komponenty, což komplikuje kontrolování pravidel, protože u každé komponenty je známá její transformace vzhledem ke scéně, ale například už není možné snadno určit, na kterém poli herního plánu se komponenta nachází.

Pokud chce implementátor reprezentovat herní situaci v kódu, musí sám nějakým způsobem uchovávat data herní situace, ta aktualizovat na základě hráčských tahů, a podle své reprezentace pak aktualizovat transformaci herních komponent. To velmi komplikuje automatizaci her.

Velmi užitečnou možností je zpracování web requestů. Existují projekty, které mají celý backend, kde se vyhodnocují statistiky a herní tahy, a TTS používají jako frontend.



Obrázek 1.2 TTS obsahuje jednoduchý editor skriptů. Každý objekt může mít vlastní skripty, které se spouští na základě eventů. Například když hráč zdvihne tento žeton, do logu se zapíše, který hráč žeton zvedl, a název žetonu.

1.3 Tabletopia

Tabletopia je filozofií velmi podobná TTS, ale mezi zásadní rozdíly patří, že běží výhradně ve webovém prohlížeči, a používá freemium obchodní model.Dalším zásadním rozdílem je chybějící možnost skriptování, a všechny hry se tedy hrají v sandboxu. [6] Z tohoto důvodu nebudeme Tabletopii rozebírat podrobně.

1.3.1 Uživatelské rozhraní

Každý hráč může přesouvat a manipulovat s libovlnými komponentami. Hráč může komponentami rotovat, míchat balíky karet, nebo komponentu zamknout. Oproti TTS jsou tedy možnosti manipulace s komponentami omezené.

1.3.2 Omezení webového prohlížeče

Tabletopia je systém, který běží primárně uvnitř webového prohlížeče. Webové prohlížeče díky webGL nejsou výrazně omezené co se týče 3D grafiky, která je potřebná k vykreslení herní situace.

Obrázek 1.3 Uživatelské rozhraní Tabletopie se podobá TTS.

Webový prohlížeč tedy simulaci deskové hry příliš nelimituje, ale přes to je hra v prohlížeči o něco méně responzivní, než hra v TTS.

1.4 Board Game Arena

Board Game Arena (dále jen BGA) je systém, který se od ostatních zmíněných systémů zásadně liší vzhledem, účelem i filozofií. Je zaměřen mnohem více na soutěžní a turnajové hraní. Všichni hráči mají pro každou hru rating, který vychází z Elo systému, a systém má sofistikovaný systém sbírání statistik z her. [7]

Filozofie se liší především v tom, že se systém nesnaží o simulaci herního stolu (jako jiné systémy), ale o co nejpříjemnější hraní samotné hry. Navíc systém kontroluje všechna pravidla, a automaticky vyhodnocuje neherní tahy.

Tato jiná filozofie znamená výrazně vyšší vstupní bariéru pro implementaci deskové hry, ale mnohem lépe hratelný výsledek.

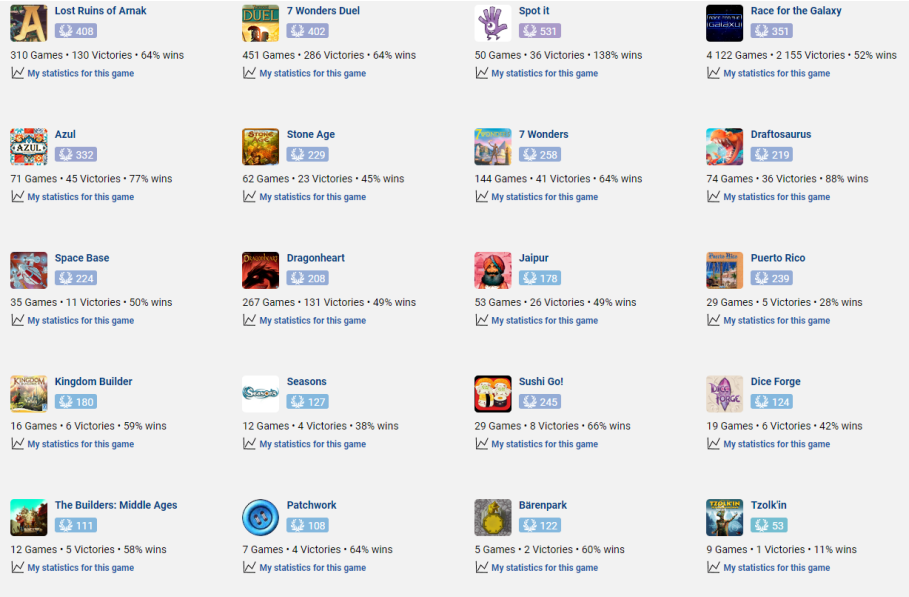
Systém je navržen, vyvíjen a udržován týmem amatérů. Komerční projekt z toho vznikl teprve postupně. Tato historie je znatelná napříč celým systémem, a mnoho věcí je navrženo a implementováno zkrátka na koleni.

1.4.1 Uživatelské rozhraní

Uživatelské rozhraní implementace je podrobně rozepsáno v BGA Studio Guidelines[8].

Mezi nejdůležitější pravidla patří:

- Veškeré herní tahy probíhají za použití klikání / tapování na komponenty
- Používání tooltipů, které se zobrazí po přejetí na herní komponentu
- Používání herního logu k informování hráčů o všech herních pohybech



Obrázek 1.4 Pro každou hru se každému uživateli počítá zvlášť rating a počet prohraných a vyhraných her, a má přístup k podrobným statistikám z partií.

1.4.2 Implementace hry

BGA vyžaduje k implementaci hry velmi nízkoúrovňové programování. Implementátor při implementaci píše SQL dotazy, PHP funkce, ze kterých vrací obecná data, a front end v klasickém JavaScriptovém kódu. Abstrakcí, které by usnadňovaly vývoj, je k dispozici velmi málo. Implementátor upravuje a nahrává .sql, .php a .js soubory na BGA servery pomocí FTP.

Vývoj pro BGA je tedy oproti ostatním systémům velmi náročný, ale dává implementátorovi mnohem větší volnost v úpravě uživatelského rozhraní pro potřeby konkrétní hry. Výsledkem tedy je mnohem pohodlnější hraní pro hráče. [9]

1.4.2.1 Databáze

Základním stavebním kamenem implementace je databáze. Každý založený stůl se hrou má založenou vlastní databázi, se kterou pak operuje kód hry. Implementátor musí definovat strukturu databáze pomocí DDL. Data v databázi musí jednoznačně popisovat herní situaci.

1.4.2.2 Backend

Systém poskytuje základní rozhraní pro řízení tahů hráčů: každý hráč může být aktivní nebo neaktivní. Toto rozdělení znamená, že aktivním hráčům se automaticky počítá čas hry.

BGA pracuje s konečným automatem, kde implementátor definuje herní stavy a přechody mezi nimi.

1.4.2.3 Frontend

Frontend se pro BGA píše v JavaScriptu bez frameworku. BGA poskytuje základní rozhraní pro přijímání zpráv ze serveru, ale zpracování zpráv je už na implementátorovi. Zásadním nedostatkem BGA je nutnost řešení synchronizace herního stavu na FE s herním stavem na BE.

Obrázek 1.5 Implementace hry Ztracený ostrov Arnak na BGA se typicky nesnaží simulovat herní stůl. Specifikem je například rychlý přehled každého hráče a počet jeho zdrojů zobrazený číslem (a nikoli graficky) nebo podrobný herní log.

Při refreshi stránky s herním stolem se načtou aktuální data z databáze, na základě kterých se vykreslí aktuální herní situace. Herní tahy se musí explicitně posílat ze serveru klientům, a klient si musí na základě tahu správně aktualizovat herní situaci.

1.5 Nativní implementace

Další možností, jak tvořit digitální adaptace deskových her, je vytvoření programu specializovaného na hraní té konkrétní hry. Je to samozřejmě nejpracnější možnost, ale zároveň výsledek je uživatelsky nejpříjemnější, protože implementátor má ještě větší volnost ve vytvoření uživatelského rozhraní a grafických efektů tvořených speciálně pro danou deskovou hru.

Tento postup je nejnáročnější, protože implementátor nemá vůbec žádné nástroje určené pro vytváření implementací deskových her. Výsledek je ale vždy uživatelsky příjemnější.

1.6 Běžný workflow vývoje deskové hry

Vývoj deskových her je komplexní proces, který má mnoho rovin, které mohou a nemusejí probíhat paralelně, a mohou a nemusí se navzájem ovlivňovat. Většina aspektů vývoje je však založená na iterativním procesu a prototypování.

Ilustrace pro hru jsou věc, která obvykle probíhá zcela nezávisle na ostatním vývoji, takže ty uvažovat nebudu.

Další roviny vývoje se dají rozdělit do 3 hlavních kategorií:

- Grafický design: jak bude hra vypadat, jak vypadá krabice, pravidla, karty, žetony.
- Návrh komponent: Výběr komponent pro hru, volba velikosti karet a ostatních komponent, způsob přemisťování a manipulace s komponenty
- Návrh herních mechanik: souvisí úzce s herním designem, a zahrnuje ladění herních komponent a pravidel. Je nejvíce náročný na testování, nejpracnější, a typicky vyžaduje nejvíce iterací.

Všechny tyto kategorie zahrnují testování a úpravy na základě zpětné vazby z odehraných her. Návrh komponent se dá testovat jedině v analogové hře, grafický design se dá testovat i v digitální hře, a herní mechaniky se dají testovat takřka rovnocenně při digitálním a analogovém hraní. Digitální hraní navíc při návrhu mechanik poskytuje mnoho výhod, jako je například možnost rychlejšího a pohodlnějšího odehrání hry, absence nutnosti hráčů se sejít fyzicky, a lepší možnosti automatického sbírání a vyhodnocování dat z her.

Kapitola 2
Analýza

S některými systémy mám bohatou osobní zkušenost s použitím při vývoji deskové hry. Podrobně rozeberu BGA, které pro vývoj je nejvhodnější, a popíšu jednotlivé problémy, které systém má. Dále popíšu výhody a nevýhody vývoje v TTS a Tabletopii. Na základě těchto zkušeností zkonstatuji, že na vývoj deskové hry žádný ze systémů není příliš vhodný.

2.1 Použití BGA pro vývoj

Systém BGA jsme ve spolupráci s vydavatelem použili při vývoji rozšíření pro hru Ztracený ostrov Arnak. Velkou výhodou byla široká hráčská základna, která se dříve vytvořila kolem adaptace základní hry Ztracený ostrov Arnak na BGA.

Po několika iteracích vývoje bylo evidentních mnoho nedostatků plynoucích z toho, že BGA nikdy nebylo navrženo pro vývoj her a rychlé iterace, ale bylo navrženo na vytváření adaptací již hotových her. Mezi tyto nedostatky patří zejména:

- Nová verze se aplikuje i na již rozehrané stoly. Při nasazování nových verzí implementace je potřeba vždy myslet na zpětnou kompatibilitu, a při výraznějších změnách znamená nasazení nové verze rozbití rozehraných stolů, které je nemožné dohrát. Mezi některými hráči je populární například takzvané asynchronní hraní, tedy formát, kdy jedna partie trvá i několik dní. Tyto partie tedy mohou být i o několik verzí pozadu oproti aktuální verzi, a ještě více to zesložiťuje zajištění zpětné kompatibility.
- Implementátor má přístup jen k těm datům, která explicitně shromažďuje, ale nemá přístup k datům celých odehraných partií.
- Reprezentace herní situace pomocí přímých dat uložených v databázi zesložiťuje přehrávání záznamů partií, komplikuje možnost vrácení tahu a vede k desynchronizaci dat na serveru a klientovi.

Kromě toho BGA obsahuje mnoho návrhových chyb a technických nedostatků, které znáročňují implementaci i již hotových her, u kterých se nečeká nutnost častých iterací. Mezi takové nedostatky a chyby patří:

- Nutnost řešit mnoho nízkoúrovňových implementačních detailů zpomaluje vývoj a přivádí na svět velké množství chyb, které často jsou náročné na debugování. Systémové chybové hlášky jsou často velmi málo informativní. Systém má nízkou úroveň abstrakce. Například pro úpravu herní situace je potřeba psát "natvrdo" dotazy v jazyce SQL nad databází hry.
- Pro každý herní stůl je vytvořena nová databáze, což jde proti tomu, k čemu by databáze měla sloužit (tj. rychlá práce s velkým množstvím dat).

