Multiplatformní karetní hra

Miroslav Páč

Bakalářská práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Miroslav Páč**
Osobní číslo:	**A17629**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Multiplatformní karetní hra**
Téma práce anglicky:	**Cross-platform card game**

Zásady pro vypracování

1. Prostudujte možnosti tvorby multiplatformních her s využitím různých frameworků, technologií a prostředí.
2. Vyberte kartení hru a v hodným způsobem definujte požadavky na její implementaci do podoby multiplatformní aplikace
3. Navrhněte stukturu aplikace a sestavte architekturu daného řešení
4. S využitím zvolených technologií implementujte požadovanou aplikaci
5. Ověřte funkčnost aplikace na minimálně dvou rozdílných platformách

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. CHROBOCZEK, Martin. Grafická uživatelská rozhraní v Qt a C++. Brno: Computer Press, 2013. ISBN 9788025141243.
2. HALPERN, Jared. Developing 2D Games with Unity: Independent Game Programming with C#. APress, 2018. ISBN 1484237714.
3. BARRERA, Ray, Aungh Sithu KYAW a Thet Naing SWE. Unity 2017 Game AI Programming. 3rd ed. Birmingham: Packt Publishing, 2018. ISBN 978-1-78847-790-1.
4. PRATA, Stephen. Mistrovství v C++. 3., aktualiz. vyd. Přeložil Boris SOKOL. Brno: Computer Press, 2007. Bestseller (Computer Press). ISBN 978-80-251-1749-1.
5. BILL EVJEN, Christian Nagel, et al. C 2008 Programujeme profesionálně. Praha : Computer Press, 2009. 1904 s. ISBN 978-80-251-2401-7.

Vedoucí bakalářské práce: **Ing. Peter Janků, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **3. prosince 2021**
Termín odevzdání bakalářské práce: **23. května 2022**

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 24. ledna 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 20.5.2022

Miroslav Páč, v. r.
podpis studenta

ABSTRAKT

Cílem této bakalářské práce je prozkoumat vývoj multiplatformní karetní hry, včetně analýzy, grafického návrhu aplikace a popisu vývoje. Teoretická část se zabývá studiem herních enginů, co by měli obsahovat, výběrem známějších enginů ve světě a jejich výhodami a nevýhodami. Dále je čtenář seznámen s pojmem UML analýza, jejím využitím, rozdělením diagramů a důvody proč se používá. Praktická část popisuje výběr karetní hry a herního enginu. Zpracování UML analýzy a následnou práci a implementaci karetní hry. Výsledným produktem je vytvořená aplikace multiplatformní karetní hry, která je otestována na více platformách.

Klíčová slova: herní engine, UML analýza, multiplatformní karetní hra, unity

ABSTRACT

The aim of this bachelor thesis is to examine the development of a multiplatform card game, including analysis, graphical application design and development description. The theoretical part deals with the study of game engines, what they should contain, the selection of more well-known engines in the world and their advantages and disadvantages. Furthermore, the reader is acquainted with the concept of UML analysis, its use, the distribution of diagrams and the reasons why it is used. The practical part describes the selection of card games and game engines. Processing of UML analysis and subsequent work and implementation of card games. The resulting product is a cross-platform card game application that is tested on multiple platforms.

Keywords: game engine, UML analysis, multiplatform card game, unity

Tímto bych chtěl poděkovat svému vedoucímu bakalářské práce Ing. Peteru Janků, Ph.D. za jeho rady a pomoc při tvorbě této práce a panu Štěpánu Štefaníkovi za umožnění programovat jeho hru Rone a poskytnutí grafických materiálů.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Obsah

ÚVOD

V dnešní době je hraní počítačových her velmi rozsáhlým a oblíbeným způsobem trávení volného času. Vývoj hry proto může být velmi zajímavý a můžeme takový produkt nabídnout široké škále lidí. Musí se však dbát na to, aby vyvíjená hra byla něčím unikátní a zaujala.

Cílem této práce je seznámit čtenáře s vývojem multiplatformní karetní hry.

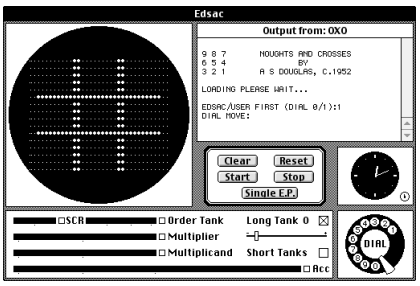
V první kapitole této práce je krátké ohlédnutím zpět do historie, kdy začali vznikat první počítačové hry ve 20 století. V druhé kapitole je popsáno vysvětlení herních enginů, jaké moduly by měly obsahovat a rozborem známějších herních enginů. U každého herního enginu jsou popsány jejich výhody a nevýhody, licenční podmínky a další důležité vlastnosti. Třetí kapitola pojednává o UML (Unified Modeling Language) analýze, na jakém principu funguje, k čemu se využívá a proč je výhodné ji používat před vývojem multiplatformní karetní hry. Na úvod v praktické části v kapitole pět jsou uvedeny důvody, proč byla vybrána k vývoji karetní hra Rone. V šesté kapitole je popsán výběr herního enginu na základě získaných znalostí z teoretické části. V další kapitole jsou definovány požadavky na aplikaci a následně vytvořeny diagramy, pro lepší pochopení co, a jak programovat. Při vývoji a zpracování analýzy multiplatformní karetní hry se využíval softwarový nástroj Enterprise Architect. V osmé kapitole je popsán celkový postup při práci s Unity, průběh samotné instalace, seznámení se s prostředím a úvodním spuštění projektu. Dále je popsán postup práce při tvoření herního menu a samotné aplikace, kde se postupně rozebírají důležité body při programování s přidanými ukázky kódu. Na závěr osmé kapitole je pár doporučení a rad pro začínající programátory v Unity.

V poslední kapitole je popsáno závěrečné otestování aplikace na více platformách.

I. TEORETICKÁ ČÁST

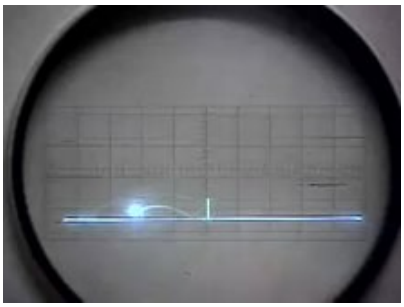
1 HISTORIE PC HER

Za počátek první počítačové hry se považuje hra OXO (Obr. 1), kterou naprogramoval student Alexander S. Douglas v roce 1952. Jednalo se o hru piškvorky omezenou na herní pole 3x3. K zobrazování se využíval osciloskop, který byl připojen k počítači EDSAC. [1]



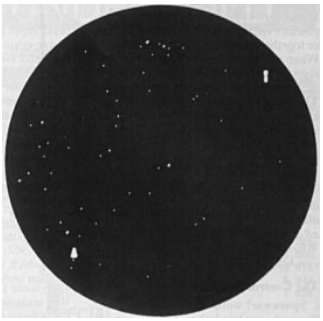
Obr. 1 OXO [2]

Další hra Tennis for Two (Obr. 2) vznikla až o 6 let později, v roce 1958. Pro zobrazení grafiky se stále ještě využívá osciloskop připojený k analogovému počítači. K ovládání se používala krabička s knoflíkem pro řízení odpalu po určitě trajektorii. Autorem je William Higinbotham. [1]



Obr. 2 Tennis for Two [1]

Největší přelom nastal v roce 1962, kdy Stephen Russell s podporou dalších studentů vytvořili hru Spacewar! (Obr. 3). Někdy bývá považována za první počítačovou hru až tato. Naprogramovaná je v assembleru na počítač PDP-1. Hra je pro 2 hráče, kdy každý ovládá vlastní vesmírnou loď a snaží se vystřelením torpéda zničit soupeře. [1]



Obr. 3 Spacewar! [1]

Z tabulky 1. jde vidět, že k největšímu rozšiřování her došlo v 70. letech, 20. století. Důvodem bylo velké rozšíření integrovaných obvodů ve světě, které pomohli k tomu, že počítače dosahovali mnohem menších rozměrů a větší výpočetní výkon za nižší pořizovací cenu.

Rok vzniku	**Název hry**
1952	OXO
1958	Tennis For Two
1962	Spacewar (+ Expensive Planetarium jako součást hry Spacewar!)
1969	Hamurabi, Lunar Lander, Space Travel
1971	Empire (Classic), Baseball, Computer Space, Star Trek (textové rozhraní)
1972	Pong, Hunt the Wumpus, Star Trek (simulace ovládaná přes dialogy)
1978	Space Invaders
1980	Pac-man, Battlezone(První 3D hra)
1983	Snipes
1985	Mario, Tetris

Tab. 1 Přehled vývoje her za 35 let vyvíjení [3]

V 90. letech začali vznikat legendární FPS (First-person shooter) hry od společnost id Software jako například Doom a hlavně Quake, který jako první začíná používat fyzikální modul pro detekci kolize. Začínají se rozvíjet real-time strategické hry s žánrem budovatelským – Caesar, hospodářkým – Railroad Tycoon nebo simulátory běžného života – The Sims. [4]

S rozšiřováním internetu přibývají hry podporující hru více hráčů po síti, např. Counter-Strike nebo Quake III Arena, který můžete vidět na Obr. 4. [4]

Obr. 4 Quake III Arena! [5]

2 HERNÍ ENGINY

Herní engine je softwarový nástroj, díky kterému máme dosáhnout komplexnějšího výsledku za nižší náklady a čas. Celková konstrukce je navržená tak, aby se dala použít nebo rozšířit při používání vývojářů třetích stran. Další výhodou je, že vývojáři nemusí dosahovat takových znalostí při vytváření her. [6]

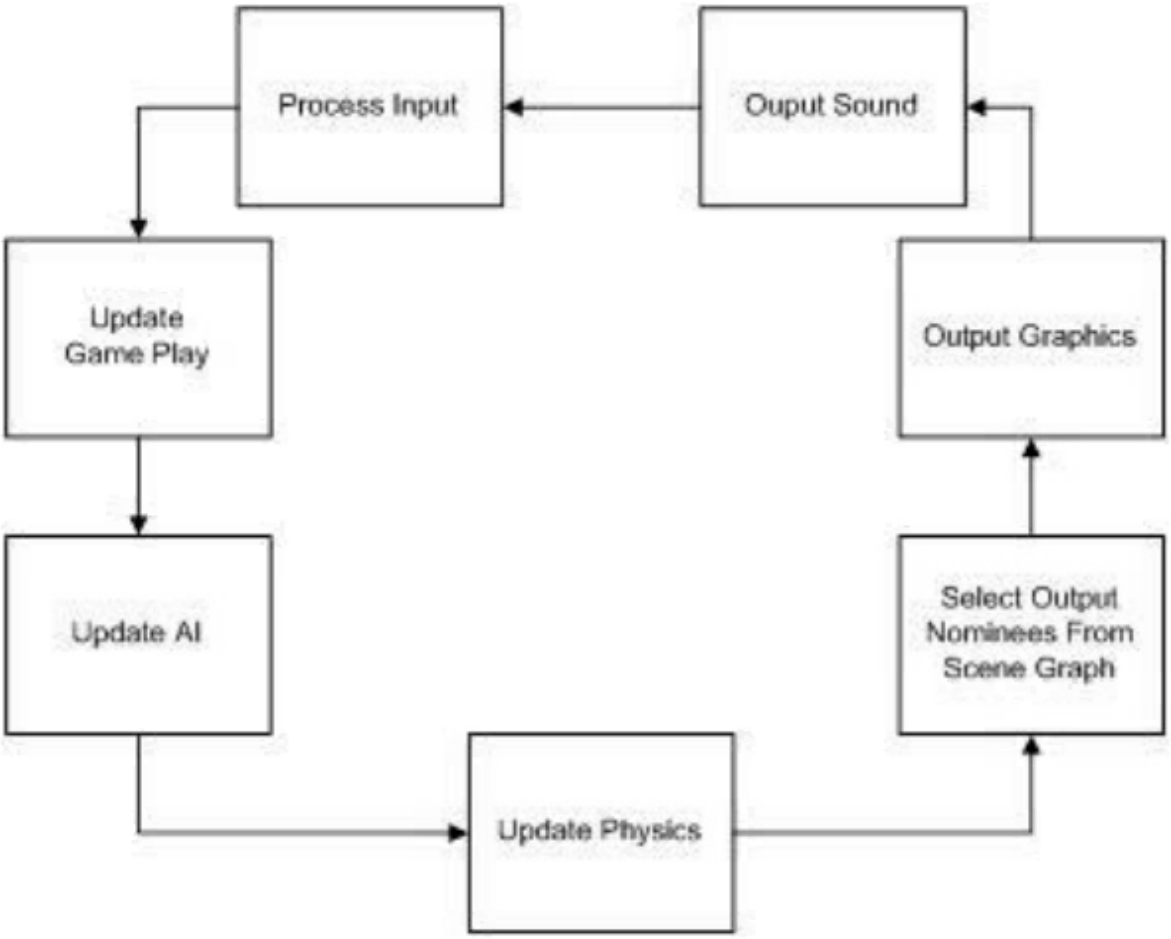
Herní enginy podporují multiplatformní programování, takže si můžete nastavit, že vyvíjíte na stolní počítač, herní konzoli, mobilní zařízení, ale budete si muset nastavit konfiguraci pro konkrétní platformu. Dále je pak potřeba ošetřit načítaní vstupů od uživatele z různých druhů periferií, které můžou být různé podle toho, na kterém operačním systému se aplikace spouští. Pokud se do aplikace vkládají obrázky, je potřeba zajistit různé velikosti obrázků z důvodu možnosti spuštění aplikace na více druhů zařízení s rozdílným rozlišením. [6]

2.1 Moduly herních enginu

Herní enginy se vzájemně liší, ale většina využívá tyto základní moduly: [7]

- Fyzikální modul, podporující detekci kolize.
- Zvukový modul pro přehrávání zvuků.
- Skriptovací modul
- AI modul *(*Artificial Intelligence)
- Modul grafické scény
- Modul pro podporu síťového připojení. [7]

Princip propojení těchto modulů je zobrazen na Obr. 5, kde herní logika AI reaguje na vstup od uživatele a díky tomu dochází ke změně herního prostředí. Dále fyzikální modul řeší fyzikální stavy a možné kolize. Dojde k výběru a aktualizaci grafických objektů na obrazovce. V pozadí se přehrávají zvukové efekty, odpovídající stavu hry. [7]



Obr. 5 Smyčka herního enginu [7]

2.1.1 Vstupní modul

Vstupní modul se zabývá zpracováním dat, které jsou přiváděny do herního systému a zapříčiňují změnu v logice hry. [7]

2.1.2 Modul hry

Herní modul se dá považovat za řídící jednotku herního enginu. Získává vstupní příkazy od uživatele a podle toho aktualizuje stavy herního prostředí. Herní modul ovlivňuje umělou inteligenci, fyzikální, grafický i zvukový modul. [7]

2.1.3 AI modul

Zabývá se chováním prvků, které hráč neovládá, aby vznikla iluze inteligentního interaktivního prostředí. Je nepřímo závislý na vstupu od uživatele. [7]

2.1.4 Fyzikální modul

Fyzikální modul zajišťuje ve hře realističtější zážitek pro uživatele, tím, že přidává do hry fyzikální zákony. Modul se skládá z fyzikálního systému, který se stará o aktualizaci hry na základě newtonovských fyzikální pravidel a systému pro detekci kolize, který řeší překrývání a interakci objektů mezi sebou. Výstup modulu je předáván do modulu grafické scény. [7]

2.1.5 Grafický modul

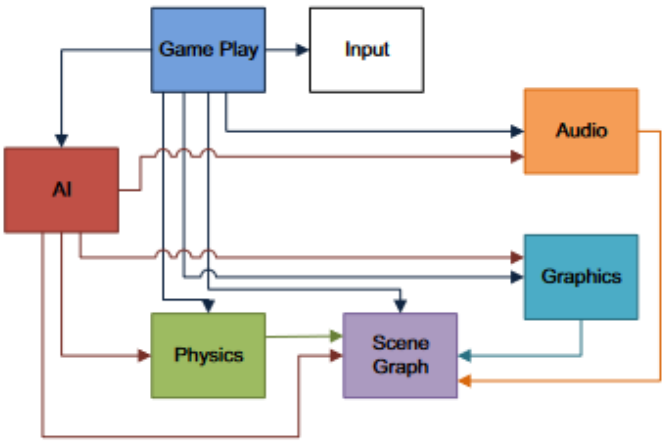
Grafický modul generuje vizuální výstupy pomocí grafického hardwaru Grafický systém je schopen pracovat ve 2D i 3D módu, kdy jeho výstup je definován prostřednictvím vizuálního tvaru a parametrů stínování. Tvarem se nastavují prostorové a objemové vlastnosti prvku a stínováním se nastavuje osvětlení a zbarvení. V průběhu hry, lze tyto parametry měnit tak, aby se vytvořil efekt animace. [7]

2.1.6 Zvukový modul

Zvukový modul se zabývá vytvářením zvukových výstupů, které jsou přímo závislé na logice AI a herního postupu. [7]

2.1.7 Vzájemné závislosti mezi moduly

Při zkoumání o modulech, jde vidět, že některé vytváří a zpracovávají většinu dat a zbylé pak pracují s výslednými upravenými daty. Na obr. 6 je znázorněna vazba mezi moduly v herním enginu. [7]



Obr. 6 Závislot modulů herního enginu [7]

2.2 Rozdělení

Herních enginů existuje velmi mnoho, proto zde uvádím jen ty nejznámější. Výběr konkrétního enginu záleží na konkrétní práci, protože každý má své výhody i nevýhody. [8] [9]

