Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Návrh a realizace 2D herního engine

Bc. Petr Martinec

Diplomová práce

2019

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2017/2018**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Petr Martinec**
Osobní číslo:	**I16233**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Návrh a realizace 2D herního engine**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

Cílem diplomové práce je návrh a implementace herního enginu pro 2D hry a realizace jedné ukázkové hry pomocí vytvořeného engine.
V teoretické části práce bude představena problematika herních engine se zaměřením na vlastnosti 2D enginů ? renderování (statické i animované objekty, částicové efekty), vstupy, podpora fyzikálního modelu (kolize), zpracování zvuků. Dále bude provedena rešerše dostupných enginů, jejich vlastností a licenčních podmínek jejich použití.
V praktické části bude vytvořen herní engine pro tvorbu 2D her. Engine poskytne minimálně funkcionality pro renderování výstupu na obrazovku, zpracování vstupů (klávesnice, myš, ...), podporu zvuku a základní podporu fyzikálního modelu objektů. Pro realizovaný engine bude dále implementována jedna ukázková hra, která bude demonstrovat možnosti navrženého engine.

Rozsah grafických prací: **10**

Rozsah pracovní zprávy: **60**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

MCSHAFFRY, Mike. Game coding complete. 4th ed. Boston, MA: Course Technology, Cengage Learning, c2013. ISBN 978-1-133-77657-4.
GREGORY, Jason. Game Engine Architecture. Second edition. Boca Raton, FL: CRC Press, 2015. ISBN 978-1-4665-6006-2.
HARBOUR, Jonathan S. Advanced 2D game development. Boston, MA: Course Technology/Cengage Learning, c2009. ISBN 978-1-59863-342-9.
NYSTROM, Robert. Game Programming Patterns. Genever Benning, 2014. ISBN 978-0990582908.

Vedoucí diplomové práce: **Ing. Roman Diviš**
Katedra softwarových technologií

Datum zadání diplomové práce: **30. října 2017**
Termín odevzdání diplomové práce: **18. května 2018**

Ing. Zdeněk Němec, Ph.D.
děkan

Univerzita Pardubice
L.S.
Česká republika

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2017

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne

Bc. Petr Martinec

PODĚKOVÁNÍ

Rád bych poděkovat panu Ing. Romanu Divišovi za odborné vedení, trpělivost a ochotu, kterou mi věnoval v průběhu vypracování diplomové práce.

Dále bych chtěl poděkovat především Alžbětě Ludínové, rodině a přátelům, kteří při mně celou dobu stáli a podporovali mě.

ANOTACE

Diplomové práce se zabývá návrhem a realizací herního engine, se kterým bude možné vytvářet 2D hry.

V úvodní části bude představen herní průmysl, následovaný teoretickým popisem herního enginu a rešerší dnes dostupných herních enginu. V závěru práce bude popsána realizace praktické části.

KLÍČOVÁ SLOVA

herní engine, C++, SDL

TITLE

Design and implementation of 2D game engine.

ANNOTATION

The thesis is about design and implementation of a game engine which it will be possible to create 2D games.

In the introductory part will be introduced the gaming industry, theoretical description of the gaming engine and list of existing game engines used todays. In the final part of the work will be introduce a realization of the practical part.

KEYWORDS

game engine, C++, SDL

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

2D	two-dimensional
3D	three-dimensional
API	application programming interface
CD	compact disc
CPU	central processing unit
EA	Electronic Arts Inc.
FIFA	Fédération Internationale de Football Association
FPS	first-person shooter
GPU	graphics processing unit
HW	hardware
JS	JavaScript
MMO	massively multiplayer online
MMORPG	massively multiplayer online role-playing game
NHL	National Hockey League
OpenGL	open graphics library
OS	operation system
PC	personal computer
PS4	PlayStation 4
RAM	random-access memory
RPG	role-playing game
SDK	software development kit
TPS	third-person shooter
USA	United States of America
UWP	Universal Windows Platform

ÚVOD

Herní engine tvoří jádro každé počítačové, mobilní či konzolové hry. Stará se o plynulý běh hry a několikrát za sekundu zpracovává vstupní signály od uživatele. Provádí implementovaná pravidla ve hře a výsledný vizuál posílá zpět na výstup grafické kartě.

Pojem herní engine, jakožto jádro hry, se začal objevovat v 90. letech 20. století. Oproti dnešní době, měl herní engine malý rozsah, co se kódu programu týče. S příchodem nových periferií s vyšším výkonem, rostou taktéž nároky koncových zákazníků tzv. hráčů.

S rostoucími nároky na množství problematik, které musí herní engine řešit, roste i jeho velikost. V době příchodů plnohodnotné 3D grafiky, herní enginy dosáhly takových rozměrů, že pro menší vývojářské celky bylo časově velice náročné realizovat vlastní herní engine. V této době na trhu vznikly společnosti, které se specializovaly pouze na vytváření herního enginu, který byl nabízen ostatním vývojářům výměnou za zakoupenou licenci.

Motivací pro používání herního enginu je zefektivnění práce herních vývojářů. Při práci s hotovým kompletním nástrojem, kterým je herní engine, lze při dobré znalosti tohoto nástroje efektivně, a relativně rychle, vytvářet hry, z čehož plynou větší zisky, tedy z prodeje zrealizovaných her.

Návrh a implementace herního enginu, ve kterém lze realizovat 2D hru, je předmětem praktické části této práce. V teoretické části bude rozebrána obecná problematika herních enginů, vznik tohoto pojmu, výčet dnes nejpoužívanějších herních enginů, a také výčet herních žánrů, které svými mechanikami kladou všemožné nároky na herní enginy.

1 HERNÍ PRŮMYSL

Jedná se o jedno z odvětví tvořící zábavní průmysl. Na jeho začátcích nesklidil až tak velký zájem. Vytvářené projekty v tomto odvětví, jsou dnes stejně finančně náročné svým rozpočtem, jako nejlepší projekty ve filmovém průmyslu. Zisk v herním průmyslu každoročně roste. Důvodem každoročního zvyšování zisků je snaha rozšířit průmysl na co nejvíce platforem a k co možná nejvíce potenciálním zákazníkům.

1.1 Hry

Produktem herního průmyslu jsou hry, které jsou určené pro herní platformy, kterými jsou dnes počítače, konzole anebo mobilní zařízení. Pojem hra bývá též někdy doplněna slovním spojením počítačová, konzolová či mobilní hra, v závislosti, na jakou platformu je určena. V této práci bude nadále hovořeno převážně o počítačových hrách. Tento pojem vznikl spojením dvou slov počítač a hra. Slovo hra je již několik století používáno v souvislosti s klasickými karetními nebo stolními hrami, příkladem takových her lze uvést šachy či poker. Základním významem slova hra je činnost, při které se člověk baví a zároveň se neustále učí [1]. Dále označení počítač, označuje elektronické zařízení, na kterém je hra zprostředkovaná.

Počítačová hra je jakousi jednoduchou simulací světa, jehož pravidla se řídí předpisy jeho tvůrců, kterými jsou vývojáři [1]. Tento svět může být simulací blížící se reálnému světu či se může jednat o čistou fantazií jeho tvůrců. To se poté odráží na dodržení pravidel, která jsou známá z reálného světa. Každý vytvořený virtuální svět má jednu a tu společnou věc. Tou věcí je matematický model, kterým je tvořen, a díky jemuž je počítač schopen takovýto svět vůbec zprostředkovat. Výsledný virtuální svět s danými pravidly a příběhem je nazýván počítačovou hrou.

Od hráče je vyžadováno, aby počítači dával instrukce prostřednictvím vstupních periferií, jako je například klávesnice nebo myš, tím pádem hra reaguje na poskytnutá vstupní data. Díky tomu je schopen hráč putovat virtuálním světem a při průchodů jednotlivými úrovněmi se učí pravidlům virtuálního světa hry.

1.2 Herní žánry

Hry lze kategorizovat do několika herních žánrů dle herních mechanik, kterými disponují. Herní žánr je podstatnou součástí hry, protože určuje základní předpisy, na kterých bude celý herní svět vybudován [1]. Konkrétní herní žánr ovlivňuje samotný vývoj hry. Různé žánry mají

i jiné požadavky na herní mechaniky. Pakliže si vývojáři nechtějí vytvořit vlastní herní engine, musí si zvolit takový, ve kterém je možné takovýto herní žánr realizovat [1]. Některé herní enginy jsou zaměřeny výhradně na určité typy žánrů a špatná volba enginu, může vést k neúspěchu vývoje.

V posledních letech se stává pravidlem, že hra disponuje více žánry, které jsou mezi sebou různě nakombinovány [2]. Kategorizace je obdobná jako ve filmovém průmyslu, kde se filmy kategorizují kupříkladu na komedie, sci-fi či horor.

V následujících podkapitolách bude zmíněno a popsáno několik nejčastějších herních žánrů.

1.2.1 Karetní

Jedná se o žánr, který přenáší reálné karetní hry, které lidstvo už nějakou dobu zná, do virtuálního světa [3]. Existují také karetní hry, které nemají přímý základ v reálném světě. Takovýmto zástupcem je kupříkladu hra *HearthStone* od společnosti *Blizzard Entertainment* [4]. V této hře hráči vykládají na stůl karty, jež reprezentují jednotky nebo kouzla a jejímž cílem je dostat životy nepřítele na nulovou hodnotu.

1.2.2 Adventury

Tento žánr je zaměřený především na vyprávění příběhu. Hráč prochází herním světem povídá si s herními postavami, sbírá různé předměty a řeší přichystané úkoly, na které v průběhu hry naráží [3]. Pro žánr je typická existence logických hádanek, kde hráč kombinuje několik herních předmětů, které následně může použít [3]. V tomto žánru se lze nejčastěji setkat s pojmem „point and click“, kde hráči stačí k ovládání hry pouze myš, se kterou kliká na herní předměty na obrazovce [3]. Zástupcem tohoto žánru, lze uvést známou českou herní sérii *Polda*.

1.2.3 Strategie

Strategie je jeden z žánrů vyžadující velkou pozornost a přemýšlení hráče. Typické pro tento žánr je pozice kamery, která je umístěna nad herní mapou a hráč se na ní dívá jakoby ze shora [1]. Nejčastěji se lze setkat se stavitelskými strategiemi a strategiemi probíhající v reálném čase.

Stavitelské strategie se zaměřují na ekonomický růst například města. Hráč si musí hlídat přísun potřebného materiálů a řídit jejich následné zpracování v jiné materiály [3].

Strategie v reálném čase jsou většinou zaměřené na velení armády, kde je hlavním úkolem zničit protivníky. Hráč se musí postarat, o dostatek potřebného materiálu na produkci staveb, nových jednotek či pro různá vylepšení [3]. Zároveň má na starost produkci a stavbu infrastruktury, taktéž hráč musí velet svým vojákům, které posílá do bitev.

1.2.4 FPS

Anglická zkratka s celým názvem first-person shooter, se dá do češtiny přeložit, jako střílečka z pohledu první osoby [1]. Děj hráč prožívá skrze pohledu hlavního protagonisty. Žánr se taktéž vyznačuje rychlou akcí, například střelbou ze zbraní, kde hráč musí rychle reagovat, aby nebyl zabit [1].

1.2.5 TPS

Žánr označovaný anglickou zkratkou TPS, celým názvem third-person shooter, do češtiny volně přeloženo jako střílečka ze třetí osoby, jedná se o odnož žánru FPS. Velkým rozdílem je zde pohled hráče na svět [1]. Hráč se zde dívá na svět jako by poletoval za zády hlavního protagonisty. Díky jinému pohledu na svět, má hráč lepší přehled o situaci okolo něj. Umožňuje mu to se dívat za rohy stěn či objektů, u kterých se právě nachází.

1.2.6 Sportovní

Typicky se jedná o převedení reálných sportů do virtuálního světa, a to i se všemi jejich pravidly, které k nim náleží [3]. Jako zástupce uvedeného žánru lze zmínit herní série *FIFA* a *NHL* od společnosti *EA*.

1.2.7 Závodní

Žánr vyznačující se simulací jízdního modelu aut či jiných silničních prostředků. Simulace jízdního modelu v závodních hrách může být různá [1]. Může se jednat o simulaci mezi dvěma extrémy. Prvním jsou závody tzv. simulátory, kde se klade velký důraz na provedení simulace jízdního modelu vozidel, které se co nejvíce blíží reálnému světu [1]. Druhým jsou arkádové závody, kde se simulace reálného jízdního modelu neřeší do hloubky, ale jedná se především o to, aby hra přinesla co nejlepší požitek [1].

1.2.8 RPG

Zkratka označující anglická slova role-playing game, do češtiny volně přeloženo jako hra na hrdiny. Jedná se o jeden z největších herních žánrů, pro který je charakteristický velký a otevřený svět s obsáhlým příběhem [3]. Hráč se zpravidla zhostí role protagonisty, se kterým prochází herním světem a odhaluje, tak jeho příběh.

K žánru taktéž patří tzv. strom zkušeností, kde se hráč učí nové schopnosti nebo vylepšuje ty, které již ovládá [3]. Podle pořadí naučených schopností si hráč sám určuje, jakým stylem bude procházet hrou. Samozřejmostí je starání se o vybavení hlavního protagonisty, které při průchodu světa hráč získává, nakupuje nové či vylepšuje ty stávající. Výbava hrdiny má pak vliv na jeho útok, odolnost nebo výdrž.

1.2.9 Simulátory

Tento herní žánr se snaží hráči dodat co nejreálnější zážitek při hraní, jako kdyby tu stejnou činnost prováděl v reálném světě [3]. Typicky se jedná o simulátory dopravních nebo leteckých her [3]. Některé herní simulátory mohou být až natolik podobné reálnému světu, že se mohou používat při školení lidí, kupříkladu řidičů.

Jedná se o minoritní herní žánr v herním průmyslu. Důvodem je malé množství hráčů, jenž považují tento žánr za zábavný. Česká *hra Euro Track Simulator 2* je jednou z her tohoto žánrů.

1.2.10 MMO

Anglická zkratka označující slova massively multiplayer online game, do češtiny lze volně přeložit jako hra pro obrovské množství hráčů. MMO se vyznačuje možností hrání hry s velkým počtem hráčů ve stejném čase [1]. Obvykle se jedná o tisíce nebo statisíce hráčů hrajících v jednom virtuálním světě. Velký počet hráčů hrajících společně v jednom čase přináší speciální herní zážitek, který jiné žánry nejsou schopné poskytnout [1]. Tento speciální herní zážitek má, ale i speciální nároky na herní engine, který musí být schopen snést nápor velkého množství hráčů. Herní engine je ve většině případů vytvořen přímo na míru dané hře.

Všichni hráči mohou zároveň provádět činnosti, které vytvoří množství herních objektů. Pakliže se tak stane je potřeba zajistit synchronizaci mezi všemi hráči, aby měli, stejné animace činností postav, stejně vygenerované herní objekty a vše se nacházelo na správném místě v daném virtuálním světě [1].

Kromě herního enginu je nedílnou součástí žánru velké množství serverů, které se starají o možnost společného hraní hráčů [1]. Servery musí být natolik výkonné, aby dokázaly bez problému komunikovat s klientem hry s co nejkratší možnou odezvou. Náklady na provoz a údržbu serverů jsou velkou finanční zátěží, které se dlouhodobě nedají financovat ze zisků prodaných kopii hry. Proto je u tohoto žánru běžné, že hráči platí měsíční poplatky [1]. Ze zaplacených poplatků je jim umožněno opět po určitou dobu hrát. Finance z poplatků slouží pro financování provozu serverů a zaměstnanců pracující jako podpora určená pro hráče.

Nejznámějším představitelem žánru MMO je počítačová hra *World of Wardraft* od společnosti *Blizzard Entertainment*, kombinující žánry MMO a RPG [1]. Aktuálně se jedná o největší a nejpopulárnější hru tohoto žánru. Další hrou je pak *Guild Wars 2* od společnosti *ArenaNet,* která se od ostatních odlišuje absencí měsíčních poplatků [1].

1.3 Kategorizace her

Finální hru lze rozdělit do dvou kategorií, které se určují podle finančních možností vývojářů. Samotná kategorie ovšem, nevypovídá nic o výsledné kvalitě, ba ani o celkové zábavnosti hry.

1.3.1 AAA

Tituly označovány jako AAA, jsou zpravidla vytvářené v týmu lidí, čítající stovky nebo tisíce vývojářů [5]. Herní projekty jsou finančně podporovány vydavateli, kteří mají na samotný vývoj určený rozpočet a jde jim hlavně o dosažení co nejvyššího výdělku [5].

Z grafického hlediska, se jedná o hry s co nejreálnějším grafickým zpracováním, která jsou v danou dobu vydáni hry možná. Z herního hlediska se jedná o hry s velkým virtuálním světem, propracovaným příběhem a s různými minihrami, které obohacují herní zážitek [5]. Z důvodu vysoké investice do takových to projektů, je ve většině případů koncová cena pro zákazníka stanovena mezi 40 $ až 60 $ [5].

1.3.2 Indie

Takovéto hry vznikají převážně v herních studiích čítající několik desítek lidí nebo se jedná o díla jednotlivců [5]. Tyto hry mají zřídka finanční podporu od vydavatele. Vývoj je většinou financován samotnými tvůrci či z crowdfundingových platforem, jako je například

Kickstarter nebo český *Startovač* [5]. Na těchto platformách samotní hráči mohou finančně podpořit vývoj titulu.

Vzhledem k omezeným finančním prostředkům, jsou Indie hry menšího rozsahu a většinou spoléhají na unikátní herní mechaniky nebo umělecký vizuál, kterým se mohou odlišit od ostatních her, a tak se zalíbit potenciálním kupcům. Výsledná cena takových to her se pohybuje okolo 20 $ [5]. Požadavky na velký výkon počítače jsou zanedbatelné na rozdíl od titulů her typu AAA [5]. Cenově jsou daleko přístupnější pro kupce a samotnou hratelností nevyžadují od hráče příliš mnoha zkušeností.

