Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Počítačová hra žánru RTS

2023

Ondřej Fremuth

Vedoucí práce:
Mgr. Radek Jánoštík, Ph.D.

Studijní program: Informatika,
Specializace: Programování a vývoj software

Bibliografické údaje

Autor:	Ondřej Fremuth
Název práce:	Počítačová hra žánru RTS
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2023
Studijní program:	Informatika, Specializace: Programování a vývoj software
Vedoucí práce:	Mgr. Radek Jánoštík, Ph.D.
Počet stran:	43
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliographic info

Author:	Ondřej Fremuth
Title:	Computer game of the RTS genre
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2023
Study program:	Computer Science, Specialization: Programming and Software Development
Supervisor:	Mgr. Radek Jánoštík, Ph.D.
Page count:	43
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Cílem bakalářské práce bylo vyvinout a naimplementovat počítačovou hru žánru RTS. Ve hře se hráč ujme role velitele, který má za úkol přežít nájezdy nepřátel. Práce byla vypracovaná v herním enginu Unity v jazyce C#. Společně s vývojem byla dále vytvořena grafika, hudební podkres a zvukové efekty.

Synopsis

The aim of the bachelor thesis was to develop and implement a computer game of the RTS genre. In the game, the player assumes the role of a commander whose task is to survive enemy raids. The work was developed in the Unity game engine in the C# language. Together with the development, graphics, background music and sound effects were also created.

Klíčová slova: Unity; C#; herní vývoj

Keywords: Unity; C#; game development

Tímto bych chtěl poděkovat mému vedoucímu doktoru Jánoštíkovi za příležitost pracovat na této bakalářské práci a svým kamarádům, kteří tuto práci testovali.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

1 Úvod **7**

2 Výběr herního enginu **7**

3 Úvod do Unity **8**

4 Úvod do hry **12**
4.1 Cíl hry . . . 13
4.2 Ukázka hry . . . 13
4.3 Tutorial . . . 21

5 Vývoj hry **22**
5.1 Jednotky a jejich chování . . . 22
5.2 Nepřátelé . . . 24
5.3 Budovy . . . 25
5.4 Hrdinové . . . 27
5.5 Komponenty hry . . . 28
5.6 UI (User Interface) . . . 29

6 A* Pathfinding **30**

7 Pomocné programy **33**
7.1 Hudba a zvuky . . . 33
7.2 Grafika . . . 36

8 Řešení problémů při vývoji **37**

9 Závěr **41**

10 Conclusion **41**

Literatura **43**

Seznam tabulek

1 Úvod

RTS (real time strategy) neboli strategie v reálném čase je žánr her, při němž hráč ovládá větší skupinu objektů, jako jsou budovy a jednotky a pomocí jejich manipulace se snaží dosáhnout vítězství. Zároveň se ale musí starat o údržbu objektů, surovin, území, staveb a jednotek. Vítězství je často dosaženo porážkou nepřítele, ale může být i více způsobů výhry, jako je třeba držení kritických míst po určitou dobu, dosažení daného množství surovin anebo i přežití několika vln nepřátel. Na rozdíl od ostatních strategických her je ovládání řešeno v reálném čase neboli hráč i nepřítel mohou hrát zároveň, takže nezanedbatelnou roli hraje i rychlost myšlení. Standardem RTS her se staly série jako jsou Age of Empires, Starcraft a Warcraft, díky kterým vzrostla jejich popularita.

Cílem bakalářské práce je vyvinout a naimplementovat RTS hru, kde má hráč za úkol přežít jednotlivé nájezdy nepřátel, s tím že každá následující vlna nájezdů je silnější nežli ta předchozí. Hráč bude schopen stavět budovy, těžit suroviny či vytvářet a vylepšovat jednotky. Zároveň bude mít speciální jednotku, hrdinu, který může používat schopnosti a zabíjením nepřátel zvyšovat svou úroveň, a tím si zajistit růst jeho atributů a zároveň odemknout nové schopnosti. Hrdinu lze vybrat před výběrem mise. Hra se bude skládat z více misí, kde v poslední bude hráč bojovat proti závěrečnému nepříteli, který se bude chovat jinak než ostatní. Nedílnou součástí každé hry je i hudební podkres a zvukové efekty.

Tato práce se tedy bude věnovat tomu, jak samotná hra s názvem Last Stand vznikala, s čím se hráči při hře setkají a jak náročné bylo tyto prvky naprogramovat. Zároveň jsou zde zmíněny problémy, které se při vývoji objevily a jaké bylo jejich řešení.

2 Výběr herního enginu

V prvé řadě je důležité si ujasnit, co si pod pojmem herní engine představit a k čemu se používá. Jedná se o vývojovou platformu, jejíž podpora a organizace ulehčuje a zrychluje vývoj her. Jeho výběr je proto nejdůležitější součástí při vývoji her obecně. Existuje mnoho různých enginů, které se používají. Ne každý engine je vhodný pro každou hru.

Enginy byly ze začátku soustředěny na jeden žánr her, takže vývojáři museli pro svou hru vyvinout vlastní engine. Velice populárním je třeba Unreal engine, který byl vytvořen pro hru Unreal. Dnes jsou enginy rozsáhlejší a univerzálnější, a tak už není nutné si vyvíjet vlastní.

Unity [1]

Unity je jeden z nejpoužívanějších enginů a je populární hlavně mezi začátečníky. Poprvé byl vydán v roce 2005 a díky jeho popularitě je snadné najít řešení k jakémukoliv problému a běží v jazyce C#. Používá se pro vývoj 2D a 3D her pro PC, konzole i mobilní telefony, ale na rozdíl od ostatních enginů má vyšší nároky na hardware, takže vytvořené hry mají většinou problémy s optimalizací.

Unity je dostupný zdarma, ale v případě, že by vývojář měl z vytvořené hry finanční prospěch, musí si dodatečně koupit licenci.

Unreal Engine [2]

Další z velice populárních enginů je Unreal Engine. Zároveň je také jeden z nejstarších, první verze byla vydána v roce 1998 pro hru Unreal a nejnovější verzí je Unreal Engine 5. Jedná se o jeden z nejvyspělejších herních enginů. Nejdříve byl vyvíjen pro střílečky z první osoby, nicméně poslední dobou lze skrz něj vyvíjet jakýkoliv žánr her. Pro kódování je používán jazyk c++, který je náročnější pro uživatele. Samotný engine je hlavně používán pro větší hry a pro jeho obsáhlost pro vývojářské společnosti. Z těchto důvodů se pro jednotlivce příliš nevyužívá. Obdobně jako Unity, lze stáhnout zdarma pod podmínkou, že pokud hra vydělá víc než 1 milion dolarů, tak 5

Godot [3]

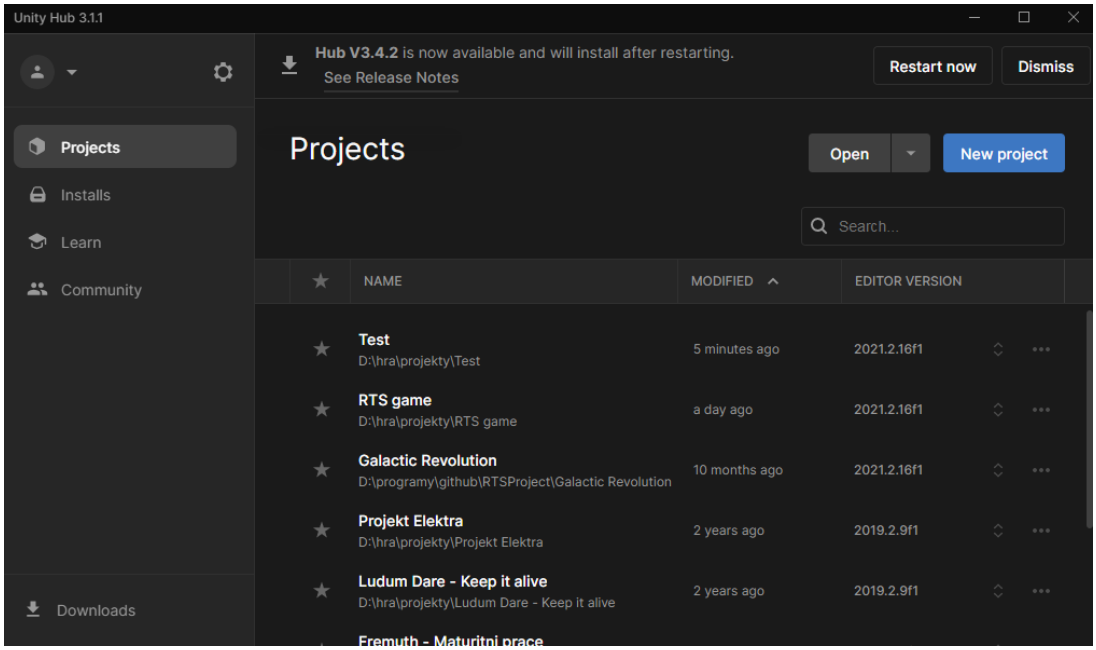
Ačkoliv Godot nepatřil mezi nejpopulárnější enginy, poslední dobou se tento trend mění a jeho obliba roste. Jako nejnovější engine byl vydán teprve v roce 2014. Ze začátku byl používán pro 2D hry, ale postupem času začal podporovat i 3D. Na rozdíl od ostatních enginů používá vlastí programovací jazyk GDScript, který je podobný Pythonu. Díky tomu používá jenom funkce, které slouží k vývoji her a je jednoduchý pro začátečníky. Podporuje také i C# a C++. Je určený spíše pro vývoj menších her, má tedy menší komunitu uživatelů, proto může být komplikovanější najít řešení některých problémů. Tento engine je naprosto zdarma bez dalších podmínek.

Závěr

Po zhodnocení všech výhod i nevýhod výše zmíněných enginů jsem se rozhodl pro Unity. Má univerzální podporu a dobře zpracovanou dokumentaci, což je pro mne, jako začínajícího vývojáře, důležité. Navíc lze jednoduše najít odpovědi na dané problémy. Kódování v C# je velmi intuitivní a mám s ním největší zkušenosti.

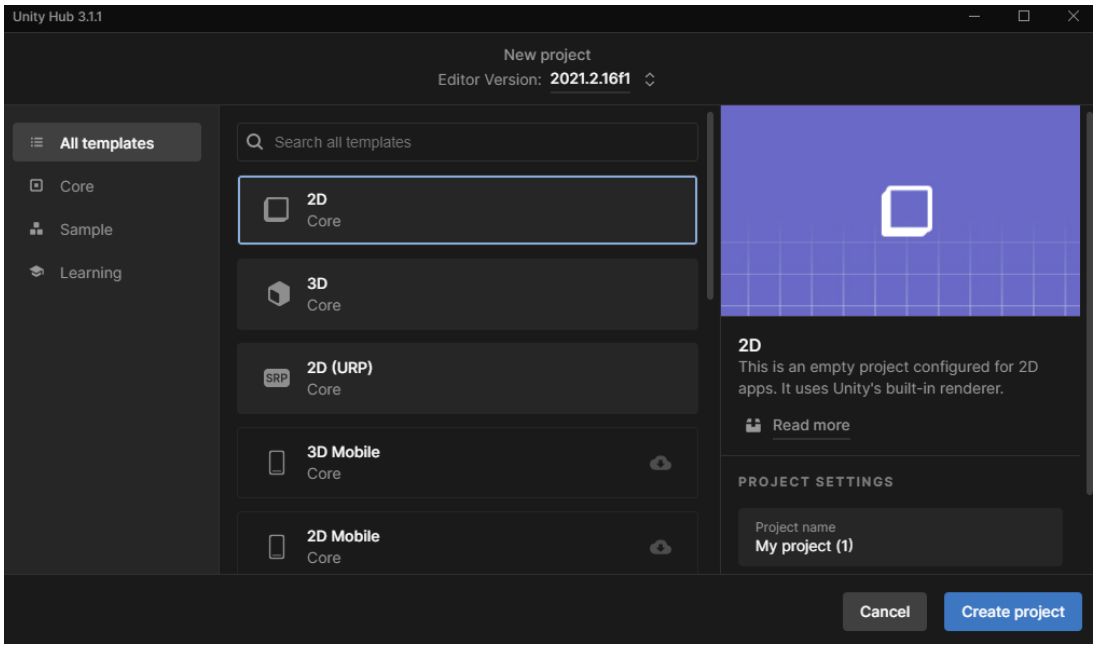
3 Úvod do Unity

První, co je potřeba, je nainstalovat Unity Hub (viz obrázek 1). Zde jsou kromě projektů uloženy také nainstalované verze enginu. Zároveň lze zde najít tutoriály a komunitní obchod, kde lze stahovat a kupovat různé modely, funkce a nástroje pro ulehčení vývoje.