2.2.1 Unity

Jedná se o herní engine, který je vyvíjený společností Unity Technologies. První verze byla vydána v roce 2005. Jednalo se o první engine, který byl nabízen široké veřejnosti zdarma. Podporuje multiplatformní programování, aktuálně přes 25 platforem. Umožňuje vyvíjet jak 2D, tak 3D hry. Hlavním programovacím jazykem je C#. [10]

Unity je ideální pro začínající programátory z důvodu rozsáhlých tutoriálů. Vzhledem k tomu, že se jedná o jeden z nejpoužívanějších enginů ve světě, tak uživatelé mají zajištěnou podporu od komunity nadšenců až po profesionály. Unity má vytvořený svůj obchod, kde můžou uživatelé nabízet své vytvořené produkty. Tím se dá dosáhnout rychlejšího a efektivnějšího vývoje her. [10]

2.2.1.1 Licence

Unity aktuálně nabízí čtyři různé licence pro vývojáře. [10]

Unity Personal

vhodné pro začínající programátory, kteří se chtějí naučit pracovat s enginem. Používání je zdarma dokud nepřesáhnete 100 000 $ ročně, pak musíte přejít na vyšší licenci. [10]

Unity Plus

Navazuje na předchozí licenci Personal. Uživatelé získávají navíc cloudovou diagnostiku v reálném čase. Více nástrojů, s kterými se dá pracovat a možnost přizpůsobit si úvodní obrazovku. Roční cena za 1 licenci je 399$, kdy jste omezeni ročním výdělkem 200 000 $, pokud ji přesáhnete, musíte přejít na vyšší licenci. [10]

Unity Pro

Navazuje na předchozí licenci Plus. Roční poplatek za 1 licenci je 1800$ a uživatelé získávají prioritu v zákaznickém servisu. K dispozici mají technickou podporu a přístup ke zdrojovému kódu Unity. [10]

Unity Enterprise

Navazuje na předchozí licenci Pro. Uživatelé mají k dispozici učební plán na míru. Možnost vytvoření licenční kapacity serveru. Je vhodné pouze pro velké organizace, minimální počet zakoupených licencí je 10, kdy cena za 1 licenci je 200$/měsíčně. [10]

2.2.1.2 Výhody

- Výborná integrace mezi platformami.
- Rozsáhlý obchod s designy, objekty a projekty.
- Rozsáhlá dokumentace a komunita lidí.
- Bezplatná verze do 100 K$/rok.
- Podpora 2D a 3D programování. [8]

2.2.1.3 Nevýhody

- Vysoké nároky na výkon počítače.
- Zdrojový kód Unity není k dispozici zdarma. [8]

2.2.1.4 Vytvořené hry s Unity

- Cycles
- Ghost of a Tale (Obr. 7)
- Hearthstone [10]

Obr. 7 Ghost of a Tale [10]

2.2.2 Unreal Engine

Je herní engine vytvořený společností Epic Games. První verze Unreal Engine 1 byla vydána v roce 1998 ve stejnojmenné hře Unreal. Aktuální verze je 4, ale v letošním roce 2021 bude vycházet verze 5. Hlavním programovacím jazykem je C++. Další možností je využívat Blueprint. Jedná se o vizuální skriptovací jazyk, kde namísto psaní kódu, používáte rozhraní, které je založeno na uzlech k vytváření herních prvků. Podporuje programování pro více platforem, něco přes 10. Engine se nejvíce zaměřuje a doporučuje na FPS hry.[12]

2.2.2.1 Výhody

- Jedná se o open-source engine.
- Podporuje 2D i 3D programování.
- Podporuje vizuální skriptování pomocí Blueprint. Umožňuje lidem bez programovacích znalostí vytvářet základní věci. Lze kombinovat s C++.
- Rychlé zpracování renderovaného obrazu.
- Výborné vizuální ladění.
- Obsahuje nástroje pro optimalizaci hry. [11]

2.2.2.2 Nevýhody

- Jakmile budete mít vyšší příjem jak 3000 $ za čtvrtletí, musíte z vydělané částky zaplatit 5 %.
- Vysoké nároky na výkon počítače.
- Nevhodný pro začínající programátory.
- Pokud chcete nabízet svůj projekt v Unreal obchodě, musí napřed projít schválením. [11]

2.2.2.3 Vytvořené hry s Unreal engine

- Fortnite (Obr. 8)
- Conan Exiles
- Robo Recall [13]

Obr. 8 Fortnite [13]

2.2.3 Godot

První verze byla vydána v roce 2014. Autoři upravili Python, aby byl vhodný pro programování s Godot engine, GDSkript. Herní engine, který je výborný pro 2D programování, kde umožňuje pracovat v pixelových souřadnicích., S verzí 3.0 už došlo i k vylepšení funkcí a zrychlení vývoje her ve 3D. Godot využívá systém uzlů, díky nim jsou projekty lépe organizované a zlepšují škálovatelnost. [14]

2.2.3.1 Výhody

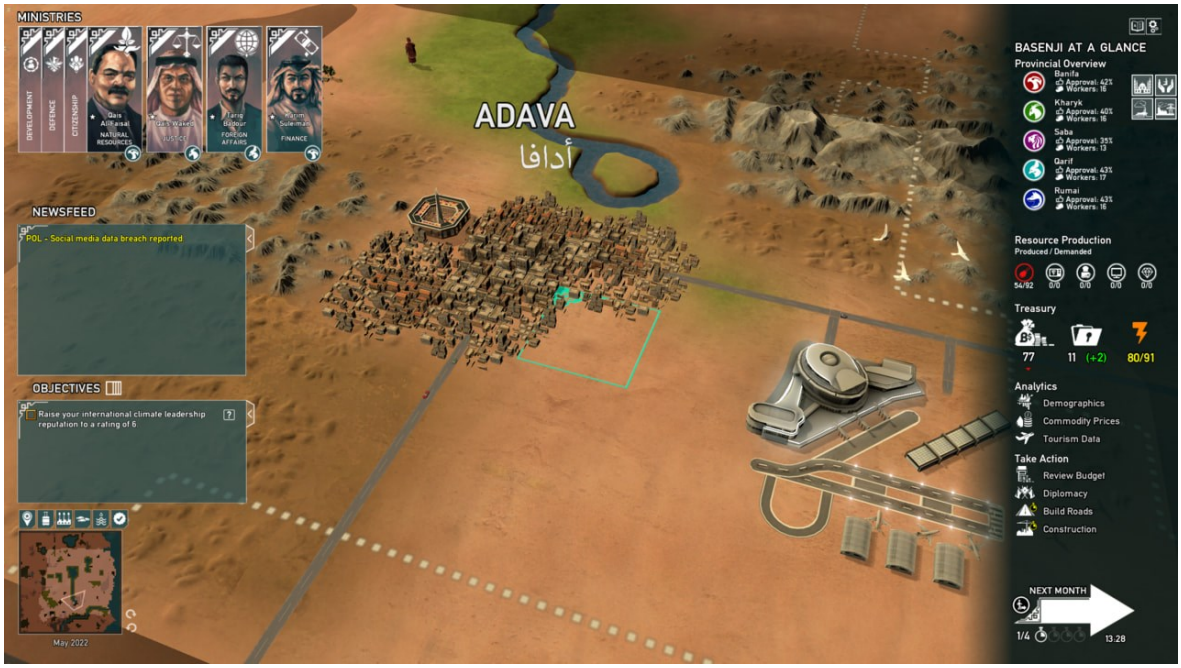
- Open source engine.
- Je k dispozici zdarma.
- Větší výběr z programovacích jazyků – C#, C++, GDScript. Vizuální skriptování
- Podporuje multiplatformní programování.
- 2D i 3D programování
- Zabere minimum místa na disku, spustitelný bez instalace [15]

2.2.3.2 Nevýhody

- Neumožňuje vývoj pro konzole. [15]

2.2.3.3 Vytvořené hry s Godot

- Gravity Ace
- Rogue State Revolution (Obr. 9)
- The Garden Path



Obr. 9 Rogue State Revolution [14]

2.2.4 CryEngine

Byl vytvořen firmou Crytek pro hru Far Cry. První verze byla vydána v roce 2004. Prioritně je engine určen pro vývoj FPS her. K dispozici je mnoho nástrojů pro urychlení vývoje. Podporuje jen 3D programování a zaměřuje se hlavně na kvalitu vizuálních prvků, aby byla vytvořena fotorealistická hra v otevřeném prostoru. Není zrovna ideální začínat s tímto herním enginem, protože komunita lidí není moc velká, není ani dostatek odborníků s praxí, kteří by Vám mohli pomáhat. Dokumentace vytvořená k enginu není dostatečná, můžete se obracet pro podporu od Crytek. [16]

CryEngine je zdarma k použití, ale jakmile se rozhodnete k publikaci komerční hry, tak využívá model založený na licenčních poplatcích. Platí se 5% licenční poplatek za roční hrubé příjmy za rok, ale prvních vydělaných 5000 $/rok je bez tohoto poplatku osvobozena. [17]

2.2.4.1 Výhody

- Výborný pro vytváření realistické grafiky.

- Skvělí fyzikální modul
- Ideální pro práci s přírodou a venkovním prostředím.
- Podporuje multiplatformní programování.
- Engine je bez komerčního použití zdarma. [18]

2.2.4.2 Nevýhody

- Náročný začátek při seznamování se s enginem kvůli nedostatečné dokumentaci a komunitě lidí.
- Zaměřuje se hlavně na FPS hry ve venkovním prostředí. [18]

2.2.4.3 Vytvořené hry s CryEngine

- Kingdom Come: Deliverance (Obr. 10)
- Crysis 3
- War of Rights [17]

Obr. 10 Kingdom Come: Deliverance [17]

2.2.5 GameMaker: Studio 2

GameMaker byl vytvořen před 20 lety firmou YoYo, která vytvořila engine, který využívá vizuální skriptovací nástroj drag-and-drop. Na tomto principu pracuje vytvořený GML (GameMaker Language) vytvořený ideálně pro tento engine. Nejnovější verze GMS2 (Ga-

meMaker Studio 2) byla vydána v roce 2017. Engine, který je vhodný i pro začínající programátory, kteří neumí programovat. Velmi snadno a rychle se v něm programuje. Snadný export na jiné platformy. Engine umožňuje vývoj pro 2D i 3D projekty. Podpora pro 3D projekty je nedostačující, ale vynahrazuje to ve vývoji 2D her, kde se dá GMS2 považovat za jeden z nejlepších enginů. [19]

2.2.5.1 Výhody

- Podporuje multiplatformní programování
- Vlastní programovací jazyk GML.
- Rychlý vývoj [19]

2.2.5.2 Nevýhody

- Není zadarmo, pouze 30denní zkušební verze.
- Chybí podpora pro vývoj 3D her.
- Málo funkcí skriptovacího jazyka GMS2 [19]

2.2.5.3 Vytvořené hry s GMS2

- 9 Till void (Obr. 11)
- Crashlands
- Sprout

Obr. 11 9 Till Void [20]

2.2.6 Shrnutí vlastností herních enginů

Výběr herního enginu při vývoji multiplatformní hry je velmi důležitou částí, protože v dnešní době je jich k dispozici mnoho a vývojář by si měl před výběrem stanovit základní požadavky v závislosti na vyvíjené hře. Proto tedy nelze všeobecně označit některý herní engine za nejlepší, protože každý má své výhody a nevýhody. V tab. 2 je vidět srovnání dříve vybraných herních enginů. [21]

	Unity	**Unreal**	**Godot**	**CryEngine**	**GameMaker: Studio 2**
Podporované programovací jazyky	C#	C++	C++	C++	GML
		BluePrint	C#	C#	
			GDSkript	LUA	
Je k dispozici zdarma	Ano	Ano	Ano	Ano	Ne
2D / 3D	Oboje	Oboje	Oboje	Oboje	pouze 2D
Podpora multiplatfomního programování	Ano	Ano	Ano	Ano	Ano
Podpora virtuální reality	Ano	Ano	Ne	Ano	Ne
Open-source engine	Ne	Ano	Ano	Ano	Ano

Tab. 2 Shrnutí vlastností herních enginů [21]

3 UML ANALÝZA

Je grafický modelovací jazyk, který se skládá z diagramů a slouží pro lepší návrh aplikace. UML je nástroj, který funguje na principu objektově orientovaného přístupu. Nejčastěji se využívá u větších projektů pro lepší komunikaci se zákazníkem, aby si obě strany rozuměli a našli odpovědi na základní otázku co naprogramovat a jak. [22]

3.1 Výhody UML

- Zjednodušuje složitost projektů, rozdělením na dílčí jednodušší úkoly.
- Zvyšuje kvalitu práce.
- Zlepšuje komunikaci se zákazníkem.
- Automatizuje vývoj softwaru a procesů.
- Zvyšuje efektivitu práce v týmech a tím se sníží čas i náklady na vývoji. [23]

3.2 Požadavky

Cílem je si definovat požadavky na vyvíjený software, pomocí kterých popíšeme funkčnost. Vytváří se v první fázi vývoje, aby vývojáři byli seznámeni s požadavky na systém. Aby se mohl lépe odhadnout čas potřebný pro vývoj takového softwaru. [24]

3.2.1 Funkční požadavky

Cílem funkčních požadavků je jednoduše popsat funkcionalitu systému, který se vyvíjí. Detailně se na tyto požadavky navazuje při tvorbě diagramu případů užití. [24]

3.2.2 Nefunkční požadavky

Nefunkční požadavek je důležitý pro zajištění použitelnosti celého systému. [24]

Př. Webová stránka se musí při návštěvnosti 20 000 uživatelů načíst do 5 sekund.

3.3 Diagramy UML

Slouží k popisu systému, kde každý diagram reprezentuje jiný úhel pohledu na vytvářený systém. Proto se jich při vývoji využívá více. Diagramy můžeme rozdělit na 4 pohledy: Statický, dynamický, uživatelský a technologický. V tabulce 3. uvádím všechny diagramy s krátkým popisem. Podrobněji se zaměřím jen na konkrétní diagramy, které budu zpracovávat pro návrh aplikace. [25]

Diagram	**Stručný popis**	**Typ**
1. Diagram případu užití	Poskytuje přehled systému z uživatelské perspektivy	Staticky behaviorální
2. Diagram aktivit	Modeluje procesy a work-flow	Staticky behaviorální
3. Diagram tříd	Reprezentuje třídy, jejich definice a vztahy	Staticky strukturovaný
4. Sekvenční diagram	Modeluje interakci mezi ob-jekty za-ložené na časové posloupnosti	Dynamicky behaviorální
5. Diagram přehledu inter-akcí	Prezentuje přehled interakcí v systému. Dále pomáhá po-chopit, jak na sobě UML di-agramy závisí.	Staticky behaviorální
6. Diagram komunikací	Zobrazuje jednotlivé ob-jekty, které mezi sebou in-teragují. Je podobný sek-venčnímu diagramu.	Staticky behaviorální
7. Objektový diagram	Zobrazuje objekty a jejich propojení.	Dynamicky behaviorální
8. Diagram složených struk-tur	Modeluje chování kompo-nentu nebo objektu, zobra-zuje uspořádání, vztahy a in-stance při spouštění.	Dynamicky behaviorální
9. Stavový diagram	Zobrazuje životní cyklus objektu, stav, ve kterém se nachází a podmínky pro změnu stavu.	Dynamicky behaviorální
10. Diagram komponent	Zobrazuje strukturovaně vy-modelované komponenty a jejich vztahy.	Staticky strukturovaný
11. Diagram rozestavení	Modeluje architekturu hard-warových komunikačních uzlů.	Staticky strukturovaný
12. Diagram balíčků	Reprezentuje subsystém a pole organizace systému. Dokáže oddělit jednotlivé entity od sebe	Staticky strukturovaný
13. Diagram časování	Modeluje způsob, jakým se mění stavy objektu v čase.	Dynamicky behaviorální

Tab. 3 Přehledu diagramů v UML [26]

3.3.1 Diagram případu užití (UC)

„*Use Case Diagram (česky diagram případů užití) zobrazuje chování systému tak, jak ho vidí uživatel. Účelem diagramu je popsat funkcionalitu systému, tedy co od něj klient nebo my očekáváme. Diagram vypovídá o tom, co má systém umět, ale neříká, jak to bude dělat.*“ [27]

Tvoří se na základě požadavků na systém, tj. funkčních požadavků. Z toho důvodu se při návrhu začíná s požadavky a následně se zpracovává UC (Use case). Případů užití musí být vytvořeno tolik, aby pokryly všechny vytvořené funkční požadavky. Ukázku UC diagramu lze vidět na Obr. 12. [27]

3.3.1.1 Specifikace

Pro podrobnější popis funkcionality UC jsou přidány další atributy. [28]

Popis

Krátký popis pár věty, co funkcionalita UC umožňuje. [28]

Aktéři

Přehled aktérů, kteří se daného případu užití účastní. [28]

Podmínky pro spuštění

Případy užití můžou mít definované podmínky, které pokud nejsou splněny, tak se daná funkcionalita nikdy neprovede. Např. Platba nemůže proběhnout, pokud nejsme přihlášeni. [28]