1.4 Počátky herního průmyslu

První počítačové hry se objevili v polovině 20. století [6]. Z dnešního pohledu se jednalo o jednoduché minihry, které si dnes lze stáhnout na mobilní telefon a zvládnou zabavit uživatele kupříkladu po cestě do školy či práce. Situace na trhu se změnila s příchodem herních konzolí v 70. letech 20. století [6]. Herní konzole je ve své podstatě drobný počítač umístěný v malé přenosné krabičce. Ty byly, a dnes stále jsou, určeny výhradně pro hraní her. Jsou vybavené svým operačním systémem a od uživatele vyžadují pouze připojení k obrazovce, bez potřeby jakéhokoliv dalšího složitého nastavování [1].

Za velkou změnu v herním průmyslu, a její následný rychlý vzestup, se zasloužila Japonská společnost *Sony* [6]. Ta v 90. letech dostala zakázku od konkurenční firmy na vytvoření nové herní konzole. Kvůli jistým okolnostem byla zakázka pro konkurenční firmu zrušena a vedení firmy *Sony* se rozhodlo, že projekt dokončí pod svou společností [7]. Důvodem proč se tak Sony rozhodlo, bylo že už samotný vývoj konzole byl téměř u konce [7].

V roce 1994 byla na trh uvedena první herní konzole od společnosti Sony, kterou pojmenovali *Playstation* [7]. Na svou dobu se jednalo o velice výkonnou herní konzoli, revoluční byla v používání CD, které bylo v té době novinkou [7]. Konzole si dokázala poradit i s vykreslováním 3D grafiky [7]. Společnost Sony učinila ještě jedno důležité rozhodnutí spojené s jejich herní konzolí. U vývojářských studií si zaplatila exkluzivitu výhradně pro svou konzoli, několik herních studií i rovnou koupila, a některá z nich fungují dodnes [7]. Vývojáři s tím neměli velký problém, protože konzole byla nejvýkonnější ve své době a vývojáři si mohli, tak dovolit věci, které na jiných konzolích nešly z důvodu omezeného hardwaru [7].

Díky zajištění dostatku herních titulů od vývojářů třetích stran se herní konzole *Playstation,* stala jednou z nejprodávanějších herních konzolí na světě. Do dnešního dne se prodalo více, jak sto milionů kusů této konzole [7].

1.5 Současná situace

V posledních letech se herní konzole staly hlavním hybatelem herního průmyslu, jelikož se jedná o uzavřený systém a tvůrcům stačí odlaďovat svoje hry pouze na konkrétní zařízení. To velice snižuje náklady potřebné pro testování a odlaďování hry.

Situace na počítačích je více komplikovanější kvůli velkému množství možným kombinacím procesoru, grafické karty, a i dostupnosti velikosti paměti RAM, kterou může mít zákazník ve svém počítači.

V posledních letech se herní průmysl více přesouvá z počítačů a herních konzolí na mobilní platformy. Za rok 2018 pocházela přibližně polovina zisku v herním průmyslu z her na těchto platformách [8]. Celkový roční zisk v herním průmyslu byl za rok 2018 137,9 miliard amerických dolarů [8]. Největší část výdělku pochází převážně z Asie, kde se má situace tak, že lidé hrají nejčastěji na mobilních platformách při cestě nebo přestávkách v práci [8].

Platební model mobilních her, je od her určených na jiné platformy naprosto odlišný. Hry jsou ve většině případů zdarma a zisky ze hry pochází z nákupu herních věcí přes integrovaný obchod přímo ve hře tzv. mikrotransakce [9]. Herní věci mohou být čistě kosmetického rázu nebo se může jednat o nákup speciální herní měny, za kterou si hráč zpřístupní speciální nedostupný obsah. V obou případech se jedná o dobrovolný nákup a hráč si je schopen hru užít i bez těchto nadstandardních služeb [9]. Mikrotransakce jsou cílené především na hráče, kterým nevadí utrácen za virtuální věci a jsou ochotni utratit velkou finanční sumu, která je schopná pokrýt náklady na vývoj a údržbu hry.

Hry často také obsahují herní měnu, kterou lze získat i pouhým hraním [9]. Tento postup je, ale časově zdlouhavý, při kterém je od hráče vyžadováno investování právě již zmíněného velkého množství volného času do hraní. Pro ty, kterým se nechce investovat svůj volný čas pro zisk herní měny, existuje možnost si tuto měnu dokoupit a urychlit tím svůj postup ve hře [9]. Jednotlivé částky za herní věci jsou oproti hrám na konzolích minimální.

2 HERNÍ ENGINE

Herní engine je termín označující část hry tvořící její jádro, a které zajišťuje běh na dané platformě [1]. Termín se do širšího podvědomí dostal v polovině 90. let 20. století [1]. V dnešní době je herní engine nedílnou součástí takřka každé nově vydané hry. Termín lze do češtiny volně přeložit jako herní motor. Vývoj a údržbu herního enginu má na starost většinou speciálně vyhrazený tým vývojářů. Ten se následně stará o kompatibilitu pro nově vydaný hardware, opravu chyb a o přidávání nových funkcionalit.

2.1 Co je to herní engine

Jedná se o sadu nástrojů určených pro torbu her [2]. Z pohledu zdrojového kódu se jedná o tu část kódu, která je univerzální a lze ji používat na různých projektech. Samotný herní engine se skládá ze samostatných komponent, které se starají o danou problematiku, například o přístup k periferním zařízením [2].

2.2 Hra a herní engine

Herní engine a hra jsou dva různé produkty, které jsou ovšem zákazníkovi dodávány, i jako komplet skrze prodávanou hru [1].

Herní engine je sada nástrojů určené pro herní vývojáře, kteří platí za jeho používání zakoupením licence nebo odváděním procentuálního zisku z prodeje [1]. Pro tvůrce, kteří jej budou používat, nabízí řešení problémů, které šetří mnoho času potřebného pro vývoj. Jedná se o tu část kódu, kterou lze opakovaně používat k tvorbě her.

Hra je výsledný produkt určený k prodeji široké veřejnosti. Jedná se o tu část kódu, která musela vzniknout na daném herním enginu a tvoří virtuální svět s příběhem, herními mechanikami a výsledným vizuálním zpracováním [1]. Tedy to, co nelze použít opětovně na odlišném projektu.

Ve většině případů lze lehce definovat, co vše ve hře má na starosti herní engine, a co bylo vytvořeno speciálně pro hru [1]. Existují ovšem případy, kdy je hra s herním enginem natolik propojena, že nelze přesně říci co je co. Je tomu především u her z žánru MMO, kde je herní enginu vytvářen přímo na míru výsledné hře [1].

2.3 Vznik prvního herního enginu

Zásluhy za vytvoření prvního herního enginu jsou přikládány herní společnosti *Id Software* [10]. Jedná se o společnost, jejíž tvůrci se zasloužili o vytvoření, dnes legendárních herních značek, kterými jsou např. *Wolfenstein* nebo *Doom* [10].

Vývojáři z firmy Id Software byli na svou dobu natolik pokrokový a geniální, že i v době stále omezeného výpočetního výkonu pro grafické operace, dokázali přijít s hrou, která na první pohled budila dojem 3D hry. Jednalo se o dobu 90. let 20. století, kdy byly na vrcholu hry s 2D grafikou a příchod 3D her byl stále limitován nedostatečným výpočetním výkonem pro grafické operace [10]. Jednou z her, u které se tomu tak stalo byl *Wolfenstein 3D* [10]. Za vznik této technologicky pokrokové hry lze vděčit hlavně dvěma lidem, a to Johnu Romerovi a Johnu Carmackovi [10]. Hra pro vykreslování grafiky používá algoritmus zvaný *ray casting* [11].

Algoritmus transformuje 2D herní svět, který je reprezentován například dvourozměrným polem, do 3D zobrazení. Každé políčko na mapě nese informaci, o jaký objekt se jedná, tedy zdali se jedná o chodbu nebo zeď [11]. Podle typu objektu je aplikovaná jiná textura. Hráči je, pak vykreslován virtuální svět podle pozice a rotace na mapě. Výsledná vykreslená grafika, poté budí dojem pravé 3D grafiky. Ve skutečnosti se jedná o tzv. pseudo 3D grafiku, protože se hra stále odehrává ve 2D prostoru. Tento pokrok v grafickém vyobrazení herního světa inspiroval další vývojáře her a než na trh přišel HW, který dokázal pracovat s pravou 3D grafikou, vzniklo mnoho dalších herních titulů inspirovaných právě hrou *Wolfenstein 3D* a jeho vykreslovacím algoritmem [1].

2.4 Potřeba existence herního enginu

Důvodů existence herních enginů je hned několik. Ty nejdůležitější budou v následujících podkapitolách rozebrány.

2.4.1 Zkrácení doby vývoje

Doba, kdy bylo možné realizovat novou hru, od úplné nuly, skončila na konci 90. let 20. století [1]. V té době se na trh pomalu dostával HW nazývaný grafický akcelerátor [1]. Ten měl na starost pouze zpracování grafického výstupu, který do té doby prováděl hlavní procesor počítače, jenž tuto činnost prováděl neefektivně. Díky tomu mohl být výkon procesoru použit k provádění jiných výpočtů. Z grafických akcelerátorů se později staly grafické karty, které jsou známé dodnes [1].

S grafickými kartami přišly i 3D hry, které vyžadovaly složitější matematické výpočty, které 2D hry nemají. Rostoucí množství kódu potřebného pro hru a s rostoucími nároky hráčů na kvalitu obsahu ve hře, vzrostly a prodloužily, tak potřebnou dobu vývoje.

Vývojáři potřebovali zkrátit dobu potřebnou pro vývoj hry na minimum, aby dokázali s hrou přijít na trh dříve než konkurence a dříve, než bude překonána jiným titulem. Herní enginy vznikly, jako nástroj potřebný pro zkrácení nutné doby na vývoj [12]. Vývojáři díky enginu nemuseli ztrácet čas implementací např. grafického API pro vykreslování, a mohli se, tak plně soustředit na tvorbu herního světa.

2.4.2 Spolupráce

Enginy, kromě zkrácení doby potřebné pro vývoj hry, přinesly i efektivnější způsob, jakým může vícero vývojářů spolupracovat na jednom projektu [1]. Všichni vývojáři pracují s jedním konkrétním nástrojem. Jakožto samostatný nástroj určený pro vývoj her, vyžaduje od vývojářů pouze znalost jeho používání, které je často dobře popsáno v dokumentaci.

Jelikož herní engine a hra jsou dva různé produkty, vývojáři si potřebují pouze zálohovat tu část kódu, která tvoří hru, protože zdrojové kódy pro danou verzi enginu si můžou stáhnout z oficiálních repozitářů. Díky tomu jsou sníženy kapacitní nároky na uložiště, na kterém se zálohuje.

2.4.3 Kompatibilita zařízení

Časově implementačně náročným úkolem je zajistit kompatibilitu s co možná největším počtem kombinací dostupného HW a operačního systému, které konečný zákazník může vlastnit. Jeden konkrétní HW může ve hře způsobovat chybu, která se na jiném HW nevyskytuje. Možnosti herních vývojářů na odhalování těchto problémů jsou značně omezené.

Herní enginy, které jsou používány na větším počtu projektů, mají větší šanci na odhalení těchto problémů díky své rozšířenosti. Pakliže se takovýto problém vyskytne, je opraven vývojáři zodpovídající za engnine v rámci opravného balíčku. Herním vývojářům, tak stačí pouze pracovat s aktuální verzí enginu.

2.4.4 Přenositelnost mezi platformami

Vývoj hry, cílící na více platforem současně, je časové náročný. Herní enginy tuto práci ulehčují díky zabudovaným systémům kompilátorů pro cílovou platformu. Problémem

na různých platformách je například používané grafické API. Na operačních systémem *Windows* je používáno především *DirectX*, které ostatní platformy nemají [1].

3 KOMPONENTY HERNÍHO ENGINU

Jak již bylo řečeno herní engine je sada nástrojů. Celý zdrojový kód se skládá z jednotlivých komponent, které mezi sebou spolupracují a jsou na sobě více či méně závislé. Samostatná komponenta je implementací určité problematiky, kterou má na starosti [1]. Komponenty mohou být na enginu nezávislé a tvůrci enginů mohou komponenty jednoduše přidávat, odebírat či zaměňovat.

V následujících podkapitolách bude zmíněno a popsáno několik takovýchto komponent. Popis bude zaměřen především na problematiku týkající se 2D her.

3.1 Herní smyčka

Je volným překladem z anglického názvu game loop. Herní smyčka tvoří základ celého herního enginu. Implementačně je herní smyčka cyklus, který aktualizuje stav všech ostatních částí herního enginu a běží po celou dobu co je hra spuštěná [1]. Některé komponenty, kterými jsou například animace, vyžadují pro vykreslování svého stavu aktualizování 30krát nebo 60krát za sekundu [1]. Oproti tomu taková simulace fyziky si může vyžadovat aktualizaci až 120krát za sekundu [1]. Jiné komponenty výší úrovně, pro kterou je typická umělá inteligence, potřebuje aktualizaci pouze párkrát za sekundu [1].

Existuje mnoho různých implementací herní smyčky. Tou nejzákladnější implementací je jeden hlavní cyklus, ve kterém se postupně aktualizuje vše. Jednotlivé činnosti, které se aktualizují, lze rozdělit do tří částí. V první části se přijímají všechny události vzniklé ze vstupních periferií od uživatele. V druhé části se provede aktualizace stavů všech herních objektů a jejich komponent. Poslední částí je vykreslení aktuálního stavu na výstupní periferie pro uživatele. Vše se děje několikrát za sekundu. Při vykreslování je nejdůležitější, aby počet vykreslovaných snímků neklesl pod 25 snímků za sekundu, pakliže by se tak stalo, uživatel by měl dojem, že se obraz seká.

Následující kód představuje základní implementace herní smyčky:

```
while (true) {
    processInput();
    update();
    render();
}
```

3.2 Simulace fyziky

Každá dnešní hra potřebuje do určité míry simulovat fyziku, výjimku tvoří například karetní hry. Samotná komponenta starající se o simulaci je natolik obsáhlá, že lze hovořit o fyzikálním enginu [1]. Ten může být implementovaný tvůrci herního enginu nebo pouze do něj zakomponovaný od vývojářů třetích stran.

Základním úkolem fyzikálního enginu je kromě simulování fyzikálních vlastností, dynamických i statických herních objektů, i detekování kolizí mezi všemi herními objekty, u kterých se provádí simulace [14].

3.2.1 Detekce kolize

Detekování kolize potřebuje engine například z důvodu, že objekt, který padá v důsledku simulované gravitace narazil na překážku, která jeho pád zastavila. Nebo, že hlavní hrdina hry si nějak ublížil, a tímto činem hra zareagovala událostí odečtením jednoho života hrdiny. Pro detekci kolizí je třeba znát několik informací o kolizním objektu, který má na sobě herní objekt. Je potřeba znát tvar a velikost kolizního objektu a následně také pozici v prostoru, kde se právě nachází herní objekt.

K vyhodnocení existence kolize mezi dvěma obdélníky je třeba znát pozici jeho čtyř bodů v prostoru, které tvoří jeho rohy [14]. Jedná se o pravý horní, levý horní, pravý dolní a levý dolní roh. Postupuje se nejprve tak, že se první obdélník dotazuje druhého, zdali se v oblasti, kterou zabírá v prostoru nehází jeden z jeho rohů. Následně se druhý zeptá prvního obdélníku. Pokud se nějaký bod nachází v prostoru toho druhého obdélníku, následuje vyvolání události o vzniku kolize.

Následující kód reprezentuje implementaci kódu pro vyhodnocení, zdali se bod nachází uvnitř obdélníku:

```
return
  (myX >= pointX1 && pointX2 >= myX &&
   myY >= pointY1 && pointY2 >= myY);
```

3.3 PhysX

Jedná se fyzikální engine ve vlastnictví společností *NVIDIA* [1]. Simulační výpočty mohou probíhat na GPU, která funguje jako druhý procesor. Tato technologie není závislá na konkrétním HW a může běžet čistě na CPU bez podpory GPU [1]. Pro vývojáře, kteří chtějí

použít tento fyzikální engine, je volně dostupné SDK na oficiálních stránkách, které je podporováno na všech hlavních herních platformách.

3.3.1 Box2D

Fyzikální engine implementovaný vývojářem Erin Catto [15]. Jedná se fyzikální engine napsaný v jazyce C++ a je sdílený pod volnou licencí zvanou zlib [15].

3.4 Zvuk

Jedná se o jednu z nejdůležitějších částí tvořící výsledný dojem hry. Bez kvalitního hudebního doprovodu a zvukových efektů nelze zaujmout mnoho potencionálních kupců výsledné hry. Problematika fyziky zvuku je obdobně náročná jako u fyzikálního enginu. Hotová řešení, která řeší tuto problematiku v oblasti zvuku se nazývá zvukový engine [1].

Zvuk se ve 2D prostoru může šířit třemi směry [1]. Prvním je směr všesměrový, kdy se zvuk šíří od místa vzniku do svého okolí všemi směry stejně. Druhý směr je kuželovitý, zde se zvuk šíří od místa vzniku ve tvaru připomínající kužel. Poslední směr je směrový, zde se zvuk šíří pouze jedním směrem, a to od místa svého vzniku. Při šíření zvuku prostorem je potřeba řešit jeho hlasitost, která se musí měnit v závislosti v jakém prostředí se šíří, a jak daleko je posluchač od místa jeho vzniku.

Typy druhů šíření zvuku jsou znázorněna na obrázku 1.

Obrázek 1: Typy šíření zvuku. Zleva doprava: všesměrový, kuželovitý, směrový [1]

3.4.1 FMOD

Jedná se o profesionální zvukový engine, který je pro nekomerční účely poskytován zcela zdarma [15]. Komerční použití vyžaduje zakoupení licenci, která pro nezávislé vývojáře

s rozpočtem do 500 000 $ je pro první rok užívání zcela zdarma, a za každý následující rok je účtováno 2 000 $ [15]. Pro větší projekty jsou ceny vyšší. *FMOD* podporuje celou řadu zvukových formátů a je podporován na všech hlavních herních platformách. Tento zvukový engine je zabudovaný v herních enginech *Unity* a *Unreal engine* [15].

