Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Dobrodružná hra pro iOS

2023

Jan Czerný

Vedoucí práce:
doc. RNDr. Michal Krupka, Ph.D.

Studijní program: Aplikovaná informatika,
prezenční forma

Bibliografické údaje

Autor:	Jan Czerný
Název práce:	Dobrodružná hra pro iOS
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2023
Studijní program:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	doc. RNDr. Michal Krupka, Ph.D.
Počet stran:	38
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliographic info

Author:	Jan Czerný
Title:	Adventure game for iOS
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2023
Study program:	Applied Computer Science, full-time form
Supervisor:	doc. RNDr. Michal Krupka, Ph.D.
Page count:	38
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Tato práce představuje implementaci 2D „adventure“ hry pro zařízení iOS a iPadOS. Porovnává možnosti vývoje her pro platformu iOS, u které zkoumá herní enginy a programovací jazyky. Vysvětluje důvod výběru frameworku SpriteKit a jazyka Swift. Rozebírá a popisuje frameworky SpriteKit a GamePlayKit, použité pro vývoj hry. Následně uvádí použití zmíněných frameworků při tvorbě hry a popisuje řešení problémů, jež mohou nastat při vývoji her pomocí zmíněných frameworků. Text této práce seznamuje čtenáře s obsahem hry, jejím příběhem a nakonec poskytuje programátorskou část, v níž vysvětluje implementaci hry i její uživatelskou dokumentaci, která seznámí uživatele s tím, jak se ve hře správně orientovat a na co se v ní zaměřit.

Synopsis

This work presents the implementation of a 2D "adventure" game for iOS and iPadOS devices. It compares game development options for the iOS platform, where possible game engines and programming languages are explored. It explains the reason for choosing the SpriteKit framework and the Swift programming language. Further on it breaks down and describes the SpriteKit and GameplayKit frameworks used for the actual game development. Subsequently, it presents the use of the mentioned frameworks during the creation of a game and describes the solutions for problems, that may arise during the development of games while using the chosen frameworks. The text introduces the reader to the content of the game and its story, and it ultimately provides a programmer's guide explaining the game's implementation and the game's user documentation, which guides the user on how to properly navigate the game and what to focus on ingame.

Klíčová slova: iOS; Swift; Hra; SpriteKit

Keywords: iOS; Swift; Game; SpriteKit

Velké díky bych chtěl věnovat svému vedoucímu bakalářské práce především za jeho trpělivost, za jeho ochotou a za pomoct s textovou částí. Dále bych chtěl poděkovat své rodině a přátelům za neustálou podporu při tvorbě a psaní této práce.

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

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Vzestup oblíbenosti mobilních her byl jedním z nejpozoruhodnějších vývojů v herním průmyslu. Mobilní hry radikálně změnily způsob, jakým lidé hrají videohry, a otevřely zcela nový svět herních příležitostí. Proto jsem se rozhodl vyzkoušet si tvorbu mobilní hry ve své vlastní režii.

Práce se zabývá vytvořením 2D hry typu „adventure“ pro iOS zařízení za pomocí frameworku SpriteKit. Úvod této práce se věnuje zkoumání herních enginů, které se používají při vývoji na zmíněné platformě a dále pak porovnává různé frameworky mezi sebou a s frameworkem SpriteKit. V části úvodu je také kromě frameworku vybrán i vhodný programovací jazyk, který bude sloužit pro vývoj hry a následně je zhodnoceno vývojové prostředí za účelem usnadnění vývoje.

V další části jsou následně uvedeny důvody podporující výběr frameworku SpriteKit, výběr programovacího jazyka Swift, ale také volbu vývojové platformy XCode. V jejím rámci je také popsán framework SpriteKit a jeho možnosti použití při vývoji her, s tím, že je poukázáno i na jeho nedostatky. Zmíněn je zde také framework GameplayKit, který rozšiřuje hru o architekturu Entity Component System (ECS) a o ovládání NPC. Architektura ECS i ovládání NPC jsou poté v práci dále okomentovány.