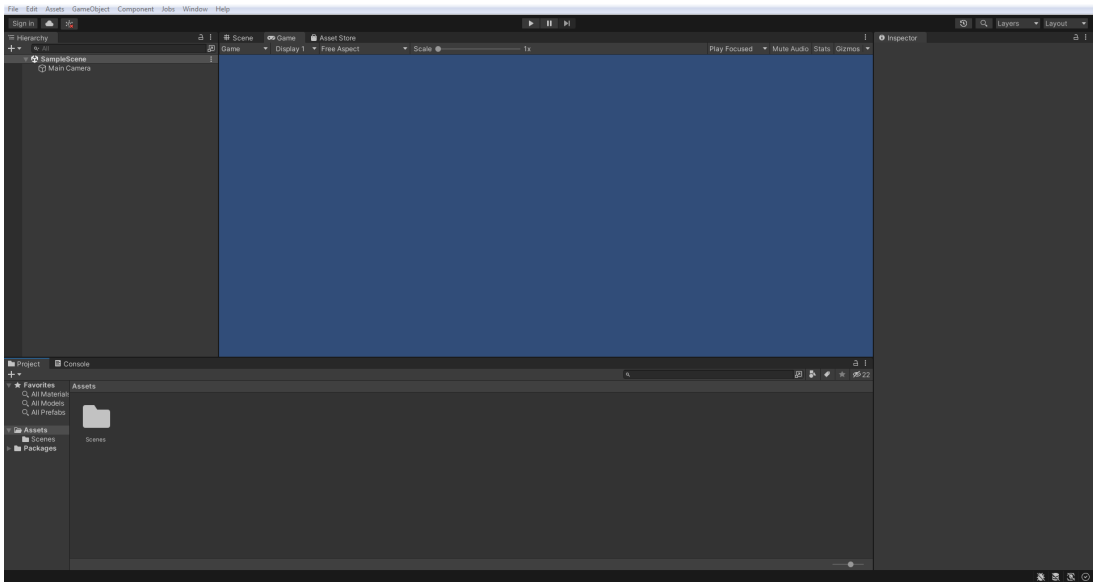
Obrázek 1: UnityHub

Při vytvoření nového projektu si lze vybrat, jakou verzi enginu a šablonu chceme používat (viz obrázek 2). Šablony jsou určeny pro typ hry, kde lze vybrat, zdali hra bude 2D, 3D, na mobilní zařízení či na virtuální realitu. Po pojmenování projektu a jeho umístění se projekt vytvoří.

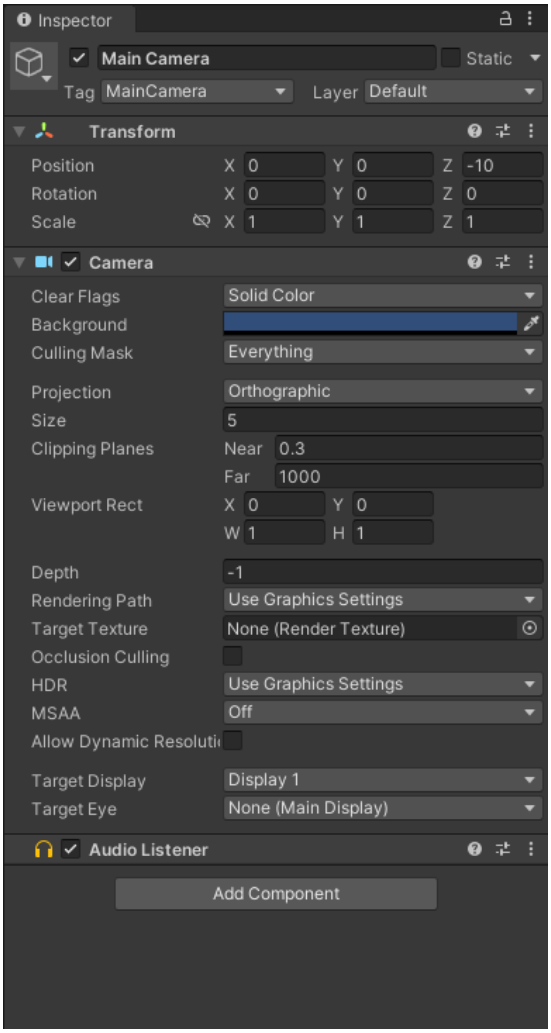


Obrázek 2: Okno při vytvoření nového projektu

Po vytvoření je uživatel přesunut do editoru (viz obrázek 3)), v němž lze už vyvíjet samotnou hru. Editor se skládá ze čtyř základních částí. V levé části obrazovky je list objektů ve scéně. Scéna je základní prostředí hry, kde většinou jedna scéna představuje například menu či úroveň. Uprostřed je obrazovka scény, která ukazuje, jak to bude vypadat ve hře. Na pravé straně je inspektor (viz obrázek 4), který ukazuje vlastnosti objektu, jakou jsou collidery, renderery, skripty a další. V dolní části obrazovky jsou soubory a složky v projektu. Zároveň se zde nachází konzole pro debugování, kde lze najít chyby, které se vyskytly při hraní. Rozložení lze upravit podle preference uživatele

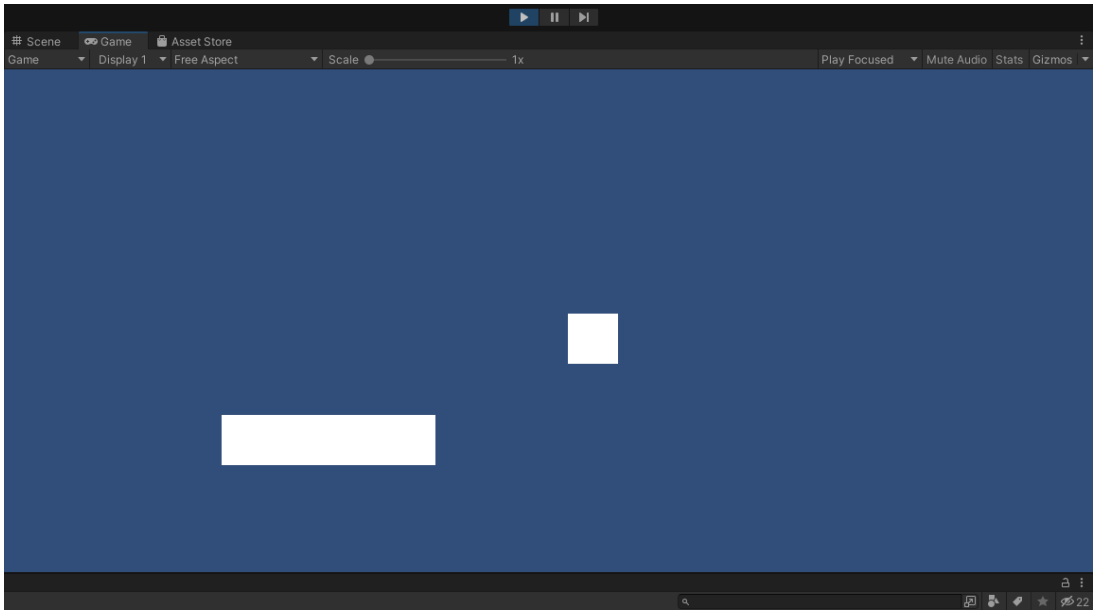


Obrázek 3: Základní obrazovka nového 2D projektu. Nový projekt má jednu základní scénu, ve které je hlavní kamera



Obrázek 4: Inspektor po výběru kamery. Lze vidět přidané vlastnosti – Transform, Camera a Audio Listener

Na horní části obrazovky ve středu jsou tři tlačítka určena pro spuštění nebo vypnutí a zastavení scény. Po spuštění je obrazovka scény změněna na obrazovku hry (viz obrázek 5). Zde lze už hru testovat. Na této obrazovce ale nelze vidět objekty na debugování, například nastavené cesty jednotky.



Obrázek 5: Spuštěná scéna

4 Úvod do hry

V této kapitole budou popsány cíle hry, jak hra vypadá, jak se hraje, a nakonec využití tutoriálu. Pro vyšší přehlednost následujících podkapitol je třeba si představit jednotlivé herní prvky.

První jsou hrdinové. Hráč může mít pouze jednoho hrdinu v misi, a tak je dobré se s nimi seznámit první. V Last Stand jsou na výběr 3 hrdinové: Acolyte (akolyt), Artificier (zbrojíř), Commander (velitel). Akolyt je agresivní hrdina, který má schopnosti určené k zneškodnění nepřátel, ale od ostatních hrdinů se liší svou zranitelností, a tak si ho hráč musí hlídat. Zbrojíř má velmi silný, ale pomalý útok a jeho schopnosti tvoří obranné struktury. Jeho poslední schopností je bomba, která zneškodní vše v okolí včetně přátelských budov a jednotek. Velitel je podpůrný hrdina, který pomáhá přátelským jednotkám. Sám o sobě je ale slabý a má pouze útok nablízko, a proto je dobré mít v okolí několik jednotek na jeho obranu.

Další prvkem jsou budovy. Hráč je schopen postavit osm druhů budov, které mu pomáhají při jeho rozvoji. Nejdůležitější budovou je Base (základna), jelikož ve chvíli, kdy hráč ztratí všechny základny, tak hra končí. Neméně důležité jsou Generator (generátor) a Refinery (rafinerie), sloužící ke generaci surovin potřebné na tvorbu dalších budov a jednotek. Útočnou budovou je Turet (věž), která může zacílit na nepřátele v okolí. Dále se zde nachází Factory (továrna), ve které již hráč může stavět pokročilé jednotky. Další dvě budovy jsou Training hall (tréninková hala) a Tech center (technologické centrum) sloužící k odemykání různých vylepšení pro jednotky. Poslední budovou jsou Barracks (kasárna), která

po postavení zvýší maximální kapacitu jednotek hráče.

V Last Stand se vyskytuje celkem pět druhů jednotek hráče. Tři z nich je možno stavět v Base. Jedná se o Drone (dron), který je sice slabý, ale na výrobu levný, rychle se staví a jako jediný dokáže opravovat budovy. Univerzálním je Initiate (novic), který má útok na dálku, stejně jako dron, ale je mnohem silnější. Poslední jednotkou ze základny je Templar (templář), který má jako jediný útok na blízko, nicméně hráč těží z jeho velké obrany a bonusem je silný útok negující obranu nepřítele. Další dvě jednotky Walker (chodec) a Odin (Odin) lze stavět ve Factory. Předností jednotky Walker je velice rychlý útok. Odin je sice pomalejší, nicméně mnohem silnější a využívá plošného útoku, který zraní více nepřátel v okolí.

Proti těmto jednotkám stojí mnoho nepřátel. Celkem je 7 druhů plus závěrečný nepřítel. Newborn (novorozený), Slasher (řezač) a Destroyer (ničitel) jsou nablízko, kde nejsilnější, Destroyer, neguje obranu cílené jednotky. Corrupted (zkažený) a Bomber (bombardér) jsou na dálku s tím, že Bomber má plošný útok. Broodmother (matka havěti) nemůže útočit a místo toho tvoří slabší jednotky. Hunter (lovec) je létající jednotka, díky čemuž může ignorovat útoky nablízko a přesunovat skrz překážky. Zároveň jeho útok zpomaluje jednotky. Závěrečný nepřítel se skládá ze tří částí, nejdůležitější z nich je Heart of Walush (srdce Walush). To se objeví pouze pokud byly zneškodněny všechny jeho části. Části jsou Eye of Walush (oko Walush) a Tentacle of Walush (chapadlo Walush).