- Na BGA je základní datovou strukturou řídící herní stav stavový automat. Na první pohled, a na velmi jednoduché hry, to může být vhodný způsob, jak reprezentovat herní stav. Implementátor definuje všechny možné stavy hry, přechody mezi nimi a možné herní akce v každém stavu. Tento systém ale vůbec neumožňuje implementaci podstavů. Podstav je vlastně stav, ze kterého se vždy chceme dostat do předchozího stavu. Protože podstav může mít i svůj podstav, stavový automat nemůže podchytit tuto funkcionalitu. Častý způsob, jak se tento nedostatek řeší, je pomocí ukládání zásobníku podstavů do databáze. To ale vede k nečekaným a těžko diagnostikovatelným, těžko replikovatelným a těžko opravitelným chybám, a velmi znáročňuje implementaci sofistikovanějších her.
- Jestliže vznikne chyba během přípravy hry, systém poskytne jen generickou chybovou hlášku bez žádných informací o tom, kde k chybě došlo. Jediný způsob, jak chybu diagnostikovat, je do různých míst v setup funkci umisťovat příkaz exit(), a takto triangulovat, kde k chybě dochází. Tento postup je předpotopní, a nutnost jeho použití nemá v moderním softwarovém nástroji co dělat.
- BGA má limit 15 MB grafických assetů na hru. Toto je velmi šibeniční limit. Je potřeba dělat mnoho kompromisů v kvalitě a rozlišení grafických materiálů, a je potřeba použití třetích aplikací pro kompresi dat. Co víc, kvůli špatné konfiguraci serveru se po vymazání konkrétního souboru stále počítá do tohoto limitu, takže je potřeba soubor, který chci smazat, nejprve nahradit prázdným souborem, a smazat až potom.
- V systému lze sbírat statistiky. Každá statistika je identifikovaná číslem, což velmi komplikuje párování statistik na jejich význam.
- Kompletní záznamy her je možné vyexportovat, ale je to velmi náročný a zdlouhavý proces, a je potřeba si záznamy her explicitně vyžádat od administrátorů BGA. Navíc, v záznamech her se ukládá obrovské množství redundantních informací: každý záznam hry je totiž exportován s kompletní HTML stránkou i skripty.
- Systém sice má funkcionalitu na vracení tahů, ale velké omezení je, že undo si pamatuje vždy jen jeden stav. Nemůžu se tedy postupně vracet vždy o jeden tah.
- Nutnost manuálně synchronizovat data klienta a serveru. Server může posílat notifikace klientům, ale na implementátorovi je, aby na základě notifikací správně aktualizovat data na klientovi. Implementátorova chyba pak znamená, že na klientovi jsou jiná data, než na serveru. To vede k mnoha duplikacím kódu, protože jeden herní tah znamená logiku pro aktualizaci databáze, ale i logiku pro aktualizaci herní situace na klientovi. Tento systém duálního vyhodnocování vede k mnoha bugům, které se velmi těžko odhalují, reprodukují a opravují.
- Dokumentace systému je velmi slabá, neaktuální a mnohdy zavádějící nebo dokonce nepravdivá.
- Není možné otevřít rozehranou produkční hru ve vývojovém prostředí. Toto extrémně zpomaluje diagnostiku chyb v implementaci. Na produkci navíc jsou chybové hlášky pouze generické, takže ani ty často nejsou moc užitečné. Je potřeba hádat, co mohlo chybu způsobit, a pokusit se nasimulovat podobnou situaci ve vývojářském prostředí.
- Velmi složité vytvoření custom herní situace ve vývojovém prostředí: je potřeba natvrdo upravovat data v databázi přes PHPMyAdmin. Je možné spouštět funkce na serveru přes herní chat, ale upravovat herní situaci tímto způsobem je neobyčejně únavné a zdlouhavé.
- Klient může posílat na server AJAX requesty společně se seznamem argumentů. Nelze ale poslat obecný string, a je potřeba ho zakódovat do Base64, a na serveru zase dekódovat. Tento postup je nutné použít například když jedním z argumentů je JSON. V logu pak nejsou lidsky čitelná data o tom, co se na server posílá za argumenty k requestům, což zpomaluje vývoj.

- Version control jede sice v základu na SVN, ale interface poskytnutý implementátorovi umožňuje pouze jednu větev.
- Klient používá k DOM manipulaci zastaralou knihovnu Dojo toolkit.
- Nutnost používání magických konstant. Všechny stavy hry je potřeba identifikovat pomocí čísel. Částečně se to dá řešit definováním konstanty pro každý stav, a v kódu místo čísel psát názvy konstant, ale pak například v databázi je jen číslo stavu, a podobně v chybové hlášce týkající se stavu hry je také jen číslo stavu. Pro zjištění, o jakém stavu se mluví, je potřeba najít příslušnou konstantu. Stejná situace je u herních statistik.

BGA navíc má mnoho dalších nedostatků, které nejsou specifické pro vývoj implementace hry:

- Soukromí: kdokoli může kdykoli o libovolném uživateli zjistit, co naposledy hrál a kdy byl naposledy online, a poslat mu zprávu. Co víc, můžu si kohokoli bez jeho souhlasu přidat do přátel, a od té chvíle dostanu upozornění, kdykoli se přihlásí do systému. To může být nepříjemné obzvlášť pro například implementátory her, kterým pak píšou soukromé zprávy všemožní uživatelé.
- BGA má notifikační systém, který ale nejde moc konfigurovat, a je pak zaplněn notifikacemi, které uživatele nemusí vůbec zajímat.
- UI je dost zastaralé a neintuitivní.
- Replay her jsou velmi uživatelsky nepřívětivé: když chce uživatel skočit na konkrétní tah, musí počkat, až se odehraní všechny tahy v rychlosti, kterou se zahrají během hry, a to může trvat až několik minut. Navíc není možné se v replay vracet zpět.

I přes mnohé nedostatky se BGA osvědčilo pro testování hry, a právě díky tomuto systému jsme mohli k testování přizvat testery z celého světa včetně nejsilnějších hráčů základní hry, kteří nám poskytli nenahraditelnou zpětnou vazbu k vyváženosti hry.

Udržování a aktualizace hry na BGA je však velmi zdlouhavý a únavný proces, což je pro svižný vývoj a nové iterace výraznou překážkou. Chybí možnost sandbox modu, kde by hráči měli možnost sami opravit případné chyby v automatizaci, a provádět herní tahy manuálně, což může zanést statisticky významnou chybu do statistik. Tyto nedostatky by nebylo složité navrhnout a implementovat v novém systému.

2.2 Použití TTS pro vývoj

Největší výhodou TTS je jeho vysoká popularita, takže není potřeba řešit vstupní bariéru pro hráče.

Vytvořit hratelný prototyp v TTS je velmi jednoduché a rychlé, a není k tomu třeba umět programovat.

Použití TTS pro vývoj je vhodné pouze ve velmi ranných fázích vývoje hry, kdy změny pravidel a komponent jsou velmi časté a významné. Rychlost a pohodlnost implementace je vykoupena nízkým pohodlím při hraní a dlouhou herní dobou. Navíc hráči musí podrobně znát pravidla každé nové iterace, takže je buď potřeba mít kvalitní a aktualizovaný soupis pravidel, nebo autor musí u partie být přítomen.

V konečných fázích, kdy jde o poslední ladění, je to nástroj celkem nepraktický, a často je už lepší vyrobit fyzický prototyp, na kterém se odehraje větší množství her.

Výrazným omezením TTS je prakticky nemožné sbírání statistik z her. Implementátor by si musel vyrobit backend, a na základě herních tahů posílat data přes fetch api. Další nevýhodou je nemožnost přehrát replay dohraných her, a vůbec absence seznamu odehraných her.

Pro vývoj deskové hry systém není vhodný, protože sbírání dat a statistik z her je velmi komplikované, a chybí možnost přehrání odehrané hry. Implementace automatizace je komplikovaná.

2.3 Použití Tabletopie pro vývoj

Systém je filozofií a základním ovládáním velmi blízký k TTS, a platí pro ni stejné věci, jako pro TTS. Navíc neumožňuje do hry přidávat skripty, takže pro vývoj je velmi nevhodná.

2.4 Shrnutí

Pro vývoj a online testování deskové hry je potřeba systém, ve kterém se hra hraje pohodlně a rychleji, než na fyzickém stole. Toho se docílí především možností implementaci automatizovat. Hry může urychlit i kontrolování pravidel, které jde ruku v ruce s automatizací. Plně automatizovaná implementace na TTS se hraje v celku pohodlně, ale vytvoření plně automatizované implementace v TTS je časově velmi náročné kvůli málo sofistikované reprezentaci herní situace. Hru navíc zdržuje snaha TTS o simulaci herního stolu, což při vývoji hry vůbec není potřeba. Nejpohodlnější na hraní je implementace na BGA, ale ta je velmi náročná a neobyčejně zdlouhavá a nepohodlná na vytvoření.

Dalším předpokladem je, aby v systému mohl implementátor provádět co nejrychlejší iterace. Na toto je vhodný TTS, kde je jednoduché implementaci upravit a nahrát novou verzi na Steam Workshop [10].

Nutností je i pohodlná možnost sbírat a vyhodnocovat data a statistiky z her, a shlédnout replay libovolné hry. BGA je v tomto ze zkoumaných systémů na tom nejlépe, ale její prvenství v tomto aspektu je dáno především tím, že ostatní systémy tuto funkcionalitu neposkytují vůbec. BGA má přehrávání partií velmi nepohodlné, a statistiky v celkem krkolomném formátu. Nedostupnost záznamů celých partií je také velmi omezující.

Všechny požadavky na systém žádný ze zmiňovaných existujících systémů uspokojivě neposkytuje. Je tedy na místě takový systém vytvořit.

Kapitola 3
Návrh

V této kapitole shrnu jednotlivé komponenty systému a životní cyklus implementace hry. Rozeberu požadavky a případy užití. Navrhnu doménový model systému a životní cyklus implementace hry v systému. Dále definuji pojmy specifické pro implementaci hry. Dále podrobně popíši reprezentaci herní situace, která je definovaná počáteční herní situací a seznamem herních tahů. Návrh zahrnuje také generátor karet, který je definován grafickými soubory a tabulkou s daty o kartách. Systém bude obsahovat grafické rozhraní pro implementaci sandboxu. Popíšu, jakým způsobem bude implementátor automatizovat implementaci. Navrhnu systém uživatelských účtů a popíšu architekturu systému jako celku. Navrhnu systém ratingu hráčů.

3.1 Přehled systému

Systém zahrnuje subsystémy pro podporu kroků při vývoji už od prvního prototypu, který bude moci herní designer vytvořit, a buď sám nebo se spoluhráči online hrát bez automatizace. Designer má k dispozici systém, kde může vytvořit balík karet, který bude definovaný grafickým souborem s vrstavmi, odkazem na tabulku s daty a pravidly generování, a volitelně soubory s ikonami, ilustracemi a fonty.

Dále k implementaci hry je API, které umožní implementátorovi přidat do hry automatizaci.

Po vytvoření implementace systém umožní implementaci zpřístupnit konkrétním uživatelům systému, kteří mohou vytvořit stůl s danou hrou, a ke stolu pozvat libovolné hráče, kteří mají implementaci zpřístupněnou.

Po vydání deskové hry bude implementace zveřejněna pro širší veřejnost. Součástí systému je metagaming, tj. sbírání bodů ELO, zisk achievementů, turnaje apod.

3.2 Požadavky

Požadavky systému jsou následující:

3.2.1 Funkční požadavky

- Systém umožní vytvoření sandboxu deskové hry bez nutnosti kódování
- Systém umožní předělání sandboxu do plně automatizované implementace pomocí kódu
- Systém umožní hraní plně automatizované implementace velkým množstvím hráčů
- Systém poskytne statistiky a data z odehraných her pro použití v designu deskové hry

3.2.2 Nefunkční požadavky

- Systém bude přístupný a použitelný skrze moderní webový prohlížeč

3.2.3 Případy užití

Případy užití popíšu společně s aktéry, protože každý aktér má svůj unikátní případ užití.