V závěru práce je podrobně popsána výsledná hra, její příběh a příběhové mechaniky, její herní prvky, ale také struktura úrovní a design jednotlivých úrovní. Část obsahu práce tvoří také problémy, které mohou při vývoji hry nastat a jejich možná řešení. Závěr práce je krom toho věnován popisu implementace hry a jejich hlavních prvků. Součástí této práce je taktéž uživatelská dokumentace, jenž seznamuje uživatele s ovládáním hry a sděluje, co je ve hře důležité a na co je potřeba zaměřit.

2 Evaluace možností vývoje her pro operační systém iOS

2.1 Herní engine

Pomocí herního enginu mohou vývojáři ušetřit čas a úsilí při vytváření svých her. Herní engine poskytuje sadu nástrojů a funkcionalit pro vývoj her, jako jsou renderování, animace, simulace fyziky, zpracovávání uživatelských vstupů a další, v závislosti na jeho velikosti [1]. Kromě toho mohou herní enginy vývojáři zjednodušit přenést hry na jinou herní platformu.

Vytváření hry bez herního enginu vyžaduje vysoké technické znalosti v oblasti grafiky, jejího renderování, animací, fyziky. Proto použití dostupných enginů pomáhá vývojáři více se soustředit na programování hratelnosti a na herní design[1].

2.2 Srovnání herních enginů a frameworků pro vývoj 2D her

K vytváření 2D her pro zařízení iOS je na výběr z několika enginů. Jmenovitě se jedná o Cocos2D, SpriteKit, Unity a Unreal Engine.

2.2.1 Unreal Engine

Unreal Engine je cross-platform herní engine podporující široké spektrum herních platforem [2]. Hry se píšou v jazyce C++, anebo vizuálním skriptováním. Díky své popularitě má Unreal Engine aktivní komunitu a velký obchod balíčků, které obsahují již naprogramovanou funkcionalitu.

K výhodám patří silná komunita vývojářů, významná podpora herních platforem, grafické možnosti a možnost vytvořit hry z velké části přes vizuální skriptování. Mezi nevýhody náleží značná velikost zkompilovaného projektu, složitost enginu především u vytváření 2D-her, na které není optimalizován, a jazyk C++ může být složitý pro začínající vývojáře.

2.2.2 Unity

Unity je multifunkční herní engine, který podporuje 2D i 3D grafiku. Unity, stejně jako Unreal Engine, obsahuje vizuální skriptování a velký obchod balíčků spolu s aktivní komunitou vývojářů [3]. Hry se vyvíjí v jazyce JavaScript, nebo v C#.

Mezi výhody se řadí podpora více než 25 herních platforem, aktivní komunita, časté aktualizace enginu, velký obchod s balíčky. K nevýhodám se počítají následující skutečnosti: může být náročný pro začínající vývojáře her pro svou velikost, obsáhlost a složitost, engine není optimalizován pro 2D-hry a má velice pomalý build, který může velmi znesnadnit ladění hry.

[1] (Video)Herní design je proces vytváření herních mechanik, úrovní, hratelnosti, příběhu, grafiky a dalších.

2.2.3 Cocos2D

Cocos2D je cross-platform open source herní engine s podporou jazyků C++, JavaScript, C#, Objective-C, Swift a Python. Obsahuje spoustu nezávislých editorů vyvinutých Cocos2D-komunitou.

Výhody představují již zmíněná podpora různých jazyků, cross-platform engine a jednoduchost. K nevýhodám se řadí slabá komunita, nízká podpora a zastaralost v porovnání se SpriteKitem. Další nevýhodou je, že hra vytvořená pomocí Cocos2D nemusí fungovat pro nově vydané verze iOS, jelikož kompatibilita frameworku a nové verze iOS není zaručena. Problematiku kompatibility lze vidět na oficiálním repozitáři[2] Cocos2D, kde se dá najít přes 1 300 otevřených problémů.

2.2.4 SpriteKit

SpriteKit je framework k vytváření 2D her pro Apple zařízení vyvinutý společností Apple. Hry se programují v jazycích Objective-C a Swift. SpriteKit je moderní a nativní verze frameworku Cocos2D [4]. Obsahuje vizuální editor, který je zakomponován v nativním vývojovém prostředí XCode.