Hlavní scénář

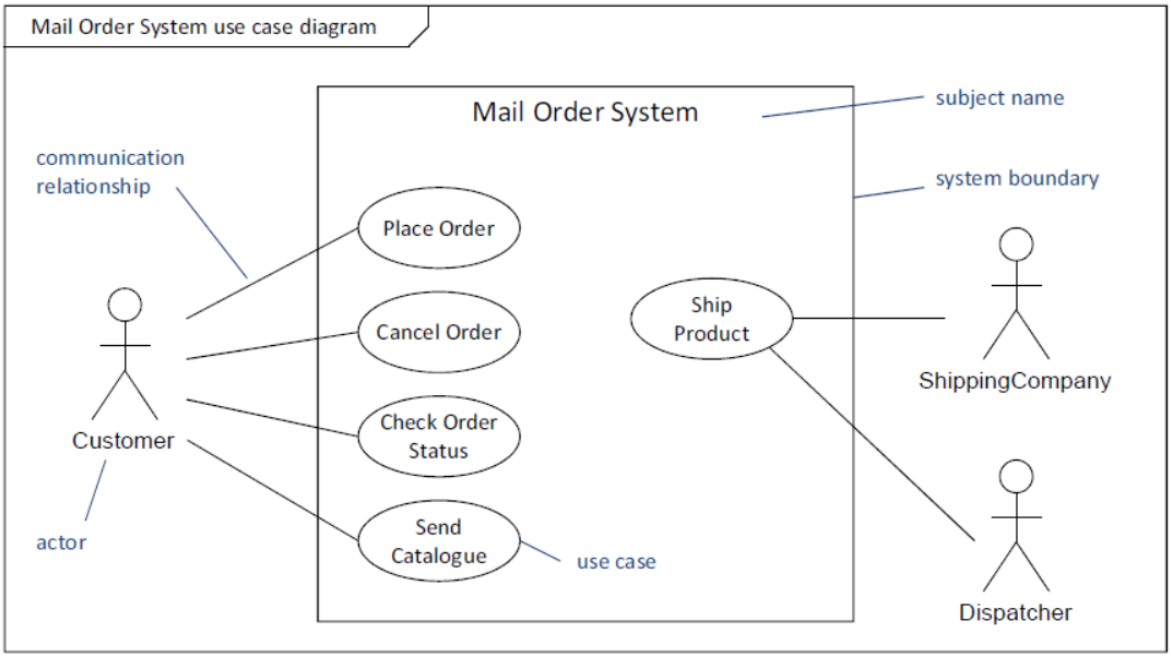
Zapisuje se stylem scénáře, kde se popisují kroky interakce uživatele a systému nad určitou funkcionalitou. Scénář musí být zapsán tak, aby se bez chyby prošlo všemi body. [28]

Alternativní scénář

Pokud uživatel může ve funkcionalitě vyvolat chybovou hlášku nebo má na výběr z více možností, musí se všechny varianty zapsat do alternativních scénářů. Z alternativního se lze odkázat zpět na hlavní. Např. Jakmile se uživateli podaří zadat správně heslo, vrátí se zpět na hlavní scénář k bodu úspěšného přihlášení uživatele. [28]

Podmínky pro dokončení

Každý UC může mít definované podmínky pro dokončení. [28]



Obr. 12 Use Case Diagram – ukázka objednávacího systému [29]

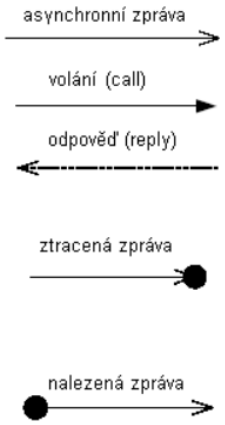
3.3.2 Sekvenční diagram

Sekvenční diagram popisuje chování a spolupráci objektů v rámci jednoho případu užití za určitý časový interval (Obr. 14).

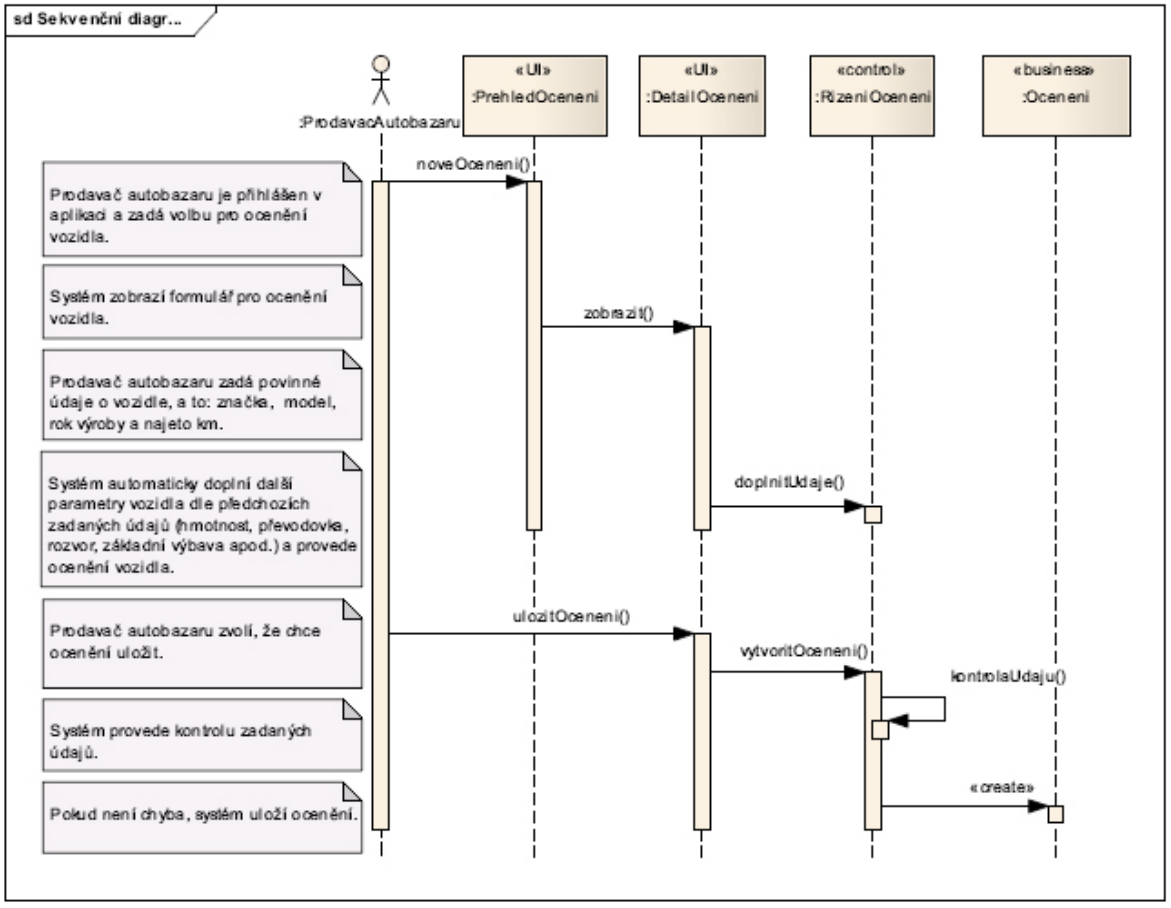
Zprávy

Zobrazují se šipkami a posílají se mezi objekty, třídami, či aktéry. Zapisují se od shora dolů a směr bývá z leva do prava.

Můžeme poslat synchronní zprávu, kde objekt čeká na odpověď, než bude pokračovat nebo asynchronní zprávu, kde pokračuje bez ohledu na reakci. Na Obr. 13 můžeme vidět šipky používané při grafickém zobrazení zpráv. [30] [31]



Obr. 13 Grafické zobrazení zpráv pomocí šipek [30]

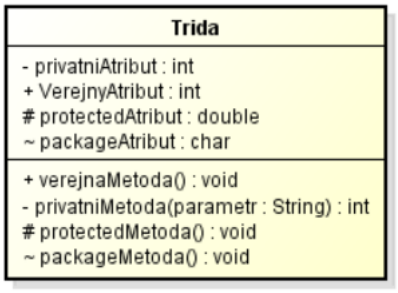


Obr. 14 Ukázka sekvenčního diagramu popisující vytvoření ocenění vozidla [31]

3.3.3 Diagramy Tříd

Diagram tříd nám reprezentuje kompletní zpracování vyvíjeného softwaru. Musí obsahovat všechny potřebné atributy a metody, aby podle toho mohl programátor naprogramovat kompletní aplikaci. Třída s atributy a metody je viditelná na Obr. 15. [32]

Cílem tohoto diagramu je přemýšlet komplexně nad vývojem a zabránit tak problémům, které by mohli nastat během programování (Obr. 19). [32]



Obr. 15 Ukázka diagramu tříd s atributy a metody [32]

Modifikátory přístupu

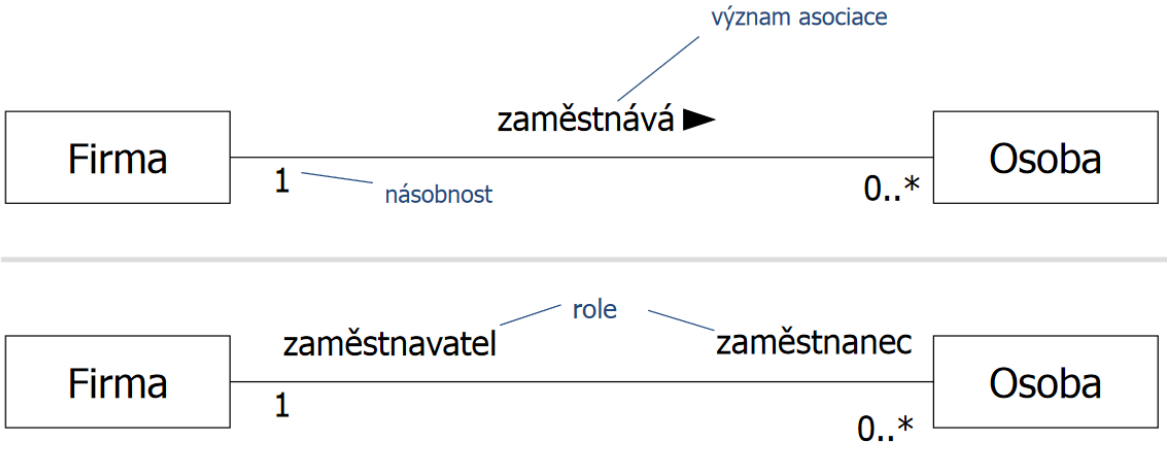
Každý atribut a metoda musí být označena modifikátorem přístupu. Na výběr jsou 4 možnosti:

- - (mínus) – Soukromý atribut
- + (plus) – Veřejný atribut
- # (hash) – Chráněný atribut
- ~ (tilda) – viditelný v rámci balíčku. [33]

Vztahy mezi třídami

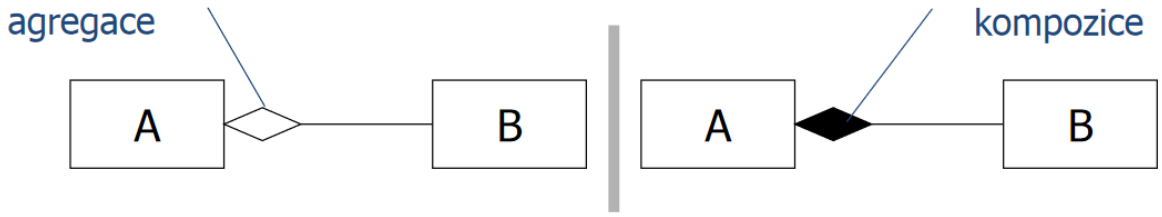
Vztah mezi třídami bývá zakreslen čárou, která propojuje třídy mezi sebou. [33]

- Asociace – Udává přímý vztah mezi propojenými třídami (Obr. 16). [33]



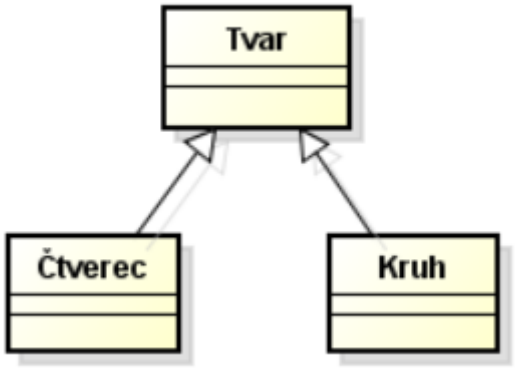
Obr. 16 Ukázka vazby typu asociace [33]

- Agregace – Určuje vztah mezi částí a celkem. Např. Kniha a list. [33]
- Kompozice – Vychází z agregace, jen dosahuje mnohem silnější vazby, kdy část by bez celku neexistovala. Například Ruka a prst. [33]

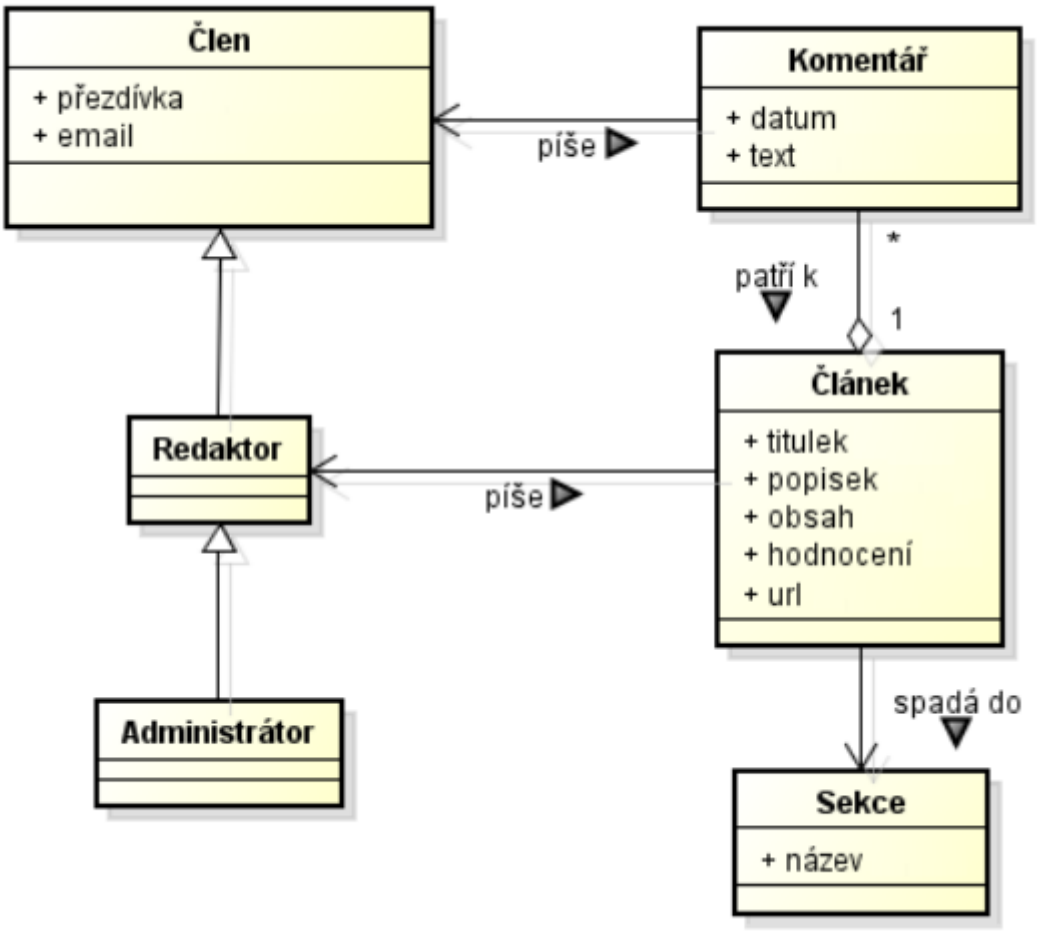


Obr. 17 Ukázka vazby typu agregace a kompozice [33]

- Dědičnost (generalizace) – Dědičnost umožňuje, že potomci přebírají metody a atributy svého předka. Ukázka jde vidět na Obr. 18. [33]



Obr. 18 Ukázka typu dědičnosti [34]



Obr. 19 Ukázka diagramu tříd [34]

II. PRAKTICKÁ ČÁST

4 ÚVOD

Cílem práce je prozkoumat proces vývoje multiplatformní karetní hry a popsat její vývoj. V prvním kroku je výběr karetní hry, která se bude vyvíjet a implementovat. Po výběru hry se definují funkční a nefunkční požadavky, které pomůže k tomu, aby aplikace byla kvalitní a stabilní. K tomu je vhodné si vybrat softwarový nástroj, který pomůže při vývoji a analýze karetní hry tím, že vývojář má přehledně k dispozici všechny zpracované diagramy. Z vytvořeným funkčních požadavků se následně vytvoří diagram případu užití, který nám hned na počátku vývoje říká, co bude vyvíjený systém umět. Vytvořené případy užití musí pokrýt všechny vytvořené funkční požadavky na systém a vytvoří se vazba mezi nimi. Vytvořený UC obsahuje krátký popis, podmínky spuštění, scénář průběhu této funkcionality, alternativní scénář a podmínky pro dokončení. Poslední tvořený diagram je diagram tříd. Ten obsahuje návrh pro všechny tvořené třídy včetně atributů a metod, aby sloužil jako návod při programování konečné multiplatformní karetní hry. Závěrem UML analýzy je tvorba grafické formy aplikace – wireframe. Na základě analýzy a požadavků se provede výběr herního enginu vhodného k vyvíjené hře. Provede se instalace potřebného vývojového prostředí Unity a Microsoft Visual Studio pro psaní kódu. Práce v unity začíná seznámením se s vývojovým prostředím a vytvořením herního menu podle grafického návrhu. Postupně je vysvětlen postup při tvorbě základní komponenty – karty. Vysvětluje se postup při tvorbě herních balíčků, ukazatelů stavu vody a hrdiny s jeho vlastnostmi a schopnostmi. Důležitým bodem je vytvoření funkčního přepínaní tahů, protože se nejedná o síťovou variantu a musí být hratelně vyřešeno předávání tahů, aniž by soupeř viděl karty. Dále je vysvětleno vykládání jednotek na bojiště, jejich útočení na soupeřovi jednotky a po zničení jejich přesun na hřbitov, kde se můžou ještě recyklovat a vrátit tak některé zpět. Závěrem jsou popsány doporučení pro začínající programátory a provedeno otestování aplikace na více platformách.