4.1 Cíl hry

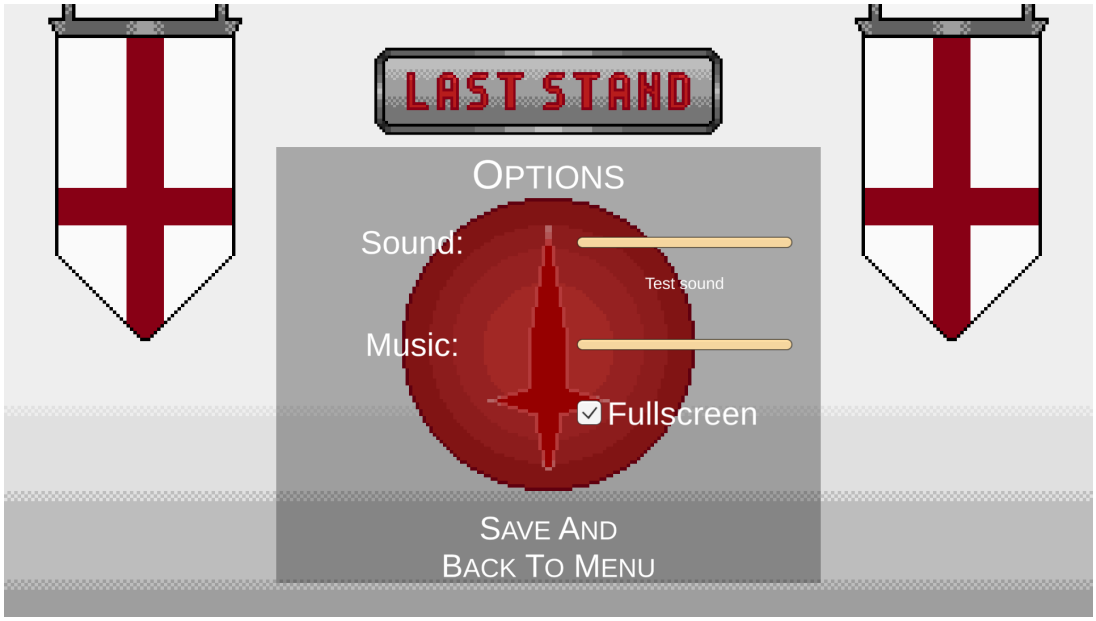
Hra byla pojmenována Last Stand. Jedná se o hru, která se ovládá podobně jako StarCraft, myší a klávesovými zkratkami. Hlavním cílem hry je přežít několik nájezdů nepřátel, nazývané vlny, a jak už bylo zmíněno, náročnost v čase roste. Hráč má na výběr několik možností, jak se bránit, ať už pomocí hrdiny, jednotek anebo věží. Každá hráčská i nepřátelská jednotka má své přednosti a slabiny. Je na hráči, jakou zvolí taktiku a zkombinuje výhody jednotlivých budov a jednotek ke zneškodnění veškerých nepřátelských vln co nejefektivněji.

4.2 Ukázka hry

Při spuštění hry se hráč ocitne v hlavním menu (viz obrázek 6). Zde má na výběr ze čtyř možností: hrát hru, hrát tutoriál, nastavení hry a ukončit hru. Při kliknutí na tutorial se spustí úroveň určená pro seznámení s ovládáním hry.

Obrázek 6: Hlavní menu hry

V nastavení hry (viz obrázek 7) lze upravovat hlasitost zvuku, hudby a možnost přepnutí hry na celou obrazovku. Také je zde tlačítko pro kontrolu hlasitosti zvuku (v ukázce Test sound).



Obrázek 7: Nastavení

Při kliknutí na Play se přepne menu na výběr hrdiny (viz obrázek 8). Zde si

hráč může vybrat hrdinu, kterého bude ovládat v úrovní. Hráč má na výběr ze tří hrdinů, přičemž každý má jiné atributy a schopnosti.

Obrázek 8: Výběr hrdiny

Nakonec si může hráč vybrat, jakou úroveň chce hrát (viz obrázek 9). Při prvním spuštění hry bude mít na výběr jen z jedné úrovně a kdykoliv danou úroveň dokončí, tak se odemkne následující úroveň. V popisku mise je zároveň největší skóre, které hráč dostal při dokončení úrovně.

Obrázek 9: Výběr mise

Poté se hráč ocitne v úrovni. Hráč bude mít na začátku vždy základnu a vybraného hrdinu. V horní liště se nachází základní a speciální suroviny, kapacita jednotek, skóre, tlačítko pro pauzu, a nakonec čas do další vlny/číslo vlny. V dolní liště se nachází minimapa, informace o vybrané jednotce a poté seznam vybraných jednotek či fronta jednotek k vytvoření. Dále zde nalezneme tlačítka pro určité akce podle toho, co má hráč vybraného. Pokud má vybranou základnu nebo továrnu, tak tlačítka jsou pro stavbu jednotek. Pokud má vybranou budovu pro vylepšení jednotek, tak tlačítka značí vylepšení, které lze odemknout. Když má vybraného hrdinu, budou tlačítka pro schopnosti. V ostatních případech jsou to tlačítka pro tvorbu budov.

Pro výběr jednotek či budov se používá levé tlačítko myši. Pro výběr více jednotek je nutné podržet levé tlačítko myši, čímž se vytvoří obdélník. Po jeho uvolnění jsou vybrané všechny jednotky uvnitř obdélníku. Více jednotek lze také vybrat podržením levého shift tlačítka při výběru. Hráč nemůže mít označených více budov anebo budovu a jednotku najednou. Hráč je schopen vytvářet takzvané „skupiny“. Jak už název napovídá, tak se jedná o skupinu jednotek, které je možné jednoduše vybrat. Skupina se tvoří stisknutím levého shift a libovolného čísla na klávesnici. Po stisknutí tohoto čísla se daná skupina vybere.

Pohyb jednotek provedeme stisknutím pravého tlačítka myši na místo, kam chceme jednotku přesunout. V případě, že se na místě nachází nepřátelská jednotka, dojde automaticky k útoku (viz obrázek 10).

Obrázek 10: Hráč vybral jednotku (lze poznat pomocí zvýrazněného stínu). Jednotka se chystá pohybovat.

Když hráč klikne na základnu nebo továrnu, tak může vytvářet jednotky pomocí tlačítek (viz obrázek 11). V případě, že má dostatek surovin a kapacitu, tak se jednotka přidá do fronty. Následně se potřebné suroviny a kapacita sníží. Stiskne-li na jednotku ve frontě, tak se z ní odstraní a hráči se vrátí suroviny i kapacita.

Obrázek 11: Hráč tvoří jednotky – Jde vidět frontu jednotek a v kolika procentech je jednotka na vrcholu fronty hotová.

Při tvorbě budov se vytvoří zástupce budovy (viz obrázek 12). Hráč musí zástupce položit na místo značené modrým kruhem a zároveň musí být místo prázdné. Položení funguje stisknutím levého tlačítka myši a zrušení akce lze stisknutím pravého. Po položení musí hráč počkat, než se budova na dané místo postaví.

Obrázek 12: Hráč tvoří budovu. Jde vidět, že na pozici základny byl vytvořen kruh, který znázorňuje místo, kde lze stavět. Zástupce budovy je zelené barvy, takže ho lze postavit na dané místo.

Jakmile časovač v levém horním rohu klesne na nulu, tak se vytvoří vlna nepřátel na předem určené pozici. Časovač se pak změní a ukazuje číslo fáze / počet fází. Nepřátelé poté zaútočí na jednotky hráče. Ten neví, kde nepřátelé jsou, dokud se neocitnou v blízkosti jeho jednotek nebo budov. Jednotky poté, co uvidí nepřítele, automaticky začnou útočit (viz obrázek 13). Lze si vybrat na koho budou útočit manuálně. Jakmile hráč porazí všechny nepřátelské jednotky, tak se znovu spustí časovač a hráč toho může využít pro vytvoření více jednotek či budov. V případě poražení poslední vlny se objeví zpráva, že hráč vyhrál a zároveň s tím tlačítko pro vrácení do menu (viz obrázek 14)

Obrázek 13: bitva mezi jednotkami

Obrázek 14: Hráč přežil všechny vlny nepřátel a dokončil úroveň. Nad listem jednotek se objevilo tlačítko pro uložení a vrácení do menu.

4.3 Tutorial

Jak je známo, tutoriál je určen zejména pro ty, kteří se chtějí seznámit s ovládáním hry. Je jim automaticky vybrán hrdina a hráč je přesunut do nulté úrovně. Oproti ostatním úrovním má v horní části text, který vysvětluje hru. Zároveň je vedle textu tlačítko, které po stisknutí změní text a nahradí ho dalším. Jakmile se hráč dostane do části, kde hra po něm něco vyžaduje, tak tlačítko zmizí a objeví se, jakmile hráč daný požadavek provede.

Hráč dostane následující úkoly:
Výběr a pohyb hrdiny
Výběr budovy a tvorba jednotek
Vybírání více jednotek najednou
Tvoření skupin
Stavba generátorů a rafinerií (viz obrázek 15 a 16)
Souboj s nepřítelem
Používání schopností

Na konci tutoriálu se spustí časovač a hráč bude pověřen se připravit na menší vlnu nepřátel a přežít.

Obrázek 15: Aby hráč pochopil, že generátory nelze stavět vedle sebe, tak dostane za úkol postavit další. Generátory nelze stavět v červené zóně

Obrázek 16: Poté musí vytvořit refinerii, která lze stavět pouze na zvýrazněném místě

5 Vývoj hry

5.1 Jednotky a jejich chování

Po vytvoření projektu byl prvně zaveden pohyb jednotek. Pro samotný pohyb je použit A* pathfinding [4], který se používá pro hledání cesty, kterou jednotka půjde na vybrané místo. Pro místo, kam se daná jednotka má posunout, je použit prázdný objekt, jež se přesune na místo dříve označené myší. Poté byl vytvořen skript PlayerManager, který řeší veškeré ovládání hry. Ten volá ostatní skripty pro ovládání. Řeší ovládání myší, použití klávesnice a vybírání jednotek a budov.

Pro ovládání všech jednotek byl vytvořen skript UnitController. Zde jsou uloženy jednotky, které hráč vybral pomocí myši. Stará se o předávání příkazů jako pohyb a útok všem vybraným jednotkám. Pokud je jednotka vybrána, tak se zavolá ve skriptu Unit funkce SetSelected, která se stará o to, aby vybraná jednotka šla vizuálně poznat na obrazovce.

Objekty, které mohou být zničeny, tedy jednotky a budovy jsou řešeny ve skriptu Target. Target používá proměnné *maxHp* a *hp*, které definují počet životů a *targetName* definující název objektu.

Pohyb jednotek je řešen skriptem UnitAI, která pracuje s A*. Samotný útok a výběr destinace je řešen ve skriptu Unit, který je potomkem Target. Ve skriptu UnitAI je uložena pozice, kam se chce jednotka posunout, rychlost jednotky, vzdálenost dalšího bodu posunutí, cestu, seeker, který hledá cestu a komponent Rigidbody2D. Jestliže jednotka nemá žádnou cestu a chce se pohybovat, tak

začne hledat cestu. Jednotka se pohybuje po bodech a jakmile se dostane do plánovaného bodu, tak se posune do dalšího, dokud se nedostane na konec cesty.