Mezi výhody SpriteKitu se počítají podpora společnosti Apple, jednoduchost učení i používání. SpriteKit tvoří součást iOS SDK, díky čemuž je zajištěna kompatibilita s novými verzemi iOS. K nevýhodám patří vývoj možný pouze pro Apple zařízení, malá komunita, dlouhá doba mezi aktualizacemi frameworku a malá nabízená funkcionalita v porovnání s ostatními enginy.

2.3 Výběr platformy pro vývoj hry

Pro výběr enginu, nebo frameworku jsem si stanovil několik podmínek. Měl by být jednoduchý na naučení a používání, mít možnost rozšíření hratelnosti, dostupnost k rozšířené funkcionalitě bez nutnosti instalace balíčků a nejlépe nativní podporu pro operační systém iOS. Tyto podmínky splňoval herní engine SpriteKit.

Unreal Engine je nevhodný pro tuto bakalářskou práci z důvodu své velikosti, složitosti a potřeby instalovat dodatečný plugin pro možnost tvorby 2D-her. Herní engine Unity rovněž nepředstavuje vhodnou volbu, protože není optimalizován pro 2D hry a pro přidání vlastní hratelnosti je složitější na naučení. Pomalá kompilace kódu v Unity může být také problém, pokud je nutné hru často ladit. Cocos2D je „zastaralá“ verze SpriteKitu bez podpory, která má problémy s kompatibilitou, proto jsem engine nezvolil. Mezi Unity a SpriteKitem rozhodly nativní podpora Apple-zařízení a zaměření SpriteKitu na vývoj 2D-her.

Výběr SpriteKitu však nebyl nejvhodnější, což jsem zjistil až v průběhu vývoje hry. V enginu chybí tlačítka, která si musí vývojář naprogramovat sám, možnost jednoduchého pozastavení všech prvků hry a zobrazení menu. Největší

[2]Otevřené problémy ve frameworku Cocos2D: https://github.com/cocos2d/cocos2d-x/issues

nedostatek se projevuje v rámci zobrazování textů a dialogů, které si vývojář musí komplikovaně naprogramovat sám. Například v kapitole 6.8 je popsáno, že pro příběhový monolog je potřeba vyřešit řadu problémů, jako například přidat text do fronty, odebrat entitám pohybovou komponentu, ukončit hráčův pohyb, odebrat hráči joystick[3] a další.

Po dokončení práce bych přidal další důležitou podmínku pro výběr enginu: zjistit, zda obsahuje veškerou funkcionalitu, kterou chci ve hře použít.

2.4 Programovací jazyky podporující SpriteKit

Hry na platformě SpriteKit mohou být psány ve dvou programovacích jazycích: Objective-C a Swift.

2.4.1 Objective-C

Objective-C je objektově orientovaný programovací jazyk, který rozšiřuje programovací jazyk C o zasílání zpráv ve stylu jazyka Smalltalk. Jazyk podporuje dědičnost, polymorfismus a zapouzdření. Obsahuje koncepty, jako jsou kategorie a rozšíření, jež umožňují přidávat již definované třídě nové metody. Kategorie, na rozdíl od rozšíření, umožňuje přidat novou metodu na jakoukoliv třídu i tu, od níž nemáme zdrojový kód (již zkompilované). Další koncepty jsou protokoly (interface) a uzávěry.

Od roku 2001 byl Objective-C standardní programovací jazyk používaný, podporovaný a propagovaný společností Apple pro vývoj aplikací pro macOS a následně pro iOS až do zavedení programovacího jazyka Swift v roce 2014 [5]. Za dobu existence Objective-C vzniklo velké množství návodů, knihoven i fór plných otázek a odpovědí něj orientovaných.

Nevýhodné na Objective-C je to, že nepodporuje jmenné prostory (namespaces), proto se vývojář v kódu často setkává s prefixem `NS`.

2.4.2 Swift

Swift je objektově orientovaný programovací jazyk speciálně vyvinutý pro Apple-platformy zahrnující prvky funkcionálního programování. Inspirace moderními jazyky přináší Swiftu funkcionalitu, jako jsou generika, tuply, odvozování typu, přetěžování operátorů, typ optional a další. Jazyk zachovává klíčové koncepty Objective-C, včetně protokolů (interface), uzávěrů a kategorií, často nahrazuje dřívější syntaxi novou a jednodušší.