3.5 Vstupy

Velice důležitým úkolem herního enginu, je umět přijímat a zpracovávat informace ze vstupních periferií zařízení. Informací může být například, že uživatel klikl na pravé tlačítko myši, engine s touto informací pracuje vyvoláním události na všech herních objektech, které čekají až bude provedena tato akce [14]. Základními vstupními periferiemi počítače je klávesnice a myš. Dalšími vstupními periferiemi, kterými může počítač disponovat, je například možnost uskutečnit hovor, joystick nebo gamepad [14]. Pro zpracování informací z těchto dalších vstupních zařízení je většinou potřeba použít knihovnu.

3.5.1 DirectInput

Jedná se o API používané pro zpracování vstupních informací pocházejících z herních ovladačů [17]. Jedná se o technologii od společnosti Microsoft a je dostupná pro programovací jazyky C a C++ [17].

3.6 Renderování

Je proces při, kterém se vytváří výsledný obraz na základě 2D nebo 3D počítačového modelu [1]. Tato část enginu má tedy na starosti vše, co hráč uvidí na svém monitoru. Aby bylo možné vykreslit výsledný obraz na monitor, je potřeba předat data grafické kartě ke zpracování. Pro komunikaci s grafickou kartou se používá tzv. grafická API [14].

3.6.1 Grafická API

Grafické API definuje způsob komunikace mezi programem a grafickou kartou, která se stará o výsledný grafický výstup zobrazovaný na monitoru [14]. API definuje funkce pro práci s grafikou kartou, tyto funkce jsou přesně definované. Definice programátorovi říká, jakou činností funkce provádí, a jaké jsou vstupní a výstupní parametry funkce.

Specifikace ovšem neurčuje konkrétní implementaci definovaných funkci [18]. Za konečnou implementaci jsou zodpovědní vývojáři, kteří jsou povinni dodržet, co možná nejpřesnější specifikaci. Tvůrci konkrétních implementací grafických API jsou obvykle vývojáři ovladačů grafických karet [18]. Pokud mají definované funkce na určitých grafických kartách nechtěného chování, jedná se o chybnou implementaci specifikace.

Grafické karty API umožňují práci s 2D i 3D grafikou.

OpenGL

Standard *OpenGL* je specifikován, vyvíjen a udržován skupinou *Khronos*. *OpenGL*. Je nejrozšířenějším grafickým API, které je podporované na každém zařízením s grafickým výstupem [18]. Tento standard je nezávislý na operačním systému. Knihovna *OpenGL* se pravidelně aktualizuje, aby bylo možné pracovat s nejmodernějšími grafickými kartami.

První verze této knihovny vznikla v roce 1992, jako otevřený standard a jednalo se, tak o alternativu pro grafické API *IrisGL* pro počítače Silicon [19]. V roce 2017 byla vydána poslední a aktuální verze této knihovny s pořadovým číslem 4.6 [19].

DirectX

Jedná se o grafické API vyvíjené společností *Microsoft* [1]. Jedná se o alternativu k *OpenGL*. Toto grafické API je možné používat pouze na operačních systémech Windows a jeho odnoži používané v herních konzolích *Xbox* [1].

Vulkan

Vulkan je obdobou standardu *OpenGL*, který je vyvíjen a udržován stejnou skupinou [20]. Jedná se o nástupce standardu *OpenGL*, se kterým postrádá jakoukoliv zpětnou kompatibilitu. *Vulkan* nabízí nižší režii, díky které je méně zatížen procesor a disponuje, tak s více přímým přístupem ke grafické kartě [20].

3.7 Animace

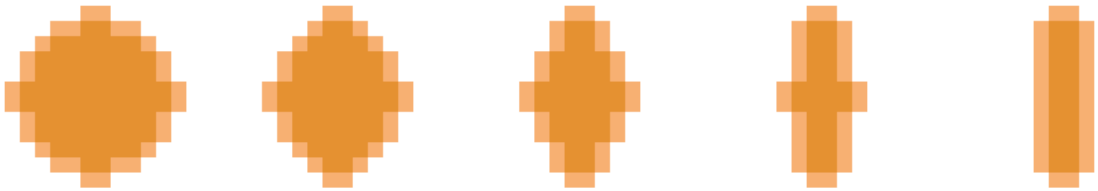
Jedná se o část enginu, jenž dodává dojem opravdové reality hernímu světu. Nejčastěji se uplatňuje u hlavních protagonistů, kteří putují světem a je potřeba, aby při jejich pohybu odpovídal i vizuál [13]. Animačních technik používaných ve hrách je několik. V následujících podkapitolách budou popsány dvě nejčastější techniky používané pro 2D herní objekty.

3.7.1 Sprite

Nejednoduší technikou pro animování 2D herních objektů je metoda v angličtině označována názvem sprite. Jedná se o techniku, kdy pro herní objekt existuje řada textur tvořící animační snímky, které jsou mu v průběhu času zaměňovány [14]. Animační snímky většinou na sebe přesně navazují, pakliže výměna textur probíhá neustále, dostávají tak dojem pohybujícího se objektu.

Problémem této techniky je existence velkého množství animačních snímků [14]. U detailnějších animací se typicky jedná o desítky snímků. Nejlepším řešením je mít všechny animační snímky v jednom obrazovém souboru, a v jednom čase vykreslovat pouze jeho určitou oblast [14]. V případě, že by každý animační snímek byl ve vlastním obrazovém souboru, vzniklo by zde riziko selhání načtení souboru. Také to zvyšuje počet souborů a nepřehlednost o obsahu všech snímků.

Příklad animace sprite je znázorněn na obrázku 2.



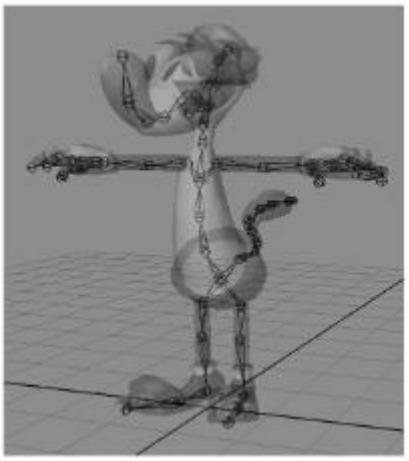
Obrázek 2: Animace otáčení mince

3.7.2 Skinned

Do češtiny lze volně přeložit jako animace pokožky. Jedná se o animační techniku, u které dochází k transformaci souřadnic a velikosti herního objektu [1]. Jeden hlavní herní objekt je tvořen skupinou dalších herních objektů, například ruce, nohy, hlava a trup. Na hlavní herní objekt je aplikována tzv. kostra, ta se skládá z jednotlivých kostí a kloubů a jsou provázané s vnitřními objekty [1]. Pokud dojde k pohybu části kostry, dojde zároveň k pohybu objektů, jenž jsou na tuto část připojeny.

Na rozdíl od předchozí metody, zde není potřeba existence desítek animačních snímků, ale je třeba vytvoření několika animací kostry, které si vyžadují přesné načasovaní, aby animace vypadala, co možná nejvěrohodněji.

Přiklad animace skinned je k dispozici na obrázku 3.



Obrázek 3: 3D model s nastavenou animační kostrou [1]

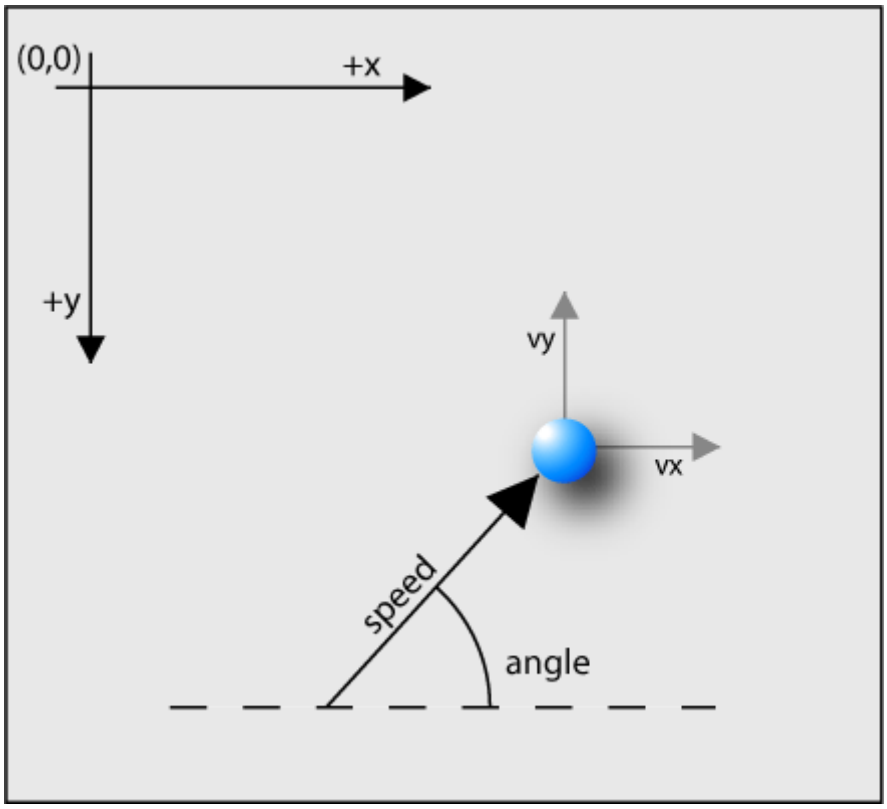
3.8 Částicový systém

Tato součást enginu se stará o simulaci částic. Technicky se jedná o kombinaci dvou částí herního enginu. Těmito dvěma částmi se rozumí simulace fyziky a renderování objektu. Částicový systém umožňuje vygenerovat velké množství malých objektů, tedy částic, s různými náhodně nastavenými parametry, které jsou jinak těžko reprodukovatelné [21]. Výsledkem vygenerovaných částic je například efekt sněžení, hoření, jiskry, poletující prach nebo vodní gejzír.

Na vytvoření částicového systému je potřeba, aby každá částice obsahovala souřadnice, rychlosti, směry úhlů a času [21]. Souřadnice určují počáteční pozici po vygenerování objektu. Vlastnost rychlosti udává rychlost, jakou se bude částice pohybovat ve virtuálním světě. Čas určuje za jak dlouhou dobu částice zanikne. Poslední vlastností je určování úhlů, jakým směrem se částice bude pohybovat ve virtuální světě. Počet úhlů je závislý na počtu dimenzí, ve kterých se hra odehrává. Každé částici jsou při jejím vzniku nastaveny vlastnosti jinak, prostřednictvím náhodně vygenerovaných dat.

Částice je ve své podstatě herní objekt, ke kterému lze přidat komponentu, jako je například komponenta pojednávající o simulaci gravitace. Například, kdyby se vytvořilo 100 částic, které by vyletěly pod úhlem 0° až 360°, docílilo by se tak efektu výbuchu.

Znázornění částice ve 2D prostoru se svými vlastnosti je k dispozici na obrázku 4.



Obrázek 4: Znázornění jedné částice a jejich vlastností [21]

3.9 Umělá inteligence

Základní implementací umělé inteligence v herních enginech je algoritmus hledání nejkratší cesty [1]. Herní objekty, které se potřebují přemístit z místa A do místa B, hledají nejkratší cestu v prostoru nazývaném navigation mesh, do češtiny přeložitelné jako navigační síť [1]. Díky této síti je algoritmus schopen najít pro každý objekt nejkratší cestu do jeho cíle, tak aby se pohybující objekt vyhnul všem překážkám. Hledaný cíl musí existovat v rámci navigační sítě, jinak bude cíl pro herní objekt nedosažitelný. Nejčastější implementací algoritmu pro hledání nejkratší cesty v herních enginech je algoritmus A* [1].

3.10 Kompilátor

Pro herní engine je velice důležité, aby umožňoval kompilaci výsledné hry na co největší množství platforem. Problém různé platformy je v jejich poskytovaných možnostech a limitech. Příkladem může být grafické API *DirectX*, které je podporované výhradně na OS od společnosti *Windows* [1]. Pro herní engine je důležité, aby tyto omezení zohledňoval a počítal s nimi.

3.10.1 Emscripten

Jedním z takovýchto kompilátorů je nástroj nazvaný emscripten. Ten umožňuje kompilaci zdrojových kódů napsaných v jazyce C nebo C++ do jazyka JavaScript, který je dnes plně podporovaný na všech moderních prohlížečích [22]. Pro fungování na webových prohlížečích, se převádí grafické API *OpenGL* na *WebGL* [22]. Díky kombinaci několika technologií je výsledný kód optimalizovaný a rychlost kódu se blíží nativní rychlosti [22].

Tuto technologii má v sobě zabudovanou například herní engine *Unity*, pro kompilaci hry do prohlížeče. V Unity se před samotným kompilováním z C++ do JS, provádí kompilace uživatelských scriptů napsaných v jazyce C# do .NET bytecode a ten je následně převeden do zdrojových souborů C++ [23].

4 HERNÍ ENGINY NA TRHU

K dnešnímu dni je na trhu spousta herních enginů, jejichž komerční použití může stát vývojáře nemalé finance nebo také vůbec nic. Aktuální finanční politika herních enginů je více nakloněná nezávislým vývojářům, než jak tomu bylo v dřívějších dobách. Herní enginy dnes ve většině případů disponují již grafickým editorem, který slouží víceméně jako takzvané „klikátko“, a hru s nimi může udělat téměř i absolutní neznalec jen se svou vizí o vysněné hře.

Následující výčet herních enginů, které jsou dnes na trhu, jsou vybrány, tak aby co nejvíce pokrývaly aktuální situaci na trhu a zastupovaly co nejvíce různých herních žánrů, se kterými se dnes hráč může setkat.

4.1 Unity

Jedná se o jeden z nejpopulárnějších enginů mezi nezávislými vývojáři. Engine začal jako malý projekt v roce 2004 [24]. Poslední verze enginu je technologicky srovnatelná s ostatními enginy, používanými pro tvorbu AAA her.

Unity dříve podporovalo několik programovacích jazyků, ve kterých bylo možné psát herní skripty. Jednalo se o jazyky Boo, C# a UnityScript, poslední zmíněný je jazyk odvozený od jazyka JavaScript [25]. Mezi tvůrci se nejvíce uchytil jazyk C#, podpora zbylých dvou jazyků byla ukončena a již se v aktuální verzi enginu neobjevuje.

Unity engine si své místo na trhu získal díky své odlišnosti v platebním modelu, který byl v té době naprosto odlišný od konkurence. Engine byl nabízen zcela zdarma, jen s menšími omezeními v podobě chybějících funkcionalit [26]. Pokud vývojář tyto funkcionality potřeboval, musel si koupit licenci v hodnotě 1500 $ [26]. Jedná se o cenu pro jednu osobu. Unity bylo díky tomu více přístupnější než konkurenční enginy, které nedisponovaly licencí zdarma.

Jelikož se *Unity* objevilo v době, kdy se od her očekával určitý stupeň kvality, převážně z grafického hlediska, tak pro vývojáře pracující samostatně, nebo v menších týmech, bylo již velice časově náročné, aby mohli investovat čas do vývoje vlastního enginu a zároveň do vývoje hry. Tito vývojáři si zvolili Unity engine, za svůj hlavní a začali na něm vytvářet své prototypy a výsledné hry. Jedním z takovýchto nezávislých herních studií bylo i české studio *MADFINGERS Games*, které vzniklo roku 2010 v Brně zaměřující se na vývoj her na mobilních platformách [27].

Aktuální verze Unity obsahuje celou řadu funkcionalit, která se dají použít na tvorbu takřka jakéhokoliv herního žánru. Engine má plnou podporu pro tvorbu 2D i 3D her. Obsahuje

dva propracované fyzikální modely, jeden slouží pro 2D objekty a druhý pro 3D objekty [24]. Disponuje i navigačním systémem na hledání nejkratších cest v 3D prostoru [24]. Dále v enginu lze nalézt podporu pro tvorbu VR aplikací, propracované osvětlení nebo možnost cílení hry pro více jak 25 různých herních platforem [24].

Veškeré funkce jsou zabudované do editoru, který je součástí enginu. Editor je navržen pro jednoduché ovládání a jednotlivé funkce jsou v něm přehledně rozvržené. Pro zpříjemnění práce s editorem, je možné si jednotlivé komponenty přeskládat, dle vlastních potřeb.

Logo herního enginu *Unity* je možné vidět na obrázku 5.

Obrázek 5: *Unity* engine logo [24]

4.1.1 Licenční politika

Platební model se od začátku existence trochu změnil, aktuálně *Unity* nabízí tři různé licence [24]. Ty jsou cenově rozdělené, tak aby co nejvíce finančně vyhovovaly vývojářům podle velikosti projektů, které se na enginu realizují.

Personal

Jedná se o licenci zdarma, je určená především novým vývojářům, kteří se potřebují s enginem teprve naučit. Používání této licence nijak neomezuje tvůrce v prodeji své vytvořené hry. Tuto licenci smí vývojář používat, pokud je roční výnos společnosti pod 100 000 $ [24]. Pokud je finanční limit překročen, je potřeba pořízení vyšší licence.

Plus

Jedná se nadstavbu licence Personal. Na rozdíl od předchozí licence je zde nabízen větší prostor na cloudovém uložišti a další benefity. Cena této licence je stanovena na 35 $ měsíčně na osobu. Na tuto licenci je nabízena aktuálně sleva. Pokud si vývojář zakoupí licenci Plus na celý rok, bude mu účtována částka 25 $ měsíčně na osobu [24]. Hlavní podmínkou pro používání této licence je roční výnos společnosti nepřesahující 200 000 $ [24]. Pokud se, tak stane je potřeba zakoupit vyšší licenci.

Pro

Aktuálně nejvyšší nabízená licence, která má stanovenou cenu na 125 $ měsíčně na osobu [24]. V rámci této licence, má vývojář v enginu zahrnuto více než 800 benefitů [24].