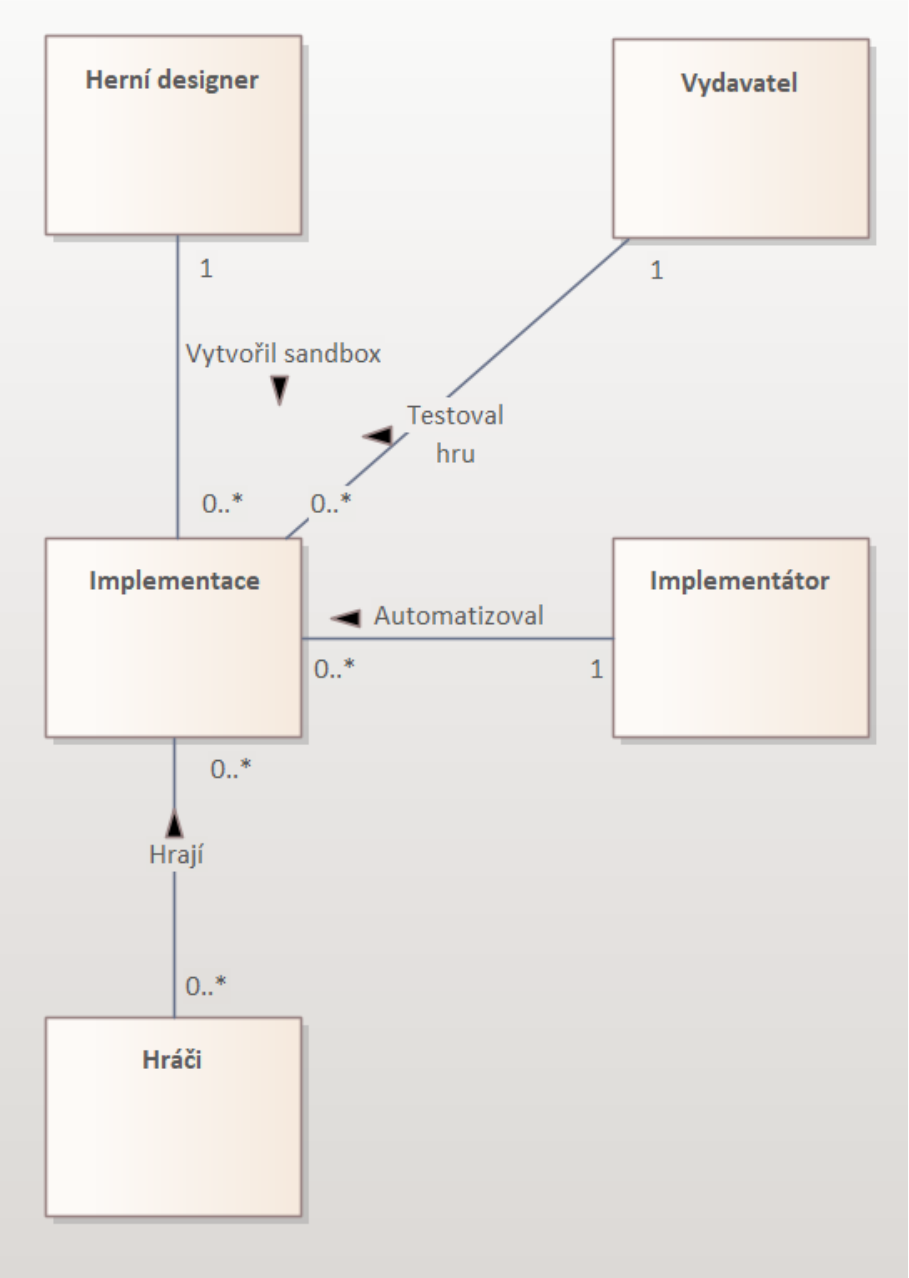
3.2.4 Aktéři

V systému budou figurovat tito aktéři s následujícími případy užití:

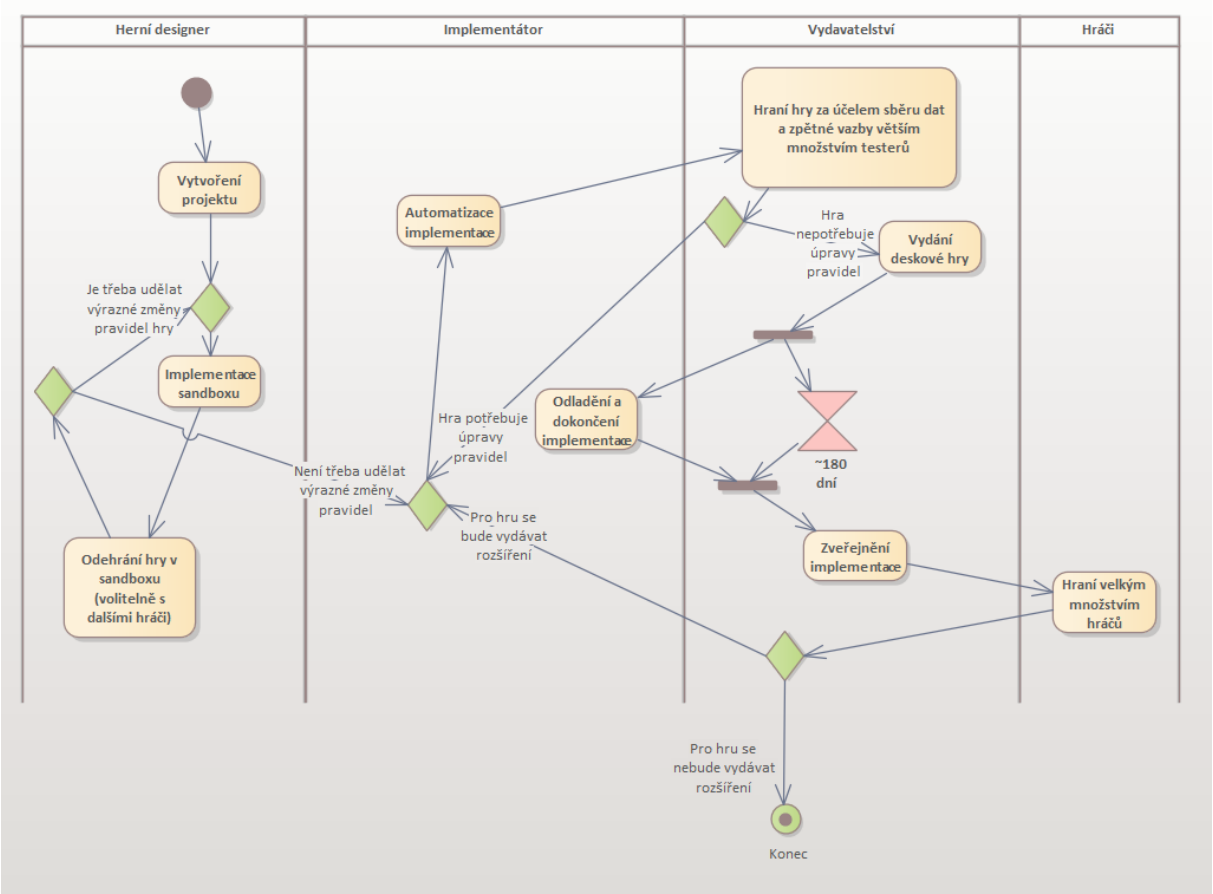
- Vydavatelství: Společnost, která se zabývá vývojem a vydáváním deskových her. Systém bude využívat na testování deskové hry, u které je hotová implementace s automatizací.
- Herní designer: autor deskové hry, který usiluje o to, aby jeho deskovou hru vydalo vydavatelství, nebo s vydavatelstvím již uzavřel smlouvu o vydání deskové hry. Systém bude používat pro vytvoření sandboxu své hry, který bude používat pro prvotní design.
- Implementátor: programátor, který ve spolupráci s designerem a vydavatelstvím bude přidávat do implementace automatizaci.
- Hráč: Koncový zákazník vydavatelství. Po zveřejnění může hru hrát v rámci systému.

3.3 Doménový model

Doménový model je zachycen v následujícím diagramu.


Herní designer
Vydavatel
1
1
Vytvořil sandbox
Testoval
hru
0..*
0..*
Implementace
Implementátor
Automatizoval
0..*
1
0..*
Hrají
0..*
Hráči

3.4 Životní cyklus implementace hry



Implementace začíná vytvořením sandboxu, pro který stačí definovat všechny herní komponenty, pro což není nutná znalost kódování. Sandbox vytváří herní designér, který může v sandboxu dělat výrazné iterace a změny pravidel. Ve chvíli, kdy je se základními obrysy herních mechanik herní designer spokojen, předá implementaci implementátorovi. Ten ze sandboxu vytvoří automatizovanou implementaci, ve které lze pohodlně odehrát větší množství her. Na implementátorovi pak je i aktualizování implementace na základě požadavků od herního designera a vydavatele. Po provedení velkého množství iterací eventuelně je desková hra připravena na vydání. Po vydání deskové hry se pak s odstupem přibližně půl roku zveřejní implementace pro širokou hráčskou veřejnost.

Pokud se vydavatelství rozhodne pro hru vyvíjet rozšíření, použije data z předchozích her pro další vývoj, a implementaci půjde použít pro vývoj rozšíření.

3.5 Reprezentace herní situace

Základním předpokladem hraní hry je vnitřní reprezentace herní situace. Existuje množství přístupů. Já jsem zvolil reprezentaci za pomoci seznamu herních tahů. Hlavní výhodou tohoto přístupu je, že umožňuje velmi přímočarou možnost implementace vracení tahů, protože není potřeba, aby systém měl záznamy o předchozích herních situacích - pro vrácení tahu stačí jednoduše vymazat poslední herní tah. Druhou výhodou je velmi jednoduchá implementace přehrávání her.

Herní situace bude v systému reprezentována dvěma entitami:

1. Počáteční herní situace
2. Seznam herních tahů

3.5.1 Počáteční herní situace

Počáteční herní situace musí obsahovat všechna data o herní situaci na začátku hry. Musí obsahovat následující data:

Herní komponenty obsahují data o všech herních komponentách. Rozlišují se herní komponenty hráčů v oblasti hráčů, a komponenty ve společné herní oblasti. Každá herní komponenta má následující příznaky:

- Jméno komponenty
- ID komponenty (unikátní u herního stolu)
- Nastavení komponenty: mapování názvu nastavení na string
- Potomky: mapování názvu pozice na seznam herních komponent, nebo dvourozměrný seznam herních komponent

Systém u každého stolu defaultně vytvoří pomocnou herní komponentu root, která samotná se nijak nezobrazuje, ale je určena k tomu, aby se do ní umisťovaly jiné herní komponenty.

▶ **Příklad 3.1.** V deskové hře Osadníci z Katanu by byla komponenta pro herní plán, která má název `herniPlan`, nastavení prázdné, a potomky pod názvy pozic `pole`, `cesty`, `osady`. Cesty a osady jsou na začátku hry prázdné (protože na herní plán je hráči přidávají až v průběhu hry), ale pole jsou v rámci přípravy připravena ještě přede hrou.

Každé herní pole má název `poleHernihoPlanu` a v nastavení příznak `zdroj`, který určuje, kterou surovinu dané pole produkuje. V potomcích by měl dva záznamy: `cislo`, což je seznam o jedné herní komponentě žetonu s číslem, a `zlodej`, což je seznam, který má v sobě položku herní komponenty zloděje v případě, že se jedná o pole s pouští.

Stav hry musí obsahovat všechna data stavu hry. Tato položka má dvě složky:

Herní stav (nebo též jen "stav") obsahuje název, zprávu pro hráče, a aktivního hráče (příp. seznam aktivních hráčů pro hry, kde může být aktivních hráčů více najednou), a volitelně další libovolná data. Obsahuje také volitelně podstav, což je také herní stav. Může takto být zanořeno neomezené množství herních stavů, což emuluje zásobník, který chybí například v BGA.

▶ **Definice 3.2.** *Aktivní stav je herní stav, který nemá žádný podstav.*

▶ Pozorování 3.3. Ve hře je vždy právě jeden aktivní stav.

▶ **Příklad 3.4.** Hráč ve hře Mars: Teraformace zahraje kartu, která mu umožní získat kartu, umístit jeden dílek oceánu na herní plán a poté získat 2 žetony rostlin. Hráč zahraje kartu a zaplatí za zahrání zdroje. Tím se hra dostane do stavu hry vyhodnocovaniKarty. Začne se vyhodnocovat efekt. Hráč si lízne kartu. Následně se může rozhodnout, kam umístí oceán. Stav vyhodnocovaniKarty si vytvoří seznam efektů, které ještě zbývá vyhodnotit, a vytvoří svůj nový podstav umistiOcean (který se tím stane aktivním stavem.) Po rozhodnutí hráče, kam oceán umístí, se podstav smaže, a aktivním stavem se stane znovu stav vyhodnocovaniKarty, který na základě seznamu zbývajících efektů dá hráči 2 žetony rostlin.

▶ Poznámka 3.5. Pojmy Herní stav a Stav hry jsou podobně znějící pojmy, ale mají různé (přesně definované) významy. Vycházejí z anglických pojmů "Gamestate" a "State of the game", kde rozdíl pojmů je lépe znatelný.

Globální proměnné obsahují data, která nejsou součástí herního stavu. Jsou to data, která nejsou spojená s konkrétním herním stavem, a mění se nezávisle na herním stavu.

▶ **Příklad 3.6.** Globální proměnné jsou například

- Číslo kola hry
- Barva začínajícího hráče

Seznam hráčů obsahuje data o hráčích. U každého hráče musíme evidovat jeho barvu, volitelně herní komponentu představující jeho herní oblast a případně libovolná další data potřebná pro hru.