Výběr více jednotek je řešen dvěma způsoby. První pomocí držení klávesy shift při kliknutí na jednotku, čímž se tato jednotka přidá do listu. Je-li tato jednotka vybraná, z listu se odstraní. Druhá možnost je přidržením levého tlačítka myši. Vezme dvě lokace, první lokace při začátku držení a druhá na konci držení. Potom kontroluje, které jednotky jsou mezi dvěma lokacemi a přidá je do listu. Pokud hráč posune více jednotek, tak se nastaví jejich pozice pohybu do tvaru kruhu. Tím se zajistí, že se jednotky nebudou zasekávat o sebe a budou mít vlastní volné místo.

Poté byla přidána do skriptu UnitController proměnná *groups*. Jedná se o list se seznamy jednotek. Pokud má hráč vybranou jednotku, tak lze tato skupina jednotek vložit do listu pomocí kombinace klávesy shift a klávesy čísla jedna až devět. Poté si hráč může kdykoliv vybrat danou skupinu jednotek pomocí stisknutí čísla znázorňující číslo skupiny.

Pro každou jednotku byl vytvořen nový objekt FiringRadius, který používá skript stejnýho názvu. Ve skriptu Unit byla přidána proměnná *enemiesInRange*, kde jsou uloženi nepřátele v blízkosti jednotky. Jakmile nepřítel vstoupí do collideru FiringRadius, tak se přidá do *enemiesInRange* a při jeho výstupu se z listu odebere. Pokud tedy jednotka porazí nepřítele, tak začne útočit na dalšího z listu a ten se z listu odebere. Pokud jednotka vybere nového nepřítele a nepřítel uložen v útoku pořád žije, tak se starý nepřítel přidá do listu nepřátel v blízkosti a jednotka může útočit na nového nepřítele.

Byla naimplementována nová proměnná do skriptu Unit jménem *defense*. Tato proměnná se používá v případě, že na danou jednotku někdo útočí. Zranění jednotky je sníženo pomocí vzorce $\frac{zranění}{100} * obrana$. Pokud je obrana jednotky například 20 a utrpí zranění o hodnotě 50, tak se sníží životy pouze o 40. Poté byla naimplementována další proměnná *armorPiercing*. Jedná se o boolean, který když je pravdivý, tak jednotka při útoku ignoruje obranu cíle.

Některé jednotky mají plošný útok neboli když útočí na nepřátele, tak zraní všechny jeho jednotky kolem něj. Vlastní jednotky jsou proti tomuto útoku imunní. Na to byl upraven skript Unit. Byla přidána proměnná *splash*, kde je uloženo, zda jednotka má plošný útok. Při útoku se zjistí, jestli je *splash* pravdivý a v případě že ano, tak místo normálního útoku se na místo nepřítele vytvoří objekt, který používá skript SplashDamage. Ten používá collider a kdokoliv, kdo je uvnitř, je zraněn. Hned potom je objekt odstraněn.

Pro opravu budov byl vytvořen skript RepairUnit, který lze přidat k jakémukoliv objektu třídy Unit. Pokud hráč klikne pravým tlačítkem na budovu a má v listu vybraných jednotek jednotky třídy RepairUnit, tak se jednotka pokusí dostat k budově a je-li dostatečně blízko, začne budovu opravovat. Jednotka přestane opravovat, jakmile má opravovaná budova plný počet životů nebo když mu je nařízena jiná akce. Oprava budov je zdarma.

Pro jednotky a budovy byl vytvořen objekt healthBar používající skript HealthBar. Ve skriptu je uloženo pole spritů, kde každý sprite znázorňuje počet

životů. Pokaždé, když je jednotka nebo budova zraněna, tak se zkontroluje, kolik životů zbývá a podle toho se přepne sprite.

5.2 Nepřátelé

Chování nepřátelské jednotky je řešeno ve skriptu Enemy. Používá proměnnou *lockedUnit*, která reprezentuje cíl, na který chce nepřítel útočit.. Po vytvoření nepřátelské jednotky si najde všechny jednotky a budovy hráče a poté zkontroluje, která je nejblíž. Poté si ji uloží do *lockedUnit*. Jakmile je v *lockedUnit* cíl, tak může začít útočit. Po smrti cíle v *lockedUnit* se najde nový cíl pro útok.

Vlny nepřátel jsou řešeny ve skriptu WaveManager. Používá proměnné *monsterCount*, , což je dvojité pole čísel, kde je uloženo počet různých typů nepřátelských jednotek v jednotlivých fázích a poté *unitList*, kde jsou uloženy typy jednotek. V dalším poli, *spawnPoints*, jsou uloženy lokace, kde budou vytvářeni nepřátelé. Proměnná *currentEnemies*obsahuje počet živých nepřátelských jednotek a nakonec *wave*, které udává, v jaké fázi jsme. Na začátku hry se zavolá funkce WaitForWave, která spustí časovač. Jakmile vyprší čas, vytvoří se nepřátelé podle toho, v jaké fázi jsme a jejich počet se uloží do *currentEnemies*. Po smrti jedné jednotky se dekrementuje *currentEnemies*. Jakmile je v *currentEnemies* nula, tak se inkrementuje *wave* a znovu se spustí WaitForWave.

Místo *monsterCount* byla naimplementována proměnná *waves*, což je pole třídy Wave, která znázorňuje jednu fázi. Má dvě proměnné *enemyCount* a *wavePos*. *enemyCount* reprezentuje pole čísel, které znázorňují počet daných nepřátel a *wavePos* reprezentuje index pozice, odkud jsou nepřátelé vytvořeni.

Aby nepřítel nemohl pronásledovat jenom jednu jednotku, která už nemusí být nejvhodnější cíl, tak byl upraven skript Enemy. První byla upravena funkce pro zranění nepřítele tak, že pokud jednotka, která zranila nepřítele je blíž než jednotka v lockedUnit, tak se do lockedUnit uloží útočící jednotka.

Poté byl upraven atribut v Enemy jménem *unitsLooking*. Ten teď má v sobě uloženy jednotky a budovy, které má právě v blízkosti. Pokud bude chtít nepřítel najít nejbližší jednotku a ví, že některé jsou už v jeho blízkosti, tak jenom prohledá tyto jednotky a z nich vybere nejbližší. Jednotky, které se nacházejí daleko ignoruje.

Nakonec je při tvorbě nepřítele spuštěna korutina, která vždy po krátkém časovém intervalu znovu spustí hledání nejbližší jednotky, pokud ovšem není v blízkosti žádná další. Tím je zaručeno, že nepřítel nebude neustále pronásledovat jednu jednotku.

Byl vytvořen skript Creator řešící nepřítele, kteří mohou vytvářet nepřátelské jednotky. Tato jednotka nemůže útočit, ale po daném čase vytvoří nového nepřítele. To se řeší tím, že má zapnutou korutinu, která vezme náhodný index listu jednotek, počká danou dobu pro vytvoření jednotky, a vytvoří ji ve své blízkosti.

Do skriptu Unit byla přidána proměnná *flying*, která značí, zdali daná jednotka může létat. V případě, že má jednotka proměnnou *flying* nastavenou na true, některé její chování je změněno. Pokud dojde k fyzické kolizi, tak ji ignoruje,

tím pádem může projít skrz struktury a jednotky. Také ho nezajímá hledání cesty a místo toho jde přímo k destinaci. Nakonec byl upraven útok jednotek proti létajícím. Když jednotka útočí na létající cíl a má malý dosah útoku, tak na cíl nemůže zaútočit.

Byl vytvořen nový nepřítel, který používá skript Boss. Po vytvoření nepřítel zmizí, vytvoří nepřátele třídy BossPart a uloží si počet vytvořených nepřátel do *currentParts*. Po jejich zneškodnění se *currentParts* dekrementuje. Jakmile proměnná klesne na nulu, tak se boss objeví na náhodném místě v okolí jednotek nebo budov hráče a lze na něj útočit. Nepřítel cyklicky útočí tak, že vezme všechny jednotky a budovy hráče v jeho okolí a zraní je. Jakmile je dostatečně zraněn nebo zůstal příliš dlouho objeven, tak znovu zmizí a vytvoří nové nepřátele. V případě zranění se také přepne do nové fáze. V poslední fázi se zvýší jeho obrana, rychlost útoku a vždy po určité době vytvoří nové nepřátele, nicméně už nemůže zmizet. Po jeho smrti zemřou i ostatní nepřátelé a hráč tím zvítězí.

Nepřátelé třídy BossPart fungují podobně jako Boss. Ze začátku se objeví na náhodném místě. Poté vybere náhodnou jednotku či budovu hráče v její blízkosti a zaútočí na ni. Po náhodném počtu útoků zmizí a objeví se na novém místě, kde se cyklus opakuje.

5.3 Budovy

Pro budovy byl jako první vytvořen skript Blueprint, který řeší, kam se daná budova postaví. První kontroluje, zda jeho collider nekoliduje s jednotkami a budovami. Jestli se nic v jeho oblasti nevyskytuje, tak se budova začne stavět. Po daném čase se objekt zničí a na jeho místě se vytvoří budova, která je už řešena ve skriptu Building.

Pro ovládání a vybírání budov je použit skript BuildingController. Má proměnnou *selectedBuilding*, kde je uložena vybraná budova.

První vytvořená budova je základna, v níž se tvoří základní jednotky. Používá skript Base, který dané akce řeší. Používá čtyři listy a jednu frontu. První list je unitList, kde jsou uloženy všechny jednotky, takže pro naverbování jednotky je potřeba znát jenom její index v listu. Potom *unitCost* a *unitSpCost*, která ukazuje na cenu jednotky v daném indexu. Poslední list *unitCreationTime* značí, jak dlouho se daná jednotka tvoří. Fronta je *unitQueue*, ve které jsou uloženy jednotky, které se čekají, než budou vytvořeny.Budova má prázdný objekt, který značí pozici, kde jednotky budou vytvořeny. Při tvorbě jednotek PlayerManager zjistí, jakou jednotku jsme zvolili. Poté index jednotky společně se surovinami hráče pošle do vybrané budovy Base, která řeší zbytek. Zde je nalezena cena vybrané jednotky a kolik volné kapacity potřebujeme. Pokud cena jednotky nepřesahuje suroviny hráče a má volnou kapacitu, jednotka je přesunuta do fronty. Jakmile se ve frontě nachází jednotka, tak se zavolá korutina. Ta po daném čase značeném v *unitCreationTime* jednotku z vrcholu fronty odstraní a vytvoří na herní ploše.

Další budova je generátor, která je potomkem třídy Building a používá skript

Generator. Po vytvoření začne pomalu generovat základní suroviny.

Aby hráč nebyl schopen stavět generátory vedle sebe, tak byl upraven skript Blueprint. Pokud Blueprint staví Generátor, tak po sestavení má ještě objekt obsahující collider, který značí oblast kolem generátoru. Generátor tedy kontroluje, jestli neleží na některých z těchto objektů. V případě, že ano, nelze je postavit.

Dále byla vytvořena další budova jménem turret. Jedná se o věž, která útočí na nepřátelé v její blízkosti. Její útok je řešen ve skriptu Turret. Také byl vytvořen objekt firingRadius třídy TurretRadius, který je potomek věže. Ten kontroluje, zda se nepřítel dostal do blízkosti věže. FiringRadius má v sobě collider, který je nastaven na trigger. Díky tomu tento collider nemůže fyzicky kolidovat s ostatními, ale ví, kdo do něj vstoupil či odešel. TurretRadius zkontroluje, jestli do collideru nevstoupil nějaký objekt a jestli ano, tak zjistí, jestli je objekt nepřítel. Pokud ano, zavolá funkci TargetEnter ve skriptu Turret a tam se řeší, jestli věž nemíří na žádného nepřítele. Pokud tomu tak není, do proměnné *selectedTarget* se uloží nepřítel a věž začne útočit. . V případě že ve skriptu TurretRadius odejde z collideru nepřítel, tak se zavolá TargetExit v Turret a jestli daný nepřítel je v *selectedTarget*, tak se proměnná nastaví na hodnotu null.