4.1.2 Hry vytvořené na enginu

Za tu dlouhou dobu, co engine existuje, byl využit k tvorbě velké řady her. Jak již bylo zmíněno výše, prvotní výtvory pocházely především od nezávislých vývojářů. Tyto hry zpočátku trpěly na kvalitu zpracování. Jednalo se spíše o prototypy nežli o plnohodnotné hry.

Dnes je již situace lepší a Unity si získává stále lepší reputaci. Jako důkazem může být například firma Blizzard Entertainment. Tato společnost, která se ve světě proslavila především jejich MMORPG hrou *World of Warcraft*, se při experimentování s novými herními značkami, rozhodla pro použití *Unity* a v roce 2014 představila karetní hru *Hearthstone*, která čerpá příběh ze světa hry *World of Warcraft* [1], [28]. To byl jeden z těch momentů, kdy velká firma, dokázala dalším vývojářům, že i takovýto engine, lze použít pro velké a potenciálně úspěšné projekty.

Následujících několik her je vybráno tak, aby demonstrovaly možnosti enginu.

Beat saber

Jedná se o hudební hru hrající se ve virtuální realitě. Pocházející od českých tvůrců ze studia *Hyperbolic Magnetism* [29]. Hra se ovládá dvěma ovladači, hráč drží v každé ruce jeden. Ovladače jsou ve hře zobrazovány jako meče, z nich má každý meč jinou barvu, a to konkrétně červenou a modrou [29].

Cílem hry je rozseknout kvádry letící na hráče ve správném směru a mečem odpovídající barvy [29]. Směr, ze kterého mají být kvádry rozseknuty je znázorněn šipkou umístěnou na kvádru. Všechny úrovně jsou ručně vytvořené, takže rytmus hudby přesně zapadá do intervalů ničení kvádrů.

Zároveň se jedná o jednu z nejlepších her ve svém žánru, díky minimálnímu pohybu ve virtuálním světe, kdy se hráč za celou dobu takřka nepohne ze svého místa. Navíc je zde sníženo riziko vzniku kinetózy neboli nemoci z pohybu na minimum.

Snímek ze hry lze vidět na obrázku 6.

Obrázek 6: Snímek ze hry *Beat saber* [29]

Old Man's Journey

Dobrodružná hra vyprávějící příběh o životě, ztrátě a naději. Jedná se o počin rakouského studia *Broken Rules* se sídlem ve Vídni [30]. Ve hře hráč pomáhá starému muži na jeho cestě za svou láskou. Na cestě napříč světem hráč řeší menší logické hlavolamy, ve kterých jde vždy o to, aby se starý muž dostal na konec cesty aktuální úrovně [30]. V jednotlivých úrovních se nachází několik kopců, úkolem hráče je upravit jejich velikosti, tak aby stařec mohl pokračovat na své cestě světem. Průchodem hrou se hráči odhaluje životní příběh starého muže a jeho lásky. Hra je vyobrazena nádherným ručně zpracovaným vizuálem a příjemným hudebním doprovodem.

Na obrázku 7 je k dispozici ukázka snímku ze hry.

Obrázek 7: Snímek ze hry *Old Man's Journey* [30]

Ori and the blind forest

Tato hra patří do žánru 2D akčních plošinových her. Ve svém žánru hra vyčnívá nádherným grafickým zpracováním a výborným hudebním doprovodem. Za vývojem stojí rakouské herní studio *Moon Studios* se sídlem ve Vídni [31].

Ve hře je vyprávěn příběh o lese Nibel, který umírá a hlavní hrdina jménem Ori musí cestovat napříč lesem, aby se postavil temné Nemesis, hlavní záporné postavě, a tak zachránil svůj domov [31]. Hráč se ujímá role hlavního hrdiny Oriho, malého stvoření. Hráč napříč cestou světem musí řešit různé hádanky a sbírat dovedností body, za které se odemykají a vylepšují jeho schopnosti [31]. S nově nabytými dovednostmi se mu zpřístupňují nové části světa do, kterých dříve nemohl. Na konci Oriho cesty hráče čeká konfrontace s Nemesis, jejímž poražením zachrání Ori celý les.

Ukázka ze hry je k dispozici na obrázku 8.

Obrázek 8: Snímek ze hry *Ori and the blind forest* [31]

4.2 GameMaker

Herní engine zaměřený na tvorbu dvou rozměrných her, od společnosti *YoYo Games*. Engine vznikl už v roce 1999 a během své existence již několikrát změnil svůj název [32]. Cílem bylo vytvořit engine pro všechny začínající tvůrce her, kteří nemají velké znalosti v oblasti programování. Aktuální verze enginu se zaměřuje i na pokročilejší tvůrce.

Součástí enginu je editor, který umožnuje veškeré potřebné věci jednoduše tzv. naklikat. Zároveň pro engine existuje i jeho vlastní skriptovací jazyk pojmenovaný *GameMaker Language* a slouží především tvůrcům, kteří potřebují vytvořit složitější logiku pro svou hru, která nejde nastavit v editoru [32].

Pro svůj běh engine používá programovací jazyk C++ a umožňuje export vytvořených her na širokou škálu různých platforem [32].

Na obrázku 9 je k dispozici grafické logo herního enginu *GameMaker*.

Obrázek 9: *GameMaker Studio 2* logo [32]

4.2.1 Licenční politika

GameMaker má na rozdíl od jiných enginů několik různých licencí lišící se způsobem placení. Kde některé licence jsou placeny měsíčně a jiné jsou permanentní.

Pro nové tvůrce, kteří nemají s enginem žádné zkušenosti, a chtějí se jej naučit, je tu licence s názvem Trial [32]. Ta ovšem neumožňuje vývojáři vytvořit finální sestavení hry. Dále je zde omezena velikost projektu, kde lze využívat pouze omezené množství možností, které engine nabízí. Zpoplatněné licence jsou již bez omezení velikosti projektu a dávají tvůrci přístup do všech funkcionalit, kterými disponují [32].

Nejlevnější licence jménem Creator Windows a Creator Mac, jsou určeny pro nové či malé tvůrce. Cena za jednu licenci je 39 $ na dvanáct měsíců [32]. První zmíněná umožňuje export na platformu *Windows* a ta druhá na platformu *macOS* [32].

Následující licence se jmenuje Desktop. Je za 99 $ a jedná se o permanentní licenci. Je zde umožněn export her na všechny hlavní operační systémy, kterými jsou *Windows, Mac* a *Ubuntu* [32].

Další licence nesou jména Fire a Web, každý za 149 $ s permanentní licencí [32]. První zmíněný nabízí export na zařízení od Amazonu s operačním systémem *Fire OS* [32]. Druhá zmíněná dovoluje vytvořit tzv. HTML5 hru, tedy provede se export hry do jazyka JavaScript, díky kterému je možné hrát hru ve webovém prohlížeči [32].

Licence Mobile a UWP, jsou nabízeny každá za 399 $ s permanentní licencí [32]. U první je umožněn export na všechny hlavní mobilní platformy, tedy *iOS, Android* a *Amazon Fire*. Druhá licence umožňuje export hry do aplikace UWP, zkratka znamená *Universal Windows Platform* [33]. Jedná se o aplikace použitelné na všech zařízeních s operačním systémem Windows 10, což zahrnuje i *Xbox One*. Hru je pak možné umístit na *Microsoft Store* nebo *Xbox One Store*.

Předposledními licencemi jsou licence jménem Nintendo Switch, Playstation 4, Xbox One [32]. Každá umožnuje export na platformu, kterou má ve svém názvu a každá je nabízena za 799 $ na dvanáct měsíců [32].

Poslední licence nese jméno Ultimate, ta je nabízena za 1500 $ na rok. Zahrnuje export na všechny platformy zmíněné výše [32].

YoYo Games nabízí ještě speciální cenovou nabídku licencí enginu pro školní použití [32]. Licence Desktop za 30 $, Web za 50 $, Mobile za 130 $ a UWP za 130 $ [32]. U všech zmíněných se jedná o licenci s platností na rok.

4.2.2 Hry vytvořené na enginu

Za dobu existence enginu, byl použit při realizaci v mnoha herních projektech. Vytvořené hry jsou k nalezení na oficiálních stránkách v sekci „showcase“. V několika následujících podkapitolách bude zmíněno pár těchto her.

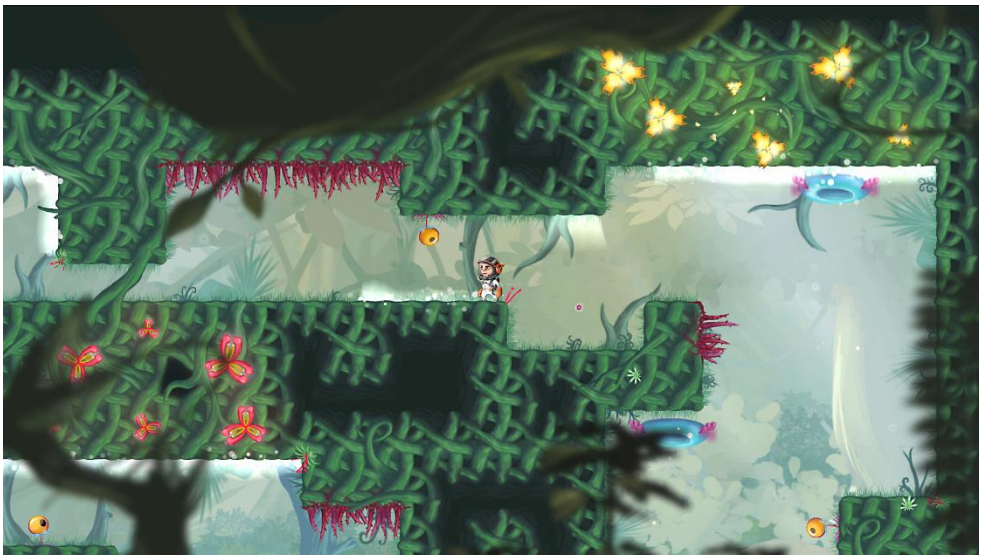
BLACKHOLE

Jedná se o českou hru vytvořenou studiem *FiolaSoft Studio* [34]. Hra vyšla v roce 2015 a dnes je dostupná na platformách *Windows, Mac, Linux, Xbox One* a *PlayStation 4* [34]. Na herní konzole se hra dostala až v roce 2017 [34]. Jedná se o dvourozměrnou hardcore logickou plošinovou hru, ve které je cílem sbírat herní předměty zvané „selfburns“ [34].

Příběh vypráví o posádce vesmírné lodi, která má za úkol uzavřít všechny černé díry. Při uzavírání poslední se vyskytnou jisté potíže a loď s posádkou ztroskotá uvnitř jedné černé díry. Hráč hraje příběh za hlavního baristu posádky, kterému se ve hře jednoduše říká „coffee guy“. Úkolem hlavního hrdiny je zachránit všechny členy posádky, kteří se při ztroskotání ztratili. Při průchodu úrovněmi hráč také sbírá herní předměty na opravu rozbité vesmírné lodi.

Hra si získala pozornost díky své unikátní herní mechanice. Svět, ve kterém se hlavní hrdina nachází se otáčí podle orientaci plošin, se kterými přijde hráč do kolize. Plošina otočí herní svět vždy tak, aby ji měl hlavní hrdina pod nohama. Tedy pokud se jedná o plošinu, která je při aktuálním otočení světa na zdi, svět se otočí o 90 stupňů. V případě, že je plošina na stropě, svět se otočí o 180 stupňů.

Na obrázku 10 je k dispozici ukázka ze hry *BLACKHOLE*.



Obrázek 10: Snímek ze hry *BLACKHOLE* [34]

Undertale

Příběhová 2D hra od amerického studia *tobyfox* [35]. Ve hře je vyprávěn příběh o malém chlapci, který spadne do podzemí, kde žijí monstra [35]. Ty tam byla vyhnána lidmi a nyní loví každého člověka, na kterého narazí. Hráč se ujme role malého chlapce, a snaží se dostat pryč z tohoto nebezpečného místa zpět na povrch. Cestou chlapec naráží na monstra, se kterými bojuje. Při soubojích se může rozhodnout, zdali monstrum zabije nebo jej nechá naživu. Bojový systém je založený na tazích, kdy vždy jeden vykonává akci a druhý čeká, než na něj přijde řada.

Samotný boj je pak založený na vyhýbání se překážkám. Hráčův život je reprezentován srdcem a během tahu jej musí ochránit před kolizí s jinými objekty. Pokud dojde ke kolizi, hráč ztrácí život.

Ukázka ze hry je k dispozici na obrázku 11.

Obrázek 11: Snímek ze hry *Undertale* [35]

Turmoil

Simulátor, od nizozemského herního studia *Gamious* [36]. Hra je inspirovaná těžbou ropy v severní Americe v 19 století [36]. Hráč se zhostí role podnikatele, který přijel na kontinent s cílem vydělat peníze. Není ovšem jediný, kdo má stejné plány, a tak má hráč proti sobě další tři rivaly podnikatele.

Samotná hra je rozdělena do čtyř částí, které jsou na mapě reprezentovány různým podnebím. Podnikatelé jsou ubytovaní v jednom malém městečku, ve kterém probíhá dražba volné půdy, kde si hráč vydraží svůj vlastní kus půdy, na kterém těží ropu a onu půdu musí, s co největším ziskem prodat. Ropa se prodává dvěma společnostem, které se nachází na mapě. Každá nabízí různou cenu za prodanou ropu. Různá podnebí mají různé překážky a množství ropy na jednotlivých pozemcích klesá. Hráč si při průběhu hrou kupuje vylepšení, která mu pomáhají překonat nové překážky nebo dopomoci k vyššímu zisku. Kromě půdy se draží i procentuální podíl nad vlastnictvím města. Hra končí v momentě, kdy už není žádná půda k prodeji a hráč s nadpolovičním vlastnění akcií na město vyhrává a stává se novým starostou.

Snímek zachycující herní úroveň je k dispozici na obrázku 12.

Obrázek 12: Úkázka ze hry *Turmoil* [36]

4.3 Unreal Engine

Jedná se herní engine od společnosti *Epic Games*, používaný především pro tvorbu 3D AAA her, tedy her s velkým rozpočtem na vývoj a reklamu [37]. V posledních letech se politika společnosti změnila, důvodem je i existence stejně dobrých herních enginů na trhu s daleko lepší licenční politikou. Aby byl engine konkurence schopný, byla jeho licenční politika pozměněna také. Nyní je engine více nakloněn nezávislým tvůrcům a menším herním studiím.

Engine se od svého vzniku zaměřoval především na tvorbu 3D akčních her s nejlepší grafikou jakou bylo možné vykreslit na nejmodernějších grafických kartách. To se v dnešní době nijak nezměnilo a engine dnes nabízí i možnosti vývoje her pro virtuální realitu a možnost kompilace hry na libovolnou platformu. Celý engine je napsán v jazyce C++.

Dvourozměrné hry jako je například již zmíněný *Ori and the blind forest* vytvořený v *Unity* enginu, jsou pro *Unreal Engine* neobvyklé a příliš mnoho takovýchto her zatím neexistuje. Engine ovšem má pro tyto hry plnou podporu a neexistuje zde důvod, který by bránil tvorbě tohoto žánru her. Rozšíření pro tvorbu 2D her se v enginu jmenuje *Paper 2D* a je součástí grafického editoru [37].

Grafické logo herního enginu *Unreal Engine* je k dispozici na obrázku 13.

Obrázek 13: *Unreal Engine* logo [37]

4.3.1 Licenční politika

Kompletní Engine je aktuálně uvolněn zcela zdarma [37]. Vývojář má přistup ke všem nástrojům, funkcionalitám, pravidelným aktualizacím, zdrojovým kódům a kompletním ukázkovým projektům. Zpoplatnění přichází až v momentě, kdy vytvořenou hru chce vývojář začít prodávat. Z prodeje následně odvádí 5 % ze zisku společnosti *Epic Games*, ale pouze v případě, že zisk za čtvrtletí činil, více jak 3 000 $ hrubého zisku [37]. V případě velkých společností mohou existovat i jiné procentuální dohody se společností *Epic Games*.

Epic Games nabízí i vlastní licenční podmínky, pro společnosti, které preferují platbu licenčního poplatku předem, za účelem snížení sazby licenčního poplatku [37].

V době psaní této práce vytvořila společnost *Epic Games* svůj vlastní obchod na prodej elektronických licencí svých her [38]. Tento obchod nazvala *Epic Games Launcher* a prodej her na něm umožnila i dalším tvůrcům. Jelikož se jedná o vlastníky herního enginu, cena za použití licence při prodeji her na tomto obchodu může být jiná.

4.3.2 Hry vytvořené na enginu

Engine je využíván především u herních projektů, které cílí na, co možná nejlepší grafický zážitek, blížící se realitě. V následujících podkapitolách jsou zmíněné některé zajímavé hry, využívající tento engine.

XCOM 2

Jedná se o hru od vývojářů ze studia *Firaxis Games* se sídlem v USA [39]. Žánrově se jedná o tahovou strategii. Hráč se zhostí role velitele, který dává rozkazy vojákům na bojišti [39]. Úrovně jsou náhodně generované a jednotlivé akce vojáků, mají vždy procentuální šanci na úspěch. Příběh hry navazuje na konec předchozího dílu, kdy hráč, jako velitel, nedokázal ubránit zemi proti mimozemské invazi a ti se infiltrovali mezi lidskou populaci, kterou následně ovládli společně se Zemí. Velitel, za kterého hráč hraje je součástí odboje a vede válku za osvobození Země. Základní princip hry vychází z šachů, kde každá jednotka má jiné možnosti pohybu a je potřeba taktizovat i několik kol dopředu.

Snímek ze hry je k dispozici na obrázku 14.