Pool komponent obsahuje mapování názvu komponenty na data komponenty. U každé komponenty musíme znát typ komponenty (karta, žeton nebo deska), její šířku, výšku, seznam proměnných, které hráč může nastavit, a URL obrázku, který má komponentu reprezentovat.

U každé položky dat komponenty můžeme definovat kondicionální data, tedy data, která závisí na nastavení komponenty jako seznam podmínek společně s výsledkem, který se má uplatnit v případě, že podmínka je splněna.

Každá komponenta má také definované pozice svých potomků, a to následujícím způsobem: pozice je mapa názvů pozice na popis pozice. Popis pozice obsahuje parametry, zda na jedné pozici může být více komponent (parametr `multi`), jaké mezi komponentami mají být mezery (parametr `gapSize`), v jakém úhlu se má více komponent vykreslovat (parametr `multiDir`), jakou konstantou se má násobit velikost komponenty na dané pozici (parametr `scaleMult`), a seznam souřadnic.

▶ **Příklad 3.7.** Ve hře Osadníci z Katanu je komponenta s názvem Settlement. Ta má nastavení "level" s hodnotou "village" a "town" a color s hodnotami "yellow", "red", "green", "blue". Na základě těchto dvou proměnných se podle definovaných podmínek u obrázku vybere jeden z osmi obrázků k zobrazení.

Nastavení hry obsahuje seznam nastavení a herních variant, která byla zvolena při založení stolu zakladatelem stolu. Jedná se o mapování názvu nastavení na hodnotu.

Seznam kontejnerů je množina kontejnerů, které mají každý svůj název, seznam herních komponent, které obsahují, a nastavení obsahující způsob vybírání herních komponent (náhodně, zásobník nebo fronta)

Data balíků karet obsahuje data potřebná k vygenerování karty

3.5.2 Seznam herních tahů

Seznam herních tahů je druhou důležitou složkou popisující konkrétní herní situaci. Každý herní tah představuje nějakou změnu herní situace. Když postupně aplikujeme na startovní herní situaci všechny herní tahy, dostaneme se do herní situace, kterou reprezentujeme.

▶ **Definice 3.8.** *Herní pohyb je atomická instrukce, která nějak upravuje herní situaci. Existují následující defaultní herní pohyby:*

component move *přesune komponentu podle ID na komponentu s jiným ID na pozici s názvem a indexem.*

component create *vytvoří novou komponentu s daným názvem, nastavením, pozicí a ID komponenty, která má být rodičem.*

component reveal *je podobný jako component create, ale navíc obsahuje informaci o původu komponenty, tedy z jakého kontejneru byla získána.*

move to bank *říká, že komponenta se přesunula do daného kontejneru.*

roll die *říká, že byla hozena kostka, společně s výsledkem hodu*

set variable *změní hodnotu daného nastavení dané herní komponenty na danou novou hodnotu*

set gamestate *nastaví herní stav na novou hodnotu*

add substate *přidá k aktivnímu stavu nový podstav*

resolve curr state *říká, že aktivní stav již byl vyhodnocen, a může se tedy smazat (tím se stane z nadstavu aktivní stav)*

set global *nastaví globální hodnotu na novou hodnotu*

set gamestate variable *nastaví danou proměnnou aktivního stavu na novou hodnotu*

set player attr *nastaví danou proměnnou daného hráče na novou hodnotu*

▶ **Definice 3.9.** *Nechť H je herní situace. Aktuální herní situace H je počáteční herní situace H po aplikování všech herních pohybů H.*

▶ Poznámka 3.10. Ve hrách je běžné, že probíhají herní pohyby nezávisle na rozhondutí hráčů (např. ve hře Osadníci z Katanu hráč dostane suroviny na základě hodu kostkami aktivním hráčem i mimo svůj tah. Dalším příkladem je doplňování karet v nabídce ve hře Splendor.)

Takové herní pohyby však budeme považovat za součást herního tahu, který jim předchází.

▶ **Definice 3.11.** *Herní tah reprezentuje jedno nějaké rozhodnutí hráče. Obsahuje barvu hráče, který tah provedl; uspořádaný seznam herních pohybů; časovou značku, kdy byl tah proveden.*

▶ **Příklad 3.12.** Ve hře Osadníci z Katanu se hráč rozhodne postavit město. To je jeden herní tah. Ten se ale skládá z několika herních pohybů: jeden herní pohyb je změna nastavení "level" dané osady z "vesnice" na "město". Kromě toho také jsou 3 herní pohyby na přesun karty kamene do balíku kamenů, a 2 herní pohyby na přesun karty obilí do balíku obilí. Tento herní tah tedy obsahuje 6 herních pohybů.

▶ Poznámka 3.13. Hráč může provést více herních tahů za sebou, aniž by ostatní hráči zahráli jediný tah. Naše definice herního tahu tedy neodpovídá běžnému intuitivnímu chápání tahu hráče v deskové hře.

Tento přístup má (oproti přímé reprezentaci, kterou používá například BGA) několik výhod a nevýhod.

3.5.2.1 Výhody

- Replay her je jednoduchý na implementaci: stačí, aby uživatel mohl měnit parametr, který říká, kolikátý tah chce zobrazit, a pak stačí přehrát jen daný počet herních tahů a zobrazit výsledek.
- Vracení tahu je jednoduché na implentaci. Když se uživatel rozhodne vrátit tah, stačí, aby se z herní situace odebral poslední herní tah. V takovém případě je potřeba, aby všichni klienti znovu přehráli celou herní situaci až na poslední tah.
- Vzniká méně nekonzistence mezi herní situací na klientovi a na serveru, protože aktuální herní situaci musíme získat jen na klientovi, a nemusíme ji získat na serveru.

3.5.2.2 Nevýhody

- Teoreticky při velmi dlouhých hrách může odehrání všech tahů znamenat delší dobu načtení stolu
- Je složitější "natvrdo" změnit herní situaci v databázi

3.6 Generátor karet

Součástí mnoha deskových her je balík karet. Součástí vývoje je častá iterace hodnot na kartách. Generátor karet by měl co nejvíce zjednodušit generování karet.

3.6.1 Uživatelské rozhraní

Uživatel nahraje přes webové rozhraní soubory potřebné ke generování karet. Po vygenerování karet se karty zobrazí v okně prohlížeče v tabulce. Pomocí posuvníků má pak možnost určit velikost karet a počet karet v řádku.

Uživatel má pak možnost aktualizovat karty po zadání nových hodnot do tabulky.

3.6.2 Definice balíku karet

Definice balíku karet se skládá ze 3 komponent: grafického souboru s vrstvami, tabulky s daty karet, a libovolného množství grafických souborů, které jsou potřeba k vygenerávní karet.

3.6.2.1 Grafický soubor s vrstvami

Základem karty je grafický soubor s vrstvami. Z grafického souboru se naimportuje rozměr karty, a u každé vrstvy její umístění, rozměry a obrázek, který se ve vrstvě nachází. S dalšími možnostmi a parametry vrstev systém nepracuje.

3.6.2.2 Tabulka s daty

V tabulce s daty jsou definované karty. Tabulka má dva listy, kde v jednom jsou data karet, a ve druhém pravidla, jak se data karet mají využít při generování.

3.6.2.3 List na data

Na listu pro definici dat jsou definované všechny atributy všech karet. V prvním řádku jsou názvy sloupců a v ostatních řádcích jsou data samotných karet. Jeden ze sloupců musí mít název `ID`, který identifikuje kartu. Ostatní sloupce mohou mít libovolné jiné unikátní názvy. Často se hodí, aby jedna karta měla víc záznamů v jednom sloupci. Sloupec `ID` je řídící ve smyslu, že řádkem, který má neprázdný sloupec `ID` začíná vždy definice nové karty. Jestliže řádek má prázdný sloupec `ID`, všechny ostatní sloupce se považují za součást poslední karty.

Data karty jsou ve formě stringů, a na stylování buňky se nebere ohled. Jestliže buňka obsahuje formuli, bere se v potaz pouze výsledek formule.

3.6.2.4 List na pravidla generování

Tento list udává relaci mezi listem s daty a grafickým souborem. V prvním řádku jsou názvy sloupců a v ostatních řádcích jsou definovaná jednotlivá pravidla pro vykreslování. Názvy sloupců jsou `ID`, `Layer`, `Rule`, `Arg` a `Arg2`.

Sloupce listu na definici pravidel vykreslování mají následující význam:

- `ID` je libovolný unikátní string identifikující dané pravidlo.
- `Layer` obsahuje název vrstvy, na kterou se pravidlo aplikuje.
- `Rule` obsahuje název pravidla, které se má aplikovat. Pravidla jsou následující:
  - `layer_visible` říká, že daná vrstva má být viditelná.
  - `set_innnerHTML` do vrsvty přidá HTML, které je v `Arg`
  - `css` přidá nové CSS pravidlo, které se aplikuje na vrstvu. Pravidlo, které se má změnit, je v `Arg`, a hodnota pravidla je v `Arg2`
- `Arg` obsahuje první argument k pravidlu.
- `Arg2` obsahuje druhý argument k pravidlu.

Pro každou kartu je vygenerována sada pravidel, podle kterých se vykresluje.

Každá hodnota použitá v tabulce může obsahovat konstrukt `#(nazev_sloupce)`. Pro každou kartu se tento konstrukt nahradí hodnotou sloupce se shodným názvem. Jestliže v daném sloupci karta má hodnot více, pravidel se také vygeneruje více.

Je zároveň možné u karty definovat obecný skript, který libovolně upraví jednotlivé vrstvy a to, jak karta vypadá.

3.7 GUI pro implementaci sandboxu

Systém musí být použitelný herním designerem k vytvoření sandboxu bez nutnosti kódování. Pro sandbox stačí, aby uživatel měl možnost vytvořit seznam herních komponent a asociovat ho se hrou, a aby mohl vytvořit počáteční herní situaci.

Toho docílí pomocí grafického editoru. Editor bude mít následující možnosti:

3.7.1 Definice komponent

Editor umožní definovat komponenty následujícím způsobem:

- Vytvořit herní komponentu: uživatel může vytvořit novou herní komponentu tím, že vyplní název komponenty

- Upravit herní komponentu: všechny herní komponenty budou zobrazené v seznamu. Uživatel může rozkliknout herní komponentu a upravit její data. Data se zobrazí jako seznam s textovými poli. Uživatel může rozhodnout, že daný parametr bude s podmínkou. V takovém případě se místo textového pole zobrazí seznam, kde každá položka bude mít 2 části: dropdown menu s výběrem parametru, a textové pole s cílovou hodnotou.
- Speciální případ úpravy dat herní komponenty je definice pozic. Uživatel může vytvořit nový seznam pozic s daným jménem, a může určit parametry pozice, tedy parametry `multi`, `gapSize`, `multiDir` a `scaleMult`. Zobrazí se komponenta s rozměry a obrázkem podle definice komponenty, a uživatel může klikáním určit souřadnice jednotlivých pozic.

3.7.2 Definice startovní pozice

Editor také umožní vytvořit defaultní startovní pozici následujícím způsobem:

- Uživatel vybere pozici a vytvoří herní komponentu, kterou chce na danou pozici umístit. Následně může upravit nastavení umístěné herní komponenty.
- Tento postup opakuje, dokud nejsou umístěné všechny potřebné herní komponenty.

3.8 Automatizace

Automatizace je proces předělání sandboxu do automatizované implementace. Automatizace bude v systému realizována tak, že k hráčským tahům budou přidány herní pohyby na základě těchto proměnných:

- Herní sitauce
- Prováděný herní pohyb
- Barva hráče, který tah provádí

Úkolem implementátora tedy bude vytvořit funkci `evalGameMove`, která na základě těchto tří proměnných k hernímu pohybu hráče přidá další herní pohyby, a vyhodí vyjímku v případě, že herní pohyb není podle pravidel s vysvětlením, které pravidlo hráč porušuje.

Kromě toho může implementátor vytvořit funkci `evalClick`, která vytvoří herní pohyby na základě těchto proměnných:

- Herní sitauce
- Komponenta, na kterou uživatel klikl
- Barva hráče, který klik provedl

Rozdělení na tyto dvě funkce je z důvodu, že kliknutím na komponentu nevzniká herní pohyb sám o sobě. Další uživatelských eventy (např. stisk klávesy) nebudeme uvažovat, aby implementace hry byla vždy co nejintuitivnější, a aby k hraní stačil jednoduchý klik. (Tento aspekt je důležitý například pro zajištění kompatibility s mobily a tablety).