Swift vznikl v roce 2014 a oproti Objective-C obsahuje jmenné prostory, REPL (read-eval-print-loop), dynamické knihovny a je open source. Oba zmíněné jazyky mohou být použity v jednom projektu. Pro použití Swift-třídy v kódu Objective-C je potřeba ji označit atributem `@objc`. U opačné kompatibility se postupuje přes takzvaný bridging header.

[3] Joystick (páčkový ovladač) je vstupní zařízení používané zejména pro ovládání her.

2.4.3 Výběr programovacího jazyka

Výhodami Objective-C jsou kompatibilita s nižšími iOS-verzemi, než je iOS 7, což Swift nenabízí, značné množství návodů a knihoven i velká pravděpodobnost, že kód aplikace nebude nutné migrovat, jelikož nové aktualizace jazyka Objective-C už neexistují. Mezi nevýhody Objective-C patří nutnost naučit se práci s pointery, složitější syntaxe ve srovnání se Swiftem[4]. Objective-C nepodporuje jmenné prostory, nemá REPL a společnost Apple ho už dále nevyvíjí, ale soustředí se na vývoj Swiftu.

K výhodám jazyka Swift se řadí jeho jednoduchá syntaxe, kompaktnost kódu oproti Objective-C, kde „*lze přepsat kód Objective-C do jazyku Swift a zmenšit projekt až na třetinu jeho původní velikosti*“ [6]. Další z výhod jsou nová funkcionalita, zmíněná v předchozí kapitole 2.4.2, aktualizace a podpora od společnosti Apple. Od Swiftu verze 5 má jazyk zpětnou kompatibilitu pro předchozí verze Swiftu a přepracovanou dokumentaci.

Oba jazyky jsou zkompilovány do stejného strojového kódu, jejich výkon je tedy identický, v určitých případech Swift dokonce vykazuje vyšší rychlost [7][8]. Swift je, na rozdíl od Objective-C, open source, čímž umožňuje vylepšování jazyka pomocí komunity nezávislých vývojářů. Swift vyřešil zásadní problém Objective-C, a to jmenné prostory. Stejný výkon, kompatibilita s Objective-C, srozumitelnější syntaxe a chytřejší kompilátor, který omezuje možnost potenciálních chyb, a tím snižuje bezpečnostní rizika aplikací, poskytují jazyku Swift významnou výhodu při rozhodování vývojáře mezi zmíněnými jazyky.

Tato práce obsahuje kód primárně v programovacím jazyce Swift, který jsem si vybral díky již zmíněným výhodám a jeho moderní funkcionalitě, syntaxi bez pointerů a jednoduššímu učení. V průběhu vývoje jsem narazil na problém, který nebyl řešitelný v jazyce Swift, a pro jednu funkcionalitu jsem použil atribut jazyka Objective-C.

2.5 Vývojové prostředí

Vývojové prostředí (IDE) je software pro zjednodušení vývoje aplikací. Pro vývoj her v jazyce Swift na frameworku SpriteKit se nabízí několik vývojových prostředí usnadňujících vývoj hry. Nativně je to XCode společnosti Apple, dále AppCode od společnosti JetBrains a editor CodeRunner od vývojáře Nikolaie Krilla.

2.5.1 CodeRunner

Jednoduchý vícejazyčný programovací textový editor IDE CodeRunner je výborný na menší kusy kódu a jednoduché aplikace. Toto IDE je vhodné v případě, kdyby vývojář chtěl přeskakovat mezi několika jazyky, nebo mezi menšími projekty. Nicméně pro tuto bakalářskou práci se příliš nehodí, jelikož je stavěný primárně pro menší projekty, proto neobsahuje zásadní funkcionalitu, jako jsou vizuální editor a migrace kódu na novou Swift-verzi, což XCode obsahuje.