Obrázek 14: Snímek ze hry *XCOM 2* [39]

A Way Out

Vývojářem je Švédské herní studio *Hazelight* [40]. Hra vypráví o příběhu dvou postav, Lea a Vincenta, kteří se poprvé potkají ve věznici, ze které následně společně utíkají, aby se mohli pomstím muži, jenž jim zničil život [40]. Žánrově se jedná o kooperační akční adventuru. Herní mechanikou, která je hlavním zdrojem herního zážitku, je nemožnost hrát hru s jedním hráčem. Při hraní jsou vždy potřeba dva hráči, pro ovládání obou hlavních protagonistů. Kvůli nutnosti přítomnosti obou hráčů, autoři učinili rozhodnutí, že pro hraní stačí, aby pouze jeden z hráčů vlastnil zakoupenou kopii hry [41]. Hra je aktuálně dostupná na PC a konzolích PS4 a Xbox One.

Hra rozděluje obraz na dva pohledy obou hlavních protagonistů, díky čemuž může první hráč vidět, co právě dělá jeho spoluhráč. Při hraní se rozložení obrazovek různě mění, v určitých částech hry je zdůrazněn příběh jedné či druhé postavy.

Snímek zachycující herní design rozdělených obrazovek pro dva souběžně hrající hráče je k dispozici na obrázku 15.

Obrázek 15: Snímek ze hry *A way Out* [40]

The Siege and The Sandfox

Hra od vývojářů *Cardboard Sword* ze spojeného království [42]. Hráč se nachází v zářivém městě v srdci rozsáhlé pouště. Jedné noci je král zavražděn a hlavní protagonista je falešně obviněn z vraždy a následně uvězněn v žaláři, ze kterého musí uniknout, aby mohl povědět lidem pravdu o králově vraždě a zachránit, tak město před zloduchy, kteří se dostali na trůn [42]. Jedná se o 2D hru s uměleckým grafickým zpracováním. Hratelností se jedná o platformovou skákačku, s důrazem na tichý postup. Hráč je veden k tomu, aby se držel více ve stínech a nebyl tak zpozorován nepřáteli a zároveň, aby se vyhýbal co nejvíce přímému střetu s nimi. Hlavní herní mechanismus je založen na metroidvanii, pro který je charakteristický velký otevřený herní svět, který hráč může nelineárně prozkoumávat a do určitých míst se dostane v momentě, až když ovládá určitou schopnost. Pořadí, v jakém se hráč bude učit novým schopnostem záleží pouze na něm samotném. Při psaní této práce je hra stále ve fázi vývoje.

Snímek zachycují stylizaci grafiky hry je k dispozici na obrázku 16.

Obrázek 16: Ukázka úrovně ze hry *The Siege and The Sandfox* [42]

4.4 Cocos2d

Jedná se open source framework pro tvorbu her a aplikací s 2D grafikou. *Cocos2d* je označení pro rodinu frameworků [43]. Jednotlivé odnože používají jiné programovací jazyky a zaměřují se, tak na jiné platformy. Cocos2d se zaměřuje na hlavní desktopové operační systémy, tedy *Windows*, *OS X* a *Linux*.

Konkrétní odnož *Cocos2d-x* je napsaná v programovacím jazyce C++ [43]. Mezi vývojáři je široce využívána pro tvorbu her a aplikací na mobilních zařízeních. Vykreslování grafiky je optimalizováno pro 2D hry a pro její vykreslování je využito grafické API OpenGL [43].

Pro framework *Cocos2d-x* existuje plnohodnotný editor jménem *Cocos Creator* [43]. Ten v sobě zahrnuje všechny možnosti, které framework nabízí včetně enginu samotného. Editor je srovnatelný s jinými editory u jiných herních enginů. Existence editoru vývoj her značně urychluje, navíc je editor vybaven otevřeným systémem rozšiřování, díky kterému existují mnohá rozšíření.

Ve frameworku se nachází technologie *SDKBOX*, která umožňuje tvůrcům jednoduše, rychle a zadarmo integrovat SDK třetích stran do svých her [43]. Takovýmto rozšířením je například *Google Analytics* nebo *Google Play Services*.

Logo herního enginu *Cocos2d* je k dispozici na obrázku 17.

Obrázek 17: Logo *Cocos2dx* [43]

4.4.1 Licenční politika

Engine *Cocos2d* i editor *Cocos Creator* mají licenci MIT. Jedná se o svobodnou licenci, takže za použití těchto technologií a následný prodej vytvořených her se nic neplatí [44].

4.4.2 Hry vytvořené na enginu

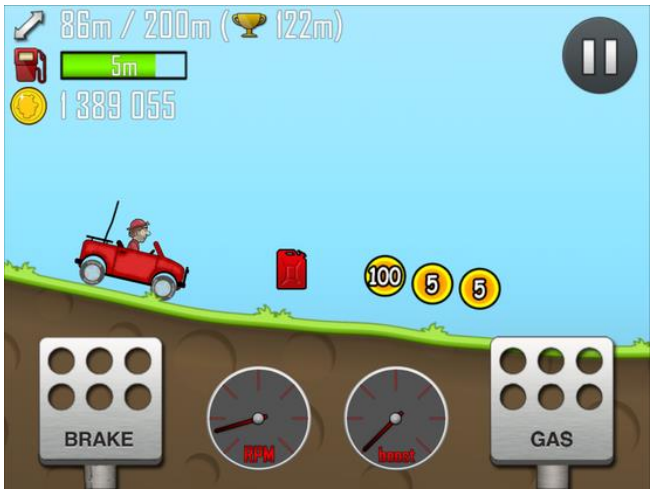
Následující výčet her pochází z mobilních platforem, z důvodu toho, že se zde engine nejvíce uchytil.

Hill Climb Racing

Jednoduchá hra z Finského studia *Fingersoft* [46]. Cílem hry je v každé úrovni projet s vozidlem ze startu do cíle [46]. Jednotlivé levely obsahují členitý terén, které musí vozidlo překonat. Pakliže dojde k nehodě a vozidlo se otočí na střechu, je vozidlo nepojízdné a hráč musí úroveň opakovat. Kromě překonání terénu si musí hráč hlídat stav paliva, pokud palivo dojde, musí hráč úroveň taktéž opakovat. V úrovni se nachází červené kanystry, které doplňují vozidlu palivo. Kromě kanystrů se v úrovni nachází, taktéž mince a diamanty. Ty slouží pro vylepšování aktuálního vozidla a nákup nových.

Hra je dostupná na všech hlavním mobilních operačních systémech. Hra má velice dobré hodnocení na všech platformách a počet stažení čítá několik milionů [45].

Snímek zachycující herní úroveň s uživatelským ovládáním je možné vidět na obrázku 18.



Obrázek 18: Ukázka úrovně ze hry *Hill Climb Racing* [45]

Geometry Dash

Akční platformová skákací hra od švédského herního vývojáře Roberta Topala a zakladatele studia *RobTop Games* [47]. Hráč ve hře ovládá malou kostku, která se žene nezastavitelně dopředu. S kostkou hráč může pouze skákat. V určitých částech úrovní kostka letí na raketce a působením gravitace padá k zemi, hráč pak místo skákání přidává plyn u raketky, a ta díky tomu letí k oblakům. Cílem hry je vyhnout se všem překážkám, které by kostku zničily a dostat se, tak na konec úrovně. Vše je doprovázeno rytmickou hudbou. Hra je dostupná na mobilních operačních systémech Android a Ios [45].

Stylizaci grafického designu hry a zároveň jednu z úrovní je možné vidět na obrázku 19.

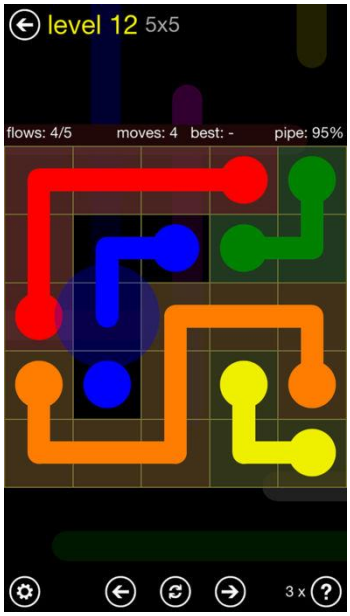
Obrázek 19: Ukázka ze úrovně ze hry *Geometry Dash* [45]

Flow

Logická hra vytvořená studiem *Big Duck Games* se sídlem na Floridě [48]. Hráč má za úkol propojit čárou dvě stejná barevná kolečka. Čáry se mezi sebou nesmí křížit a pro úspěšné zvládnutí úrovně musí být všechna stejně barevná kolečka propojená. Levely jsou

řazeny do několika úrovní podle složitosti, tak aby hra vyhovovala, co největšímu množství hráčů. Hra je dostupná na operačních systémech *Android* a *iOS*. Na obou platformách má hra velice dobrá hodnocení [45].

Jednu z úrovní hry je k dispozici k nahlédnutí na obrázku 20.



Obrázek 20: Ukázka úrovně ze hry *Flow* [45]

4.5 Shrnutí

Herní enginy zmíněné v předchozích kapitolách jsou jedny z nejpoužívanějších na dnešním herním trhu, především pak mezi tvůrci 2D her. Žádný ze zmíněných herních enginů není zcela dokonalý a má své klady i zápory, které jej odlišují od ostatních. Rozdíly jsou, jak v licenčních podmínkách, tak i v poskytovaných možnostech, jež ovlivňují možné herní žánry, které v nich lze realizovat.

Tabulka 1 stručně shrnuje licenční politiku jednotlivých enginů.

Tabulka 1: Licenční politika enginů

Herní engine	**Licenční politika**
Unity	Zdarma do ročního zisku 100 000 $.
	Od ročního zisku 100 000 $ se platí měsíční licence za 35 $.
	Od ročního zisku 200 000 $ se platí měsíční licence za 125 $.
GameMaker	Různé licence podle druhu cílené platformy a druhu trvání licence. Ceny licencí se pohybují mezi 30 $ až 799 $.
Unreal Engine	Zdarma v případě menšího zisku než 3000 $ za čtvrtletí.
	V případě většího zisku než 3000 $ se odvádí 5 % ze zisku.
Cocos2d	Zcela zdarma.

Tabulka 2 zobrazuje oficiální možnosti, které zmíněné enginy, v této práci, poskytují vývojářům her.

Tabulka 2: Porovnání jednotlivých enginů

	Unity	***GameMaker***	***Unreal Engine***	***Cocos2d***
Grafický editor	ANO	ANO	ANO	ANO
Plná podpora pro 2D hry	ANO	ANO	ANO	ANO
Plná podpora pro 3D hry	ANO	NE	ANO	ANO
Hry pro PC	ANO	ANO	ANO	ANO
Hry pro Android	ANO	ANO	ANO	ANO
Hry pro iOS	ANO	ANO	ANO	ANO
Hry pro konzole	ANO	ANO	ANO	NE
Hry pro prohlížeč	ANO	ANO	ANO	ANO
Hry pro VR	ANO	NE	ANO	NE
Zcela zdarma	NE	NE	NE	ANO
Možnost vytvoření hry bez zakoupené licence	ANO	NE	ANO	ANO

Pro vývoj 2D her jsou si enginy velice konkurence schopné. Nejlépe jsou na tom herní enginy *Unity* a *Unreal Engine*. Finanční tlak firem vlastnící tyto enginy, zajišťuje pravidelný přísun nových funkcionalit nebo i opravu vyskytujících se chyb. Menší nevýhodou je zde existence potřeby placení za použití enginu, jenž je vyžadováno zaplatit po určitém finančním zisku plynoucího z jeho používání.

Obdobně je na tom *GameMaker*, který má, avšak tu nevýhodu, že pro vytvoření finální hry musí být zakoupená licence, a navíc je potřeba zakoupit takový typ nebo typy licencí pro cílenou platformu.

U enginu *Cocos2d* je velkou výhodou jeho open source licence, která tvůrcům, především těm menším, šetří potřebné finance na vývoj. Velkou nevýhodou je zde, ale neexistence finančního tlaku ze strany vlastníka na vylepšování a opravu chyb, která u tohoto enginu neexistuje v takové míře jako u komerčních enginů.

5 NÁVRH VLASTNÍHO 2D HERNÍHO ENGINU

Následující kapitola se zabývá představením implementovaného herního enginu, specifikováním jeho požadavků a návrhem, na kterém bude implementován.

5.1 Představení aplikace

Implementovaný herní engine poskytuje možný nástroj herním vývojářům, se kterým je možné implementovat vlastní 2D hru. Poskytuje řešení všech základních požadavků na jednoduchou hru, kterými jsou vykreslování grafiky, zvukový výstup, fyzikální model a zpracování vstupních signálů.

5.2 Požadavky

Před samotným vývojem je potřeba provést analýzu, která zodpoví otázky, co se za produkt konktrétně vyvíjí, a co je třeba implementovat pro splnění všech požadavků na produkt. Pro zodpovězení těchto otázek je potřeba určení funkčních a nefunkčních požadavků. Tyto požadavky pro vývoj 2D herního enginu jsou představeny v následujících podkapitolách.

5.2.1 Funkční požadavky

V následující tabulce je uveden seznam funkčních požadavků, které definují výsledné chování a funkcionalitu vyvíjené aplikace. Seznam funkčních požadavků je zobrazen v tabulce 3.

Tabulka 3: Funkční požadavky aplikace

Identifikační číslo	Popis
F01	Engine poskytne možnost vykreslování statických obrázků.
F02	Engine poskytne možnost pro zpracování vstupních signálů z klávesnice a myši.
F03	Engine poskytne možnost pro zvukový výstup.
F04	Engine poskytne funkcionalitu pro jednoduchou simulaci gravitace.
F05	Engine poskytne možnost vykreslování sprite animací.
F06	Engine poskytne možnost detekce kolizí mezi kolizními komponentami herních objektů.

F07	Engine poskytne funkcionalitu herním objektům pro odchycení událostí vzniklých při kolizi s jinými objekty.
F08	Engine poskytne funkcionalitu herním objektům pro odchycení událostí vzniklých při stisknutí kurzoru myši.

Jak již bylo v úvodu práce napsáno, počítačová hra je zábavný produkt a cílem hry je vytvářet činnost, při které se bude hráč bavit. Herní engine, jako nástroj, musí poskytnout minimální funkcionality k vytvoření hry.

Aby hráč měl vizuální přehled o dění ve hře, je potřeba, aby herní engine umožňoval minimálně vykreslovat statické obrázky, které by reprezentovaly např. různé herní objekty terénu nebo prvky uživatelského rozhraní. Herní engine by měl umět pracovat s obrázky v obrazovém formátu PNG a měl by umět vykreslit obrázek i s transparentním pozadím. Tuto funkcionalitu by měl vývojářům poskytnou skrze komponentu (F01).

Dále je potřeba aby herní engine uměl reagovat na vstupní signály, které jsou vytvářené vstupními periferiemi hráče. Je tedy potřeba, aby herní engine disponoval mechanismem detekce vstupních signálů a dokázal je zpracovat a předat dále implementované hře. Herní engine by měl poskytnou mechanismus, kterým by hře oznámil, která klávesa na klávesnici nebo myši je právě stisknutá, nebo naopak které klávesa byla aktuálně uvolněná (F02).

Kromě vizuálního výstupu je dobré, aby hráč měl přehled o dění ve hře i pomocí zvukového doprovodu. Proto je potřeba, aby herní engine disponoval základním mechanismem pro zvukový výstup. Důležité je, aby bylo možné přehrávat více zvukových efektů, tedy aby existovalo více zvukových kanálů, které lze zároveň alokovat. Taktéž je pro hru důležité, aby herní engine umožňoval spuštění hudby na pozadí. Hudba na pozadí slouží jako zvukový doprovod a více, tak zvýrazňovala herní zážitek. U veškerého přehrávaného zvuku je také potřeba rozhraní pro ovládání, aby se daly jednotlivé zvuky spustit, pozastavit nebo zastavit. Situací, kdy ve hře je potřeba zvukový výstup je například, když je hlavní hrdina ve hře zasažen a hráč je, kromě jiného vizuálu, informován také zvukovým efektem. Děje se tomu z důvodu, aby hráč byl vždy co nejvíce informovaný o dění ve hře. Soubory se zvukovým doprovodem nebo zvukovými efekty by měly být uložené ve standardním formátu, kterým je například MP3 nebo WAV. Tuto problematiku a její ovládání by měl herní engine řešit skrze speciálně vytvořenou komponentu (F03).

Pro potřeby her založených na skákání po platformách je potřeba, aby herní engine disponoval základní simulací gravitace. Jedná se o jeden ze základních požadavků na skákací

hry, takže je potřeba, aby touto vlastností disponoval samotný herní engine. Simulaci gravitace by měl herní engine poskytnout vývojářům skrze komponentu (F04).

Kromě vykreslování statických obrázků je ve hře dobré, aby důležité herní objekty, které se ve hře pohybují nebo mají nějakým způsobem upoutat pozornost hráče, neměly pouze statický vizuál. Proto je potřeba, aby herní engine disponoval základní animační mechanikou zvanou *sprite*. Herní engine by měl tedy poskytnout možnost pracovat s tímto druhem animace a umět pracovat s animačními snímky. Pro každou animaci by měl poskytnout mechanismus, kterým by se dalo nastavit pořadí jednotlivých snímků. I jakou rychlostí se budou jednotlivé snímky střídat. Samotná animace a její ovládání by mělo být realizováno jako komponenta (F05).

Pro řízení děje ve hře je důležité, aby herní engine disponoval mechanismem detekce kolizí mezi jednotlivými herními objekty. Pro základní detekci kolizí postačí, když mají herní objekty kolizní prostor ohraničený obdélníkem. Následně mechanismus detekce kolizí se bude ptát jednoho obdélníku, zdali v jeho prostoru neleží jeden z bodů druhého obdélníku. Detekování kolize by mělo být realizováno pomocí speciální komponenty, která by reprezentovala kolizní prostor herního objektu (F06).

Kromě detekce samotné kolize je potřeba, aby herní engine disponoval mechanismem, který dává vědět herním objektům, které se dostaly do kolize s jinými herními objekty a ty mohly na tuto událost dále reagovat. Herní engine by měl tedy disponovat mechanismem pro řízení výskytů těchto událostí. Mělo by se jednat o mechanismus, který by herním objektům řekl, ať informují své komponenty a ty podle situace zareagují (F07).