Implementátor bude kromě již zmíněných proměnných mít k dispozici ve funkcích objekt `eventManager`, který poskytuje rozhraní k přidávání herních pohybů.

3.8.1 Vyhodnocování automatizace

Jsou dvě možnosti, kde se budou funkce `evalGameMove` a `evalClick` vyhodnocovat: na serveru, a nebo na klientovi, který daný event vyvolal.

3.8.1.1 Vyhodnocování na serveru

Vyhodnocování na serveru je konvenčnější způsob, a využívá ho například i BGA. Má ale několik nevýhod v naší reprezentaci situace. Hlavní nevýhodou je, že při každém novém tahu by bylo potřeba "přehrát" celou hru, aby se získaly informace o aktuální herní situaci, což by znamenalo velké nároky na výkon serveru. Byla by možnost na serveru mít cache s aktuální herní situací, ale tím dojde k redundanci dat a připravíme se o výhody, které nám naše reprezentace herní situace poskytuje. Navíc, na klientovi je potřeba taky držet aktuální herní situaci, takže by došlo k duplikaci dat a větší šanci k desynchronizaci mezi klientem a serverem.

Výhodou tohoto přístupu je, že máme velmi dobrou kontrolu nad tajnými, náhodnými a skrytými informacemi, a je jednodušší znepřístupnit hackerovi informace o herní situaci, ke kterým jako hráč nemá mít přístup.

3.8.1.2 Vyhodnocování na klientovi

V tomto přístupu vyhodnocujeme obě funkce na klientovi. Hlavní výhodou je, že na klientovi máme aktuální herní situaci, a je přímočařejší nad ní vyhodnocovat funkce.

Velkou nevýhodou je ale těžší zabránění podvádění. Případný hacker může na server poslat libovolné herní pohyby, které přidá do herního pohybu, který na server posílá. Některé takové pohyby by byly obtížné na detekci - například hacker může určovat, kterou kartu ze zamíchaného balíčku si lízne.

3.8.1.3 Kombinace

Každý ze zvažovaných přístupů má výhody a nevýhody. Jako řešení zvolím kombinaci těchto přístupů, která kombinuje jejich jednotlivé výhody.

Funkce `evalGameMove` a `evalClick` se budou vyhodnocovat na klientovi. Klient ale nebude moct v herních pohybech zahrnout neznámé a skryté informace daného hráče, konkrétně `component reveal` a `roll die`. Tyto informace doplní server. Když na server přijde herní pohyb `component reveal`, server ze všech dosavadních herních pohybů, které se týkaly kontejneru, ze kterého se komponenta má odhalit rekonstruuje aktuální stav kontejneru, a do herního pohybu před odesláním všem klientům dodá informaci o odhalené komponentě. Podobně do herního pohybu `roll die` přidá náhodně stěnu kostky, která padla.

Po této úpravě herních pohybů server uloží nový herní tah na konec seznamu dosavadních herních tahů, a nový herní tah včetně upravených herních pohybů pošle všem připojený klientům, kteří nový tah vyhodnotí ve své herní situaci.

3.8.2 Herní log

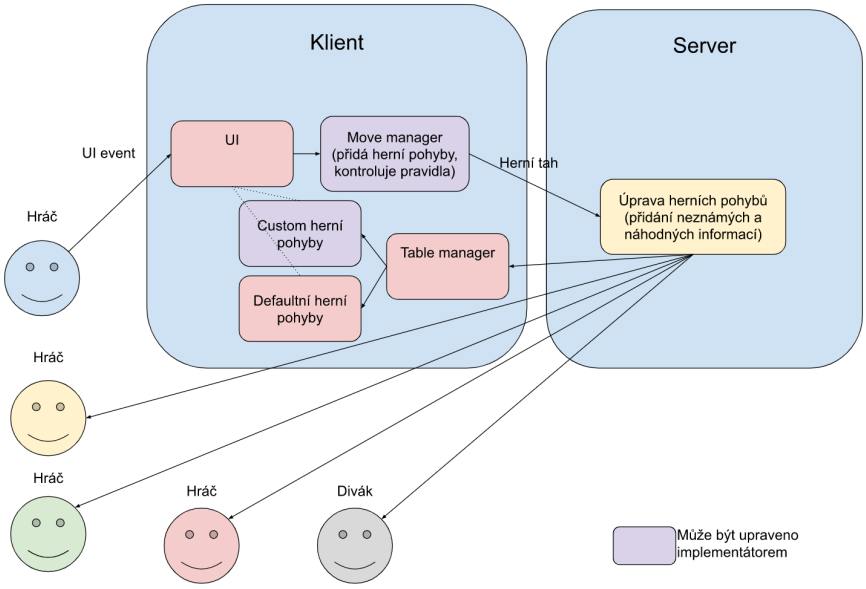
Herní log je záznam hry v textové formě. Každý herní pohyb může mít svoji textovou zprávu v parametru `logMessage`. Tato zpráva může být parametrická: text v dvojitých závorkách bude nahrazen hodnotou stejného jména z herního pohybu.

Kromě toho v logové zprávě může být obrázek v textu. Když je ve zprávě text v hranatých závorkách, a zároveň je v seznamu komponent definovaná příslušná ikonka, bude text nahrazen danou ikonkou.

V parametru herního pohybu `logHover` jsou data herní komponenty, která se má zobrazit v případě, že uživatel najede myší na zprávu v herním logu.

V parametru herního pohybu `logStyle` je objekt s CSS vlastnostmi, který se aplikuje na danou zprávu v logu. Tímto způsobem je například možné zvýraznit některé důležité herní tahy, nebo naopak znevýraznit zprávy doplňkového informativního charakteru.

Občas se hodí přidat zprávu do logu i když se neváže ke konkrétnímu hernímu pohybu (například na vysvětlení, proč se nějaký herní pohyb neprovedl, nebo proč se provedl nějaký



Obrázek 3.1 Vývojový diagram herního tahu

netradiční herní pohyb). V takovém případě stačí vytvořit herní pohyb bez definice typu pohybu, ale s definovanou `logMessage`.

3.8.3 Edit mode

Edit mode je způsob, jak měnit herní situaci bez ohledu na automatizaci. Během hraní hry v systému je možné každý tah udělat v režimu automatizace, nebo zapnout Edit mode. Když uživatel provede tah se zapnutým Edit mode, na server se pošle herní tah s tím jediným herním pohybem, který uživatel provedl, a tah se nebude prohánět funkcemi na automatizaci.

3.8.4 Undo

Undo je speciální druh eventu. Hráč může kliknout na Undo. Pokud tak udělá v Edit mode, znamená to, že chce vrátit poslední zahraný herní tah. Z databáze se odstraní poslední odehraný tah, a všem připojeným klientům se pošle informace o tom, že hráč udělal undo přes edit mode, a odstraní ze své herní situace poslední tah.

Jestliže však hráč provede undo mimo edit mode, logika bude trochu sofistikovanější. Pokud poslední herní tah má příznak “no undo past”, neprovede se žádná akce, a uživateli, který se o undo pokusil, se zobrazí chybová hláška o nemožnosti undo. Jinak se odstraní poslední herní tah. Pokud některý z herních tahů (kromě posledního tahu) za posledním tahem “no undo past” (včetně tahu “no undo past”) má příznak “undo breakpoint”, odstraní se všechny herní tahy za posledním tahem s příznakem “undo breakpoint”.

3.9 Systém uživatelských účtů

Důležitou součástí systému jsou uživatelské účty, a různé role, které mohou být uživatelským účtům přiděleny.

3.9.1 Vytvoření uživatelského účtu

Standardním způsobem bude možné vytvořit účet zadáním uživatelského jména, hesla a emailu. Každý nově vytvořený účet má status "external player". Tedy může u projektů se statusem Public a Unlisted vytváře stoly, přidat se ke stolu a provádět tahy.

3.10 Projekt jako entita v systému

V dosavadních sekcích bylo popsáno, jak bude fungovat jednotlivá partie a automatizace jednotlivé hry. Dále popíšeme systém na vyšší úrovni z pohledu výběru projektů a vytvoření stolu. Každý projekt v systému (tedy každá implementace hry) u sebe má následující informace:

- Název projektu
- Slug projektu (tj. jak se bude zobrazovat v URL)
- Seznam balíků karet, které se v implementaci používají
- Nastavení hry, která lze zvolit při zakládání stolu
- Popis hry
- Status viditelnosti, tedy jednu hodnotu z následujících
  - Public: Libovolný uživatel projekt vidí a může vytvářet nové stoly
  - Unlisted: Projekt se nezobrazí v seznamu projektů, ale jinak je stejný jako Public.
  - Testing: Pouze uživatelé, kteří mají roli "internal tester", vidí projekt a mohou vytvářet nové stoly
  - Private: Projekt vidí pouze administrátoři a uživatelé, kteří mají přístup explicitně povolen

3.10.1 Stránka projektu

Stránka projektu obsahuje popis hry, umožní připojit se k viditelnému lobby, umožní zobrazit a filtrovat odehrané partie a umožní vytvořit nové lobby.

3.10.2 Založení lobby, založení stolu, přidání se ke stolu

Uživatel, který má oprávnění k vytvoření stolu projektu, může vytvořit lobby. Vytvořené lobby má status viditelnosti "unlisted", tedy do něj může vejít pouze uživatel, který lobby vytvořil, a každý uživatel, který má odkaz. Uživatel může změnit viditelnost na "visible", čímž se lobby zobrazí v seznamu lobby na stránce projektu. Ostatní uživatelé se mohou připojit do lobby na stránce lobby po kliknutí na "join game".

Zakladatel lobby může v lobby navolit nastavení hry a spustit hru. U každé hry může nastavit, zda se spustí v režimu "sandbox" nebo "normal", a viditelnost stolu ostatními uživateli. Před spuštěním systém přidělí každému hráči barvu. Spuštění hry přidá do databáze nový rozehraný stůl, a všechny uživatele, kteří jsou v lobby, ke stolu automaticky připojí.

Jestliže je hra v režimu "sandbox", libovolný hráč, co se připojí ke stolu, může používat edit mode a hrát tahy za libovolnou barvu hráče. V režimu "normal" tyto dvě věci dělat nemůže, a všechny tahy uživatele musí být za tu barvu, která mu byla přidělena, bez použití edit mode.

3.11 Metagaming

Uživatelé mají možnosti komunikace s ostatními uživateli a soutěžení i mimo jednotlivé partie. Základem této metasoutěže je Elo rating.

3.11.1 Elo rating

Elo rating je způsob hodnocení hráčů, hojně používaný v závodním Šachu [11], na určení síly hráče. Základní charakteristikou je, že Elo rating bere v potaz Elo rating soupeřů. Tedy hráč, co vyhraje proti hráči s výrazně větším Elo ratingem, získá více Elo ratingu, než kdyby vyhrál proti hráči s nižším Elo ratingem.

Systém vytvoříme na základě rovnice pro výpočet Elo ratingu:

$$r'(p) = r(p) + K(S(p) - \frac{e^{r(p)/400}}{e^{r(p)/400} + e^{r(o)/400}})$$

kde $r'(p)$ je nový rating hráče p, $r(p)$ je aktuální rating hráče p, $S(p)$ je výsledek hry pro hráče P (od 0 do 1). K faktor je proměnná, která je v dané partii pro všechny hráče a výpočty stejná. Určuje, jak významná partie je pro rating.

V systému použijeme obdobu takového ratingu. Pro potřeby systému ale musíme mít na paměti rozdíly oproti šachu, kvůli kterým je potřeba výpočet ratingu upravit. Rozdíly postupně rozebereme:

3.11.1.1 Variabilní počet hráčů u jednotlivých partií

Některé partie se budou hrát ve dvou hráčích. Potřebujeme systém ratingu ale zobecnit tak, aby fungoval i ve hrách více hráčů. Při výpočtu nového ratingu budeme porovnávat všechny jednotlivé hráče mezi sebou. Jestliže hráč skončil v konečném pořadí nad jiným hráčem, budeme počítat, že nad ním zvítězil.

3.11.1.2 Variabilní průměrná délka partie u jednotlivých her

Aby byly ratingy srovnatelné mezi jednotlivými hrami, je potřeba vzít v potaz průměrnou délku partie. Jsou hry, u kterých partie může trvat několik hodin, a je na místě, aby u takových her jedna partie měla větší váhu, než u her s průměrnou dobou partie v řádu jednotek minut.

K faktor bude přímo úměrný průměrné délce hry. Průměrnou délku hry na začátku určí implementátor, a po odehrání statisticky významného počtu partií se hodnota nahradí geometrickým průměrem délky všech odehraných partií.

3.11.1.3 Větší vliv náhody u některých her

Jsou hry, kde o výsledku výrazným způsobem rozhoduje náhoda. Implementátor může definovat, jak velký vliv náhoda ve hře má. Větší náhodnost hry znamená, že očekávaný výsledek hry se více přiblíží k hodnotě 0.5.