[4]Ukázka složitější syntaxe Objective-C: http://goshdarnblocksyntax.com/

2.5.2 AppCode

AppCode je velice pokročilé IDE, které má ve srovnání s XCodem celou řadu výhod, například chytřejší doplňování kódu, lepší unit-testování, intuitivní navigace a pokročilé refaktorování kódu. I přes všechny silné stránky má AppCode jednu velkou nevýhodu v porovnání s XCodem. Spočívá v neexistujícím editovacím uživatelském prostředí [9]. V AppCodu nemůžeme zobrazit a editovat soubory storyboard, scény a další vizuální prvky. Scéna představuje velice důležitý prvek SpriteKitu, absence možnosti vizuálně editovat scény činí vývoj hry na platformě SpriteKit náročným. Další nevýhodou AppCodu je také potřeba instalace XCodu pro kompilaci a linkování.

2.5.3 XCode

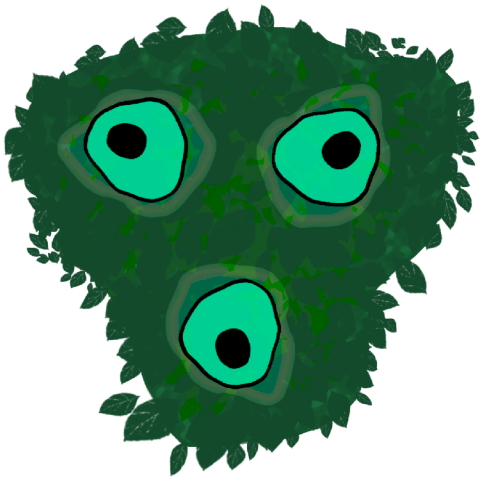
Nativní vývojové prostředí XCode společnosti Apple je určeno pro vývoj aplikací na Apple zařízení [10]. Obsahuje všechno, co je potřeba pro vývoj, testování a distribuci na všechny Apple platformy. Zároveň má funkcionalitu, kterou u IDE očekáváme: možnost refaktorizace kódu, označení chyb před spuštěním aplikace (statická analýza), navigace v projektu a chytré doplňování kódu.

Největší výhodou XCodu oproti ostatním IDE je možnost zobrazení a editace vizuálních prvků (storyboard, scéna a další), které se při vývoji hry ve SpriteKitu často používají. Nativní vývojové prostředí XCode tedy představuje jednoduchou volbu pro vývoj této práce, popřípadě lze využít výhody AppCodu a přepínat mezi IDE.

2.6 Designování hry

Cílem práce je vytvoření 2D hry, což znamená, že je potřeba dodat grafiku pro úrovně a herní prvky. Tuto část jsem se rozhodl vytvořit sám, jelikož tím mohu přidat příběhu hloubku, graficky ho doplňovat a sladit herní postavy s grafikou úrovní. Grafika hry byla vytvořena a upravena v softwaru Adobe Photoshop a GIMP. Exportované 2D obrázky neboli sprity (ty si přesněji definujeme v kapitole 3.2) jsem poté umístil do hry editorem scén, který nabízí XCode, konkrétně to popisuji v kapitole 6.1.

Dalšího pozvednutí zážitku ze hry lze dosáhnout pomocí hudebního doprovodu. Ten jsem také vytvořil sám a použil k tomu software Logic Pro X společnosti Apple a FL Studio společnosti Image-Line. Spolu s grafickou částí lze hráče takzvaně vtáhnout do hry. Prostředí úrovní, vzhled postav a hudba se postupem hry vyvíjí, stejně jako interakce herních postav s hráčem. Hru, stejně jako její design, podrobněji popisuji v kapitole 5. Ukázka vytvořené grafiky pro NPC postavu je na obrázku 1.



Obrázek 1: Grafika 2D NPC postavy použité v této hře.

3 SpriteKit

SpriteKit je společností Apple oficiálně podporovaný framework usnadňující vytváření 2D her pro Apple zařízení [11] [12]. Nabízí jednoduché programovací rozhraní, kde vývojář může rozšiřovat funkcionalitu enginu, čímž umožňuje snadný vstup do světa herního vývoje, bez zbytečné složitosti velkých herních enginů a bez nutnosti zdlouhavého učení. SpriteKit obsahuje veškerou základní funkcionalitu od vykreslování tvarů, obrázků a animací přes fyzikální simulace, tvorbu dlaždicových map až po částicové a texturové efekty.