Poslední důležitou funkcionalitou, která je pro herní engine důležitá, je detekce vstupních signálů přicházejících z myši a identifikace místa kliknutí v herním světě. Po kliknutí následuje zjištění, které herní objekty se v daném místě nachází, pakliže jsou zjištěny, konkrétní herní objekty jsou o vzniklé situaci informovány. Tato funkcionalita je důležitá především pro prvky uživatelského rozhraní, kdy je pro uživatele daleko jednoduší a přirozenější tyto akce provádět pomocí počítačové myši. Herní engine by měl poskytovat mechanismus pro řízení této události, mělo by se jednat o podobný způsob jako u výskytu kolizí, tedy že herní objekt informuje všechny své komponenty (F08).

5.2.2 Nefunkční požadavky

V následující tabulce číslo 4 je uveden seznam nefunkčních požadavků, které definují omezení či vlastnosti vyvíjené aplikace.

Tabulka 4: Nefunkční požadavky aplikace

Identifikační číslo	Popis
N01	Použité technologie nesmí být závislé na jednom typu operačního systému.
N02	Implementace se řídí návrhem *Entity component system.*
N03	Výsledný herní engine je statickou knihovnou.
N04	Herní engine umějící pracovat s obrazovými soubory ve formátu PNG.
N05	Herní engine umějící pracovat se zvukovými soubory ve formátu WAV.

Pro herní engine je velice důležité, aby nebyl závislý na jednom konkrétním hardwaru nebo operačním systému. Proto je důležité, aby byly použity takové technologie, které dovolují přenositelnost výsledné hry na co největší možný počet zařízení. Jedná se typicky o grafické API, které není na všech operačních systémech zcela stejné. Pokud jsou zvolené správné technologie je možné, aby herní engine poskytoval sestavení finální hry na co nejvíce možných druhů zařízení, a tím dostal hru k co nejvíce hráčům. Dalším možným problémem při zvolení špatné technologie je nekompatibilita s novým hardwarem, nemusí se jednat o ty nejdůležitější prvky, kterým je např. grafický karta, ale může se jednat i nové herní periferie. Zástupcem nejmladší herní periferie je například *Steam Controller*. Je tedy od herního enginu vyžadováno, aby nebyl omezený použitou technologií a umožňoval rozšiřování svých vlastností podle potřeby a jednotlivé použité technologie nebyly na sobě přímo závislé (N01).

Kromě použitých technologií je potřeba, aby byl herní engine postaven na dobrém návrhu, který bude umožňovat rozšiřování herního enginu i do budoucna. Jedním z takových to návrhů je právě návrh *Entity component system*. Jedná se o velice lehko pochopitelný návrh. Jeho základní princip tkví v přidávání funkcionality skrze komponenty. Díky tomu neomezuje herní engine v jeho budoucím rozšiřování o novou funkcionalitu, což je naprosto nezbytné, pokud je herní engine vydáván v pravidelných intervalech s aktualizacemi (N02).

Pro herní engine je potřeba, aby dokázal pracovat se standardními formáty pro obrazové a zvukové soubory. Těmito formáty jsou například PNG a WAV. Důvodem pro použití standardních formátů je ten, že vytvářený obrazový vizuál nebo zvukové formáty jsou vytvářené v různých programech, které taktéž umí pracovat se standardními formáty. Na internetu existuje celá řada databází s herní grafikou nebo zvukovými efekty. Takže pro herní engine je velice důležité, aby dokázal pracovat se standardními formáty stejně jako ostatní programy, protože tyto formáty představují jednoduchý způsob, jak předávat soubory mezi různými programy nebo systémy (N04), (N05).

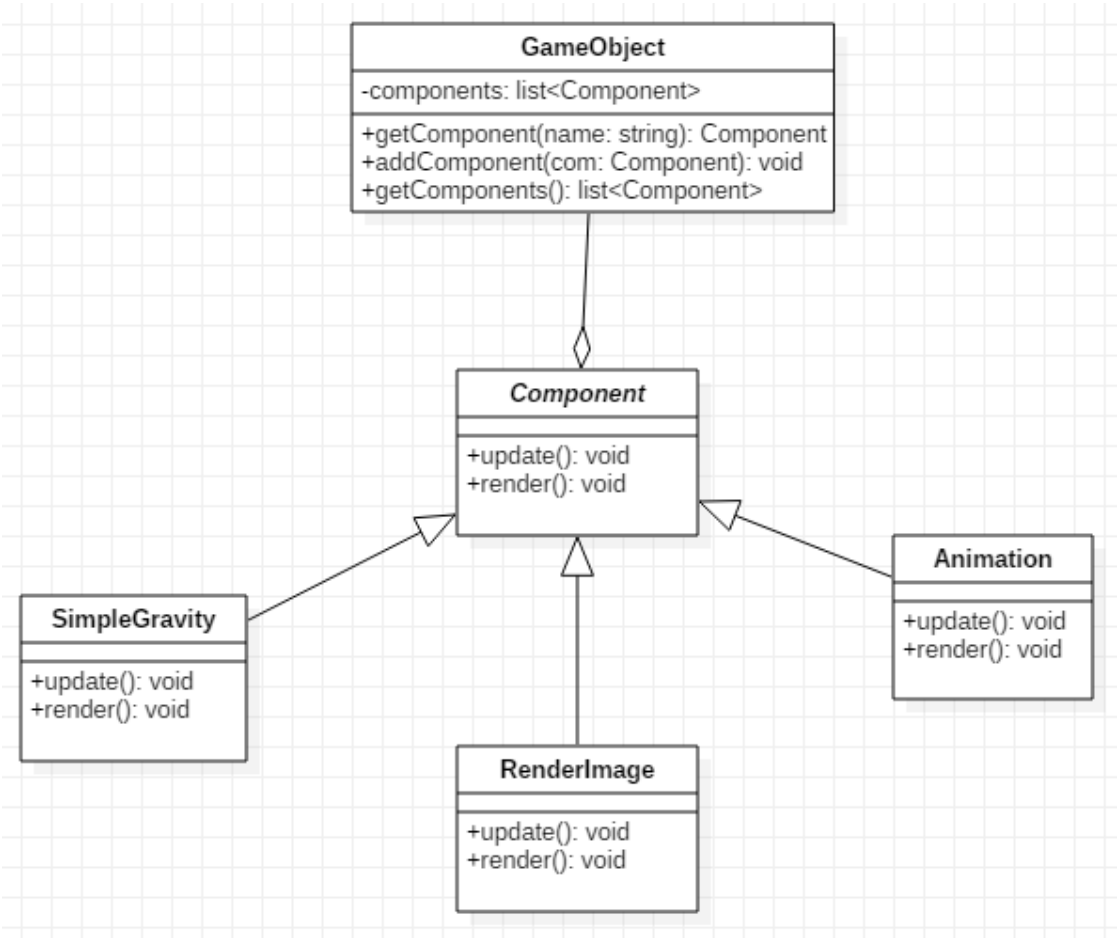
5.3 Návrh

Po stanovení požadavků na vývoj následuje návrh řešení. Návrh řešení by měl být takový, aby splnil všechny definované požadavky.

5.3.1 Struktura projektu

Engine je inspirovaný herním enginem *Unity*, který je postaven na návrhu zvaném *Entity component system* [49]. Tento návrh je založený na entitách, které jsou herními objekty, a ty disponují jednou nebo více komponentami, které definují chování a vlastnosti herního objektu [49]. V tomto návrhu je velice jednoduché engine do budoucna rozšiřovat o nové funkcionality skrze komponenty.

Na obrázku 21 je vyobrazen UML digram představující základní implementace návrhu *Entity component system*. Třída *Component* je abstraktní třídou, od které všechny ostatní komponenty dědí její vlastnosti a funkce. Díky polymorfismu má herní objekt ve své kolekci uložené jednotlivé instance existujících komponent.

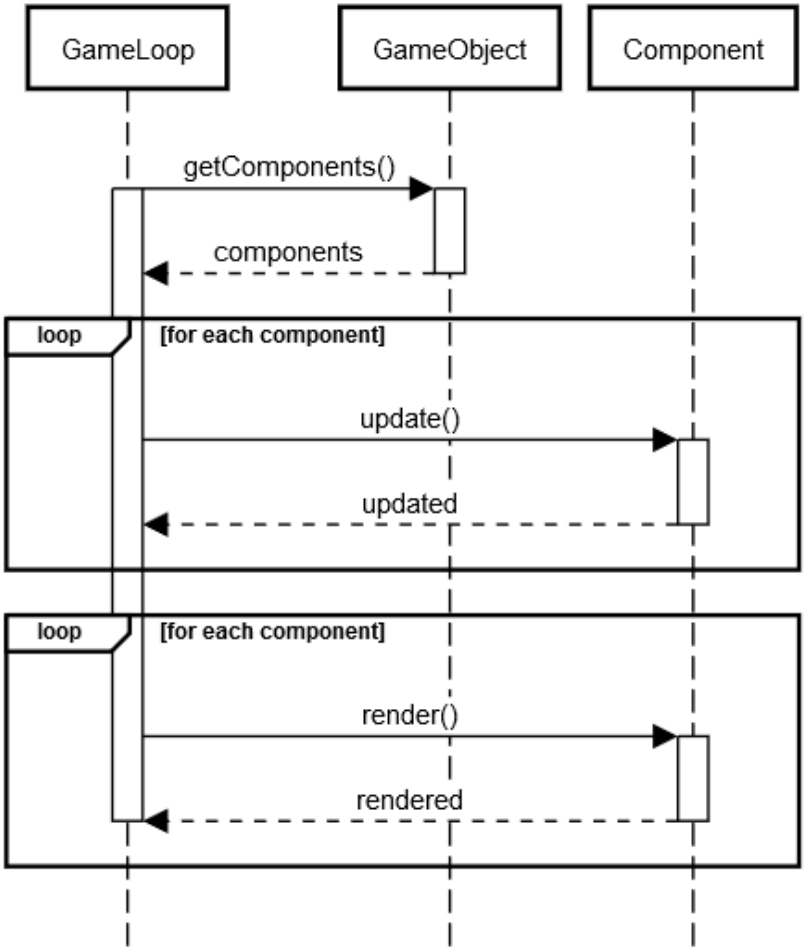


Obrázek 21: Přiklad návrhu *entity component systém*

Na obrázku 22 je k dispozici ukázka, jak tento návrh funguje při iteraci herní smyčky. Herní smyčka, která má přístup ke kolekci herních objektů, prochází každý herní objekt a jeho komponenty. Nejdříve požádá o předání kolekce komponent herního objektu. Následně každou tuto komponentu projde a zavolá nad ní funkci *update*, která má na starost provádění činnosti konkrétní komponenty. Následně projde komponenty ještě jednou, ale tentokrát u každé komponenty zavolá funkci *render*, která vykreslí grafické znázornění herního objektu na plátno hry.

Důvodem proč se komponenty prochází dvakrát je ten, že různé komponenty můžou ovlivnit stav objektu, například pozici, a pokud by došlo k vykreslení dříve než by se aktualizovat stav herního objektu všemi komponentami, mohlo by dojít k vykreslení grafiky na jiném místě, než se ve skutečnosti herní objekt nachází.

Tato činnost se děje pro každý herní objekt v kolekci. V momentě, kdy se dojde nakonec kolekce, herní smyčka přechází do další iterace a celý cyklus se opakuje.



Obrázek 22: Přiklad principu herní smyčky při jedné iteraci jednoho herního objektu v návrhu *entity component system*

6 IMPLEMENTACE

V rámci praktické části byl implementován jednoduchý herní engine umožňující tvorbu 2D her. Kromě samotného implementovaného enginu byla vytvořena jedna ukázková hra demonstrující možnosti herního enginu.

6.1 Použité technologie

Pro implementaci finálního herního enginu bylo použito několika technologií, které budou popsány v následujících podkapitolách.

6.1.1 Programovací jazyk

Pro realizaci byl zvolen programovací jazyk C++. Jedná se moderní jazyk s podporou pro objektově orientované programování. Jazyk poskytuje programátorům větší kontrolu nad pamětí a její alokování či uvolnění manuálně. V případě, že je manuální správa nad pamětí v určitých situacích nežádoucí, C++ disponuje i dynamickou správou nad pamětí pomocí třídy zvané *smart pointer*, jejíž alokaci či uvolnění nemá na starosti programátor. Pro tento jazyk existuje, také celá řada knihoven a API, se kterými se dá lehce pracovat.

6.1.2 Vývojové prostředí

Jako vývojové prostředí, bylo zvoleno *Visual Studio 2017 Community* edice. Jedná se o produkt od společnosti *Microsoft* a vývoj aplikací v jazyce C++ je v něm plně podporován [1]. Kromě podpory jazyka C++, byl dalším důvodem pro použití tohoto vývojového prostředí jeho edice *Community*. Ta je poskytována zcela zdarma, a kromě několika chybějících vlastností je srovnatelná s placenou verzí.

6.1.3 SDL

Pro práci s HW byla použita knihovna *Simple DirectMedia Layer* označována zkratkou *SDL* [50]. Je šířena pod volnou licencí *zlib* [50]. Knihovna je podporována na všech hlavních operačních systémech a umožňuje vývojářům nízkoúrovňový přístup k vstupním a výstupním periferiím. Pro vykreslování grafického výstupu se používá grafické API *OpenGL* nebo *DirectX*, podle platformy, na které je knihovna používána. Knihovna byla původně napsána

v jazyce C a nativně pracuje v jazyce C++. Knihovna je taktéž dostupná pro několik dalších programovacích jazyků.

Společně s knihovnou *SDL* byly použity dvě doplňkové knihovny *SDL_mixer* a *SDL_image*. První zmíněná se stará o zvukový výstup. Druhá zmíněná zase podporuje různé formáty obrázků a umožňuje jejich grafický výstup.

6.2 Implementování enginu

Samotný herní engine je postavený na návrhu *Entity component system* dále jen ECS, který tvoří jádro celého návrhu implementace. Herní engine je ve výsledné kompilaci statickou knihovnou, kterou stačí importovat do projektu se hrou. Implementovaný engine se skládá ze tří hlavních částí.

6.2.1 Komponenta

Základní třídou, která tvoří engine je samotná komponenta. Ta má ve svém kódu implementaci určité problematiky. Základní třída komponenty je abstraktní třída, která slouží jako předek pro všechny existující komponenty, se kterými se v herním enginu pracuje. Samotná implementovaná komponenta má přístup k hernímu objektu, který ji využívá a skrze něj, může přistupovat i k ostatním jeho komponentám.

Komponenty existující v rámci enginu jsou následující:

Abstraktní třída *Component*

Jedná se o abstraktní třídu, ze které dědí všechny komponenty v rámci herního enginu nebo na míru vytvořené pro hru. V rámci implementovaného enginu, má tato třída vlastnost uchovávající informaci, kterému hernímu objektu patří. Tato informace je velice důležitá, a to z důvodu potřeby měnit stav jiných komponent v rámci jednoho herního objektu. Samotný herní objekt, pak může předat komponentě požadovanou svou komponentu, a ta může následně ovlivnit její stav.

Dále třída obsahuje funkce *update, render, resetRender, onTrigger, onCollision, onMouseClick, onLoad a clone*. Všechny tyto funkce si potomci dané třídy mohou libovolně podědit, výjimkou je pouze funkce *clone,* jejíž implementace v potomku je povinná. Tato funkce se stará o vytvoření hluboké kopie komponenty.

Funkce *update* je jednou z funkcí, která se volá při každé iteraci herní smyčky a je v ní obsažen kód, který aktivně nějakým způsobem ovlivňuje její stav. Takovouto situací může být například změna textury herního objektu při animaci.

Druhou funkcí, která se volá při každé iteraci herní smyčky, je funkce *render*. Tato funkce se stará o vykreslování grafického vizuálu na plátno herní scény. K vykreslování dochází několikrát za sekundu. V implementovaném enginu dochází k překreslení herní scény maximálně 90krát za sekundu, pokud je okno aplikace hry v popředí a aktivní.

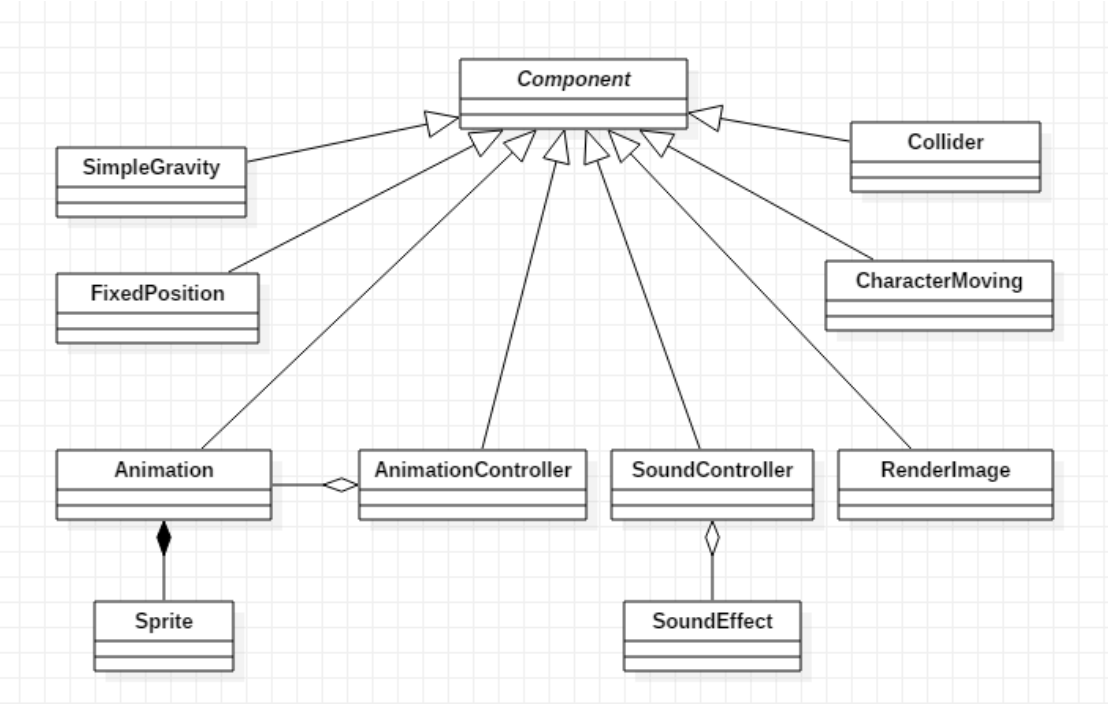
Funkce *resetRender* se stará o uvolnění alokovaných grafických prvků. Tato funkce se volá například v situaci, kdy dojde ke změně okna aplikace hry, a tím pádem dochází k novému načtení, a tím se tvoří nové plátno, na které se vykresluje. K opětovné alokaci grafických prvků dojde opět při zavolání funkce *render*.