3.11.1.4 Různé herní varianty v rámci jedné hry

Jsou případy, kdy různé herní varianty jsou natolik odlišné, že je na místě počítat pro ně elo zvlášť. Implementátor může definovat, že pro určité nastavení hry se má počítat separátní Elo. Implementátor definuje libovolný počet variant s popisem a funkcí, která na základě nastavení vrátí, jestli se daná konfigurace počítá do daného Ela. Každý hráč pak u hry bude mít Elo pro každou definovanou variantu, a pak i celkové Elo.

Kapitola 4
Implementace

V implementaci jsem se soustředil hlavně na umožnění hraní automatizované hry. V systému je možné vytvořit implementaci deskové hry, pomocí generátoru karet vytvořit balík karet, vytvořit herní stůl v režimu sandbox, ke stolu připojit více uživatelů a odehrát hru.
Vynechal jsem tedy systém lobby a uživatelských účtů, editor komponent a Elo rating.

4.1 Architektura

Jako všechny běžné webové aplikace, i nový systém se dá rozdělit na dvě základní komponenty: frontend a backend. Pro frontend jsem zvolil knihovnu React [12]. Důvodem k této volbě je hlavně to, že reprezentace herní situace docela dobře pasuje na React state.

Pro backend jsem zvolil Node.js [13] společně s frameworkem Express. [14]

Aplikace běží nad databází MongoDB [15]. V databázi jsou následující collections:

- `decks` obsahuje data všech balíků karet
- `projects` obsahuje data všech existujících projektů v systému
- `tables` obsahuje data všech běžících i dokončených herních stolů

4.2 Generátor karet

Generátor karet byla první komponenta systému. Pro definici tabulek jsem použil Google Sheets hlavně kvůli jejich popularitě, jednoduchosti použití i osobním sympatiím.

4.2.1 Backend

Backend má API endpoint, který přijímá .psd soubor, množinu dalších grafických souborů, ID Google sheetu a volitelně ID balíku karet v databázi. Vrací seznam karet a pravidel, které jsou vytvořeny na základě Google Sheetu.

.psd soubor je zpracován za použití package `psd` [16]. Jednotlivé vrstvy souboru jsou uloženy do Google Cloud Storage. Do odpovědi za použití toho package přidá informace o velikosti obrázku a seznam vrstev. U každé vrstvy odešle

- Šířku a výšku
- Pozici levého horního rohu vrstvy

- URL obrázku ve vrstvě

Jestliže bylo posláno ID balíku v databázi, záznam o balíku v databázi je upraven seznamem vrstev, karet a pravidel. Tento záznam se použije při hraní hry s daným balíkem karet.

4.2.2 Frontend

Na stránce s generátorem karet je formulář, kam lze vyplnit ID google sheetu a nahrát .psd soubor a další grafické soubory. Po odeslání formuláře zobrazí vygenerované karty.

Karty se generují jednotlivě jedna po druhé na základě odpovědi serveru. U každé karty se projde seznam jejích pravidel, a do pravidel se do hodnot se substitucí doplní data dané karty. Karta je prachobyčejný `div`, a na každou vrstvu, která má pravidlo `layer_visible`, se vytvoří `div`, který je potomkem karty. Všechna pravidla se následně aplikují na příslušné vrstvy.

Po vygenerování karet se zobrazí posuvník na úpravu velikosti karet, a posuvník na úpravu počtu karet na řádek. Kromě toho přidá i možnost karty vytisknout stlačením tlačítka. Po stlačení tohoto tlačítka se zobrazí standardní tiskový dialog prohlížeče.

4.3 Projekt

Projekt je základní stavební kámen implementace. Každý projekt zastupuje jednu implementaci a všechny vytvořené stoly u této implementace.

4.3.1 Seznam projektů

Na úvodní stránce je seznam projektů. U každého projektu je obrázek a název. Každá položka je odkaz na detail projektu.

4.3.2 Projekt jako součást aplikace

Všechny soubory k dané implementaci se nacházejí ve složce `projects`. Každý projekt má zde svoji zvláštní složku. Aplikace je navržená jako single-page. Je tedy nesmysl při spuštění načítat všechny zdrojové soubory všech projektů, a proto se dynamicky importují vždy jen soubory těch projektů, se kterými uživatel interaguje. Zdrojové soubory jsou takto “blízko” zbytku systému (a neimportují se nějak externě) z důvodu pohodlnějšího vývoje a debugování. Když jsou implementátorovy funkce definovány tímto způsobem, je například možné použít k debugování IDE jako třeba VSCode.

Ve složce projektu jsou následující soubory:

- **setup.js** obsahuje jedinou funkci `setup`, která v argumentu přijímá `gameSituation`, která obsahuje informace o počtu hráčů a nastavení hry. Tato funkce pomocí `gameSituation` vytváří komponenty, které přidává do herní situace.
- **components.js** obsahuje definice všech herních komponent, které jsou uložené do herní situace při založení stolu.
- **moveManager** obsahuje funkce `evalGameMove` a `evalClick`, které na základě herního pohybu, respektive kliknutí, upravují herní situaci.
- **highlight.js** obsahuje funkci, která říká, jestli daná komponenta v dané herní situaci má být zvýrazněná
- **customMoves.js** obsahují definice custom herních pohybů

4.3.3 Stránka projektu

Stránka projektu obsahuje obrázek projektu, název projektu a popis projektu. Navíc obsahuje seznam stolů. U každého stolu jsou informace o počtu hráčů u daného stolu, počet odehraných herních tahů u daného stolu a možnost připojit se ke stolu. Na této stránce je i formulář, který umožňuje založit nový stůl s nastavením.

4.3.4 Založení nového stolu

Při založení nového stolu se zavolá funkce, kterou dodal implementátor. Tato funkce je dynamicky importovaná ze složky s projektem.

4.4 Stránka herního stolu

Do herního stolu se dá dostat buď přes seznam rozehraných stolů, anebo přímo přes odkaz.

Stránka herního stolu nahoře obsahuje lištu s popisem aktuálního herního stavu, `<select>` s možností výběru, za koho budu hrát. Kromě barev hráčů je v nabídce i "spectator" a "always active". "Spectator" nemůže dělat žádné herní tahy, a nikdy se nepovažuje za aktivního hráče. "Always active" se vždy považuje za aktivního hráče. Dále stránka stolu obsahuje toggle tlačítko na zapnutí a vypnutí edit modu.

4.4.1 Herní situace

Hlavní součástí stránky herního stolu je zobrazení herní situace. Komponenta herní situace umí zobrazit aktuální herní situaci, aktualizovat herní situaci na základě herních pohybů, co dostává od serveru, a poskytuje uživatelské rozhraní k provádění herních pohybů uživatelem.

Vnitřně je tato komponenta rozdělena na `Table` a `GameSituation`, která je potomkem `Table`.

4.4.1.1 Table

Komponenta `Table` má v kompetenci logiku za herní situací. Má funkce na přidávání a vyhodnocování herních pohybů a komunikaci se serverem. Pro komunikaci se serverem používá knihovnu socket.IO [17].

socket.IO má jako primární komponentu `room`, která velmi dobře odpovídá využití v systému, kde všichni uživatelé připojení k jednomu stolu jsou ve stejné `room`. I z toho důvodu jsem se rozhodl použít socket.IO a ne pouze WebSocket, protože by bylo stejně potřeba implementovat podobnou funkcionalitu, kterou nabízí socket.IO.

4.4.1.2 GameSituation

Komponenta `GameSituation` je dceřinou komponentou komponenty `Table`. Má na starosti renderování herní situace a zpracování uživatelských eventů v herní situaci.

4.4.1.3 ComponentComp

`ComponentComp` je komponenta, která reprezentuje jednu herní komponentu. Jejími props jsou `x` a `y`, které určují její souřadnice oproti matčinné komponentě. Dále to jsou `data`, ve kterých je název, nastavení a případné děti komponenty. Dále obsahuje reference na globální funkce a hodnoty, jako je aktuální herní situace, seznam komponent, callbacky na jednotlivé eventy, seznam balíků karet, edit mode, aktuálně vybraný komponent atd.

Každá komponenta se na základě dat v seznamu komponent vykreslí. V seznamu komponent najde podle svého jména svůj záznam, a na základě svých nastavení vybere podle podmínek své údaje. Přidá do stránky svůj obrázek, a rekurzivně vytvoří všechny své potomky. Do nich pošle pozici `x` a `y` na základě svých definovaných pozic.

Kapitola 5
Testování

Testování systému proběhlo implementací několika her a jejich hraním. Starship Captains je hra, pro kterou byl systém použit až během vývoje. Ztracený ostrov Arnak je hra, která byla vyvinuta mimo systém. V systému byla implementována s cílem použít implementaci na testování rozšíření. Slow Machines a Mining on Mars jsou deskové hry, které vznikají přímo v systému společně s herním designem. Šachy jsou klasická desková hra.

5.1 Implementace hry Starship Captains

Starship Captains je pilotní projekt, pro který byl systém využit. Systém byl z velké části implementován paralelně s implementací Starship Captains. Starship Captains je tahovou kompetitivní hrou s velkým množstvím různých efektů, které se různě kombinují. Systém tedy byl pro toto vcelku vhodný.

5.1.1 Starship Captains

Starship Captains je desková hra od Petera Hoffgaarda, kterou vydalo CGE v roce 2022. Jedná se o tahovou kompetitivní deskovou hru zasazenou do lehkého a vtipného sci-fi vesmírného tématu.

Hra je o létání vesmírem, plnění misí, boji s vesmírnými piráty a vylepšování posádky a vesmírné lodě. Každý hráč má svoji vesmírnou loď, kde může mít panáčky, které mohou být v jedné ze 4 barev, a to buď na můstku, a nebo ve frontě. Každý hráč má také nákladový prostor, kde může skladovat artefakty a pirátské lodě.

Hra obsahuje herní plán, kde každý hráč má svoji vesmírnou loď. Herní plán je neorientovaný graf, kde na každém vrcholu může být karta mise, a na každé hraně může být žeton piráta.

Hra se hraje na 4 kola. Na konci každého kola se všichni až na poslední 3 panáčky přesunou z fronty na můstek.

Hráč se střídají na tahu, a hráč má ve svém tahu 2 základní možnosti: buď pošle jednoho svého panáčka na můstku pracovat do nějaké místnosti odpovídající barvy na vesmírné lodi, nebo pošle jednoho až tři panáčky z můstku na misi, na které zrovna má svoji vesmírnou loď (počet panáčku určuje mise). Všechny použité panáčky hráč umístí do fronty.

Kompletní pravidla hry jsou k dispozici na stránkách vydavatele. [18]

5.1.2 Zasazení do kontextu

Starship Captains je hra, kterou firma CGE přijala k vývoji a vydání v roce 2021, a intenzivně pracovala na vývoji v roce 2022. Při vývoji byl systém intenzivně využíván. Hra byla původně

jako sandbox implementovaná v TTS, a na testování přes systém se přešlo až ve chvíli, kdy byla hotová automatizace. Pak implementace na TTS přestala být udržovaná.

Generátor karet při vývoji nebyl příliš hojně využíván, protože na přiblížení se grafickému standartu CGE prototypů bylo potřeba výrazné zapojení grafika pro přípravu grafických podkladů pro generátor karet. Generátor karet není stavěn na sofistikovanou sazbu karet, o kterou se starají grafici. V implementaci tedy byly použity celé vysázené karty od grafiků, protože karty se musely sázet tak jako tak pro výrobu fyzických prototypů.

5.1.3 Automatizace implementace

Vzhledem k velkému množství různých efektů je vhodné implementovat obecné provedení efektu na jednom místě, a na to pak odkazovat, ať už se efekt spustil aktivací místnosti nebo splněním mise.

Jádrem automatizace je funkce `gainRes(resName, player)`, která říká, že hráč `player` má získat surovinu `resName`. Získání suroviny bylo teprve později zobecněno na provedení libovolného efektu.

Může se stát, že při získávání suroviny, resp. provádění efektů, musí hráč provést nějaké rozhodnutí. V takovém případě musíme pro toto rozhodnutí vyrobit nový podstav, a po vyhodnocení podstavu pokračovat ve vyhodnocování nadstavu.

Zároveň často hráč vyhodnocuje více efektů najednou (např. při splnění řádku mise).

Z toho důvodu byla vytvořena funkce `createEffectQueue`. Ta má jako argument seznam efektů, které postupně provádí pomocí funkce `gainRes`. Pokud narazí na efekt, který vytvoří nový podstav, vytvoří k tomuto podstavu nový nadstav, který v argumentu má seznam efektů, které se po vyhodnocení efektu, který vytvořil podstav, musí ještě vyhodnotit.