5 VÝBĚR KARETNÍ HRY

Při výběru karetní hry bylo důležité, aby se jednalo o komplexnější hru, na které se může demonstrovat vývoj multiplatformní karetní hry. Proto byla vybrána karetní hra Rone (Obr. 20). Jedná se o existující deskovou karetní hru od autora Štěpána Štefaníka, která byla vydána v roce 2015.

Hlavní důvody výběru karetní hry Rone:

- Spolupráce s autorem, který umožnil použití karetní hry Rone k vytvoření aplikace v digitální verzi.
- Podpora od autora s poskytnutím grafických podkladů všech karet a komponent.
- Jedná se o skvělou strategickou karetní hru, která ještě není naprogramovaná v digitální verzi.
- Umožňuje hrát hru ve 2 nebo 4 hráčích. Při hře 4 hráčů umožňuje i hraní v týmech.
- Jedná se o vhodnou hru, na které se dá demonstrovat vývoj multiplatformní karetní hry.

Obr. 20 – Karetní hra Rone [35]

6 VÝBĚR HERNÍHO ENGINU

Při výběru vhodného herního enginu k implementaci karetní hry Rone byl vytvořen seznam požadavků, které by měl vybraný engine splňovat při vývoji aplikace.

Požadavky pro výběr herního enginu:

- Pro nekomerční účely musí být k dispozici zdarma.
- Umožňuje programování ve 2D i 3D variantě, abych získané zkušenosti mohl uplatnit při dalším vývoji her.
- Umožňuje vývoj aplikací na více platformách.
- Využívá programovací jazyk C#.
- Má k dispozici modul pro síťové připojení.

Na základě vytvořených požadavků a získaných znalostí o herních enginech, které jsou shrnuté v tab. 4, bylo pro vývoj multiplatformní karetní hry vybráno Unity.

	Unity	**Unreal**	**Godot**	**CryEngine**	**GameMaker: Studio 2**
Podporované programovací jazyky	C#	C++	C++	C++	GML
		BluePrint	C#	C#	
			GDSkript	LUA	
Je k dispozici zdarma	Ano	Ano	Ano	Ano	Ne
2D / 3D	Oboje	Oboje	Oboje	Oboje	pouze 2D
Podpora multiplatfomního programování	Ano	Ano	Ano	Ano	Ano
Podpora virtuální reality	Ano	Ano	Ne	Ano	Ne
Open-source engine	Ne	Ano	Ano	Ano	Ano

Tab. 4 Shrnutí vlastností herních enginů [21]

7 UML ANALÝZA – ŘEŠENÍ

Pro vytváření diagramů, funkčních a nefunkčních požadavků byl použit softwarový nástroj Enterprise Architect, který slouží k vytváření UML modelů a umožňuje pokrytí životního cyklu aplikace pomocí diagramů, které jsou pro vývojáře k dispozici přehledně. [42]

7.1 Instalace Enterprise Architect

Stažení nástroje EA (Enterprise Architect) je k dispozici na internetu od oficiálního výrobce. Nástroj je k dispozici zdarma, ale je potřeba se registrovat a nechat si vygenerovat klíč.

Systémové požadavky tohoto softwaru na hardware jsou pro Windows:

- Minimálně 2 GB paměti. [42]
- 800 MB místa na disku. [42]
- Minimální rozlišení displeje 1280 x 720 pixelů. [42]

7.2 Funkční a Nefunkční požadavky

Na úvod analýzy multiplatformní karetní hry Rone se provedlo vytvoření funkčních a nefunkčních požadavků. Vytvořené požadavky jsou vidět v Tab. 5.

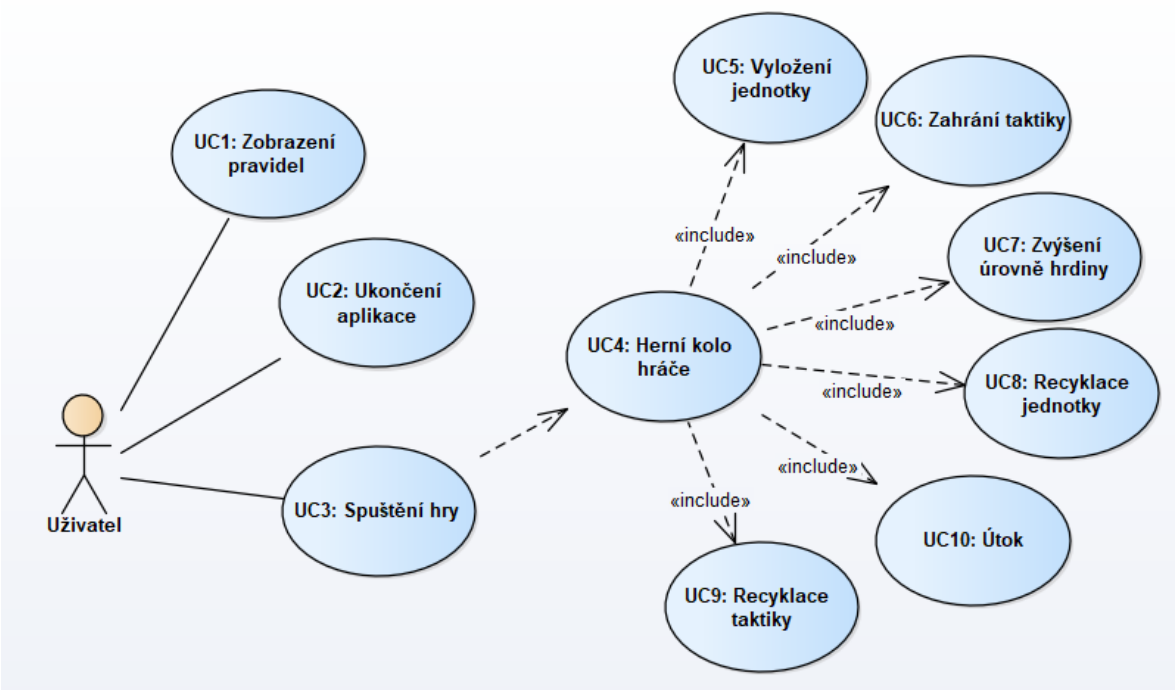
Funkční požadavky		
ID	Název	Popis
R01	Zobrazení pravidel	Uživatel si může před zahájením hry přečíst pravidla.
R02	Ukončení aplikace	Uživatel stisknutím tlačítka „Konec“ zavře aplikaci.
R03	Spuštění hry	Uživatel stiskem tlačítka „Nová hra“ spustí hru.
R04	Tah hráče	Prochází herní kolo hráče
R05	Zahrání jednotky	Hráč zaplatí cenu vody za kartu a vyloží jednotku ze své ruky na bojiště.
R06	Zahrání taktiky	Hráč zaplatí cenu a zahraje taktiku ze své ruky.
R07	Útok	Hráč si vybere jednotku na bojišti a zvolí si svůj cíl útoku.
R08	Recyklace karet	Hráč kliknutím na hřbitov aktivuje recyklaci karty a zaplatí se příslušná recyklační cena.
R09	Zvýšení úrovně hrdiny	Hráč zaplacením vody provede vylepšení hrdiny na vyšší úroveň.
Nefunkční požadavky		
ID	Název	Popis
R10	Výkon	Je zajištěn plynulý běh aplikace.

R11	Rozšiřitelnost	Umožňuje rozšiřovat aplikaci o další funkce bez nutnosti zásahu do stávajícího systému.

Tab. 5 Funkční a nefunkční požadavky

7.3 Diagram případu užití (UC)

Vytvoření případu užití bylo na základě vytvořených požadavků, aby každý požadavek byl obsažen alespoň jednou v use case. Na Obr. 21 jsou vidět tyto vytvořené případy užití a vazby mezi nimi.



Obr. 21 Use case model multiplatformní karetní hry Rone

7.3.1 UC specifikace

Pro každý use case se vytvořil scénář, na kterém jde vidět, co daný use case potřebuje a jaký bude výsledek.

7.3.1.1 Ukázka UC specifikace

UC5: Vyložení jednotky

Krátký popis: Hráč zaplatí cenu za kartu a vyloží jednotku ze své ruky na bojiště.

Aktér: Uživatel

Vstupní podmínky: Hráč zaplatí příslušnou vodu za kartu jednotky na ruce.

Předpoklady: Hráč je na tahu.

Hlavní scénář

1. Systém zaplatí vodu v hodnotě ceny karty.

2. Hráč vyloží kartu na bojiště.

Alternativní scénář:

Výstupní podmínky: Hráč má vyloženou jednotku na bojišti.

UC07: Útok

Krátký popis: Hráč si vybere vyloženou jednotku na bojišti a zvolí si svůj cíl.

Aktér: Uživatel

Priorita:

Vstupní podmínky: Hráč na tahu má jednotku na bojišti, která není unavená a soupeř má jednotku na bojišti.

Předpoklady:

Hlavní scénář

1. Hráč si vybere svou neunavenou jednotku na bojišti.

2. Hráč si vybere neprátelskou jednotku na bojišti.

3. Systém vzájemně sníží počet životů, podle střeleckého útoku soupeřovy jednotky.

4. Systém vzájemně sníží počet životů, podle útoku na blízko soupeřovy jednotky.

5. Systém unaví útočící jednotku o únavu uvedenou na kartě.

6. Systém unaví bránící se jednotku o 1.

Alternativní scénáře:

3a. Útočící jednotka nemá životy a přesune se na hřbitov.

4a. Systém unaví bránící se jednotku o 1.

3b. Bránící jednotka nemá životy a přesune se na hřbitov.

3b. Systém unaví útočící jednotku o únavu uvedenou na kartě.

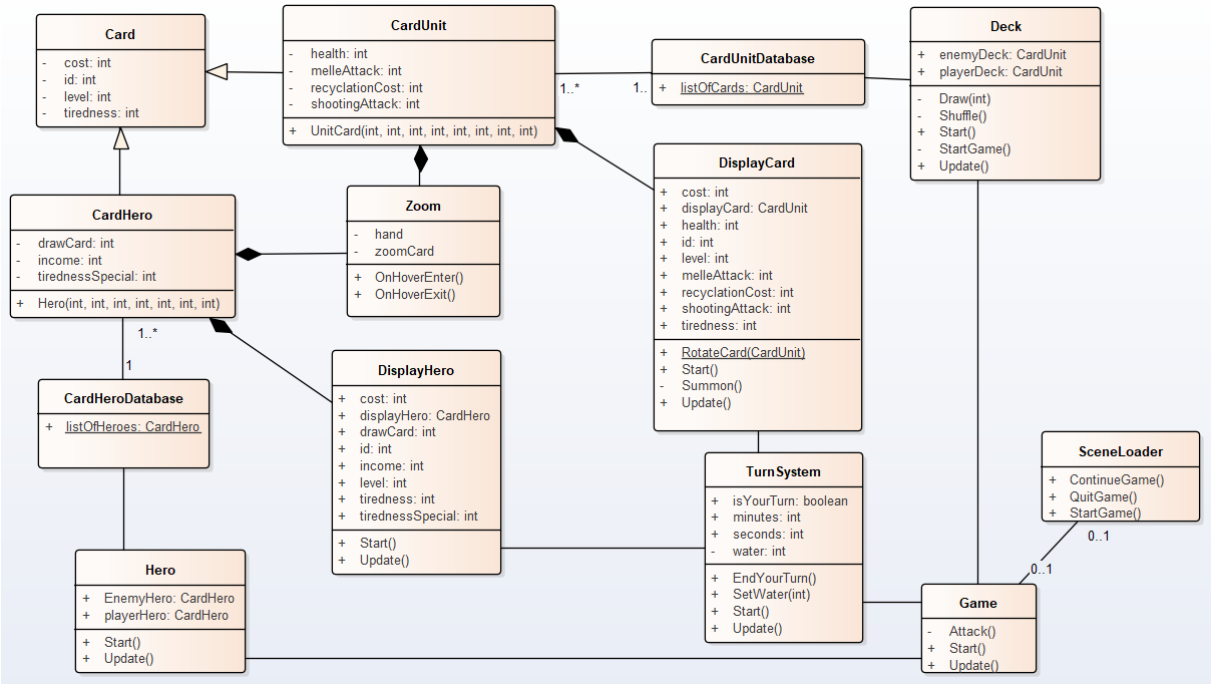
5c. Útočící jednotka nemá životy a přesune se na hřbitov.

6d. Bránící jednotka nemá životy a přesune se na hřbitov.

Výstupní podmínky: Zaútočení s jednotkou

7.4 Diagram Tříd

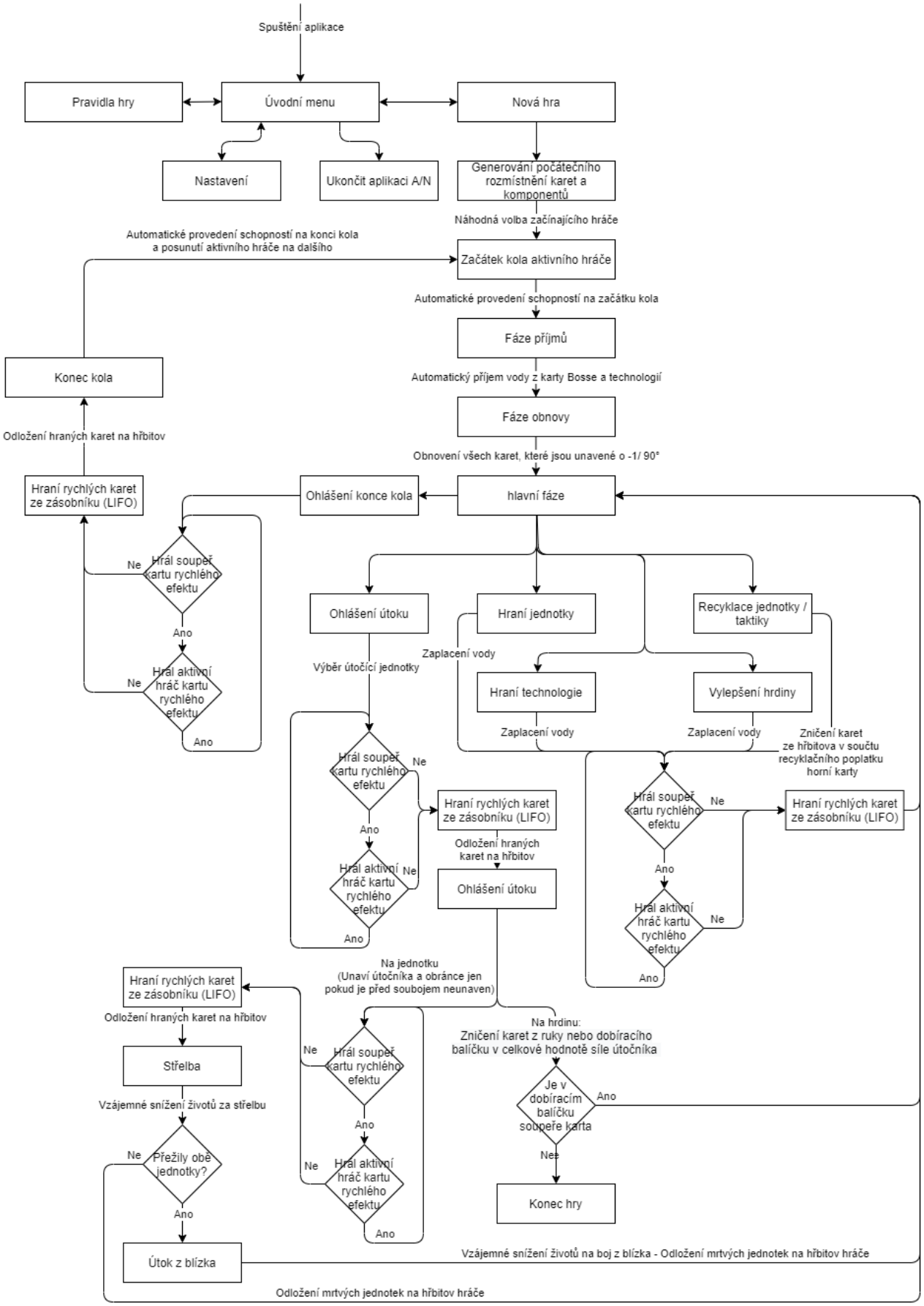
Navrhovaný diagram tříd by měl být kompletním návrhem vyvíjeného systému. Vytvořený diagram tříd na Obr. 22 je návrhem, jak by měly být vytvořeny třídy a určitě nebude kompletní z důvodu neznalosti vývojového prostředí Unity a postupu při programování aplikace. Základní vytvořenou třídou je Card, která je rodičem tříd CardUnit a CardHero a obsahuje jejich společné atributy. Třída CardUnit a CardHero mají své další atributy a kontruktor. Slouží k následnému vytváření instancí jednotek a hrdinů. Třída CardUnitDatabase obsahuje list karet, ve kterém se nachází všechny herní karty. Stejné je to pro třídu CardHeroDatabase jen pro hrdiny. Herní balíčky spravuje třída Deck, která vytvoří náhodně balíčky každého hráče. Metoda StartGame() rozdá hráči postupně 6 počátečních karet s rozestupem 1 sekundy. Třída Zoom obsahuje metodu OnHoverEnter(), která inicializuje novou zvětšenou kartu, na kterou hráč najede šipkou. Následně při opuštění prostoru karty, se zavolá metoda OnHeverExit(), která dočasně zvětšenou kartu zničí. Třídy DisplayCard a DisplayHero zajišťují zobrazení karet, vykládání na bojiště a jejich rotaci při změně stavu únavy. Třída TurnSystem udržuje přehled o hráči na tahu, jeho zbývající čas na tah a upravování změnu stavu vody. Třída Game zajišťuje kompletní běh hry s metodou Attack(), která se volá při útoku na nepřátelskou jednotku na bojišti nebo hrdinu. Ve třídě SceneLoader jsou metody pro spouštění nové hry, ukončení aplikace a samotné hry v průběhu.