Framework je zakomponován přímo do vývojového prostředí XCode, díky čemuž je ihned připraven k používání a nevyžaduje žádné další instalace ani nastavování. Ve vývojovém prostředí XCode je možné zobrazit a editovat scény, které reprezentují kolekci herních objektů. Herní design tak může být vytvořen pomocí vizuálního editoru a být doplněn kódem, který přidává hratelnost.

SpriteKit je dobře doplňován frameworkem GamePlayKit, který se používá pro implementaci hratelnosti a herní logiky, například při hledání cesty, rozhodování pomocí stavového stroje, umělé inteligence a chování NPC[5]. SpriteKit lze kombinovat i s frameworkem SceneKit, který je určen pro vývoj 3D her, například pro zobrazení 2D tlačítek.

3.1 Uzel

Uzly (`SKNode`) představují základní stavební kameny SpriteKitu. Každý herní objekt je reprezentován jako uzel, který je zároveň kontejnerem, do něhož lze umístit další uzly jako potomky. Uzel má souřadnice popisující, kde se ve scéně nachází, a souřadnice spolu s dalšími vlastnostmi sdílí se svými potomky.

Uzly mohou mít vizuální podobu a také fyzické tělo, které může být ovlivněno

[5]NPC je postava ve video hře, kterou neovládá hráč, ale počítač.

gravitací, nebo interagovat s dalšími uzly a jejich fyzickými těly. Dále lze uzlům definovat, jak budou reagovat na vstupy uživatele a spouštět akce jako pohyb na určené souřadnice. Všechny datové zdroje na obrazovce jsou instancí `SKNode` nebo jejich podtřídou. Uzly samy o sobě však neposkytují žádný vizuální obsah. Veškerý vizuální obsah je vykreslován pomocí jedné z mnoha předdefinovaných podtříd `SKNode`.

Uzly jsou hierarchicky uspořádány do stromů uzlů. Nejčastěji je kořen stromu definován uzlem scény (`SKScene`) a dalšími uzly jako jeho potomky. Každý uzel ve stromu uzlů poskytuje svým potomkům souřadnicový systém.

3.2 Sprite

Sprite je dvojrozměrný obraz nebo animace integrované do scény. Jedná se o typ počítačové grafiky, která se často používá ve videohrách. Ve dvojrozměrné hře se sprity používají k reprezentaci postav, objektů i dalších prvků. Animovaný sprite se skládá ze série obrazů, které jsou přehrávány za sebou, aby vytvořily iluzi pohybu neboli animaci. Ukázka spritu lze vidět na obrázku 1.

Sprity se také využívají k vytvoření speciálních efektů, například výbuchů, kouře, ohně a dalších. Sprity se často používají k vytvoření pocitu hloubky ve hře, protože mohou být vrstveny a kombinovány, aby vytvořily 3D efekt. Kromě použití ve videohrách se sprity používají ve webovém designu i dalších interaktivních médiích.

3.3 Renderování

Vykreslování scény se provádí v nekonečné smyčce, kde se vyhodnocuje veškerá logika, simuluje fyzika, reakce na vstupy uživatele, provádí se veškeré akce a následně se vykreslí snímek. Obnovovací frekvence pro všechna Apple-zařízení je 60 snímků za vteřinu. Optimalizované hry pak mohou běžet v této frekvenci a uživatelům přinést hladký průběh hry. Metody pro správu vykreslování a manipulaci s uzly se nachází ve scéně (`SKScene`). Průběh vykreslování scény ovšem není možné přímo změnit. Každý snímek je vykreslován v následujícím cyklu:

1. v `update` funkci se aplikuje herní simulace; ta se skládá z herní logiky, aplikace změn běžících uzlů, umělé inteligence, řešení vstupů a dalších,
2. scéna začne procházet všechny akce uzlů ze stromu uzlů, a jestliže najde běžící akce, pak je aplikuje,
3. metoda `didEvaluateActions` se zavolá po provedení všech akcí pro daný snímek,
4. scéna simuluje fyziku pro všechny uzly, které mají fyzické tělo, a aplikuje na ně fyziku definovanou ve třídě `SKPhysicsBody`,
5. zavolá se metoda `didSimulatePhysics`, po simulaci všech fyzických těl,

6. scéna použije všechna omezení spojená s uzly ve scéně; omezení se dá použít pro navázání vztahu mezi jednotlivými uzly ve scéně,

7. následně se zavolá metoda `didApplyConstraints` po provedení předchozího kroku,

8. scéna zavolá metodu `didFinishUpdate`, kde lze provést poslední změny před vykreslením snímku.