Funkce *onTrigger* a *onCollision* jsou funkce volaná při kolizi a k jejich zavolání je potřeba, aby herní objekt měl kolizní komponentu. Při události kolize, mezi herními objekty, je tato funkce zavolána nad všemi vlastněnými komponentami. V parametru funkce je předán druhý herní objekt, se kterým došlo ke kolizi.

Předposlední funkcí je *onMouseClick*. Tato funkce je zavolána nad všemi komponentami herního objektu. Funkce je zavolána při události, kdy uživatel klikne do prostoru v herním světě tlačítkem myši a v místě, kde uskutečnil stisknutí tlačítka na myši se nachází herní objekt.

Poslední funkcí je *onLoad*. Jedná se o funkci, která je zavolána při načtení nové úrovně.

UML diagram na obrázku 23 zobrazuje všechny komponenty, které v rámci implementovaného herního enginu dědí tuto abstraktní třídu.



Obrázek 23: UML diagram tříd komponent v implementovaném enginu

Komponenta *Animation*

Tato komponenta, jak je z jejího názvu patrné, se stará o animaci objektu. Jedná se o implementaci tzv. animace *sprite*, která byla již představena v předešlé kapitole.

Pro použití teto komponenty je potřeba existence obrazového souboru obsahující animační snímky. Je velice důležité, aby každý jednotlivý snímek měl stejnou velikost, jelikož implementovaný engine neumí pracovat s různě velkými animačními snímky v rámci jednoho souboru. Tento soubor společně s informací o velikosti jednoho animačního snímku je předán třídě *Sprite*, která tyto informace v enginu uchovává.

Samotná třída *Sprite* slouží jako jeden z parametrů animační komponenty, tím druhým je rychlost animace. Rychlostí animace se rozumí, jak rychle se budou střídat jednotlivé animační snímky. Rychlost se udává v desetinném číslu, takže se snímky mohou vystřídat i několikrát za sekundu.

Pořadí snímků, ve kterém se budou opakovat, se nastavuje pomocí funkce *addFrame*. Funkci se předávají dva parametry, index řádku a sloupce. Jedná se o souřadnice, kde se daný animační snímek nachází na obrázkovém souboru. Index je počítán od nuly. Tyto informace se ukládají do kolekce.

Samotné střídání jednotlivých snímků, podle uplynutého času, se odehrává ve funkci *update*, která je enginu volána nad každou komponentou herního objektu v aktivní scéně.

Komponenta *AnimationController*

Jedná se o komponentu, která ve své vnitřní kolekci uchovává komponenty *Animation*. Každá animace je zde uložena i se svým jménem. Úkolem této komponenty je umožnit střídání různých animací v rámci jednoho herního objektu. Animace je potřebovat měnit například v situaci, kdy herní objekt postavy hlavního hrdiny utíká po mapě a následně vyskočí, aby přeskočil překážku. V ten moment, kdy dojde k výskoku změní aktuální animaci běhu za animaci výskoku a při dopadu zpět na zem se aplikuje zase jiná animace.

Jednotlivé animace lze přidávat před funkci *add,* kde je v prvním parametru očekáván název animace a v druhém samotná komponenta *Animaiton*. Přidané animace lze odebrat přes funkci *remove*, kde parametrem je název animace. Samotná změna aktuální animace se provádí přes funkci *play,* kde se v parametru předá název požadované animace.

Komponenta *Collider*

Tato komponenta je jednou z nejdůležitějších v celém enginu. Hernímu objektu na sebe přidá obdélník, jenž má svou výšku a šířku. Tento obdélník je dále využívám pro detekci kolize s ostatními herními objekty, které mají taktéž tuto komponentu. Samotná kontrola, zdali došlo ke kolizi se provádí ve třídě *ColliderManager*.

Vlastností této komponenty dále využívají komponenty *CharacterMoving* a *SimpleGravity*. Kde je detekce kolize potřeba například pro zjištění, zdali se herní objekt pohybuje nebo stojí na místě.

Kromě výšky a šířky kolizního obdélníku, kde jeho pozici na herním objektu lze posunout pomocí vlastností *offsetX* a *offsetY*. Samotný obdélník vždy začíná v levém horním rohu herního objektu, pomocí těchto vlastností, pro které existují funkce *get* a *set* lze posunout. Posunutí obdélníku je potřeba například v situaci, kdy by detekce kolize v daném místě neodpovídala grafickému vizuálu, který nese herní objekt.

Komponenta obsahuje ještě vlastnost *trigger,* česky spoušť. Jedná se o vlastnost, která při kolizi mezi herními objekty říká enginu, že nemá řešit posunutí herních objektů, tak aby jejich kolizní komponenty nebyly v kolizi. Místo toho se vyvolá pouze událost, kterou obdrží herní objekt, jenž komponentu vlastní.

Implementovaný engine aktuálně dovoluje detekci kolizí pouze obdélníkových tvarů.

Komponenta *FixedPosition*

Komponenta slouží především pro účely grafického rozhraní pro uživatele. Třída disponuje vlastním výčtovým typem, jehož hodnoty reprezentují pozici umístění v prostoru okna aplikace

hry. Těmito hodnotami jsou umístění pravý horní roh, levý horní roh, pravý dolní roh, levý dolní roh a střed obrazovky.

Při vytvoření komponenty se jí předá jedna z hodnot tohoto výčtového typu. Pro případ, že by bylo potřeba umístit herní objekt na jiné místo, je možné pozici posunout pomocí odsazení. Stejně jako v případě komponenty *Collision,* zde existují vlastnosti *offsetX* a *offsetY,* které se dají nastavit patřičnými funkcemi.

Situací pro využití této komponenty je například herní menu, které je ve většině případů umístěno uprostřed obrazovky.

Komponenta *SimpleGravity*

Jak je z názvu patrné, jedná se o komponentu, která má v herním enginu na starosti jednoduchou simulaci gravitace. Komponentě se při jejím vytvoření nastavuje váha, kterou bude mít herní objekt. Čím větší je váha, tím rychleji herní objekt bude padat směrem k zemi.

Rychlost pádu není nekonečná a existuje zde maximální konečná rychlost, kterou může herní objekt padat. Jedná se o 50 pixelů za jednu iteraci herní smyčky.

Samotná herní komponenta nutí herní objekt neustále padat k zemi a měnit, tak jeho pozici v prostoru. Při kolizi s jiným herním objektem je rychlost pádu vynulována a zrychlování pádu začíná opět od znova.

Pro jiné herní komponenty tato komponenta poskytuje informaci o tom, zdali je herní objekt uzemněn, tedy zdali herní objekt stojí pevně na místě a nedochází, tak k možnému pádu.

Komponenta *CharacterMoving*

Tato komponenta má na starost pohyb herních objektů v herním světě. Je založena na pohybu herního objektu reprezentující humanoidní postavu. Komponenta disponuje vlastnostmi, rychlost chůze a rychlost výskoku, které jsou komponentě nastavené při jejím vytvoření. Rychlost výskoku se zde myslí, jakou rychlostí herní objekt poletí vzhůru. Pro tuto komponentu je velice důležité, aby herní objekt disponoval také komponentou *SimpleGravity*. Síla gravitace, při výskoku herního objektu, bude růst až překoná rychlost výskoku a herní objekt začne padat.

Komponenta dovoluje vyskočit pouze jednou, pro opětovné vyskočení je třeba, aby se herní objekt dotkl země, tedy jiného herního objektu vlastnící kolizní komponentu. Ke kolizi ovšem musí přijít se spodní částí.

Komponenta *RenderImage*

Komponenta zajišťující vykreslování grafických prvků, kterými jsou obrázkové soubory na plátno okna aplikace. Jedná se o její jediný a primární úkol. Při vytvoření se komponentě předá cesta k obrázkovému souboru, který má být vykreslen.

Komponenta *SoundController*

Tato komponenta se stará o zajištění hudebního doprovodu pro hru prostřednictvím zvukových efektů. Jednotlivé zvukové efekty jsou uvnitř komponenty uchovávaný v kolekci, kde má každý zvukový efekt svůj jedinečný název.

Komponenta pro uložení jednotlivých zvukových efektů, využívá třídu *SoundEffect,* která v sobě uchovává všechny informace o zvukovém souboru.

Komponenta disponuje několika funkcemi, jednou z hlavních je funkce *add*, která se stará o přidávání zvukových efektů do kolekce. Parametry této funkce jsou jméno a cesta k souboru se zvukovým efektem. Dále funkcí *remove* je možné odstranit zvukový efekt z kolekce podle jejího jména předané v parametru funkce.

Zvukové soubory v kolekci lze přehrát pomocí funkce *play*, která na svém parametru přímá jméno zvukového efektu. Se zvukovým efektem lze dále pracovat s funkcemi *pause*, která přehrávaný zvukový efekt pozastaví. Dále je tu i funkce *resume*, která opět spustí pozastavený zvukový efekt. Poslední funkcí je *stop,* ta přehrávaný soubor úplně zastaví.

Implementovaný herní engine aktuálně dokáže pracovat pouze se zvukovými soubory s efekty ve formátu WAV.

6.2.2 Herní objekt

Druhou hlavní důležitou třídou v enginu, je třída samotného herního objektu, která má své vlastnosti a funkcionalitu definovanou právě podle vlastněných komponent.

Třída *GameObject*

Jedná se o samotný herní objekt, jenž si své komponenty uchovává v kolekci. Komponenty jsou v kolekci uloženy podle jména třídy komponenty a v kolekci, tak může existovat pouze jedna instance komponenty od každé třídy.

Kromě samotných komponent nese třída i informace o pozici v prostoru herního světa, své velikosti, jméno, značku, zdali je aktivní a pozici na ose Z.

Informace o jménu a značce jsou důležité například při výskytu kolize, kdy druhý herní objekt, se kterým došlo ke kolizi může zjistit, co je herní objekt zač k dané situaci reagovat jinak.

Velikost, tedy výška a šířka, herního objektu je důležitá především pro vyvolání události kliknutím tlačítkem myši na pozici herního objektu. Při této události se počítá oblast, kterou zabírá herní objekt z těchto vlastností, není zde využívána kolizní komponenta.

Pozice na ose Z určuje, v jakém pořadí budou herní objekty vykreslovány a zaručuje, že nedojde k překrytí jiným herním objektem v případě, že je to nechtěné.

Zdali je herní objekt aktivní ovlivňuje jeho volání. Pokud je herní neaktivní nevolají se nad jeho komponentami funkce *update* a *render*. Dále se u něj nekontrolují kolize.

Kromě funkcí, které nastavují stav vlastností herního objektu, třída disponuje funkcemi *getComponent, callOnTrigger, callOnCollision, callOnMouseClick.* Funkce getComponent vrací požadovanou komponentu z kolekce. Zbylé tři funkce volají nad všemi komponentami funkce, které mají v názvu. Tyto funkce již byly popsány u abstraktní třídy komponenty.

6.2.3 Herní smyčka

Samotná herní smyčka je implementována v nejednodušší formě. Jedná se o cyklus zpracovávající v každé iteraci vstupy od uživatele, následně provede logiku všech komponent, všech herních objektů a ve finále vykreslí stav všech těchto komponent.

Herní smyčka se nachází ve třídě *Core*, kde se také nachází všechny důležité informace o všech existujících úrovních, včetně aktuální úrovně, kterou lze vyměnit a načíst jinou z úrovní v kolekci.

6.2.4 Ostatní třídy

Kromě tříd vytvořených v rámci návrhu ECS existuje v enginu i několik dalších tříd.

Třída *ColliderManager*

Jedná se o statickou třídu, jež je využívána v herní smyčce ve třídě *Core*. První z funkcí této třídy kontroluje, zdali došlo ke kolizi mezi herními objekty, které vlastní komponentu kolize a jejichž pozice se od předchozí iterace herní smyčky změnila. V případě výskytu kolize informuje herní objekty, kterých se to týká, a řešení kolize mají následně na starosti jejich určité komponenty.

Druhá funkce kontroluje kolizi místa, kliknutí myši a herní objekty, kterých se to týká informuje obdobným způsobem jako při kolizi mezi objekty.

Třída *SoundManager*

Statická třída, která se stará o hlasitost zvukových efektů a hudby. Kromě hlasitosti má třída na starost spouštění a ovládání hudebního doprovodu hry.

Třída *TimeManager*

Jedná se o statickou třídu, které má na starost čas v herním enginu. Tato třída umožňuje zrychlovat či zpomalovat rychlost plynutí času. Ve třídě je taktéž uložena informace o době, jak dlouho hra běží od doby jejího spuštění.

Třída *WindowManager*

Statická třída, jež má na starost aplikační okno hry. Lze zde nastavit výšku a šířku okna aplikace. Taktéž lze nastavit okno na celou obrazovku nebo naopak do okna s rámem.

Třída *Position*

Třída nese informace o poloze v herním světě. Tato třída je využívána v herním objektu.

Třída *SoundEffect*

Jedná se o třídu, které nese informace o souborech se zvukovými efekty, se kterými lze pracovat operacemi spuštění, zastavení nebo pozastavení přehrávaného zvuku. Třída je využívána v komponentě *SoundController,* která v sobě nese více instancí této třídy.

Třída *Rect*

Pomocná statická třída, která zjednodušuje sestavení obdélníku pro vykreslování na plátno.

Třída *SharedMemory*

Statická třída, které umožňuje uložení hodnot na souborový systém. Uložený soubor je při spuštění hry ze souborového systému načten do paměti, a tak lze s načtenými hodnotami dále pracovat.

Uložené hodnoty jsou v souboru uložené v textovém tvaru, takže nejsou nikterak chráněné proti přepisu. V případě, že dojde k odstranění souboru s uloženými hodnotami, je vytvořen nový soubor.

Jedná se o jednoduchý způsob, kterým se ve hře dá ukládat například nejvyšší dosažené skóre nebo nejrychlejší čas.

Třída *Camera*

Třída se stará o sledování zvoleného herního objektu a posouvá obraz, tak aby byl požadovaný herní objekt uprostřed vykreslované scény. Posunutí je docíleno změnou pozice herních objektu, kde je k jejich pozic přičtena pozice aktuálního středu obrazovky.

Kamera nemusí sledovat pouze střed obrazovky. Třídě lze přes vlastnosti nastavit odsazení od středového bodu.

Jedná se o statickou třídu, která je využívána při vykreslování grafiky.

Třída *DebugLogging*

Statická třída poskytující možnost vývojářům vypisovat požadované hodnoty do konzole nebo do souboru. Důvod existence této třídy je pouze pro vývojářské potřeby.

Třída *ErrorLogging*

Třída zapisující chyby vzniklé při běhu hry do konzole a do souboru.

Třída *Input*

Statická třída, jež se stará o zpracování vstupních signálů od uživatele zadaných prostřednictvím klávesnice a myši. Poskytuje mechanismus pro zjišťování, která z kláves na klávesnici či myši je aktuálně stisknutá, nebo která klávesa byla uvolněna.

Třída *Level*

Třída reprezentuje jednu herní úroveň. Kromě svého jména vlastní také herní objekty, které má uschované v kolekci.

Jednotlivé herní objekty lze do kolekce přidávat a odebírat. Při přidávání nového herního objektu dochází ke kontrole, zdali se v úrovni nenachází již jiný herní objekt se stejným jménem. Pakliže tato situace nastane je za jméno přidáno do závorky počet herních objektů již obsažených v kolekci.

Podobně se postupuje v případě, že vkládaný herní objekt nemá vyplněné jméno. Takovému hernímu objektu je pak přiřazeno jméno „GameObject (1)“, v závorce je pak opět číslo reprezentující počet herních objektů v kolekci.

Třída *Sprite*

Jedná se o třídu, jež v sobě uchovává obrázkový soubor s animačními snímky, jeho rozměry a počet snímků nacházejících se na řádku a ve sloupci. Tato třída je využívána v komponentě *Animation.*

Výčtový typ *KeyCode*

Výčtový typ, který v sobě uchovává hodnoty sloužící pro třídu *Input*. Jedná se o podporované klávesy na klávesnici a myši.

6.3 Implementování ukázkové hry

V rámci praktické části práce bylo úkolem demonstrovat možnosti enginu. Byla proto vytvořena jednoduchá skákací hra inspirovaná dnes již klasickými hrami, kterými jsou *Mario* a *Sonic* z konce 20. století.

6.3.1 Příběh hry

Ve hře se hráč ujme role robota, jehož úkolem je sesbírat všechny mince na mapě, kterých je v každé úrovni celkem tři. Po nasbírání všech mincí se musí robot dostat k vlajce, aby dokončil danou úroveň. Pokud se k vlajce dostane dříve, než nasbírá všechny mince, nemůže aktuální úroveň opustit a musí se vydat zpět a sesbírat chybějící mince.