Obr. 22 Diagram tříd

7.5 Vývojový diagram

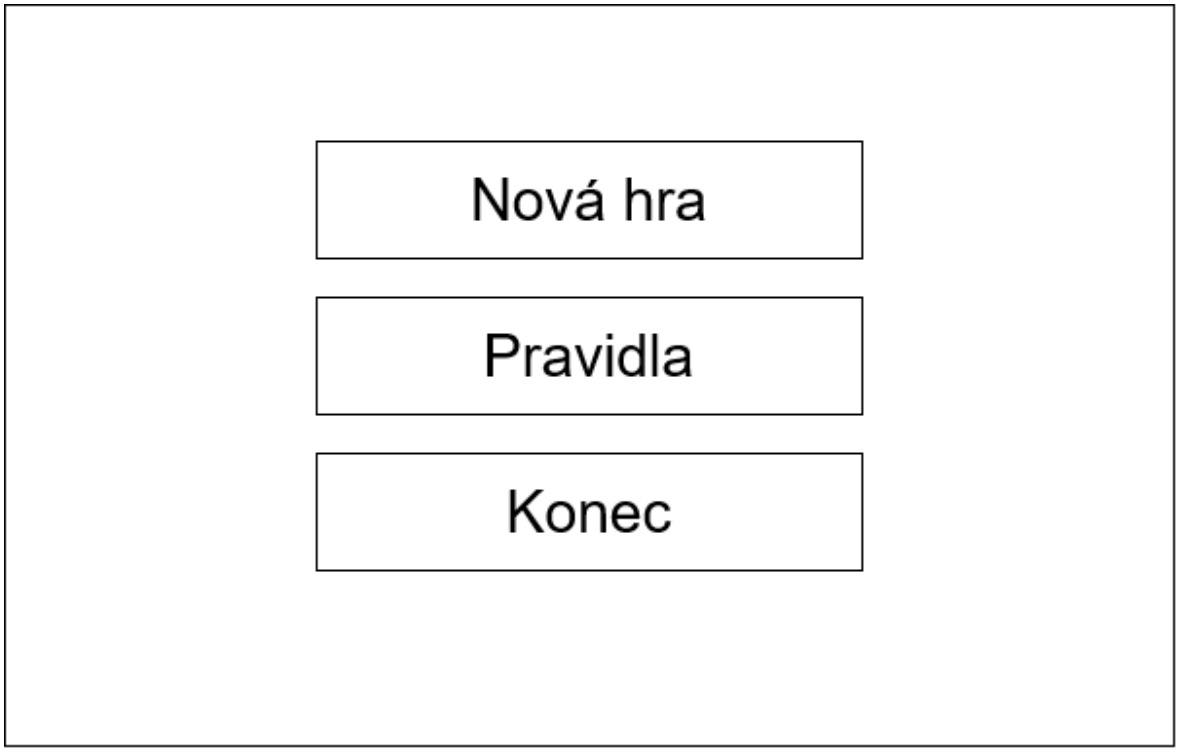
Pro lepší pochopení programování aplikace byl vytvořen vývojový diagram (Obr. 23), který slouží jako stavový automat a zachycuje všechny možné varianty, které můžou pro uživatele nastat při používání aplikace a hraní navrhované multiplatformní karetní hry.



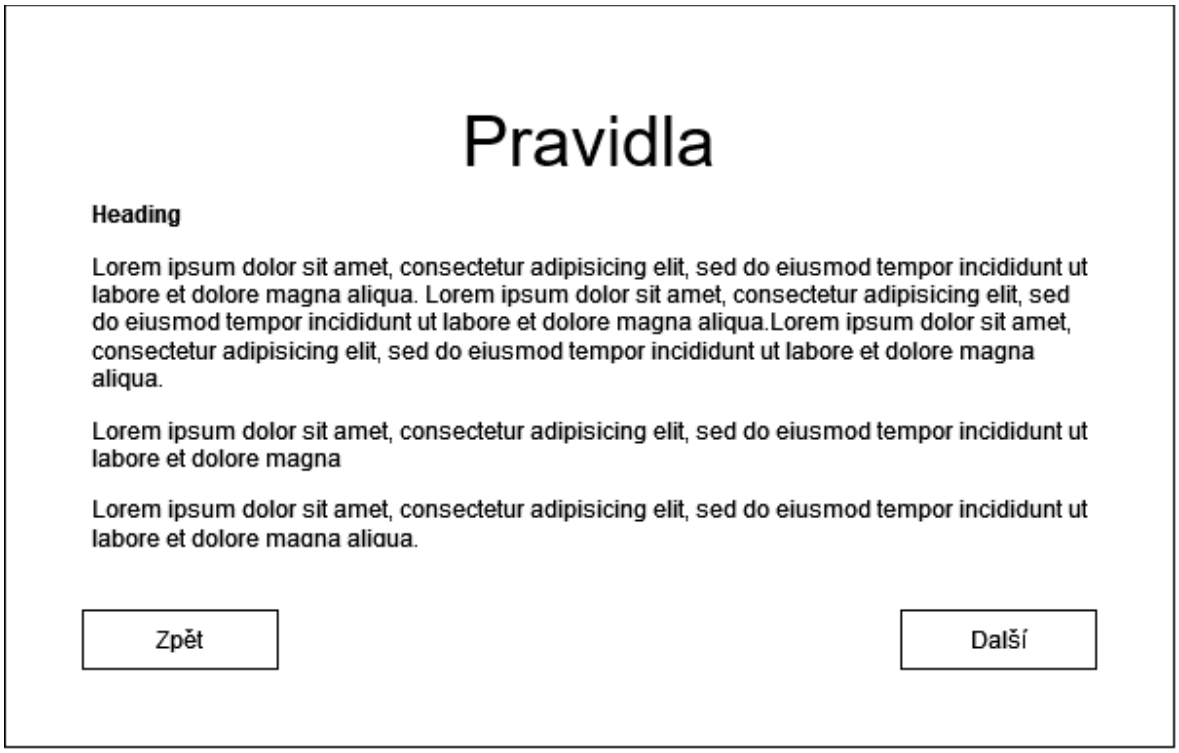
Obr. 23 Vývojový diagram funkčnosti hry

7.6 Grafický návrh aplikace

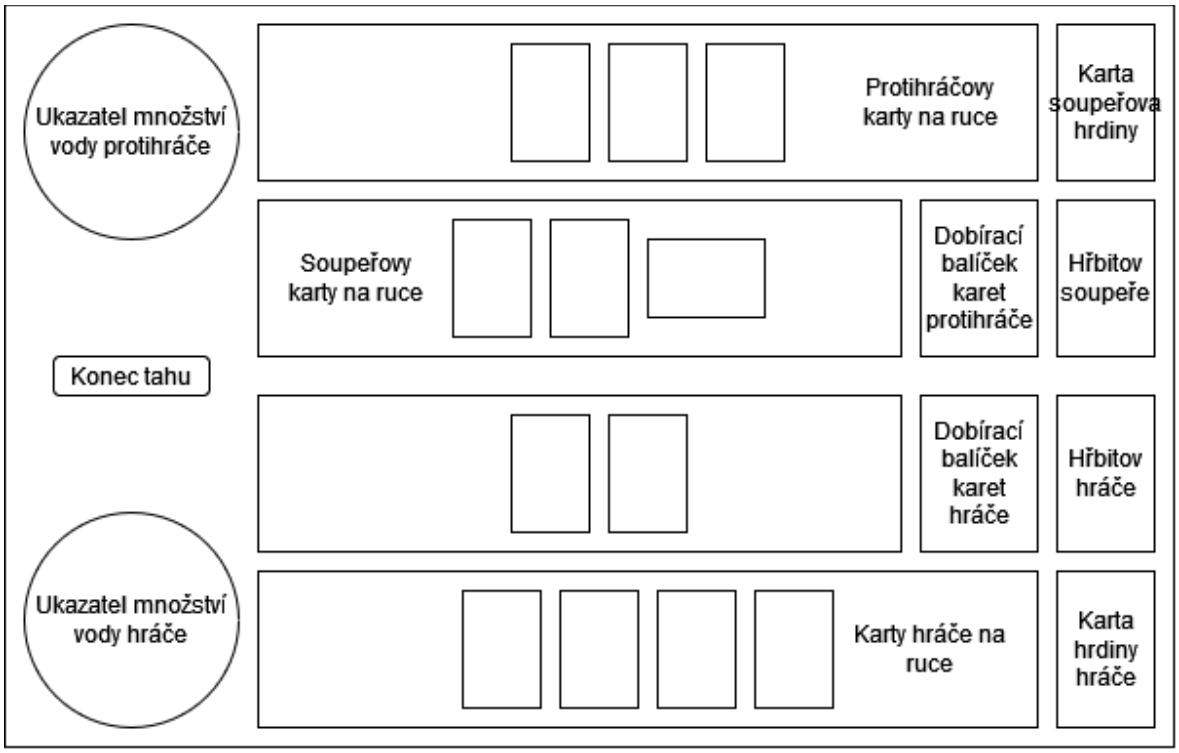
K vytvoření grafického návrhu aplikace byl použitý online software pro tvorbu vývojových diagramů „draw.io“. Návrh jednoduchého hlavního menu je vidět na Obr. 24. Dále byl vytvořen návrh pro pravidla (Obr. 25) a vizuální podoba herní plochy aplikace z pohledu obou hráčů (Obr. 26).



Obr. 24 Hlavní menu aplikace



Obr. 25 Zobrazení pravidel aplikace



Obr. 26 Návrh herní plochy aplikace

8 PRÁCE S UNITY

V této kapitole je popsán postup při instalaci vývojového prostředí Unity, postupný vývoj a implementace multiplatformní karetní hry Rone.

8.1 Systémové požadavky pro Unity

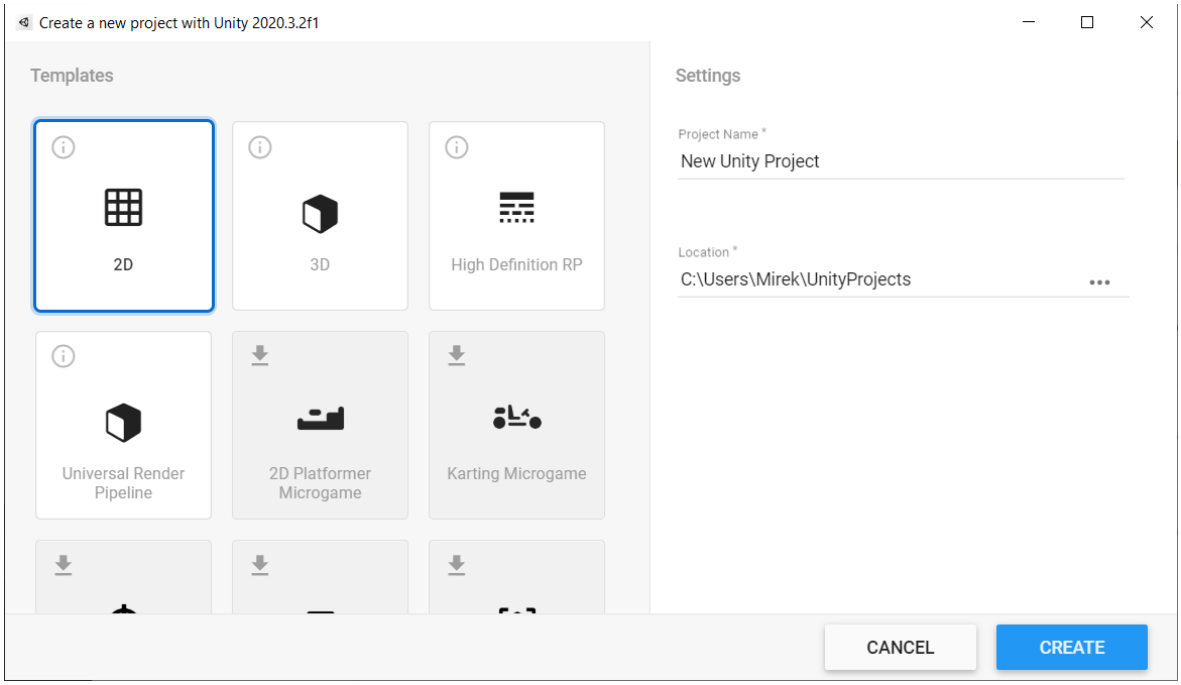
Instalaci Unity se provedla na počítači, který má procesor Intel(R) Core(TM) i3-6100U CPU @ 2.30GHz a paměť RAM 4 GB. Jedná se o minimální požadavky, které by měl hardware počítače splňovat, protože při změnách ve skriptu Microsoft Visual Studia je čekací doba klidně i 30 sekund, než dojde ke zkompilování a znovunačtení změněných skriptů.

8.2 Instalace Unity

Před instalací Unity se provede instalace Unity Hub. Jedná se o aplikaci, která zjednodušuje práci s Unity. Spravuje uživatelský účet, umožňuje instalovat a udržovat přehledně různé nainstalované verze i s jejich moduly. K dispozici je pro uživatele nabídka tutoriálů a informace o komunitě unity. Instalace samotné Unity Hub vám při instalaci Unity nabídne doporučenou verzi a pak už jen zbývá vybrat si moduly, které si přejete nainstalovat. Unity s modulem pro sestavování Windows zabírá 10 GB.

8.3 Vytvoření projektu

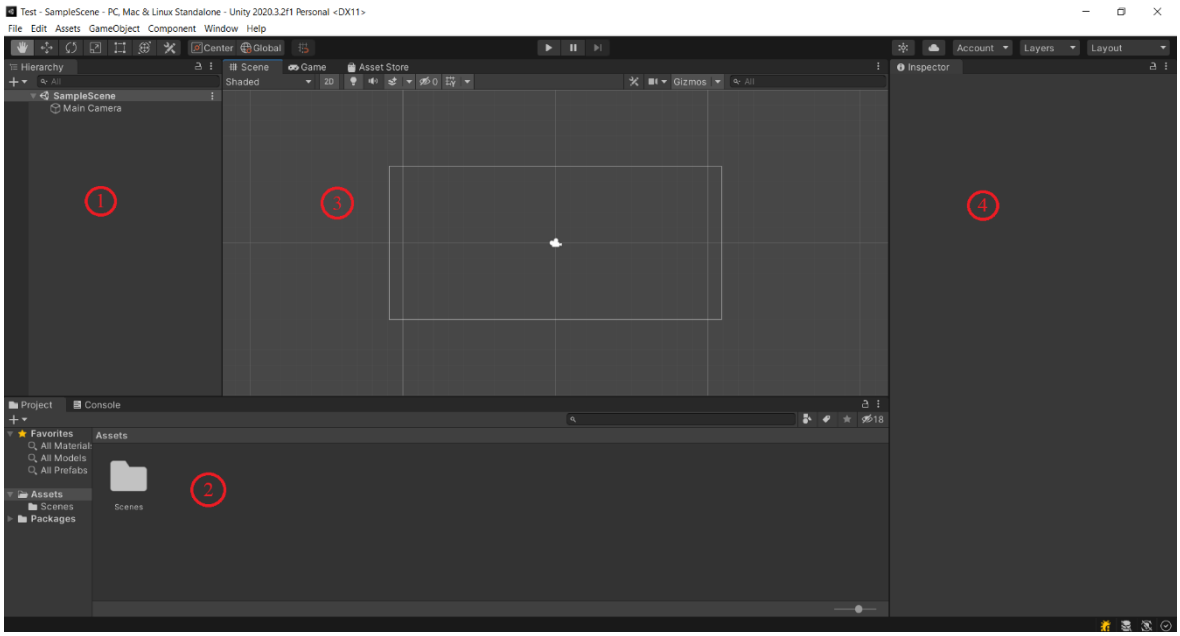
Na Obr. 27 jde vidět, že při vytváření projektu si uživatel vybere jednu z dostupných šablon. Vzhledem k tomu, že se jedná o první práci v Unity, bylo upřednostněno programovat hru ve 2D.