Budova refinery funguje jako generátor, ale generuje jinou surovinu. K tomu byl upraven skript Generator tak, že byla zavedena proměnná *resourceType* typu boolean. Poté se jen řeší, jakou má hodnotu a podle toho zjistí, jakou surovinu má generovat. Tato budova má vlastní blueprint, který používá skript RefineryBlueprint. Na rozdíl od ostatních budov, se refinery může postavit jen na určeném místě, kde je objekt zvaný Geyser. RefineryBlueprint tedy kontroluje, zda na místě, kde chceme umístit budovu, je určený objekt s tagem Resource. Pokud ano, položí se daná budova na místo a blueprint se odstraní.

Pro budovu třídy Base byl přidán objekt BuildingArea. Pouze v oblasti BuildingArea lze stavět budovy. Blueprint poté kontroluje, jestli se nachází na povolené oblasti. Tím může hráč stavět pouze v okolí základen. Také bylo nastaveno ve skriptu PlayerManager, že pokud hráč nemá žádné budovy jménem Base, tak hra končí, neboť nemůže nic jiného stavět.

Poté do Base jsme přidali proměnnou *movePosition*. Zde se nachází objekt, kam se jednotky při vytvoření posunou. MovePosition se dá nastavit vybrané budově pomocí stisknutí pravého tlačítka myši na mapě. Objekt má svůj vlastní sprite, který se objeví kdykoliv hráč vybere danou budovu a díky tomu bude vždy vědět, kam se jednotky přesunou.

Pro vylepšování jednotek byla vytvořena budova, která používá skript UpgradeStation. Poté byl vytvořen skript Upgrades, který má statickou proměnou *upgrades*. Pokud je zavolána funkce pro zakoupení vylepšení, tak se nastaví hodnota upgrades v daném indexu. Následně se dané vylepšení uzamkne, čímž hráč nemůže vylepšení znovu koupit. Nově vytvořené jednotky poté zvýší některé své atributy o hodnotu nastavenou v upgrades.

5.4 Hrdinové

Hrdina používá skript Hero, který je potomek Unit. Hrdina je uložen v proměnné hero ve skriptu Player. Po jeho vytvoření je uložen do listu jednotek stejně jako ostatní jednotky. Používá proměnnou *level* zobrazující jeho úroveň, která se zvyšuje poražením nepřátelských jednotek. Nepřátelé mají proměnnou *xp*, která ukazuje, kolik zkušeností dostane hrdina po zabití daného nepřítele. Hodnota *xp* se poté přičte do proměnné *currentXP*, která reprezentuje, kolik zkušeností konkrétně má. Jakmile hodnota v *currentXP* přesáhne požadované hodnoty určené v poli *nextLevel*, tak se zvýší úroveň hrdiny, čímž se zlepší některé jeho atributy a případně odemknou další schopnosti. Každý hrdina má vlastní skript, který řeší jeho schopnosti a jeho zvýšení atributů po dosažení nového levelu.

Každý hrdina má odlišné schopnosti, které byly rozdělené do tří typů. První z nich jsou schopnosti, které vylepší jednotky v blízkosti hrdiny. Druhý typ se aktivuje položením na mapě v případě, že je schopnost v dosahu hrdiny. Poslední typ schopností vytvoří jednotku či objekt.

První typ je řešen ve skriptu Aura a InvincibleAura. Schopnost se skriptem Aura po použití aktivují collider a jakákoliv přátelská jednotka, která je uvnitř aury, má vylepšený útok. Aura se po aktivaci znovu vypne. InvincibleAura po aktivaci zapne collider po dobu 10 sekund a zase se vypne. Jakákoliv jednotka v dosahu je nesmrtelná: Kdykoliv nepřítel danou jednotku zraní, tak se funkce pro zranění nepoužije.

Druhý a třetí typ používá skript Ability. Ten po vytvoření dává možnost hráči nastavit, kde se má schopnost použít. Jakmile hráč klikne na místo, tak zavolá událost a skript daného hrdiny na to poté zareaguje podle toho, jakou schopnost vyvolal. Každá schopnost má poté svůj vlastní skript, který se aktivuje daným hrdinou.

Třetí typ používá skript Summon. Po aktivaci schopnosti se vytvoří jednotka, s níž se může pracovat s jako normální s tím rozdílem, že hned po jejím vytvoření začnou její životy klesat.

Poté byly vytvořeny skripty pro jednotlivé schopnosti:

Skript ElectricArc po aktivaci zavolá korutinu. Zde po uplynutí daného času vezme všechny nepřátele v okolí objektu a zraní je. Tato akce se poté opakuje. Po daném počtu útoků objekt zanikne.

Skript ArtificierObject po použití vytvoří na daném místě objekt. Na rozdíl od skriptu Blueprint neřeší, kam je objekt postaven, nepotřebuje suroviny na postavení a jeho vytvoření je instantní.

Skript Mines je použit tak, že po použití je objektu, který má tento skript, aktivován skript Explosion. Ten zvětší collider objektu a kdokoliv je uvnitř, tak je zraněn. Poté je objekt odstraněn.

FreezeBlast po použití zraní všechny nepřátelé uvnitř collideru a spustí korutinu Freeze. Ta potom nastaví proměnnou *canMove* u skriptu Enemy na true. Tím nepřátelé nemohou danou dobu útočit nebo se pohybovat. Objekt se poté odstraní.

Grenade funguje podobně jako freezeBlast, ale pouze spustí koroutinu Debuff

pro nepřátele, která zpomalí jejich pohyb a sníží jejich zranění na polovinu.

Po aktivaci schopnosti se daná schopnost uzamkne a nelze použít po určitou dobu. To, jestli je schopnost zamknutá, je uloženo v proměnné *onCooldown* ve skriptu Hero a doba zamknutí v proměnné *abilityCooldown*.

5.5 Komponenty hry

Byl vytvořen skript GameEvents, který řeší interakci mezi skripty pomocí událostí. Pokud je někde v kódu zavolán tento skript, tak GameEvents zavolá skripty, které jsou zaregistrovány do dané události a ty danou informaci zpracují. Příklad volání událostí: Po vytvoření budovy je zavolána událost BuildingCreation, který GameEvents zpracuje tak, že tuto akci pošle skritpu PlayerManager, který danou budovu uloží do listu vytvořených budov.

Bylo naimplementováno skóre, ve kterém se na konci hry ukáže, kolik hráč získal bodů. Kdykoliv hráč postaví budovu, jednotku nebo porazí nepřátelskou jednotku, tak se do skóre přičítají body. Počet bodů, které hráč dostane je definován ve skriptu Target pod názvem *points*.

Pro vytvoření mlhy, která je tak typická ve strategických hrách, byl prvně vytvořen objekt Fog, který má jenom sprite černého obdélníku. Ten byl poté roztáhnut po celé mapě a nastaven tak, aby byl transparentní, čímž se celá obrazovka jen ztmavila. Poté pro každou jednotku a budovu hráče byl přidán objekt revealer, který používá sprite mask. Poté byl nastaven objekt Fog tak, aby interagoval se sprite mask, čímž se část objektu zneviditelní v okolí objektů revealer.

Poté byli upraveni nepřátelé tak, aby šli vidět pouze, pokud jsou v blízkosti hráčské jednotky či budovy. Ve skriptu Enemy byla přidána proměnná *unitsLooking*, kde je uloženo, kolik jednotek hráče jsou v blízkosti nepřítele. Jakmile se nepřítel dotkne collideru firingRadius, tak zkontroluje, zda se jedná o hráčskou jednotku či budovu. Poté zavolá funkci Reveal a Vanish podle toho, jestli do collideru vstupuje nebo vystupuje. V Reveal a Vanish se inkrementuje (dekrementuje) *unitsLooking*. Ve funkci Reveal pokud je *unitsLooking* jedna, tak se jednotka zviditelní a pokud ve Vanish je *unitsLooking* nula, tak zase zmizí.

Ve skriptu Player jsou přidány nové proměnné *unitCapacity* a *currentCapacity*. V *unitCapacity* je kapacita jednotek, které hráč může v daný okamžik mít a *currentCapacity* je aktuální kapacita hráče. Každá jednotka hráče má teď proměnnou *capacityCost*, která při vzniku jednotku zvýší currentCapacity o danou hodnotu. Pokud součet ceny kapacity jednotky a *currentCapacity* přesáhne *unitCapacity*, tak danou jednotku nelze vytvořit.

Pro zvýšení *unitCapacity* byla vytvořena nová budova, která používá skript Barracks. Po vytvoření téhle budovy je zvýšena hodnota *unitCapacity* a pokud je tahle budova zničena, tak je zase tato hodnota opět snížena.

Byl vytvořen skript GameData, kde jsou uloženy odemknuté mise a nejvyšší skóre, které hráč za misi dostal. K samotnému uložení těchto informací byl vytvořen skript ProgressionSystem. Za pomocí skriptu lze uložit a načíst herní data.

Pokud data neexistují, tak se při startu hry vytvoří nový soubor, kde jsou zakódována herní data. Zakodování je řešeno třídou BinaryFormatter, která zkonvertuje data na jiný typ, který lze snadněji přečíst a zapsat programem. Poté při dalším spuštění se tato data zkonvertují zpátky a načtou do hry. Pokaždé, když hráč dohraje misi, tak se načtou stará data, zkonvertují do originální podoby, odemkne se další mise a pokud skóre v dané misi je vyšší než staré skóre, tak se do dat uloží nové skóre. Nakonec se data zakódují a uloží se do souboru. Zároveň jsou v GameData hlasitost hudby a zvuku, které se nastaví, když hráč změní hlasitost v nastavení a poté uloží, jakmile se přepne znovu do hlavního menu.

5.6 UI (User Interface)

Hru je možné zastavit pomocí tlačítka escape. Rychlost hry se zastaví na nula a zobrazí se menu. Zde lze přejít do hlavního menu, ukončit hru nebo pokračovat s hrou. Herní menu je řešeno skriptem PauseMenu. Hlavní menu hry má skript MainMenu využívající funkce PlayGame, která přepne do scény hry a ExitGame, která celý program ukončí.

Nastavení hry je řešeno ve skriptu SettingsMenu. Zde se nastavuje hlasitost zvuku a hudby a přepnutí hry na celou obrazovku nebo v okně. Pro nastavení hlasitosti byl vytvořený objekt typu AudioMixer. Ten může rozdělit různé zvuky do skupin a tím lze nastavit hlasitost pro různé zvuky. Zvuky byly rozděleny na normální herní zvuky a hudbu.

Poté byla vytvořena tlačítka pro stavbu budov a jednotek. Každé tlačítko volá dané funkce v PlayerManager. Také jsou vytvořena tlačítka pro zastavení hry, vylepšení a schopnosti hrdiny.

Byla vytvořena tlačítka pro výběr dané jednotky v právě vybraných jednotkách. Pokud si hráč vybere jednu či více jednotek, tak se objeví právě tolik tlačítek, kolik má vybraných jednotek. Po kliknutí na tlačítko se ukáže informace o jednotce a pokud se jedná o hrdinu, tak se objeví tlačítka pro schopnosti jednotky. Poté byl nastaven limit vybraných jednotek na deset.

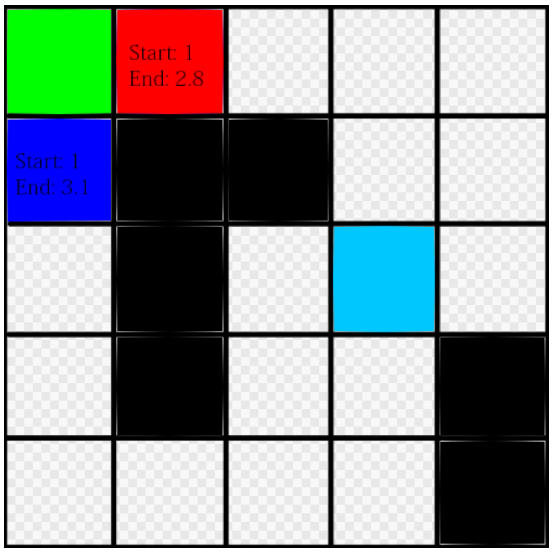
Také byla vytvořena tlačítka pro právě se vytvářející jednotky v budovách se skriptem Base. Obdobně jako tomu bylo v předchozím odstavci, pokud hráč bude chtít vytvořit jednotky z budovy, tak se objeví tlačítka, které značí, jaké jednotky se právě budují. Po stisknutí na tlačítko jednotky se tvorba zvolené jednotky zruší. Poté byl nastaven limit velikosti fronty jednotek na deset.