6.3.2 Herní úrovně

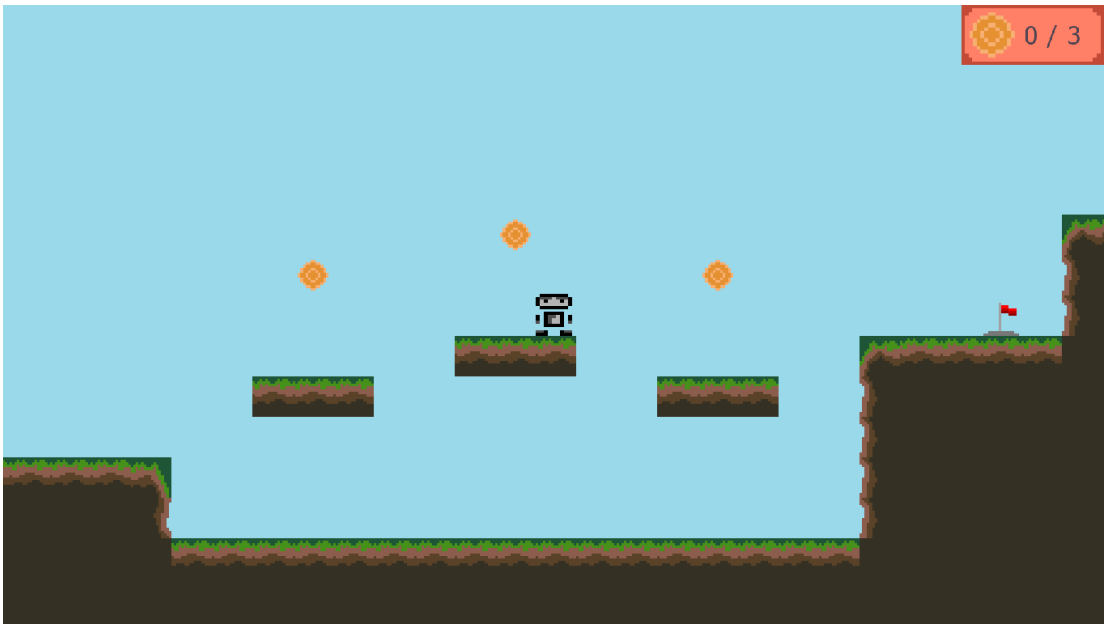
Hra obsahuje celkem tři herní úrovně. V každé úrovni se nachází vždy tři mince, vlajka a robot, za kterého hraje hráč. Po úspěšném dokončení úrovně může hráč pokračovat okamžitě do další úrovně nebo se vrátit zpět do menu.

V každé úrovni má hráč v pravém horním rohu ukazatel počtu sesbíraných mincí, který mu pomáhá se zorientovat kolik mincí ještě chybí k sesbírání.

Úrovně jsou ve hře vytvářené pomocí dvourozměrného pole, které v sobě nese číselné hodnoty. Každá taková hodnota má jiný význam a určuje, jaký herní objekt, s jakými komponentami se bude na daném místě nacházet. Jedná se především o pozice herních objektů terénu a herních objektů sbíraných mincí. Kupříkladu hodnota „1“ je herní objekt s komponentou textury trávy po celé své délce a komponentou kolize. Hráč tedy po tomto herním objektu může přejít a pokračovat, tak ve své cestě.

Pro iterování celého tohoto dvourozměrného pole má za následek vytvoření velkého množství herních objektů, která jsou následně vložené do požadované úrovně.

Na obrázku 24 je možné vidět snímek z druhé úrovně hry. Na obrázku lze vidět kromě terénu hry, robota, za kterého hráč hraje, tři mince a vlajku, ke které se musí dostat.



Obrázek 24: Ukázka druhé úrovně

6.3.3 Herní logika

Aby se hra stala hrou, bylo potřeba vytvořit pro nový herní svět pravidla, kterými se bude řídit. To bylo doprogramována pomocí komponent, které vychází z návrhu ECS, jenž se řídí implementovaným enginem. Vytvořené komponenty nesou na sobě herní objekty v různých úrovních a svou naprogramovanou logikou vytváří herní svět.

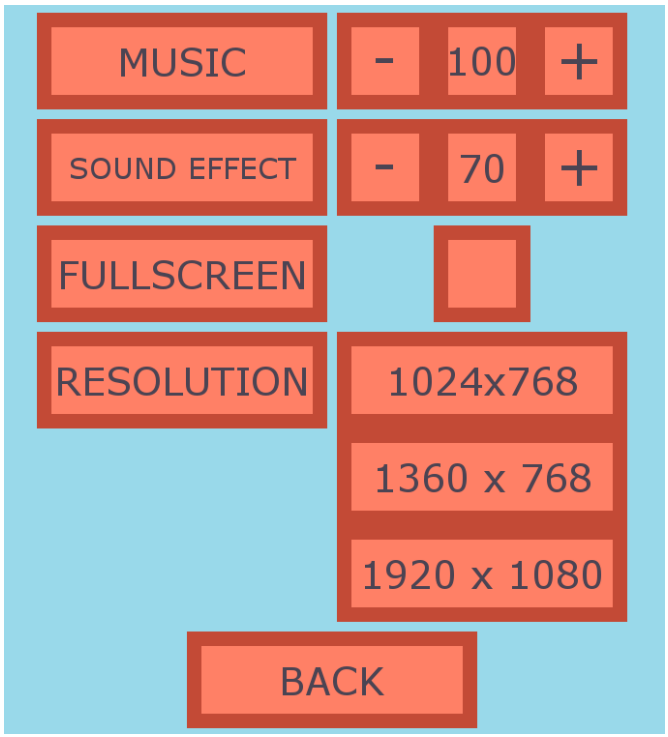
Jednou z takovýchto komponent je například *PlayerController*, která se stará o pohyb robota v herním světe. Dále se stará o počítání skóre sesbíraných mincí, se kterými přijde herní objekt do kolize.

6.3.4 Nastavení

Hra disponuje i části s nastavením, ve kterém si hráč může změnit hlasitost hudby a zvukových efektů. Dále je zde možnost zapnutí či vypnutí režimu přes celou obrazovku. Posledním, co lze nastavit je rozlišení hry. Aktuálně lze nastavit celkem tři možná rozlišení, a to konkrétně 1024 na 768, 1360 na 768 nebo 1920 na 1080.

Provedené nastavení je uloženo do paměti a při opětovném načtení hry je aplikováno. Uložení a načtení nastavení je realizováno pomocí třídy z herní enginu *SharedMemory*.

Na obrázku 25 je možné vidět ukázku nastavení takového, jak vypadá v rámci hry.



Obrázek 25: Ukázka nastavení ve hře

6.3.5 Vizuál hry

Vizuální stránka vychází z druhu počítačové grafiky zvaný *pixel art*. Jedná se o metodiku, kde samotný obrázek je vytvářen kreslením pixel po pixelu. Detailnější obrázky, vytvářené touto formou, vyžadují velké množství času na tvorbu a grafický cit. Z tohoto důvodu existují ve hře pouze obrazové soubory s jednoduchým grafickým provedením.

Pro tvorbu tohoto typu počítačové grafiky, lze použít takřka jakýkoliv grafický editor. Finální grafický vizuál pro hru byl vytvořen v programech *Pyxel Edit* a *GIMP*.

6.3.6 Zvukové efekty

Pro ozvučení hry byly použity zvukové soubory s efekty ze stránky freesound.com. Konkrétně byly použity následující záznamy:

- 07114 hindu fail ding od uživatele Robinhood76,
- Coins 1 od uživatele ProjectsU012,
- Jump02 od uživatele sharesynth.

Na stránce lze nalézt velké množství zvukových záznamů s různou licencí omezující jejich použití.

ZÁVĚR

Herní enginy vytvořené přímo na míru nebo vytvořené jinou vývojářskou firmou jsou nedílnou součástí každé vytvořené hry. Jedná se o rozsáhlé nástroje, které herním tvůrcům umožňují efektivně a rychle vytvářet hry. Tímto tématem se zabývá i tato diplomová práce, jejímž cílem bylo vytvořit jednoduchý herní engine, v němž by bylo možné vytvářet 2D hry.

V teoretické části diplomové práce byla představena problematika herních enginů, dále byl představen herní trh a herní žánry, které svými požadavky velice ovlivňují nároky na vlastnosti použitého enginu zvoleného pro realizaci konkrétní hry.

V praktické části diplomové práce byl vytvořen jednoduchý 2D herní engine v nástroji *Microsoft Visual Studio 2017*, s využitím technologie C++ a *SDL*. Součástí praktické části práce byl také návrh herního enginu a vytvoření jedné ukázkové hry, demonstrující možnosti enginu. Výsledný engine umožňuje zpracování vstupních signálů z periferií, kterými jsou myš a klávesnice, dále umožňuje vykreslování statických i animovaných obrázků, detekci kolizí, jednoduchou simulaci gravitace a zvukový výstup.

Vytvořený herní engine je díky použitým technologiím možné používat a vytvářet hry na všechny hlavní typy operačních systémů.

Dalšími možnými vylepšeními aktuálního vytvořeného herního enginu, jsou například implementace vlastního grafického editoru, který by umožnil tvorbu i lidem bez hlubších programátorských znalostí či podpora dalších vstupních zařízeních, kterým je například gamepad. Za zváženou by stálo i přidání kompilace her do jazyka JS pomocí technologie *Emscripten*, která si rozumí s použitou knihovnou *SDL*. Vytvářené hry by díky tomu byly pro hráče přístupné i přímo skrze webový prohlížeč.

POUŽITÁ LITERATURA

[1] GREGORY, Jason. *Game engine architecture*. Second edition. Boca Raton: CRC Press, 2014. ISBN 978-1466560017.

[2] DOAN, Daniel. GameDev Protips: How To Effectively Combine Multiple Genres In Game Development. *Medium* [online]. Mar 16, 2017 [cit. 2019-03-14]. Dostupné z: https://medium.com/@doandaniel/gamedev-protips-how-to-effectively-combine-multiple-genres-in-game-development-b6a2cdceebc4

[3] MATTHEWS, Vince. *The Many Different Types of Video Games & Their Subgenres* [online]. Apr 12, 2018 [cit. 2019-03-14]. Dostupné z: https://www.idtech.com/blog/different-types-of-video-game-genres

[4] *Card Life: Hearthstone by Blizzard Entertainment* [online]. June 19. 2014 [cit. 2019-05-05]. Dostupné z: https://unity3d.com/showcase/case-stories/hearthstone

[5] LOWRY, Brendan. *The major differences between 'indie' and 'AAA' video games* [online]. 29 Nov 2017 [cit. 2019-03-14]. Dostupné z: https://www.windowscentral.com/indie-vs-aaa-which-type-game-you

[6] *Video Game History* [online]. September 1, 2017 [cit. 2019-03-14]. Dostupné z: https://www.history.com/topics/inventions/history-of-video-games

[7] COHEN, David. *The History of the Sony PlayStation* [online]. February 14, 2019 [cit. 2019-03-14]. Dostupné z: https://www.lifewire.com/history-of-sony-playstation-729672

[8] WIJMAN, Tom. *Mobile Revenues Account for More Than 50% of the Global Games Market as It Reaches $137.9 Billion in 2018* [online]. Apr 30 2018 [cit. 2019-03-14]. Dostupné z: https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/

[9] RATHI, Nimisha. *MOBILE GAME MONETIZATION MODELS FOR 2018* [online]. Apr 4, 2018 [cit. 2019-03-14]. Dostupné z: https://medium.com/greedygame-media/mobile-game-monetization-models-for-2018-5c49a2978dca

[10] LOWOOD, Henry. *Game Engines and Game History* [online]. [cit. 2019-03-18]. Dostupné z: https://www.kinephanos.ca/2014/game-engines-and-game-history/

[11] VANDEVENNE, Lode. *Lode's Computer Graphics Tutorial* [online]. [cit. 2019-03-18]. Dostupné z: https://lodev.org/cgtutor/raycasting.html

[12] HALPERN, Jared. *The What and Why of Game Engines* [online]. [cit. 2019-03-18]. Dostupné z: https://medium.com/@jaredehalpern/the-what-and-why-of-game-engines-f2b89a46d01f

[13] MCSHAFFRY, Mike. *Game coding complete*. 4th ed. Boston, MA: Course Technology, Cengage Learning, c2013. ISBN 978-1-133-77657-4.

[14] HARBOUR, Jonathan S. *Advanced 2D game development*. Boston, MA: Course Technology/Cengage Learning, c2009. ISBN 978-1-59863-342-9.

[15] CATTO, Erin. *Box2D: A 2D Physics Engine for Games* [online]. [cit. 2019-03-18]. Dostupné z: https://box2d.org/

[16] *FMOD* [online]. [cit. 2019-03-18]. Dostupné z: https://www.fmod.com/

[17] *DirectInput* [online]. 09/10/2011 [cit. 2019-03-18]. Dostupné z: https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ee416842(v=vs.85)

[18] *OpenGL* [online]. [cit. 2019-03-18]. Dostupné z: https://learnopengl.com/Getting-started/OpenGL

[19] *History of OpenGL* [online]. [cit. 2019-03-18]. Dostupné z: https://www.khronos.org/opengl/wiki/History_of_OpenGL

[20] *Industry Forged* [online]. [cit. 2019-03-18]. Dostupné z: https://www.khronos.org/vulkan/

[21] GREER, Matt. *Particle Systems From the Ground Up* [online]. Nov 21st, 2012 | [cit. 2019-03-18]. Dostupné z: http://buildnewgames.com/particle-systems/

[22] *Emscripten* [online]. 2015 [cit. 2019-03-18]. Dostupné z: https://emscripten.org/

[23] *Getting started with WebGL development* [online]. 2018–03–19 [cit. 2019-03-18]. Dostupné z: https://docs.unity3d.com/Manual/webgl-gettingstarted.html

[24] Unity. *Unity* [online]. Copyright © 2019 Unity Technologies [cit. 19.03.2019]. Dostupné z: https://unity.com/

[25] FINE, Richard. *UnityScript's long ride off into the sunset* [online]. August 11, 2017 [cit. 2019-03-19]. Dostupné z: https://blogs.unity3d.com/2017/08/11/unityscripts-long-ride-off-into-the-sunset/

[26] BRODKIN, Jon. *How Unity3D Became a Game-Development Beast* [online]. June 3, 2013 [cit. 2019-03-19]. Dostupné z: https://insights.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast/

[27] *Exploding visuals on the go: Shadowgun by Madfinger Games* [online]. July 27. 2011 [cit. 2019-03-19]. Dostupné z: https://unity3d.com/showcase/case-stories/madfinger-shadowgun

[28] *Card Life: Hearthstone by Blizzard Entertainment* [online]. June 19. 2014 [cit. 2019-03-19]. Dostupné z: https://unity3d.com/showcase/case-stories/hearthstone

[29] *Hyperbolic Magnetism* [online]. [cit. 2019-03-19]. Dostupné z: http://presskit.hyperbolicmagnetism.com/sheet.php?p=beat_saber

[30] *Old Man's Journey Presskit* [online]. [cit. 2019-03-19]. Dostupné z: http://www.brokenrul.es/presskit/?Old_Mans_Journey

[31] *Ori and the Blind Forest* [online]. [cit. 2019-03-19]. Dostupné z: https://www.thqnordic.com/games/ori-and-blind-forest

[32] *GameMaker Studio 2* [online]. [cit. 2019-03-19]. Dostupné z: https://www.yoyogames.com/gamemaker

[33] *What's a Universal Windows Platform (UWP) app?* [online]. 05/07/2018 [cit. 2019-03-19]. Dostupné z: https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide

[34] O HŘE | BLACKHOLE [online]. [cit. 19.03.2019] Dostupné z: https://www.blackhole-game.com/cs/o-hre

[35] Undertale - Press Kit. [online]. [cit. 19.03.2019]. Dostupné z: https://www.igdb.com/games/undertale/presskit

[36] *TURMOIL* [online]. [cit. 2019-03-19]. Dostupné z: https://gamious.com/turmoil/

[37] *What is unreal engine 4* [online]. [cit. 2019-03-19]. Dostupné z: https://www.unrealengine.com/en-US/what-is-unreal-engine-4

[38] SWEENEY, Tim. *Announcing the Epic Games store* [online]. December 4, 2018 [cit. 2019-03-19]. Dostupné z: https://www.unrealengine.com/en-US/blog/announcing-the-epic-games-store

[39] *XCOM 2 Press kit* [online]. [cit. 2019-03-19]. Dostupné z: https://www.igdb.com/games/xcom-2/presskit

[40] *A Way Out Press kit* [online]. [cit. 2019-03-19]. Dostupné z: https://www.igdb.com/games/a-way-out/presskit

[41] *Play A Way Out with a Friend, Even if They Don't Own the Game: The A Way Out Friend Pass lets you bring your best friend behind bars for free.* [online]. [cit. 2019-03-19]. Dostupné z: https://www.ea.com/games/a-way-out/news/a-way-out-friends-pass?isLocalized=true

[42] *Cardboard Sword* [online]. [cit. 2019-03-19]. Dostupné z: http://presskit.siegeandsandfox.com/

[43] *Products* [online]. [cit. 2019-03-19]. Dostupné z: https://cocos2d-x.org/products

[44] *About cocos2d-x* [online]. [cit. 2019-03-19]. Dostupné z: https://docs.cocos2d-x.org/api-ref/cplusplus/V3.2/

[45] SID, James. *Top 10 Cocos2d-x Games Ever Made* [online]. [cit. 2019-03-19]. Dostupné z: http://blog.soomla.com/2015/10/top-10-cocos2d-x-games-ever-made.html

[46] *Hill Climb Racing 2 Press kit* [online]. [cit. 2019-03-19]. Dostupné z: https://www.igdb.com/games/hill-climb-racing-2/presskit

[47] *Geometry Dash Press kit* [online]. [cit. 2019-03-19]. Dostupné z: https://www.igdb.com/games/geometry-dash/presskit

[48] *Flow Press kit* [online]. [cit. 2019-03-19]. Dostupné z: https://www.igdb.com/games/flow--1/presskit

[49] SEFTON, Daniel. *Developing a Data-Oriented Game Engine (Part 1)* [online]. [cit. 2019-05-07]. Dostupné z: https://www.danielsefton.com/2016/05/developing-a-data-oriented-game-engine-part-1/

[50] *Simple DirectMedia Layer* [online]. [cit. 2019-05-05]. Dostupné z: https://www.libsdl.org/