Obr. 27 Vytvoření projektu

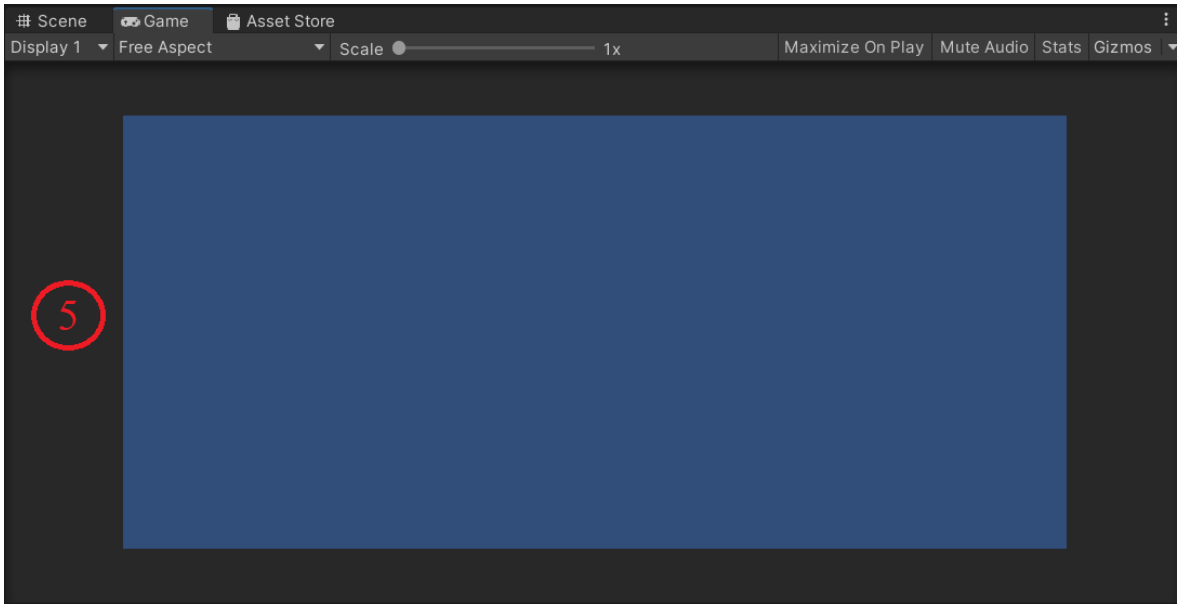
8.4 Rozhraní Unity

Po vytvoření projektu se nám zobrazí úvodní plocha (Obr. 28), která se dá rozdělit do pěti základních částí.



Obr. 28 Úvodní plocha Unity

1. Hierarchy - zobrazuje herní objekty v hierarchii, s kterými vývojář pracuje na aktuální scéně. V případě, že se objekty ve scéně překrývají, vždy rozhoduje pořadí v hierarchii. Na popředí bude zobrazen vždy ten, který je níže v hierarchii.
2. Project - okno, které zobrazuje všechny komponenty uložené v projektu. Usnadňuje tak jednodušší přístup k souborům a není potřeba využívám správce souborů operačního systému.
3. Scene - zobrazuje herní objekty, s kterými vývojář pracuje.
4. Inspector – zobrazuje podrobnosti o vybraném objektu z hierarchie, souboru nebo objektu ze scény.
5. Game – zobrazuje plochu aplikace v editoru (Obr. 29).



Obr. 29 Herní zobrazení úvodní plochy

8.5 Vytvoření hlavního menu

Na začátku si vytvoříme novou scénu s názvem „MainMenu“. Dále se vytvoří herní objekt Canvas, který by měl obsahovat veškeré uživatelské rozhraní a nastaví se, aby se scéna měnila podle velikosti displeje, na kterém se aplikace spouští. Provede se změna nastavení rozlišení na 1920x1080. Do projektu si vložíme obrázek, který se nastaví jako pozadí herní plochy.

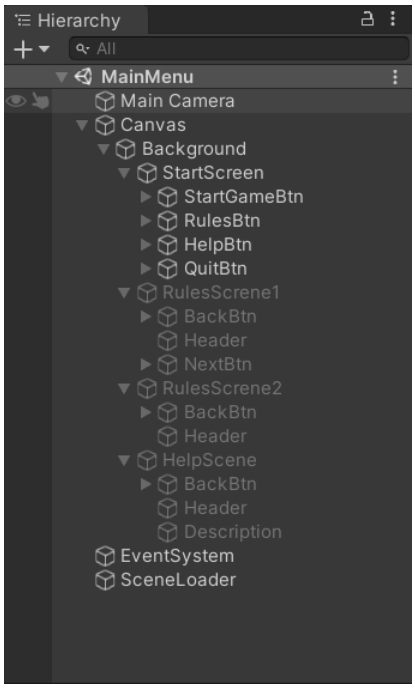
Vytvoříme si prázdný objekt StartScreen do kterého se vloží tlačítka pro ukončení aplikace, zobrazení pravidel a spuštění hry. Obdobně se postupuje i při vytvoření obrazovky pro pravidla. Unity umožňuje nastavení viditelnosti objektu, proto není třeba vytvářet více scén, ale

postačí, když se po kliknutí na tlačítko změní viditelnost potřebných objektů, a tak dojde k potřebné vizuální změně.

Pro funkčnost tlačítek v hlavním menu se vytvoří třída „SceneLoader“, která jde vidět na Obr. 30. Pro psaní skriptů byl použitý softwarový nástroj Microsoft Visual Studio. Ukázka výsledné hierarchie a hlavního menu je na obr. 31 a obr. 32.

```
public class SceneLoader : MonoBehaviour
{
    public GameObject Quit;
    Počet odkazů: 0
    public void LoadGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }
    Počet odkazů: 0
    public void QuitGame()
    {
        Application.Quit();
    }
    Počet odkazů: 0
    public void EndGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 1);
    }
    Počet odkazů: 0
    public void ContinueGame()
    {
        Quit.SetActive(false);
    }
    Zpráva Unity | Počet odkazů: 0
    public void Update()
    {
        if (Input.GetKeyDown(KeyCode.E))
        {
            Quit.SetActive(true);
        }
    }
}
```

Obr. 30 Vytvořené metody pro tlačítka v hlavním menu



Obr. 31 Hierarchie hlavního menu

Obr. 32 Výsledné hlavní menu

8.6 Vytvoření Rone

Stejně jako u hlavního menu si vytvoříme novou scénu, nazveme ji MainGame a v ní vytvoříme objekt Canvas a pozadí aplikace.

8.6.1 Vytvoření hrací karty

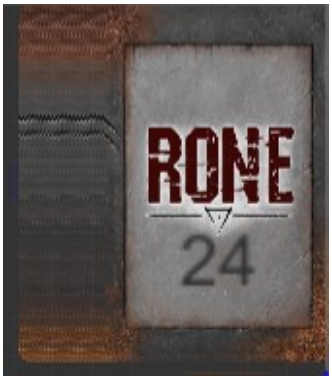
Vytvoříme si obrázek, který bude sloužit jako podklad pro kartu. Kartě si přiřadíme potomka obrázku o stejné velikosti pro zobrazování zadní strany karty. Změnou viditelnost pak půjde jednoduše otáčet kartu. Dále se vytvoří grafické zobrazování proměnných hodnot, jako je zdraví jednotky, útok na blízko, útok na dálku, cena a recyklační cena. Kartě přidáme do nejvyšší pozice další obrázek, který bude sloužit jako ohraničení karty. V případě, že je ohraničení zelené, hráč splňuje všechny podmínky pro vyložení karty na bojiště. Pokud bude karta na bojišti a ohraničení bude svítit červeně, má hráč možnost zaútočit s touto jednotkou. Samotný objekt karty se všemi předky přesuneme do vytvořené složky

Obr. 33 Vytvořená karta jednotky

Prefabs v projektu. Z objektu kartu se tak stane prefab karty, což umožňuje při vytvářet libovolné množství instancí takové karty s již dříve nastavenými vlastnostmi a komponenty. Ukázka takto vytvořené karty je na Obr. 33.

8.6.2 Vytvoření herního balíčku

Při vytváření dobíracího balíčku je třeba si vytvořit databázi všech jednotek z které se pak bude vytvářet hráčův herní balíček. Každý hráč pak při začátku hry obdrží náhodně 24 karet, které tvoří dobírací balíček a můžete ho vidět na obr. 34. Karty jsou překryty tak, aby bylo viditelné, jak se balíček vizuálně zmenšuje. Pro přesný počet karet v balíčku přidáme jednoduché počitadlo, aby měli hráči přesnější přehled o zbývajícím počtu karet, protože to jsou zároveň i životy.



Obr. 34 Dobírací balíček hráče

8.6.3 Vytvoření ukazatele vody

Ukazatel vody je vytvořen ze dvou částí, a to vnější kruh s čísly a vnitřní s ukazatelem. Podle aktuálního stavu vody hráče nastavuji rotaci vnitřního kruhu, aby ukazoval na správné čísla. Vytvořena funkce, která se volá při každé změně stavu vodu (Obr. 35).

Obr. 35 Ukazatel vody

8.6.4 Hrdina

Vytvoření grafické karty hrdiny je stejným postupem jako u karty jednotky.

Každý hráč na začátku hry obdrží svého hrdinu, tak aby každý měl jiného. Hrdina má k dispozici speciální možnosti. Jejich aktivace je nastavena na klávesách:

W – Získej 1 vodu a unav hrdinu o 1.

C – Dober si jednu kartu a unav hrdinu o 1.

Vytvořený skript při stisknutí kláves uvedených výše jde vidět na Obr. 36.

```
// Pokud je hrdina neunavený a hráč je na tahu, tak po stisknutí klávesy C si dobere kartu. Hrdina se unaví o 1.
if (Input.GetKeyDown(KeyCode.C) && TurnSystem.staticIsYourTurn == 0 && PlayerHero.staticHero[0].actualTiredness == 0)
{
    PlayerHero.staticHero[0].actualTiredness = 1;
    StartCoroutine(Draw(1));

    foreach (Transform child in HeroPanel.transform)
    {
        RotateHero(PlayerHero.staticHero[0].actualTiredness, child.GetComponent<HeroToHand>().gameObject);
    }
}

// Pokud je hrdina neunavený a hráč je na tahu, tak po stisknutí klávesy W získá 1 vodu. Hrdina se unaví o 1.
if (Input.GetKeyDown(KeyCode.W) && TurnSystem.staticIsYourTurn == 0 && PlayerHero.staticHero[0].actualTiredness == 0)
{
    PlayerHero.staticHero[0].actualTiredness = 1;
    TurnSystem.yourWater += 1;
    TurnSystem.SetWaterPointer(TurnSystem.yourWater, false, TurnSystem.staticYourPointer);

    foreach (Transform child in HeroPanel.transform)
    {
        RotateHero(PlayerHero.staticHero[0].actualTiredness, child.GetComponent<HeroToHand>().gameObject);
    }
}
```

Obr. 36 Ukázka kódu pro spuštění akcí hrdiny

Vylepšení hrdiny může každý hráč provést stisknutím klávesy U ve svém tahu. Aby se provedlo vylepšení hrdiny, musí mít hráč dostatek vody ve své zásobě.

8.6.5 Vytvoření přepínání tahů

Pro přepínání tahů je na scéně vytvořeno tlačítko, časovač a popis, který hráč je na tahu. Po stisknutí tlačítka dojde k otočení všech karet v ruce obou hráčů, aby se mohli hráči vyměnit po dobu 5 sekund. Následně se otočí hráči na tahu karty na ruce a časovač se nastaví na 2 minuty. Dále je naprogramováno, aby hráč obdržel příjem vody a karet, podle aktuální úrovně hrdiny. Dojde ke snížení únavy všech vyložených a unavených karet na bojišti o 1, stejně tak i u hrdiny (Obr. 37 a Obr. 38).

```
isYourTurn = 3;
yourTurn += 1;
yourWater += PlayerHero.staticHero[0].income;
if (yourWater > maxWater)
    yourWater = maxWater;
// Nastavení rotace obrázku, aby to odpovídalo aktuálnímu stavu vody
SetWaterPointer(yourWater, false, yourPointer);

// Pokud je hrdina unaven, dojde ke snížení únavy o 1
if (PlayerHero.staticHero[0].actualTiredness > 0)
{
    PlayerHero.staticHero[0].actualTiredness -= 1;
    foreach (Transform child in HeroPanel.transform)
    {
        PlayerDeck.RotateHero(PlayerHero.staticHero[0].actualTiredness, child.GetComponent<HeroToHand>().gameObject);
    }
}
// Prochází všechny vyložené karty na bojišti, pokud je některá unavená dojde ke snížení únavy o 1
foreach (Transform child in PlayerBattleField.transform)
{
    if (child.GetComponent<DisplayCard>().actualTiredness > 0)
    {
        child.GetComponent<DisplayCard>().actualTiredness -= 1;
        DisplayCard.RotateCard(child.GetComponent<DisplayCard>().actualTiredness, child.GetComponent<CardToHand>().gameObject);
    }
}
```

Obr. 37 Ukázka kódu po předání tahu dalšímu hráči

```
public static void RotateCard(int actualTiredness, GameObject card)
{
    if (actualTiredness == 3)
        card.transform.eulerAngles = new Vector3(0, 0, -270);
    else if (actualTiredness == 2)
        card.transform.eulerAngles = new Vector3(0, 0, -180);
    else if (actualTiredness == 1)
        card.transform.eulerAngles = new Vector3(0, 0, -90);
    else if (actualTiredness == 0)
        card.transform.eulerAngles = new Vector3(0, 0, 0);
}
```

Obr. 38 Procedura pro změnu únavy karet

8.6.6 Vykládání jednotek

Aby se dalo pohybovat s herními kartami vytvoříme si skript Draggable, který umožňuje pohyb karet. Další skript DropZone zajišťuje změnu rodiče, podle toho, kam se karta přesune. Dané skripty se vždy přiřadí herním objektům, u kterých chcete, aby se to na ně vztahovalo. Z ukázky kódu (Obr. 39), který se nachází v Update metodě se vždy kontroluje, zda je možné kartou pohybovat a vyložit na bojiště. Dále se nastavuje viditelnost ohraničení karty.

```
if((TurnSystem.yourWater >= cost) && (summoned == false) && (TurnSystem.staticIsYourTurn == 0) && (this.level <= PlayerHero.staticHero[0].level)
    canBeSummon = true;
else
    canBeSummon = false;

if(canBeSummon == true)
    gameObject.GetComponent<Draggable>().enabled = true;
else
    gameObject.GetComponent<Draggable>().enabled = false;

battleField = GameObject.Find("YourBattleField");

if ((summoned == false) && (this.transform.parent == battleField.transform))
{
    Summon();
}

if (canBeSummon == true)
    summonBorder.SetActive(true);
else
    summonBorder.SetActive(false);

if ((this.transform.parent == Hand.transform) && (TurnSystem.staticIsYourTurn != 0))
    cardBack = true;
else
    cardBack = false;
```

Obr. 39 Ověření, že karta může být vyložena na bojiště

8.6.7 Útok

Hráč má možnost zaútočit na hrdinu, tím dojde ke snížení počtu karet v balíčku podle útoku útočící jednotky. Další naprogramovaná možnost je zaútočit na nepřátelské jednotky, které se vzájemně napadnou. Začíná se útokem na dálku a v případě, že obě jednotky přežijí se pokračuje s útokem na blízko. Útočící jednotka se pak unaví podle své únavy a přeživší bránící jednotka je unavena o 1. Hlavní část tohoto kódu jde vidět na Obr. 40.

```
public void Attack()
{
    if((canAttack == true) && (summoned == true))
    {
        if (Target != null)
        {
            if (Target == Enemy)
            {
                EnemyDeck.sizeOfDeck -= shootingAttack;
                EnemyDeck.sizeOfDeck -= melleAttack;
                targeting = false;
                cantAttack = true;
                actualTiredness = tiredness;
                RotateCard(actualTiredness, this.gameObject);
            }
        }
        else
        {
            foreach(Transform child in EnemyBattleField.transform)
            {
                if(child.GetComponent<EnemyCardToHand>().isTarget == true)
                {
                    child.GetComponent<EnemyCardToHand>().hurted = child.GetComponent<EnemyCardToHand>().hurted + shootingAttack;
                    hurted = hurted + child.GetComponent<EnemyCardToHand>().shootingAttack;
                    if((child.GetComponent<EnemyCardToHand>().hurted < child.GetComponent<EnemyCardToHand>().health) && (hurted < health))
                    {
                        child.GetComponent<EnemyCardToHand>().hurted = child.GetComponent<EnemyCardToHand>().hurted + melleAttack;
                        hurted = hurted + child.GetComponent<EnemyCardToHand>().melleAttack;
                    }

                    // Pokud útočná jednotka přežila nastaví se únava podle hodnoty jednotky.
                    if (hurted < health)
                    {
                        actualTiredness = tiredness;
                        RotateCard(actualTiredness, this.gameObject);
                    }

                    // Bránící přeživší jednotka získá únavu 1.
                    if (child.GetComponent<EnemyCardToHand>().hurted < child.GetComponent<EnemyCardToHand>().health)
                    {
                        child.GetComponent<EnemyCardToHand>().actualTiredness = child.GetComponent<EnemyCardToHand>().actualTiredness + 1;
                        RotateCard(child.GetComponent<EnemyCardToHand>().actualTiredness, child.GetComponent<EnemyCardToHand>().gameObject);
                    }
                    cantAttack = true;
                }
            }
        }

    }
}
```

Obr. 40 Procedura Attack()

8.6.8 Hřbitov

Na hřbitov se přesouvají všechny zničené karty. Jakmile je jejich zdraví na 0 zavolá se procedura Destroy (Obr. 41). V případě, že se na hřbitově nachází dostatek karet, má hráč možnost provést recyklaci horní karty v balíčku kliknutím myši na kartu. Karta se přesune hráči na bojiště s únavou 2 a dojde ke zničení karet ve hřbitově podle recyklační ceny karty (Obr. 42).

```
public void Destroy()
{
    canBeDestroyed = true;
    if (canBeDestroyed == true)
    {
        this.actualTiredness = 0;
        this.hurted = 0;
        RotateCard(this.actualTiredness, this.gameObject);
        this.transform.SetParent(Graveyard.transform);


        if (Graveyard.transform.childCount == 1)
            this.gameObject.SetActive(true);
        else
            this.gameObject.SetActive(false);

        canBeDestroyed = false;
        summoned = false;
        beInGraveyard = true;
        this.tag = "Graveyard";
    }
}
```

Obr. 41 Procedura Destroy()

```
public void ReturnFromGraveyard()
{
    if (this.tag == "Graveyard")
    {
        int recyclationCost = 0;
        Graveyard = GameObject.Find("YourGraveyard");
        battleField = GameObject.Find("YourBattleField");

        summoned = true;
        beInGraveyard = false;
        int x = 0;
        foreach (Transform child in Graveyard.transform)
        {
            if (x == 1 && recyclationCost > 0)
            {
                // Odstranění karet ze hřbitova
                Destroy(child.gameObject);

                recyclationCost--;
            }

            if ((Graveyard.transform.childCount > child.GetComponent<DisplayCard>().recyclationCost) &&
               (child.GetComponent<DisplayCard>().level <= PlayerHero.staticHero[0].level) && x == 0)
            {
                x++;
                // Vezmi horní kartu ze hřbitova a dej jí hráči do pole s únavou 2
                child.transform.SetParent(battleField.transform, false);
                child.GetComponent<DisplayCard>().actualTiredness = 2;
                RotateCard(child.GetComponent<DisplayCard>().actualTiredness, child.gameObject);
                child.tag = "Untagged";
                recyclationCost = child.GetComponent<DisplayCard>().recyclationCost;
            }

        }
    }
}
```

Obr. 42 Procedura ReturnFromGraveyard()

8.6.9 Konec hry

Jakmile jeden z hráčů nemá žádné karty na ruce a v herním balíčku, dojde k ukončení hry.