U každé karty mise pak v balíku karet je definován seznam efektů u každého řádku jako JSON, který je posílán do funkce `createEffectQueue`.

5.1.3.1 Plnění mise

Při automatizaci plnění mise se velmi osvědčil systém podstavů. Ve chvíli, kdy se hráč rozhodne plnit misi, vytvoří se podstav `resolve mission` s podstavem `select mission meeple` a karta se přesune k hráčově lodi:

```
function startMission(cardData) {
    eventManager.addGameMove({
        type: "add substate",
        newState: {
            name: "resolve mission",
            activePlayer: sourcePlayer,
            missionId: cardData.options.id,
            message: "{activePlayer} player must evaluate the mission"
        }
    }, gameSituation);

    eventManager.addGameMove({
        type: "add substate",
        newState: {
            name: "select mission meeple",
            resolvingRow: 1,
            missionId: cardData.options.id,
            activePlayer: sourcePlayer,
            message:
            "{activePlayer} player must send a meeple to the mission"
        }
```

```
    }, gameSituation);
    eventManager.addGameMove({
        type: "component move",
        sourcePlayer,
        componentId: cardData.id,
        componentData,
        destinationId: playerBoard.id,
        positionName: "missions",
        positionIndex: 0,
        logMessage: "{sourcePlayer} goes on a mission"
    }, gameSituation);
}
```

Výpis kódu 5.1 Funkce na plnění mise

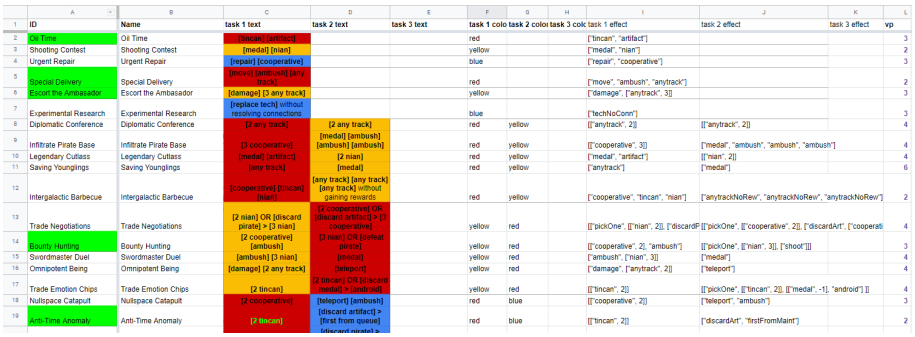
Když hráč klikne na svého panáčka na můstku, když je hra ve stavu `select mission meeple`, pošle tím panáčka na řádek mise, který se má vyhodnotit. V tu chvíli je potřeba

- Přesunout panáčka do fronty
- Zjistit, jestli barva panáčka odpovídá řádku mise
- Provést efekty řádku, pokud barva odpovídá
- Pokud na misi hráč poslal commandera, dát hráči na výběr buď vykopnout jiného panáčka z fronty, nebo znovu vyhodnotit řádek mise
- Pokud to byl poslední řádek mise, zrušit podstavy pro plnění mise
- Pokud to nebyl poslední řádek mise, vytvořit podstav pro další řádek mise.

```
function sendMeepleToMissionRow() {
    let isOfficer = componentData.options.level === "officer";
    eventManager.addGameMove({
        type: "component move",
        sourcePlayer,
        componentId: componentData.id,
        componentData,
        destinationId: playerBoard.id,
        positionName: "queue",
        positionIndex: 0,
        officerText: isOfficer ? "officer " : " ",
        logMessage: "{sourcePlayer} sends {componentData.options.color}
        {officerText}meeple to the mission"
    }, gameSituation);
    let missionId = gameSituation.getActiveState().missionId
    let cardData = gameSituation.deckData["missions(faction)"].cards[
        missionId];
    let row = gameSituation.getActiveState().resolvingRow;
    let done = true;
    let rowColor = cardData["task " + row + " color"][0];
    let effect = [];
    let colorMatches = rowColor === componentData.options.color;
    let androidSent = componentData.options.color === "green";
    if (colorMatches || androidSent) {
        effect = JSON.parse(cardData["task " + row + " effect"][0]);
        if (isOfficer) {
            effect.push(["pickOne", JSON.parse(JSON.stringify(effect)),
```

```
27                 [["kickFromMaint", componentData.options.color]]]);
28             }
29         }
30         else if (isOfficer) {
31             effect = ["kickFromMaint", componentData.options.color];
32         }
33         done = createEffectQueue(effect, sourcePlayer);
34         if (done) {
35             resolveState(sourcePlayer, false);
36         }
37         return done;
38     }
```

Výpis kódu 5.2 Funkce na poslání panáčka na řádek mise



ID	Name	task 1 text	task 2 text	task 3 text	task 1 colo	task 2 colo	task 3 colo	task 1 effect	task 2 effect	task 3 effect	vp
Oil Time	Oil Time	[tincan] [artifact]			red			["tincan", "artifact"]			3
Shooting Contest	Shooting Contest	[medal] [nian]			yellow			["medal", "nian"]			2
Urgent Repair	Urgent Repair	[repair] [cooperative]			blue			["repair", "cooperative"]			3
Special Delivery	Special Delivery	[move] [ambush] [any track]			red			["move", "ambush", "anytrack"]			2
Escort the Ambasador	Escort the Ambasador	[damage] [3 any track]			yellow			["damage", ["anytrack", 3]]			3
Experimental Research	Experimental Research	[replace tech] without resolving connections			blue			["techNoConn"]			3
Diplomatic Conference	Diplomatic Conference	[2 any track]	[2 any track]		red	yellow		[["anytrack", 2]]	[["anytrack", 2]]		4
Infiltrate Pirate Base	Infiltrate Pirate Base	[3 cooperative]	[medal] [ambush] [ambush] [ambush]		red	yellow		[["cooperative", 3]]	["medal", "ambush", "ambush", "ambush"]		4
Legendary Cutlass	Legendary Cutlass	[medal] [artifact]	[2 nian]		red	yellow		["medal", "artifact"]	[["nian", 2]]		4
Saving Younglings	Saving Younglings	[any track]	[medal]		red	yellow		["anytrack"]	["medal"]		6
Intergalactic Barbecue	Intergalactic Barbecue	[cooperative] [tincan] [nian]	[any track] [any track] [any track] without gaining rewards		red	yellow		["cooperative", "tincan", "nian"]	["anytrackNoRew", "anytrackNoRew", "anytrackNoRew"]		2
Trade Negotiations	Trade Negotiations	[2 nian] OR [discard pirate] > [3 nian]	[2 cooperative] OR [discard artifact] > [3 cooperative]		yellow	red		[["pickOne", [["nian", 2]], ["discardP	[["pickOne", [["cooperative", 2]], ["discardArt", ["cooperati		4
Bounty Hunting	Bounty Hunting	[2 cooperative] [ambush]	[3 nian] OR [defeat pirate]		yellow	red		[["cooperative", 2], "ambush"]	[["pickOne", [["nian", 3]], ["shoot"]]]		3
Swordmaster Duel	Swordmaster Duel	[ambush] [3 nian]	[medal]		yellow	red		["ambush", ["nian", 3]]	["medal"]		4
Omnipotent Being	Omnipotent Being	[damage] [2 any track]	[teleport]		yellow	red		["damage", ["anytrack", 2]]	["teleport"]		4
Trade Emotion Chips	Trade Emotion Chips	[2 tincan]	[2 tincan] OR [discard medal] > [android]		yellow	red		[["tincan", 2]]	[["pickOne", [["tincan", 2]], [["medal", -1], "android"]]]		4
Nullspace Catapult	Nullspace Catapult	[2 cooperative]	[teleport] [ambush]		red	blue		[["cooperative", 2]]	["teleport", "ambush"]		3
Anti-Time Anomaly	Anti-Time Anomaly	[2 tincan]	[discard artifact] > [first from queue] [discard pirate] >		red	blue		[["tincan", 2]]	["discardArt", "firstFromMaint"]		2

Obrázek 5.1 Definice balíku misí. Sloupec `task N effect` obsahuje definici efektů, se kterou pracuje automatizace. Texty efektů byly dodané herními designery.

5.1.4 Hráči

Implementace byla hrána vyššími desítkami unikátních hráčů. V ranných fázích byla použita k internímu hraní zaměstnanci firmy, jako alternativa k TTS. V pozdějších fázích byla hrána třemi skupinami hráčů:

- Externí testeři: externí hráči, kteří mají zájem dobrovolně se zapojit do vývoje deskové hry a poskytnout zpětnou vazbu
- Obchodní partneři: hra byla v systému prezentována obchodním partnerům CGE, kteří lokalizují a distribuují deskové hry.
- Demotéři: hra byla v systému prezentována demotérům, tedy brigádníkům, kteří na herních veletrzích prezentují deskovou hru

V poslední fázi vývoje, kdy se úsilí vývojového týmu soustředilo na implementaci režimu hry pro jednoho hráče, opravdu zazářily přednosti systému. Bylo potřeba rychle vyzkoušet různé varianty sólové hry lidmi v Praze, Brně a Dánsku. Zde se ukázalo, jak je systém flexibilní, a umožňuje dělat rychlé iterace, kdy byly dny, kdy ráno proběhla designerská porada, dopoledne jsem provedl implementaci výsledků porady, a po obědě už jsme mohli zkoušet nové varianty sólového režimu a diskutovat klady a zápory našich nápadů. Věci, které byly jednoduché na automatizaci jsem rychle implementoval, a pravidla, která na automatizaci byla náročnější, se nasimulovala při hře pomocí edit mode.

Obrázek 5.2 Hra Starship Captains v systému. Červený hráč právě poslal modrého panáčka na druhý řádek mise Repair the Repair Station. Hra je tedy v podstavu, kdy si hráč má vybrat, které zranění odstaní ze své lodi. Po vyhodnocení tohoto efektu se vyrobí ještě postupně další dva podstavy na opravu lodi (protože efekt dává celkem 3 opravy). Po vyhodnocení těchto podstavů se odstraní podstavy na vyhodnocování řádku mise a na vyhodnocování celé mise.

5.2 Implementace hry Ztracený ostrov Arnak

U tohoto projektu je to specifikum, že již byl implementován s plnou automatizací na BGA, a výsledek může poskytnout přímé srovnání implementace v novém systému a na BGA.

5.2.1 Ztracený ostrov Arnak

Ztracený ostrov Arnak je desková hra, která byla vydaná v roce 2020. Rozhodl jsem se tuto hru implementovat pro BGA, kde se rychle dostala do 20 nejhranějších her, a vytvořila se okolo ní komunita závodních hráčů.

V roce 2021 pro hru začalo vznikat rozšíření Velitelé Expedic. CGE se rozhodlo k vývoji použít implementaci na BGA. K testování byli pozváni nejsilnější hráči základní implementace. Vývoj ale byl náročný, protože BGA není určeno k vývoji nových deskových her, a implementace základu nebyla vyvíjena s tím, že by měla být použita k vývoji rozšíření. Tyto dva problémy se mezi sebou násobily, a působily překážky na každém kroku vývoje. Přes to bylo BGA úspěšně použito ke hraní, testování a vývoji hry.

V roce 2022 se CGE rozhodlo k vydání dalšího rozšíření. Testování jsme začali na BGA. Po zkušenostech z vývoje prvního rozšíření jsme došli k závěru, že Arnak dále budeme testovat v novém systému, a provedeme tranzici na nový systém

5.2.2 Automatizace implementace

Základem je Arnak podobný již zmiňované hře Starship Captains. Arnak používá stejný systém podstavů a zobecněnou funkci zisku zdroje, která může vytvořit nový podstav.

Obrázek 5.3 Rozehraná hra Arnaku v systému

5.2.2.1 Objevení lokace

Zajímavou specifickou akcí v Arnaku je objevení lokace. Objevení má množství kroků, které občas potřebují a občas nepotřebují rozhodnutí hráče. Objevení lokace má tyto kroky:

- Zaplacení cestovních symbolů
- Přesunutí panáčka na lokaci
- Provedení efektu sošky
- Přesunutí sošky na hráčovu desku
- Odhalení nové destičky lokace
- Provedení efektu nově objevené lokace
- Odhalení strážce na lokaci

Protože některé kroky vyžadují rozhodnutí hráče, je potřeba pro ně vytvořit podstav. Konkrétně při objevení lokace automatizace vytvoří podstavy, které jsou zanořené v tomto pořadí:

- Odhalení strážce
- Vyhodnocení efektu lokace
- Odhalení lokace
- Vyhodnocení sošky
- Zaplacení cestovních symbolů

Stav dole je nejnižší stav. Po zaplacení cestovních symbolů se tedy "probublává" nahoru skrz stavy. Jestliže soška obsahuje efekt vylepšení, zahození karty nebo obnovení asistenta, musí podstav počkat na rozhodnutí hráče. Přidání speciálního podstavu pro odhalení lokace je potřeba v případě, že Průzkumnice odhaluje lokaci, a má na výběr ze dvou. V takovém případě se v tomto stavu čeká na rozhodnutí hráče.