Dále byla upravena tlačítka pro tvorbu budov, jednotek, výběr schopností a vylepšení jednotek tak, že se přidala pro každé tlačítko vlastnost Event Trigger. Pokud jsme najeli na tlačítka myší, tak tlačítko volá skript PlayerManager, který potom na obrazovku vypíše, co dané tlačítko dělá.

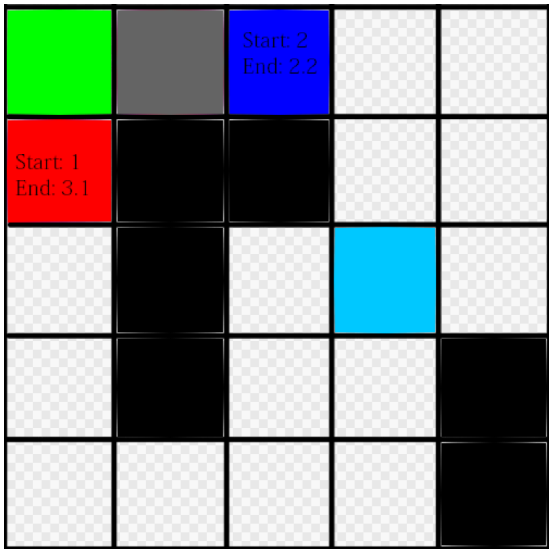
Výběr hrdiny je řešen ve skriptu MainMenu. Po stisknutí tlačítka play je hráč pobídnut, aby si vybral svého hrdinu. Vybraný hrdina je poté uložen ze skriptu MainMenu do skriptu Player ve statické proměnné hero. Při startu hry je ve skriptu PlayerManager hrdina zavolán z daného skriptu a uložen v PlayerManager pod proměnnou hero. Poté lze hrdinu ovládat.

6 A* Pathfinding

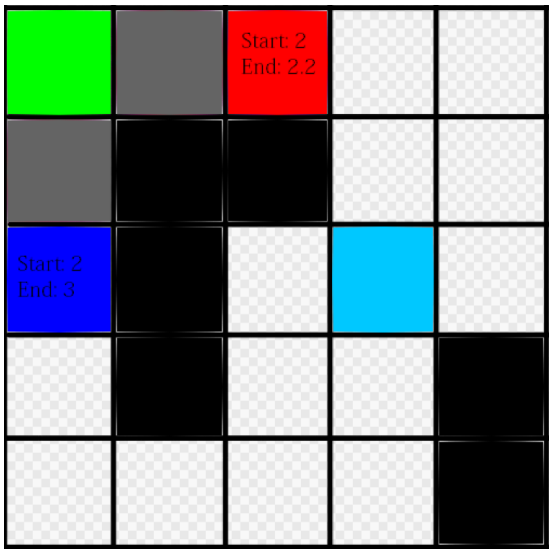
A* algoritmus je upravený Dijsktrův algoritmus hledání cesty. Graf je reprezentován dvourozměrným polem, kde každé místo v poli je buď volné anebo plné. Přes plná políčka nelze projít. Na začátku cesty si vezme své volné sousedy a poté vypočítá vzdálenost nového místa od začátku a pak až do konce pomocí Pythagorovi věty. Následně si vypočítané vzdálenosti uloží a vybere místo, kde je součet těchto vzdáleností nejmenší. Toto místo se označí jako navštívené a poté vezme jeho sousedy, kde se znovu vypočítají vzdálenosti. Zároveň si tato nová políčka pamatují svého rodiče (políčko, které ho stvořilo). Toto se opakuje, dokud se nedostane na konec. Poté se cyklicky vezme rodič políčka, dokud se nedostane znovu na start. Tím se získala nejkratší cesta. Výhodou algoritmu je, že ignoruje cesty, které se mu zdají dlouhé.



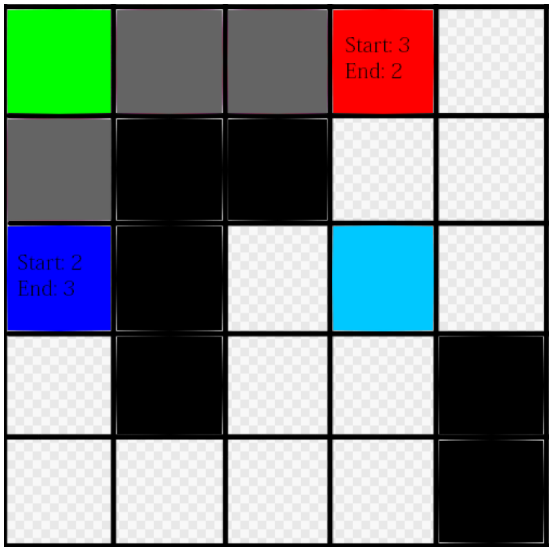
Obrázek 17: Příklad A* algoritmu. Zelené políčko je start, světle modré cíl a černá políčka jsou plná, nemůžeme jimi projít. Vezmeme ze startu dva jeho sousedy a zkontrolujeme, který má nejmenší součet vzdáleností. Políčko zdola od startu má součet 4,1 a políčko zprava má součet 3,8. Vezmeme políčko zprava od startu a označíme ho.



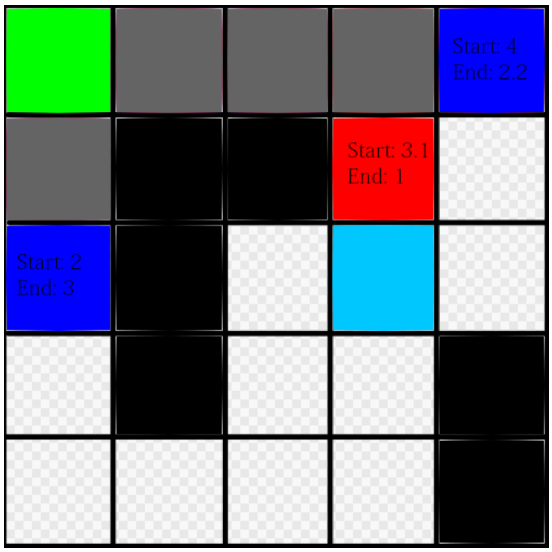
Obrázek 18: Vezmeme sousedy vybraného políčka a zbarvíme ho na šedo. Nové políčko má součet 4,2. Vybere políčko pod startem



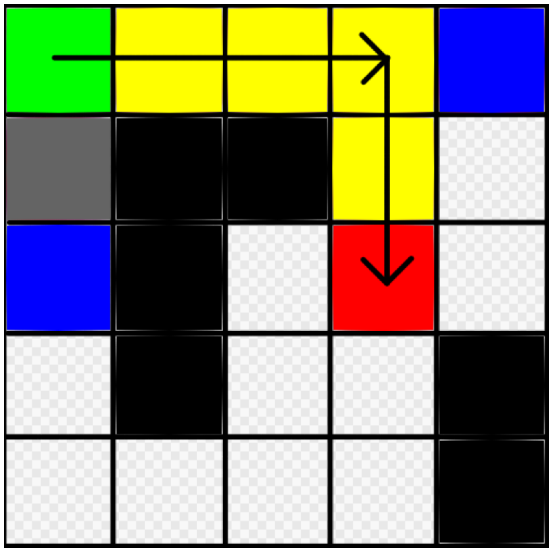
Obrázek 19: Znovu vezmeme sousedy vybraného políčka. Nový soused má součet 5, tak vybereme políčko se součtem 4,4



Obrázek 20: Nové políčko má součet 5. Vybereme ho



Obrázek 21: Máme dvě nová pole, které mají součet hodnot 4,1 a 5,2. Vybereme políčko s hodnotou 4,1

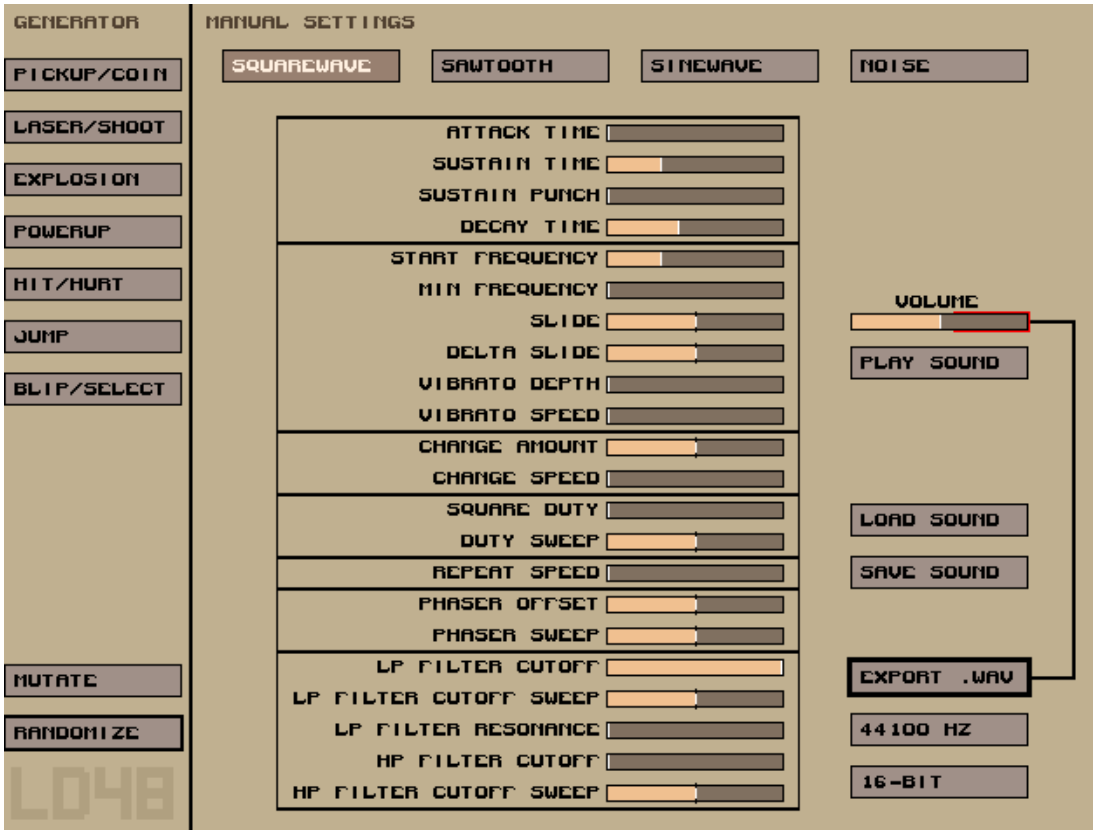


Obrázek 22: Soused vybraného políčka je už cíl. Vezmeme políčka, pomocí nichž jsme se dostali do cíle. Vytvořili jsme nejkratší cestu k cíli.