8.6.10 Výsledný stav aplikace

Výsledný stav aplikace je viditelný na Obr. 43. Kdy hráč na tahu vidí pouze své karty a má zeleně označené ty, které může hrát. Na pravé straně je jeho hrdina a dobírací balíček ukazující aktuálně zbývající počet karet v balíčku. Aktuální stav vody je 6.

Obr. 43 Konečný stav vytvořené aplikace

8.7 Moje doporučení pro práci v Unity

8.7.1 Požadavky na hardware

Systémové požadavky uváděné na oficiální stránce Unity jsou považovány za minimální a pro práci na rozsáhlejším projektu nedostatečné. Proto zde pro představu uvedu hardware používaného počítače.

Hardware používaného počítače

Procesor - Intel(R) Core(TM) i3-6100U CPU @ 2.30GHz, 2304 Mhz, jádra: 2, logické procesory: 4.

Operační paměť – 4 GB

Kompilace skriptů mně trvala od 10 sekund až po 30 v závislosti na počtu změněných skriptů. Sestavování celého projektu trvalo v řádech minut.

Doporučené požadavky na hardware

Uvedené požadavky jsou mým doporučením na základě mých zkušeností při práci v Unity.

Procesor – AMD Ryzen 5 1600 / Intel Core i5-12600

Počet jader procesoru – alespoň 6.

Operační paměť – minimálně 8 GB.

8.7.2 Používání ScriptableObjects

Doporučuji si rozhodně nastudovat skriptovací objekty, protože se jedná o vhodný způsob uchovávání dat a pro budoucí programování v Unity je to nezbytná znalost.

8.7.3 Používání Asset Store

K dispozici pro všechny uživatele je Asset store, kde máte možnost si stáhnout zdarma nebo zakoupit již vytvořené produkty. Určitě si před programováním projděte, zda to již někdo nenaprogramoval, může to být zdarma nebo v dostupné ceně a v některých případech se určitě vyplatí investovat, pokud plánujete komerční využití vyvíjené aplikace.

8.7.4 Unity community

Pokud si nevíte s něčím rady, určitě vždy odpovědi hledejte na fórum Unity. Komunita lidí je hodně rozsáhlá, takže v případě, že nenajdete hledanou odpověď, tak na Váš položený dotaz brzy někdo odpoví.

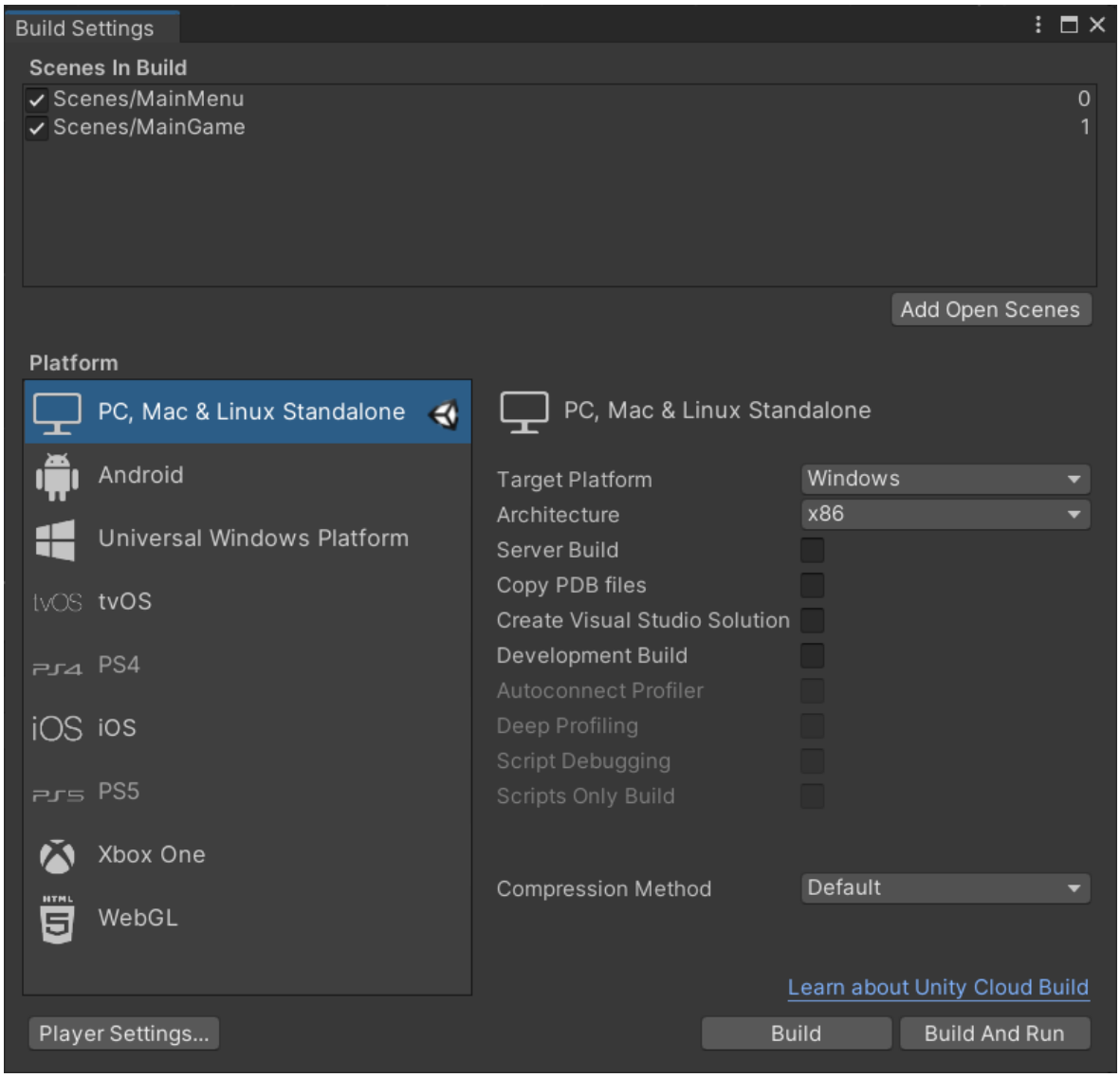
9 TESTOVÁNÍ FUNKČNOSTI NA VÍCE PLATFORMÁCH

Na závěr byl provedeno testování aplikace na více platformách.

9.1 Testované platformy

Testování aplikace bylo prováděno na dvou operačních systémech. Před každým testováním se v „Build Settings“, které je viditelné na Obr. 44, provede sestavení projektu pro konkrétní OS (Operační systém).

Aby byla možnost sestavit aplikaci pro Linux musí být přes Unity Hub v používané verzi Unity nainstalovány moduly „Linux Build Support (IL2CPP)“ a „Linux Build Support (Mono)“. Každý takový modul zabírá přibližně 1 GB místa paměti na disku.



Obr. 44 Nastavení Build settings v Unity

9.1.1 Windows 10

Po spuštění aplikace Rone.exe došlo k zobrazení herního menu, jak bylo vytvořeno. Funkčnost všech tlačítek menu byla otestována a fungují dle očekávání. Po spuštění hry dojde k zobrazení herní plochy a k rozdání herních karet pro oba hráče. Komponenty zobrazují všechno v pořádku. Pohyb karet je plynulý a herní mechanismus funguje, jak byl naprogramován. Testování proběhlo v pořádku. Velikost sestavené složky je 178 MB.

9.1.2 Linux

Při testování aplikace na Linuxu se spuštění Rone.x86_64 provedlo přes příkazový řádek z důvodu nastavení oprávnění pro spuštění aplikace. Testování proběhlo se stejnými výsledky jako u Windows 10 s jediným zaznamenaným rozdílem ve zhoršené grafice aplikace. Při hledání důvody bylo zjištěno, že aplikace má horší grafiku, protože test byl proveden na display s HD kvalitou a u Windows 10 byl použit display s full HD. Velikost sestavené složky je 225 MB.

ZÁVĚR

Cílem této bakalářské práce bylo seznámit čtenáře s vývojem multiplatformní karetní hry. Seznámit je se správným postupem, kdy před samotným programováním aplikace si rozmyslí, jaké mají požadavky na aplikaci a její funkčnost. Pomocí UML dosáhneme, aby byl vytvořen úspěšný systém bez zbytečných a zásadních změn v průběhu implementace.

V teoretické části se zaměřuji na nejznámější herní enginy ve světě, které se používají a rozebírám jejich hlavní výhody a nevýhody. Na základě těchto teoretických znalostí si pak vybírám vhodný engine pro svoji aplikaci. Dále se zabývám rozborem UML analýzy, která by se měla vždycky používat u projektů, protože nám umožňuje si graficky vizualizovat, specifikovat a konstruovat struktury a vztahy rozsáhlejšího projektu za pomocí diagramů.

Praktická část se nejprve zabývá výběrem a odůvodněním výběru herního enginu a stolní karetní hry Rone, která ještě nebyla digitalizována. Dále jsou popsány pravidla této hry a vypracována grafická analýza projektu za pomocí sepsaných požadavků a diagramů.

Nejdůležitější praktickou částí je programování aplikace, kdy pro její implementaci byl použit herní engine Unity 2020.3.2f1 a vývojové prostředí Visual Studio 2019. Postupně je popsána strukturu projektu a důležité části projektu. Konkrétně vytvoření objektu karty, herního balíčku, ukazatele stavu vody, hrdiny, zpracováním funkčnosti tahu hráče, přepínání tahů mezi hráči, vykládání jednotek z ruky na bojiště, vzájemné útočení jednotek mezi sebou, přesun zničených karet na hřbitov a vítězství ve hře.

Dále je popsáno pár doporučení pro budoucí programátory v Unity, co by měli používat a čemu se vyvarovat.

Na závěr vytvořenou aplikaci testuji na více platformách s úspěšným provedením.

Závěrem můžu tedy říct, že se jedná o funkční aplikaci, ale některé její části nejsou ideálně naprogramovány a s nabitými zkušenostmi po prvním programování v Unity bych sám provedl mnoho změn a vylepšení. Z časových důvodů a náročnosti projektu se nepodařilo implementovat karty taktiky do projektu a ze stejného důvodu nebyla provedena implementace plánované síťové varianty.

SEZNAM POUŽITÉ LITERATURY

[1] TIŠŇOVSKÝ, Pavel. Historie vývoje počítačových her (1.část - první milníky). Root [online]. 2011, 10. 11. 2011 [cit. 2021-01-12]. Dostupné z: https://www.root.cz/clanky/historie-vyvoje-pocitacovych-her-1-cast-prvni-milniky/

[2] OXO. Wikiwand [online]. [cit. 2021-01-12]. Dostupné z: https://www.wikiwand.com/en/OXO

[3] TIŠŇOVSKÝ, Pavel. Historie vývoje počítačových her (2.část - věk simulací). Root [online]. 2011, 22. 11. 2011 [cit. 2021-01-12]. Dostupné z: https://www.root.cz/clanky/historie-vyvoje-pocitacovych-her-2-cast-vek-simulaci/

[4] SLÁMA, David. Chléb a hry: Historie počítačových her. Zive [online]. 2009, 6. 7. 2009 [cit. 2021-01-12]. Dostupné z: https://www.zive.cz/clanky/chleb-a-hry-historie-pocitacovych-her/sc-3-a-147762/default.aspx

[5] Quake III Arena. In: *Steampowered* [online]. c2021 [cit. 2021-02-27]. Dostupné z: https://store.steampowered.com/app/2200/Quake_III_Arena/

[6] HALPERN, Jared. The What and Why of Game Engines. Medium [online]. 2018, 11. 12. 2018 [cit. 2021-03-05]. Dostupné z: https://medium.com/@jaredehalpern/the-what-and-why-of-game-engines-f2b89a46d01f

[7] MOHEBALI, Ali a Thiam KIAN. *Redefining Game Engine Architecture through Concurrency* [online]. Malaysia, 2014 [cit. 2022-05-13]. Dostupné z: https://eprints.um.edu.my/13036/1/somet201456.pdf. Faculty of Computer Science and Information Technology, University of Malaya.

[8] TYLER, Dustin. The 25 Top Video Game Engines (For Beginners). *GameDesign* [online]. 2022, 2022 [cit. 2022-05-10]. Dostupné z: https://www.gamedesigning.org/career/video-game-engines/

[9] DAR, Renana. Top 7 Gaming Engines You Should Consider for 2021. *IncrediBuild* [online]. 2022, 1.2.2022 [cit. 2022-05-10]. Dostupné z: https://www.incredibuild.com/blog/top-7-gaming-engines-you-should-consider-for-2020

[10] Unity [online]. Unity Technologies, c2021 [cit. 2021-03-06]. Dostupné z: https://unity.com/

[11] Unity vs. Unreal: How to Choose the Best Game Engine. Medium [online]. N-iX, 2018, 11. 09. 2018 [cit. 2021-03-07]. Dostupné z: https://medium.com/@N_iX/unity-vs-unreal-how-to-choose-the-best-game-engine-d3dbb4add73c

[12] UnrealEngine [online]. Epic Games, c2004-2021 [cit. 2021-03-09]. Dostupné z: https://www.unrealengine.com/en-US/

[13] DRAKE, Jeff. 15 Great Games That Use The Unreal 4 Game Engine. In: *Thegamer* [online]. 2020, 08.06.2020 [cit. 2021-03-09]. Dostupné z: https://www.thegamer.com/great-games-use-unreal-4-game-engine/

[14] Godot Engine [online]. Linietsky, c2007-2021 [cit. 2021-03-09]. Dostupné z: https://godotengine.org/

[15] BUCKLEY, Ian. 10 Reasons to Use Godot Engine for Developing Your Next Game. Makeuseof [online]. 2019, 25. 04. 2019 [cit. 2021-03-10]. Dostupné z: https://www.makeuseof.com/tag/reasons-godot-engine-game-development/

[16] DEALESSANDRI, Marie. What is the best game engine: is CryEngine right for you? Gameindustry [online]. 2020, 16. 01. 2020 [cit. 2021-03-18]. Dostupné z: https://www.gamesindustry.biz/articles/2020-01-16-what-is-the-best-game-engine-is-cryengine-the-right-game-engine-for-you

[17] CryEngine [online]. Crytek, c2021 [cit. 2021-03-18]. Dostupné z: https://www.cryengine.com/

[18] CRYENGINE Reviews [online]. In: . [cit. 2021-03-22]. Dostupné z: https://www.capterra.com/p/210664/CRYENGINE/reviews/

[19] DEALESSANDRI, Marie. What is the best game engine: is GameMaker right for you? Gameindustry [online]. 2020, 16. 01. 2020 [cit. 2021-04-03]. Dostupné z: https://www.gamesindustry.biz/articles/2020-01-16-what-is-the-best-game-engine-is-gamemaker-the-right-game-engine-for-you

[20] 9 Till Void. In: *Yoyogames* [online]. [cit. 2021-04-12]. Dostupné z: https://www.yoyogames.com/en

[21] DAR, Renara. Top 7 Gaming Engines You Should Consider for 2021. *Incredibuild* [online]. Incredibuild Software, 2021, 1.2.2021 [cit. 2022-05-13]. Dostupné z: https://www.incredibuild.com/blog/top-7-gaming-engines-you-should-consider-for-2020

[22]

[23] ČÁPKA, David. Lekce 1 - Úvod do UML. Itnetwork [online]. Čápka, c2021 [cit. 2021-04-16]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-uvod-historie-vyznam-a-diagramy

[24] Jednoduchý návod k UML diagramům a modelování databází. Microsoft [online]. Microsoft 365, 2019, 24. 9. 2019 [cit. 2021-04-16]. Dostupné z: https://www.microsoft.com/cs-cz/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling

[25] Analytická specifikace a její zpracování [online]. , 4-10 [cit. 2021-04-19]. Dostupné z: http://ecom.ef.jcu.cz/web2/download/teorie/04_analyticka_specifikace_a_jeji_zpracovani.pdf

[26] Definice [online]. [cit. 2021-04-17]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=18435

[27] JAKUBEC, David. Využití softwarového jazyka UML pro programování řídicích systémů PLC [online]. Praha, 2018 [cit. 2021-04-17]. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/77593/F2-BP-2018-Jakubec-David-vyuziti-softwaroveho-jazyka.pdf?sequence=-1&isAllowed=y. Bakalářská práce. České vysoké učení technické v Praze Fakulta strojní. Vedoucí práce Ing. Mgr. Jakub Jura, Ph.D.