Po vyhodnocení efektu lokace se konečně stav dobublá až k odhalení strážce. Zde zase analogicky Sokolnice má efekt, kdy si může vybrat jednoho strážce ze dvou. Stav tedy zase čeká na rozhodnutí hráče.

5.2.3 Srovnání s BGA

Velmi zajímavé je srovnání s implementací na BGA. Všechny zdrojové soubory implementace na BGA mají 350 KB. Oproti tomu zdrojové soubory v novém systému mají 104 KB. (Pokud budeme ignorovat zdrojové soubory zajišťující vzhled implementace a budeme brát v potaz pouze soubory s herní logikou, BGA má 300 KB souborů, ale implementace v novém systému má 87 KB). Přitom je implementovaná stejná sada pravidel hry. To velmi jasně ukazuje problém implementace BGA, a to je nutnost psaní obrovského množství kódu, který neobsahuje herní logiku, a navíc velké množství redundandního kódu.

Vývoj byl navíc oproti BGA výrazně urychlen lepšími nástroji pro debugování, a doba implementace hry na BGA trvala přibližně 5x déle, než v novém systému.

5.3 Implementace hry Slow Machines

Slow Machines je hra, která je v ranné fázi vývoje, a zatím nemá vydavatele. Ve hře je 5 druhů surovin a peníze. Základním principem hry jsou karty, kde každá má svojí pořizovací cenu, cenu aktivace, dobu aktivace a zisk. Hlavní myšlenkou hry je to, že každý tah se všechny aktivované karty posunou o kousek blíže k zisku.

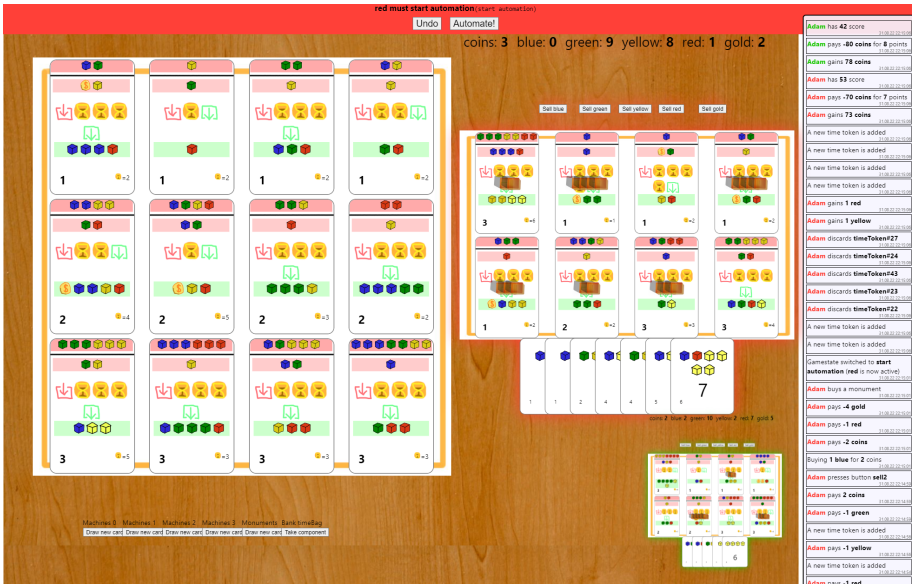
Byl to první projekt, kde vznikla implementace ještě před vznikem fyzického prototypu, a byla použita pro hraní v sandboxu v prvotní fázi vývoje. Na toto použití se systém osvědčil i přes chybějící GUI pro definici komponent. Jedná se o hru s velkým množstvím údržby, protože v každém tahu je potřeba provést údržbu každé karty, kterou hráč vlastní, a to může znamenat až 8 zcela automatických úkonů. Ty byly dobře automatizovatelné. Po výrobě fyzického prototypu a vyzkoušení hry na fyzickém stole se však ukázalo, že velké množství nutné údržby je celkem náročné a nezábavné.

5.4 Implementace hry Mining on Mars

Mining on Mars je hra o těžení surovin na Marsu, prodeji a postupném vylepšování hráčova rypadla. Hra je založená na velkém množství karet s různými efekty. Dává tedy smysl udělat částečnou automatizaci základních pravidel, ale většinu efektů karet nechat vyhodnotit hráče manuálně.

Pravidla hraní karet, která byla implementována, jsou ta pravidla, která mají společné všechny karty. Každá karta má cenu paliva na zahrání, a palivo získáné jestliže se hráč rozhodne efekt karty nevyužít. Při kliknutí na kartu se karta přesune mezi hráčovy zahrané karty, a odečte se mu příslušné množství paliva. Pokud hráč ale klikne na číslo, které určuje, kolik paliva z karty získám, karta se přesune do oblasti zahraných karet, ale hráči se příslušné množství paliva přidá.

Při vynoření na povrch hráč prodá všechny minerály, co natěžil pod zemí. Proto jsem vytvořil makro, které prodá všechny minerály přesunutím na discard a přidání příslušného množství peněz. Další makro jsem vytvořil na dobírání karet. Na začátku každého kola si hráč lízne pevné množství karet, a místo abych karty musel přesouvat jednotlivě jich na jeden klik do své ruky odhalím dané množství.



Obrázek 5.4 Rozehraná hra Slow Machines v systému

U tohoto projektu bylo otestováno, jak si systém poradí s přístupem postupné automatizace dalších a dalších pravidel, zatímco se hra kontinuálně hraje po každém dalším naimplementovaném pravidlu. Tento postup byl velmi příjemný, a pravidla, která byla náročná na manuální vyhodnocení, bylo často jednoduché naimplementovat. Čas ušetřený manuálním prováděním tahů se vyrovnal času strávenému na implementaci automatizace již po několika odehraných partiích.

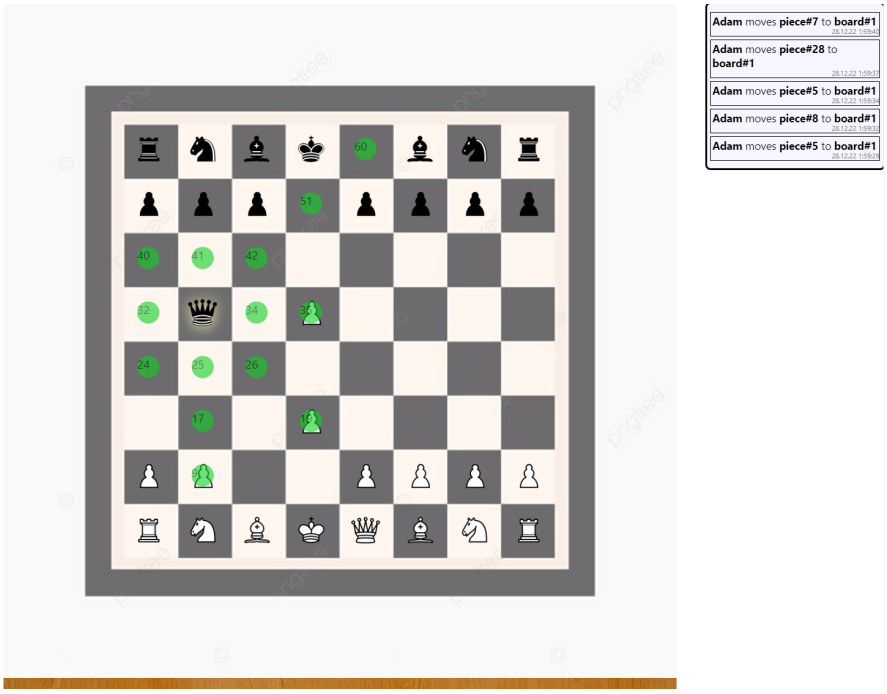
V tomto projektu zazářil generátor karet. Vzhledem k velké různorodosti karet je potřeba je často upravovat. Přes tabulku úprava dat probíhala pohodlně a přehledně. Aktualizace dat karet v projektu byla velmi jednoduchá, a rychle bylo možné hrát novou parti s novou sadou karet.

5.5 Implementace hry Šachy

V systému vesměs cvičně a jako ukázkový projekt byla implementovaná zjednodušená hra Šachy. Zjednodušená varianta má implementované základní pohyby figur bez speciálních tahů jako rošáda a braní mimochodem.

Na pozicích, kde je místo pro pouze jednu komponentu, platí obecné pravidlo, že když se na pozici přesouvá nová komponenta, přepíše předchozí komponentu. To přesně odpovídá tomu, co se stane při vyhození figury. Jediné, co tedy bylo potřeba implementovat, bylo definování pozic, kam se může každá figura přesunout.

Obrázek 5.5 Rozehraná hra Mining on Mars v systému



Obrázek 5.6 Rozehraná hra Šachy v systému. Zvýrazněné jsou pozice, kam se může přesunout černá dáma.

Kapitola 6
Závěr

Dle zadání bylo cílem práce několik věcí. Proběhla rešerše existujících systémů, většinou formou vyzkoušení systému pro vývoj deskové hry v praxi. Na základě rešerše jsem navrhl systém pro testování a vývoj deskových her online. Na základě návrhu jsem implementoval část systému a otestoval jej implementaci dvou komplexních, dvou středních a jedné jednoduché deskové hry.

Zadání tedy bylo splněno.

Bibliografie

1. *A look into the golden age of boardgames* [online]. 2018. [cit. 2022-11-12]. Dostupné z: https://boardgamegeek.com/thread/1943195/look-golden-age-boardgames.
2. *TTS Steam Preview* [online]. Berserk Games, 2015 [cit. 2022-09-27]. Dostupné z: https://store.steampowered.com/app/286160/Tabletop_Simulator/.
3. *TTS Documentation* [online]. Berserk Games, 2015 [cit. 2022-07-24]. Dostupné z: https://kb.tabletopsimulator.com/.
4. *Lua documentation* [online]. 2022. [cit. 2022-08-16]. Dostupné z: https://www.lua.org/docs.html.
5. *TTS API Documentation* [online]. Berserk Games, 2015 [cit. 2022-08-19]. Dostupné z: https://api.tabletopsimulator.com/.
6. *About Tabletopia* [online]. 2020. [cit. 2022-09-12]. Dostupné z: https://tabletopia.com/about.
7. *About BGA* [online]. 2020. [cit. 2022-10-12]. Dostupné z: https://boardgamearena.com/join?aboutmode.
8. *BGA Studio Guidelines* [online]. 2022. [cit. 2022-10-06]. Dostupné z: https://en.doc.boardgamearena.com/BGA_Studio_Guidelines.
9. *Bga Studio Docs* [online]. 2019. [cit. 2022-08-12]. Dostupné z: https://boardgamearena.com/doc/Studio_file_reference.
10. *Steam Worksop and TTS*. 2021. Dostupné také z: https://kb.tabletopsimulator.com/custom-content/steam-workshop/.
11. ELO, Arpad E. The USDF Rating System - A Scientific Achievement. *Chess Life*. 1961, roč. XVI, č. 6, s. 8–9. Dostupné také z: http://uscf1-nyc1.aodhosting.com/CL-AND-CR-ALL/CL-ALL/1961/1961_06.pdf#page=8.
12. *React* [online]. 2022. [cit. 2022-09-21]. Dostupné z: https://reactjs.org/.
13. *Node.js* [online]. 2022. [cit. 2022-04-18]. Dostupné z: https://nodejs.org/en/about/.
14. *Express* [online]. 2022. [cit. 2022-05-15]. Dostupné z: https://expressjs.com/.
15. *MongoDB* [online]. 2022. [cit. 2022-08-03]. Dostupné z: https://www.mongodb.com/home.
16. *PSD.js* [online]. 2022. [cit. 2022-09-04]. Dostupné z: https://www.npmjs.com/package/psd.
17. *Socket.IO documentation* [online]. 2020. [cit. 2022-06-12]. Dostupné z: https://socket.io/docs/v4/.

18. *Starship Captinas rules EN*. 2022. Dostupné také z: `https://czechgames.com/files/rules/starship-captains-rules-en.pdf`.

Obsah přiloženého média

```
readme.txt ..... stručný popis obsahu média
simulator_deskovych_her.pdf ..... text práce ve formátu pdf
src ..... Zdrojové kódy implementace
src-tex ..... Zdrojové kódy pro sazbu
```