7 Pomocné programy

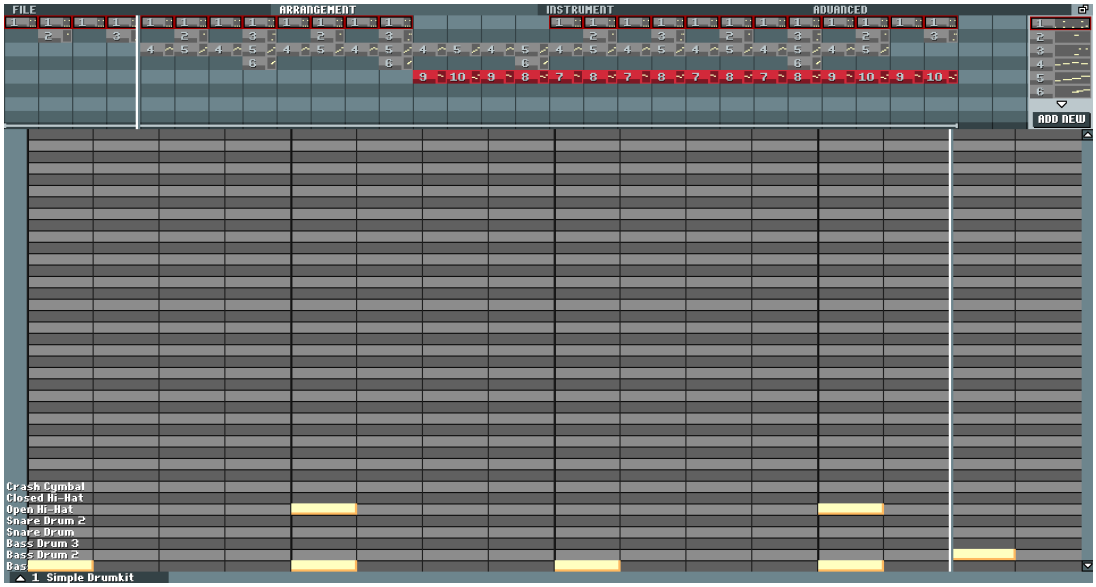
7.1 Hudba a zvuky

Pro zvuky byl použit program sfxr [5]. Jedná se o freeware program, který dokáže generovat náhodné zvuky pro různé akce, jako jsou například střelba nebo exploze. Po generaci zvuku je možné daný zvuk opravit. Zvuky jsou 8 nebo 16 bitové hodící se do retro her (viz obrázek 23).



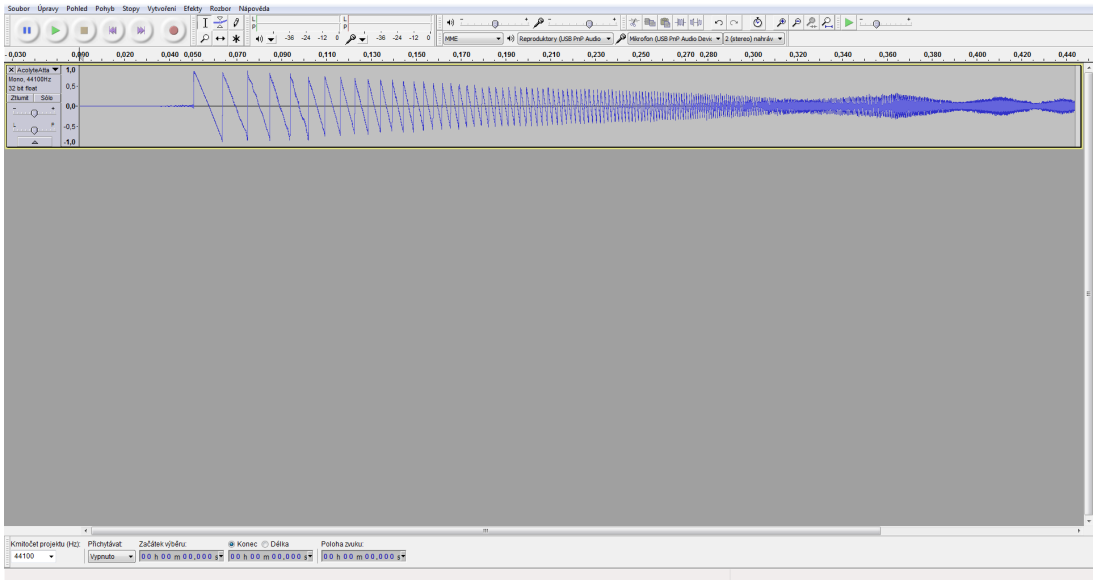
Obrázek 23: Hlavní obrazovka sfxr. V levé části jsou typy přednastavených generátorů, které můžeme použít např: pro útok, exploze, výběr atd. V centru obrazovky jsou různé proměnné, které můžeme změnit pro úpravu nebo tvorbu zvuku.

Pro hudbu byl poté použit freeware program BoscaCeoil [6]. Jedná se o program s intuitivním ovládáním a velkým množstvím hudebních nástrojů. Má však svoje limity, proto není vhodný pro složitější kompozice. V programu lze vytvořit jednotlivé hudební stopy a ty následně složit dohromady (viz obrazek 24).



Obrázek 24: Hlavní obrazovka BoscaCeoil. V horní části obrazovky jde vidět různé menu a místo pro stopy hudby. Zbytek obrazovky je pro editaci stop.

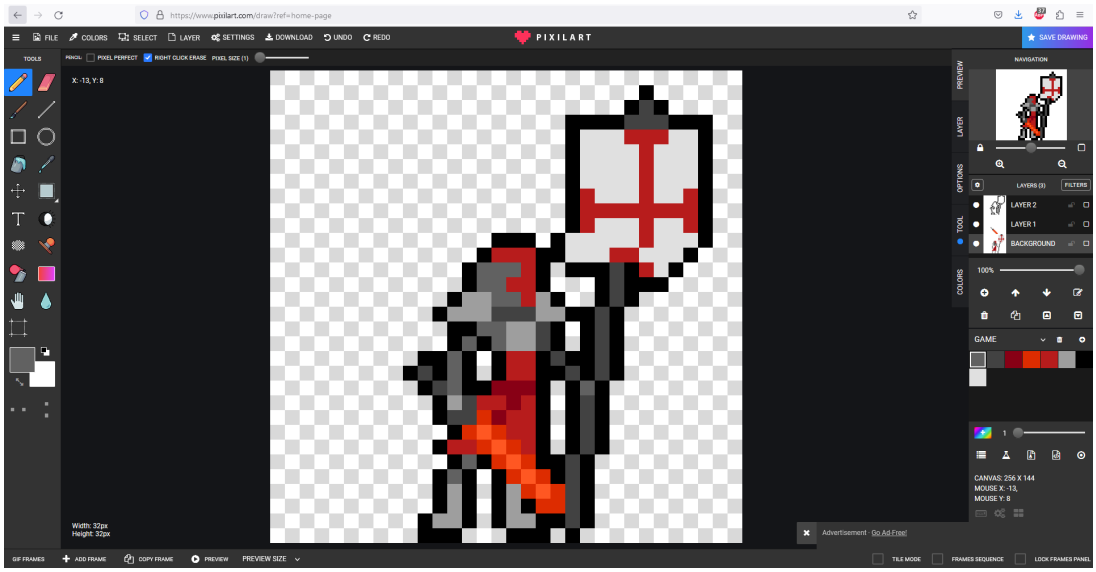
Nakonec byl pro poslední úpravy hudby a zvuku použít program Audacity [7], populární freeware program, který kromě úpravy může zvuky i nahrávat. Uživatel má tak mnohem více svobody nežli u předchozích dvou programů. Audacity je primárně určen pro pokročilé uživatele. Při tvorbě hry Last Stand byl použit převážně na konvertování typu souboru zvuků a menší úpravu jakou jsou zeslabování zvuků či odstraňování šumu (viz obrazek 25).



Obrázek 25: Hlavní obrazovka Audacity. Byl přidán zvuk, který lze následně upravovat.

7.2 Grafika

Veškeré sprity byly vytvořeny v programu Pixilart [8]. Jedná se o webovou aplikaci, která funguje podobně jako grafický editor. Velikost plátna je omezena na 700 pixelů, tudíž není určený na příliš detailní grafiku. Aplikace se hlavně používá pro styl grafiky zvané pixel art. Cílem stylu je reprezentace objektu za pomocí limitovaného plátna a v některých případech i limitovaného množství barev. Tento styl byl používán převážně u starších her, jelikož počítače neměli sílu renderovat velké množství pixelů a barev. V dnešní době se pixel art používá jako preferovaný styl u velkého množství malých vývojářů kvůli své jednoduchosti. Kromě tvorby spritů Pixilart podporuje tvorbu animací (viz obrazek 26).



Obrázek 26: Hlavní obrazovka Pixilart. Jdou vidět základní nástroje známé z ostatních grafických editorů.

Pro tvorbu efektů jako jsou exploze byl použit částicový systém v Unity. Jedná se o jednoduchý systém, který vytváří objekty reprezentující částice. Lze měnit například velikost, rychlost, barvu částic, jak se budou chovat při kolizi a rychlost jejich vytvoření. Díky tomu lze rychle a jednoduše oživit hru.

8 Řešení problémů při vývoji

Při vývoji bylo objeveno značné množství problémů, u kterých nebylo na první pohled patrné řešení. Některé lze samozřejmě najít na internetu a snadno je vyřešit, nicméně ne všechny problémy lze takto najít. Zde jsou popsány problémy, které se vyskytly a jejich řešení.

Problém s výběrem jednotek

Našel se problém, kde jednotky a budovy nešli zvolit myší. Následně bylo zjištěno, že jednotky s kladným číslem na pozici z nešly zvolit. Vyřešilo se to nastavením pozice z na -1 pro všechny budoucí jednotky.

Problém UI

Po stisknutí některých tlačítek, jako je tlačítko zastavení hry a tlačítko pro výběr hrdiny nastal problém, že nefungovaly tak, jak by měly. Ukázalo se, že společně se stisknutím tlačítka se také zavolá ve skriptu PlayerManager Update, že se stisknulo tlačítko myši, což daný problém způsobovalo. Vyřešilo se to tím, že byl pro kameru vytvořený objekt, který má collider v místech kde je UI a když hráč klikne na toto místo, PlayerManager stisknutí ignoruje. Poté se upravil skript CameraController tak, že jestli se kamera přiblíží nebo oddálí, změní se i velikost collideru.

Problém se zastavením hry

Když byla hra zastavena, tak i přes menu šlo vybírat jednotky a budovy. Opravilo se to přidáním proměnné isPaused ve skriptu PlayerManager. Pokud je isPaused pravdivé, tak skript nelze použít.

Problémy s automatickým útokem

Jakmile jednotka porazí nepřítele, měla by najít dalšího nepřítele, ale nefungovalo to. Ze začátku byl upraven útok tak, že během cooldownu útoku se zjistí, jestli byl nepřítel zabit. Jestli ne, tak na něj útočí znovu a pokud ano tak se zmenší radius útoku a následně opět zvětší, čímž nepřítele znovu najde. Toto by šlo obejít pomocí funkce OnTriggerStay2D, která se volá, když je někdo uvnitř collideru, ale jelikož je tato funkce náročná pro systém, použilo se řešení s radiem. Nakonec byl opraven útok tak, že se radius útoku nezmenšuje, ale místo toho jednotka má list jednotek, který se naplňuje, dokud se nepřítel nedostane do blízkosti jednotky a odebere, pokud zmizí z blízkosti. Pokud jednotka někoho porazí a list není prázdný, tak se vezme nepřítel z listu, odebere se z něj a jednotka začne na něj útočit.

Druhým problémem bylo, že pokud hráč vybere jednotku a chce, aby jednotka útočila na jiného nepřítele než právě vybraný jednotkou, tak po zneškodnění toho nepřítele nebude znovu útočit na původně vybraného, i když je v dosahu jednotky. To se vyřešilo tím, že pokud je vybrán nový nepřítel a jednotka už na nějakého útočí, tento nepřítel se uloží do listu nepřátel v blízkosti a nově vybraný nepřítel bude zaměřen jednotkou.

Problém se stavěním budov

Při stavbě budov byly suroviny pro stavbu odstraněny od hráče, jakmile vytvořil Blueprint pro danou budovu. Tady nastal problém, že pokud hráč budovu nechtěl postavit a zrušil tuto akci, tak se mu nevrátí suroviny. To se upravilo tak, že skript Blueprint řeší, jestli hráč budovu postavil. Pokud ano, budova se postaví na místo, odstraní se suroviny z hráče a blueprint se následně odstraní. Pokud ne, tak se pouze odstraní.