[28] ČÁPKA, David. Lekce 2 - UML - Use Case Diagram. Itnetwork [online]. Čápka, c2021 [cit. 2021-04-18]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram

[29] ČÁPKA, David. Lekce 3 - UML - Use Case Specifikace. Itnetwork [online]. Čápka [cit. 2021-04-20]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-use-case-specifikace-diagram

[30] Návody a příklady USE CASE DIAGRAM [online]. , 2 [cit. 2021-04-19]. Dostupné z: http://ecom.ef.jcu.cz/web2/download/podklady/use-case-diagramy.pdf

[31] Sekvenční diagram. *Orca* [online]. [cit. 2021-04-20]. Dostupné z: http://orca.xf.cz/ooms/010/010.htm

[32] Sekvenční diagram. *Uml* [online]. Rejnková, c2009 [cit. 2021-04-20]. Dostupné z: http://uml.czweb.org/sekvencni_diagram.htm

[33] ČÁPKA, David. Lekce 5 - UML - Class diagram. *Itnetwork* [online]. Čápka [cit. 2021-04-20]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-class-diagram-tridni-model

[34] KOMÁREK, Martin. UML diagram tříd -hledání analytických tříd, relací, atributů. *Moodle.fel.cvut* [online]. [cit. 2021-04-20]. Dostupné z: https://moodle.fel.cvut.cz/pluginfile.php/103006/mod_label/intro/Prednask5UMLdiagram-Trid_a_Stavu.pdf

[35] Rone - Race of New Era (česky). *Ostrov-her* [online]. © 2022 [cit. 2022-05-14]. Dostupné z: https://www.ostrov-her.cz/d/rone-race-of-new-era-cesky-1005023/

[36] ČÁPKA, David. Lekce 4 - UML - Doménový model. *Itnetwork* [online]. Čápka [cit. 2021-04-20]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-domenovy-model-diagram

[37] CHROBOCZEK, Martin. Grafická uživatelská rozhraní v Qt a C++. Brno: Computer Press, 2013. ISBN 9788025141243.

[38] HALPERN, Jared. Developing 2D Games with Unity: Independent Game Programming with C#. APress, 2018. ISBN 1484237714

[39] BARRERA, Ray, Aungh Sithu KYAW a Thet Naing SWE. Unity 2017 Game AI Programming. 3rd ed. Birmingham: Packt Publishing, 2018. ISBN 978-1-78847-790-1.

[40] PRATA, Stephen. Mistrovství v C++. 3., aktualiz. vyd. Přeložil Boris SOKOL. Brno: Computer Press, 2007. Bestseller (Computer Press). ISBN 978-80-251-1749-1.

[41] BILL EVJEN, Christian Nagel, et al. C 2008 Programujeme profesionálně. Praha : Computer Press, 2009. 1904 s. ISBN 978-80-251-2401-7.

[42] System Requirements. *Sparxsystems* [online]. [cit. 2022-04-26]. Dostupné z: https://www.sw.cz/enterprise-architect-professional-edition/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

FPS First person shooter (Střílečka z pohledu první osoby)

AI Artificial intelligence (Umělá inteligence)

GMS2 GameMaker Studio 2.

GML GameMaker Language

UML Unifield Modeling Language (Unifikovaný modelovací jazyk)

UC Use case (diagram případu užití)

EA Enterprise Architect

OS Operační systém

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: PRAVIDLA KARETNÍ HRY RONE

Karetní hra Rone je zasazena do postapokalyptického světa, kde se přeživší lidé snaží přežít následky nukleární války.

Hráči jsou ve hře představováni kartou svého hrdiny a hrají jednotky nebo jednorázové efekty (taktiky) s cílem zničit všechny karty svého soupeře.

Celkový počet karet v balíčku a v hráčově ruce představuje celkový počet životů hráčova hrdiny. Za každé utržené zranění musí hráč zničit kartu ze své ruky nebo svrchu svého balíčku.

V okamžiku, kdy hráč přijde o všechny karty v ruce nebo balíčku, ihned prohrává hru. Protože každá karta, která je zahrána v průběhu hry, představuje jeden ztracený život, jsou hráči nuceni přemýšlet a zvažovat přínos každé karty.

Hráči musí hledat rovnováhu mezi snahou o získání kontroly nad bojištěm, což jim umožňuje hrát karty efektivněji, a pokusy o zabití protihráčova hrdiny, což je hlavním cílem hry.

Přehled komponent

Počitadlo vody

Ve hře se vyskytuje pouze jedna surovina a tou je voda. Aktuální stav vody je zaznamenáván na tomto počítadle.

Karty

Karetní hra Rone obsahuje 156 karet. Společnými prvky na všech kartách jsou:

1. **Úroveň** – Představuje úroveň karty a vyjadřuje, jakou minimální úroveň musí mít hráčův hrdina, aby tuto kartu mohl zahrát nebo recyklovat.
2. **Aktivační cena** – Množství vody, které musí hráč zaplatit ze své zásoby vody, aby mohl tuto kartu zahrát.
3. **Recyklační cena** – Množství karet, které se musí odstranit svrchu hřbitova, aby se dala karta recyklovat.
4. **Jméno** – Název karty.
5. **Symbol edice**: Karty se stejným symbolem vytvářejí dohromady kompletní edici. Tento symbol nemá žádný herní význam a používá se pouze pro rozlišení karet různých edic.

6. **Typ karty** – Představuje druh karty (hrdina, technologie, jednotka, taktika) a určuje tak její způsob využití.

Obr.45 Karta jednotky Auza

7.**Podtyp karty** – Podtypy karet se vyskytují pouze u jednotek a hrdinů a vyjadřují, k jaké rase daná jednotka/hrdina patří.

8.**Text karty** – Představuje zvláštní schopnosti karty. Text psaný kurzívou pouze dokresluje příběh a nemá vliv na hru.

9.**Ilustrátor** – Jméno ilustrátora obrázku na kartě.

10. **Číslo karty** – Toto číslo nemá herní význam, slouží pouze k doplňující identifikaci.

Ve hře se nacházejí čtyři typy karet rozdělené do dvou kategorií: Trvalé karty (hrdinové, technologie a jednotky) a Jednorázové efekty (taktiky). Ukázka jednotky je na Obr. 18.

Trvalé karty

Trvalé karty jsou karty, které po zahrání či recyklaci zůstanou na bojišti do doby, než jsou zničeny, či jinak odstraněni z bojiště. Trvalé karty mohou mít schopnosti, které vyžadují jejich otočení o určitý počet fázi únavy – každá fáze únavy je vyjádřena otočením karty o 90° po směru hodinových ručiček. Jakmile karta využije svoji aktivní schopnost, je X-krát unavena, kde X je cena únavy takové schopnosti. Hráčům s tímto neobvyklým systémem pomáhají tzv. rotační symboly, které jsou zobrazeny na třech stranách každé trvalé karty, takže hráči dokáží jednoduše zjistit, jak moc je daná karta unavená.

Obr. 46 Karta jednotky Žoldačka

Společné prvky na trvalých kartách:

11. **První rotační symbol** – Karta se považuje za 1x unavenou.

12. **Druhý rotační symbol** – Karta se považuje za 2x unavenou.

13. **Třetí rotační symbol** – Karta se považuje za 3x unavenou.

Aktivní stav – Pokud není karta unavená (tzn. Je otočena názvem směrem k hráči), považuje se za aktivní a může provádět aktivní schopnost.

Jednorázové efekty

Jednorázové efekty (taktiky) jsou karty, které se po zahrání a vyhodnocení efektů odloží na hřbitov. Tento typ karet je jednoduše rozpoznatelný od ostatních, protože nemá žádné rotační symboly.

Hrdinové

Každý hráč hraje za jednoho hrdinu (Obr. 20), kterého ve hře zastupuje sada tří karet. Tyto karty představují schopnosti hrdiny na třech úrovních. Karty hrdiny nabízejí hráči aktivní a pasivní schopnosti a také poskytují příjem vody a karet. Hráčův hrdina začíná na první úrovni a po zaplacení aktivační ceny dojde k vylepšení hrdiny. Pozice a fáze únavy hrdiny zůstávají nezměněny.

Obr. 47 Karta hrdiny

Společné prvky na kartách hrdiny:

14. **Příjem vody** – Představuje množství vody, které hráč dostane na začátku každého kola.
15. **Příjem karet** – Představuje množství karet, které si hráč může dobrat ze svého herního balíčku do ruky na začátku každého kola. Hráč si nemusí dobrat žádnou kartu.
16. **Ekonomická schopnost** – Každý hrdina má základní aktivní schopnost, pomocí které může získat více vody či karet.
17. **Pasivní schopnost** – Každý hrdina může mít pasivní schopnost, která se ve většině případů spouští za určité podmínky.
18. **Aktivní schopnost** – Hrdina může mít další aktivní schopnost.

Jednotky

Jednotky vyložené na bojišti se mohou účastnit soubojů s ostatními jednotkami či hrdiny, buďto jako útočníci nebo jako obránci. Jednotky jsou od ostatních typů karet rozpoznatelné podle toho, že mají ukazatele životů, útoku a střelby.

19. **Životy** – Představují počáteční počet životů jednotky po vyložení. Počet životů se v průběhu hry může měnit, dokonce může přesáhnout svůj počáteční stav.

Obr. 48 Karta jednotky Behemot

20. **Útok** – Ukazuje hodnotu útoku jednotky, představující počet zranění při boji tváří v tvář, které udělí protivníkovi, s kterým bojuje.

21. **Střelba** – Ukazuje hodnotu střelby jednotky, představující počet zranění na dálku protivníkovi, s kterým bojuje.

22. **Rychlost útoku** – je žlutě podbarvený rotační symbol, představující kolikrát musí být jednotka unavena, pokud je s ní proveden útok.

Technologie

Technologie jsou bonusové karty, které mají hráči k dispozici již od začátku hry a je pouze na nich, kdy je využijí. Tyto karty se nepočítají mezi karty, které má hráč v ruce. Hráč může mít na bojišti současně maximálně 3 karty technologií.

Taktiky

Taktiky jsou jednorázové efekty, které mohou být zahrány kdykoliv a v reakci na jakoukoliv událost ve hře.

Herní zóny

Herní balíček

Herní balíček obsahuje karty jednotek a taktiky, celkový počet karet je 24. V průběhu hry si hráči dobírají svrchu svého herního balíčku do ruky.

Hráčova ruka

Všechny dobrané karty jdou do hráčovi ruky, odkud můžou být zahrány. Ve hře není žádné omezení na maximální počet karet v ruce hráče.

Bojiště

Každá trvalá karta se pokládá po zahrání či recyklaci na bojiště, na kterém zůstane, dokud není zničena nebo jinak odstraněna z bojiště. Každá recyklovaná trvalá karta se po příchodu na bojiště 2x unaví.

Hřbitov

Zničené karty se odkládají do hráčova odhazovacího balíčku lícem vzhůru (hřbitov). Jakmile má být karta odložena na hřbitov, hráč musí rozhodnout, zda se karta položí navrch či dospod hřbitova. Recyklovat a tím vrátit kartu zpět do hry lze pouze vrchní kartu na hřbitově.

Balíček technologií

Balíček technologií obsahuje až 5 karet technologií. Tyto karty jsou dostupné hráči od začátku hry. Karty v balíčku technologií se nepočítají mezi karty, které má hráč na ruce.

Libovolná karta technologie může být zahrána po zaplacení ceny aktivační ceny kdykoliv v průběhu hráčovy hlavní fáze.

Hráč může mít na bojišti současně maximálně 3 karty technologií.

Základní herní princip

Propojení karet s životy hrdiny

Počet karet v herním balíčku každého hráče je propojen s počtem životů hráčova hrdiny. Za každé zranění, které hrdiny utrží, musí hráč zničit jednu ze svých karet – volí mezi vrchní kartou ze svého herního balíčku nebo libovolnou kartou ze své ruky. Všechny takto zničené karty se odkládají na hřbitov. Jakmile hráč přijde o všechny své karty na ruce a v herním balíčku, prohrává hru.

Před první hrou

První hru doporučuji hrát bez technologií. Technologie vyžadují větší hráčovu pozornost a občas využívají složitějších efektů, které můžou v začátcích hru komplikovat. Bez technologií je mnohem snazší naučit se základní herní principy a mechaniky.

Fáze kola

RONE se hraje na kola, v nichž se hráči střídají. Každé hráčovo kolo je rozděleno do 5 fází.

1. Začátek kola
2. Fáze příjmu
3. Fáze obnovy
4. Hlavní fáze
5. Konec kola

Kolo je u konce, jakmile je odehráno všech pět fází daného kola a další hráč začne své vlastní kolo. Hráč, jehož kolo právě probíhá, se nazývá aktivní hráč.

Začátek kola

Některé karty na bojišti mohou mít schopnosti, které se aktivují na začátku kola. Aktivní hráč musí provést všechny efekty těchto karet v libovolném pořadí.

Fáze příjmu

V této fázi hráč obdrží vodu do své zásoby. Hodnota se bere podle úrovně hrdiny a dalších vyložených technologií. Následně si aktivní hráč může dobrat až X karet, kde X je číslo uvedené na aktuální kartě hrdiny v sekci příjmu karet.

Fáze obnovy

Každá karta na bojišti, která je unavená, se 1x obnoví.

Hlavní fáze

Během hlavní fáze může hráč hrát karty a využívat jejich schopnosti. Seznam akcí je rozdělen na 2 kategorie – pomalé efekty a rychlé efekty.

Pomalé efekty

- Hraní jednotky
- Hraní technologie
- Zvýšení úrovně hrdiny
- Recyklace jednotky
- Recyklace taktiky
- Ohlášení útoku

Pomalé efekty můžou být použity pouze v průběhu hráčova kola.

Rychlé efekty a zásobník

- Hraní taktiky
- Provedení aktivní schopnosti

Rychlé efekty mohou být zahrány kdykoliv, i v průběhu protihráčova tahu, a to v reakci na cokoliv, co se stalo ve hře. Všechny tyto zahrané karty jsou umístěny do pomyslného zásobníku a poté, co žádný z hráčů již nechce přidat další efekt do zásobníku, jsou vyhodnocovány postupně od posledního až po první (FILO).

Hraní karet

Kartu z ruky lze zahrát, pokud má stejnou nebo nižší úroveň než je aktuální úroveň hrdiny a hráč má dostatek vody pro zaplacení aktivační ceny.

Jednotky a technologie se vyloží v aktivním stavu na bojiště.

Taktika se vyhodnotí a odstraní se na hřbitov.

Zvýšení úrovně hrdiny

Aktivační cena na kartě hrdiny představuje cenu, kterou musí hráč zaplatit, aby jeho hrdina zvýšil svou úroveň. Pozice a fáze únavy hrdiny zůstávají nezměněny.

Recyklace karet

Hráč může vždy recyklovat pouze tu kartu, která je vrchní kartou hřbitova. Kdykoliv, kdy se má přidat karta na hřbitov, hráč rozhoduje, zda se dá dospod nebo navrch balíčku.

Aby mohl hráč recyklovat kartu, musí zaplatit recyklační cenu. To znamená odstranit požadovaný počet následujících vrchních karet ze svého hřbitova.

Stejně jako u hraní karet, musí být hráčův hrdina alespoň na stejné úrovni jako recyklovaná karta.

Recyklovaná jednotka se na bojiště vrátí 2x unavená.

Taktika se po recyklaci vyhodnotí a následně se odstraní ze hry.

Ohlášení útoku

Hráč může ohlásit útok, pokud má alespoň jednu jednotku na bojišti v aktivním stavu. Útočit lze i s jednotkou, která má útok z blízka i z dálky nula.

Aktivní hráč vybere jednotku, se kterou chce útočit a určí cíl svého útoku. Unaví útočící jednotku X-krát, kde X je její rychlost útoku (žlutě podbarvený rotační symbol).

Jednotky si navzájem udělí zranění podle hodnoty střelby jednotek.

Pokud je počet životů jednotky snížen na 0 nebo méně, je okamžitě zničena a odchází na hřbitov.

Pokud obě jednotky přežijí střelbu, následuje útok z blízka. Jednotky si navzájem udělí zranění podle hodnoty útoku na blízko.

Pokud jednotka útočí na hrdinu, udělí mu zranění v součtu hodnoty střelby a útoku na blízko.

Pokud bránící se jednotka přežije souboj, 1x se unaví.

Strážce - pokud má soupeř aktivní jednotku na bojišti, nemůže aktivní hráč zaútočit na soupeřova hrdinu, dokud je alespoň jedna soupeřova jednotka aktivní.

Konec kola

Aktivní hráč musí provést všechny schopnosti, které se spouští na konci kola, v libovolném pořadí.

Konec hry

Hra končí v okamžiku, kdy jeden z hráčů přijde o všechny karty ze svého herního balíčku a o všechny karty ze své ruky.

PŘÍLOHA P II: OBSAH CD

K práci je přiloženo CD, které obsahuje vytvořené diagramy, funkční a nefunkční požadavky v Enterprise Architect. Sestavené verze aplikace spustitelné pro Windows 10 a Linux. Vytvořené skripty, prefabs, scény a použité grafické podklady jsou uloženy ve složce Assets. Vytvořená bakalářská práce BP_PacMiroslav.