Problém s útokem na blízko

Jednotky, které mají krátký dosah mají problém s útokem na větší jednotky nebo budovy. Ukázalo se, že když se měří vzdálenost mezi jednotkou a jeho cílem, tak se bere střed jednotky i cíle. Jelikož je ale cíl veliký, tak i když je jednotka blízko, jeho vzdálenost k cíli je větší než jeho dosah. To se nakonec upravilo tak, že se místo středu cíle bere pozice nejbližšího bodu v collideru.

Problém se zmizením nepřátelských jednotek

Nepřátelské jednotky jsou ze začátku neviditelné a zviditelní se až jsou v blízkosti jednotky nebo budovy hráče. Ze začátku se použila proměnná unitsLooking, kde byl uložen počet jednotek, které ho vidí. Ale stávalo se, že byl nepřítel viditelný, i když nebyla žádná jednotka či budova hráče v blízkosti. Nakonec se to upravilo tím, že v proměnné unitsLooking je teď uložen list jednotek, a ne její počet.

Problém s hledáním jednotek

Pokud nepřítel hledá jednotky, které jsou mu nejblíže a v proměnné unit-

sLooking nějaké jednotky jsou, začne hledat pouze z tohoto listu. Ale stávalo se, že pokud nepřítel porazí jednotku a začne hledat jinou jednotku, zasekne se v hledání. Důvod, proč se to stávalo byl ten, že poražená jednotka nebyla z listu unitsLooking odstraněna a když prohledává list, tak se zasekne. To bylo upraveno tím, že před prohledáním se zkontroluje, jestli je nějaká jednotka z listu v hodnotě null. Pokud ano, je tato jednotka odstraněna z listu.

Problém s pronásledováním jednotek

Když už nepřítel pronásleduje jednotku, tak nebylo možné poté změnit na jinou dokud první nebyla poražena, i když třeba není nejbližší. Toto chování bylo způsobeno proměnnou attackQueued ve skriptu Unit, která ve funkci Attack nastaví její hodnotu na jednotku, na kterou chce útočit. To bylo dobré u jednotek hráče, ale ne u nepřítele. Proto se nakonec kontroluje, jestli funkci Attack volá jednotka se skriptem Enemy a pokud ano, tak se attackQueued nenastaví.

Problém s vylepšováním jednotek

Vylepšení jednotek bylo ze začátku řešeno změnou hodnoty dané proměnné ve skriptu Unit, což mělo za následek, že změněné hodnoty trvale zůstaly i po skončení hry. To se poté upravilo tím, že bylo do skritpu Unit přidána proměnná upgrades, kde byly potom uloženy vylepšení, které se přidali k daným proměnným. To se však projevilo neefektivním. Nakonec se vytvořil skript Upgrades, kde bylo uloženo statické pole s hodnotami. Poté, co se odemklo vylepšení, se tato daná hodnota v poli zvýší. Toto pole se zároveň vždy na začátku hry znovu inicializuje na výchozí hodnoty. Jednotky nakonec po jejich vytvoření vezmou dané hodnoty z pole a přidají si to do vlastních proměnných.

Problém s oživením hrdiny

Aby hrdina po své smrti neztratil svoji úroveň a zkušenosti, není odstraněn, ale deaktivován. Poté ho lze znovu aktivovat přes budovu se skriptem Base. To mělo ale za následek hned několik problémů. Hrdina někdy nemohl útočit, používat schopnosti a jeho textura zůstala ve stavu zranění. Problém byl v tom, že deaktivací hrdiny byly pozastaveny i všechny korutiny. Proto byla většina proměnných přenastavena tak, aby funkce znovu fungovali.

V některých případech, kdy chtěl hráč oživit hrdinu v základně, tak se místo hrdiny začne stavět normální jednotka. Není známo, proč se tento problém objevoval. Ze začátku, když hrdina zemřel a hráč ho chtěl oživit, tak ve funkci oživení se nastaví hrdina jako jednotka, kterou může základna vytvářet. Řešení bylo takové, že se hrdina v základně nenastavoval, jakmile ho hráč chtěl oživit, ale už při postavení základny. Jakmile se vytvoří základna, tak volá událost, kterou převezme PlayerManager. Ten pak nastaví základně hrdinu.

Problém s tutoriálem

Tutoriál byl ze začátku nastaven tak, že byl obrazovku vypsán text, který byl po nějakém čase odstraněn a nahrazen novým. Pokud hráč musel něco udělat, tak cyklicky čekal, dokud hráč neprovedl správnou akci, což bylo nastaveno ve skriptu Tutorial. To se projevilo špatně, jelikož ostatní buď nestíhali číst nebo museli čekat. Bylo tedy naimplementováno tlačítko, po jehož stisknutí text pokračoval. Text byl poté celý napsán do textového souboru, kde jeden řádek textu byl právě

text, který se ukáže právě v jednu dobu. Pokud byl na začátku řádku speciální symbol, tak bylo tlačítko deaktivováno a čekalo se, než hráč provedl správnou akci. Potom se tlačítko znovu aktivuje a hráč může pokračovat.

Problém s úrovněmi hrdinů

Aby se hrdinům zvýšila jejich úroveň, musí porazit nepřátelské jednotky, díky nimž se zvýší proměnná experience ve skriptu Hero. Jakmile hrdina dosáhne požadované hodnoty, tak se zvýší jeho úroveň. Nastal ale problém, kdy někteří hrdinové mají mnohem těžší porazit nepřítele, a navíc bylo mnohem těžší porazit nepřítele hrdinou, jelikož ji může kdykoliv porazit přátelská jednotka anebo schopnosti hrdiny. To se upravilo následujícími změnami. Hrdina se skriptem Commander může pomocí jeho první schopnosti, která zvýší zranění ostatním přátelským jednotkám, navíc nastaví, že jakmile taková jednotka zneškodní nepřítele, tak se bude počítat, že ho porazil hrdina. U ostatních hrdinů bylo nastaveno, že každá jejich schopnost, pokud zneškodní nepřítele, se bude počítat, že ji porazil hrdina.

Problém s tlačítky schopností

Tlačítka schopností měly být původně pro každou schopnost. To se jevilo neefektivní, a navíc znamenalo více kódu. Nakonec byly vytvořeny tři tlačítka, které mají index schopnosti. U skriptu Hero bylo naimplementována proměnná, která znázorňuje pole tlačítek. Ve skriptech Artificier, Acolyte a Commander je potom ve funkci Start nastaven text u tlačítek znázorňující, co se jedná za schopnost. Po použití schopnosti byl přes tlačítko nastaven text, který znázorňuje čas, dokud nelze daná schopnost použít znovu.

Problém se zasekáváním jednotek

Pohyb jednotek je řešen pomocí A* algoritmu tak, že je přes celou mapu dvourozměrné pole políček. Pokud je v políčku překážka nebo budova, tak se přes políčko nelze přesunout. Problém nastává, když velké jednotky se snaží přesunout přes místo vedle dvou budov a místo mezi nimi je pro jednotku moc malé. Jednotka se tedy zasekne mezi dvěma překážkami a hráč je musí manuálně přesunout kolem. Proto bylo rozhodnuto, že kdykoliv se jednotka zůstane ve styku s budovou, tak začne ignorovat kolizi a může projít.

Další problém nastal se zasekáváním se o jednotky. Jelikož dvourozměrné pole na mapě pracuje pouze s budovami a překážkami, tak se stává, že se jednotky snaží přesunout přes další jednotku, což samozřejmě nefunguje. Dalo by se to taky vyřešit ignorováním kolize, ale to by mělo za následek několik problémů, například když se nepřátelé spojí dohromady, kde všichni mohou útočit, ale hráč si z nich může vybrat pouze jednoho nepřítele. Nakonec se to vyřešilo tím, že pokud jednotka zůstane ve styku s dalším, vezme se dvourozměrné pole z A* algoritmu, vybere se nejbližší políčko k pozici jednotky, které jsme se dotkli, a nakonec se toto políčko nastaví tak, že přes něj nelze projít. Poté se vytvoří nová cesta a tím už může jednotka obejít další jednotky. Aby se poté změněné políčko vrátilo zpět, byl vytvořen skript GridManager. Kdykoliv jednotka nastaví políčko, tak zavolá událost, kterou převezme objekt třídy GridManager a ta poté po krátkém čase zase změní políčko, aby se šlo přes něj přesunout.

9 Závěr

Cílem práce bylo vytvořit počítačovou hru žánru RTS, ve které se hráč ujme role velitele a za pomocí svých schopností se pokusí přežít několik vln nepřátelských jednotek. Pro dosáhnutí cíle je zapotřebí postavit několik druhů budov, vygenerovat dostatečné množství surovin a samozřejmě tvořit jednotky, přičemž je dobré brát v úvahu jejich kvalitu i počet. Svou roli při snaze o vítězství hraje i zvolení počátečního hrdiny, plné využití jeho schopností a přizpůsobení taktiky.

Hra samotná má krom tutoriálu tři úrovně, přičemž náročnost se lineárně zvyšuje. Vzhledem k značné rozličnosti jednotlivých herních prvků, jako jsou budovy, jednotky i nepřátelé skýtá dostačující možnosti pro zábavu hráče. Nemalou měrou se na tom podílí i závěrečný nepřítel, který se svými vlastnosti od klasických nepřátel odlišuje.

V budoucnu bych rád na hře nadále pracoval, neboť si myslím, že má zajímavý potenciál růstu. Příkladem by mohlo být zavedení nových jednotek, bojových vozidel a i nepřátel. To by se samozřejmě podepsalo i na množství a podobě vln nepřátel. Za zvážení by také stálo přidat nové i silnější závěrečné nepřátele, kteří by opět měli mít odlišné chování. To vše bych rád doplnil upravenou, popřípadě novou grafikou jednotlivých herních prvků a i animacemi.

Práce s jednotlivými programy byla inspirativní, neboť jsem si mohl vyzkoušet vícero různých nástrojů a sám pak vyhodnotit, který bude pro mé záměry nejvhodnější. Samotné řešení problémů při vývoji hry mi dalo mnoho zkušeností a nový pohled na práci vývojářů.

10 Conclusion

The goal of the work was to create a computer game of the RTS genre, in which the player takes on the role of a commander and, using his abilities, tries to survive several waves of enemy units. To achieve the goal, it is necessary to build several types of buildings, generate a sufficient amount of raw materials and, of course, create units, taking into account their quality and number. The selection of the initial hero, the full use of his abilities and the adaptation of tactics also play a role in the pursuit of victory.

The game itself has three levels in addition to the tutorial, with the difficulty increasing linearly. Due to the considerable variety of individual game elements, such as buildings, units and enemies, it offers enough possibilities for the player's entertainment. The final enemy also plays a large part in this, which characteristics make it different from classic enemies.

I would like to continue working on the game in the future as I think it has interesting growth potential. An example could be the introduction of new units, combat vehicles and even enemies. This, of course, would also affect the number and form of enemy waves. It would also be worth considering adding new and stronger final enemies, which again should have different behaviors. I would like to supplement all this with modified or new graphics of individual game elements

and animations.

Working with individual programs was inspiring, as I could try several different tools and then evaluate myself which would be the most suitable for my purposes. Solving problems during game development alone gave me a lot of experience and a new perspective on the work of developers.

Literatura

[1] Dostupný z: ⟨www.unity.com⟩.

[2] Dostupný z: ⟨www.unrealengine.com⟩.

[3] Dostupný z: ⟨www.godotengine.org⟩.

[4] Dostupný z: ⟨www.arongranberg.com/astar/⟩.

[5] Dostupný z: ⟨https://www.drpetter.se/project_sfxr.html⟩.

[6] Dostupný z: ⟨www.boscaceoil.net⟩.

[7] Dostupný z: ⟨www.audacityteam.org⟩.

[8] Dostupný z: ⟨www.pixilart.com⟩.