9. scéna je vykreslena.

3.4 Scéna

SpriteKit používá hierarchickou strukturu pro organizaci herních prvků a definování jejich chování. Základem hierarchie je scéna (`SKScene`), která je kontejnerem pro všechny herní prvky neboli uzly a definuje globální nastavení hry. Scéna tedy zobrazuje a spravuje sbírku uzlů. Obsahuje také metody pro aktualizaci a vykreslování scény, stejně jako metody pro zpracování vstupu uživatele. Scény můžeme vytvářet jak programováním, tak skrze „*drag and drop*“[6] editor scén, zabudovaný ve vývojovém prostředí XCode. Najednou lze zobrazit pouze jednu scénu. Pro přechod do další herní úrovně je potřeba načíst novou scénu.

3.4.1 Načítání scén

Pro načtení, případně přechod do další scény se používá třída SKTransition. Ta nejprve pozastaví scénu, použije námi zvolený animovaný přechod a následně po dokončení animace načte novou scénu. Hry se často skládají z více scén. I jednoduché hry mívají většinou alespoň dvě scény, jednu pro menu hry a druhou pro hru samotnou.

3.4.2 Akce

Přidáním akcí (`SKAction`) uzlům lze do hry vznést život, a ta se tak stává více interaktivnější. Akce umožňují ze statické hry udělat dynamickou vykonáváním pohybu, otáčení, či změnou tvaru uzlů, dále přehráváním zvuků, nebo vykonáním akce, kterou si vývojář nadefinuje. U určitých akcí lze nastavit dobu, po kterou budou vykonávány, a také jestli se po skončení akce spustí blok kódu.

Více akcí lze přidat do sekvence, a vytvořit tak například složitou animaci. Sekvenci můžeme spustit postupně, nebo najednou (akce se provedou paralelně). Uzlům můžeme přidat pozorovatele, případně delegáta (`SKSceneDelegate`)), jejichž pomocí můžeme detekovat změny v každém kroku akce. Přesnou kontrolu

[6] drag and drop je operace používaná v grafickém uživatelském rozhraní, kdy uživatel v počítači „uchopí“ pomocí ukazovacího zařízení virtuální objekt a přesune ho „přetažením“ na jiné místo.

nad akcemi získáme nastavením atributu jména akce, přes který můžeme přistoupit k akci. Atribut jména je pro přístup k akci lepší, protože je méně paměťově náročný než přístup přes instanci akce.

3.5 Vizuální reprezentace uzlů

Jak už bylo zmíněno v dřívější kapitole 3.1, uzly samy o sobě neposkytují žádný vizuální obsah. Pro vykreslení objektu na scénu je nutné vybrat vhodnou třídu uzlů. Mezi základní třídy pro vykreslení objektů se řadí `SKSpriteNode`pro sprity neboli obrázky, `SKShapeNode` pro geometrické tvary, `SKLabelNode` pro text a na speciální efekty existuje třída `SKEmitterNode`.

3.6 Simulace fyziky

Pro nastavení fyzické hranice hry, gravitace, nebo možnost interagovat s uzly podobně jako s objekty ve fyzickém světě je potřeba simulovat fyziku. Pro aktivaci simulování musíme povolit fyziku ve scéně (`SKPhysicsWorld`) a uzlům přidat objekt fyzické tělo (`SKPhysicsBody`). Uzlům můžeme nastavit hmotnost, díky které pak reagují rozdílně na další fyzické objekty a gravitaci. V základu máme tři typy fyzických těl: dynamické, statické a hranu.

Dynamické tělo	Statické tělo	Hrana
Objekt s objemem a hmotností	Statický objekt s hmotností a objemem	Statický objekt bez objemu
Může být ovlivněn gravitací, sílou a kolizí	Není ovlivněn gravitací, sílou ani kolizí	Stejně jako statické tělo
Využití může být například pohyblivé NPC	Využití může být například zeď v bludišti	Využití je především pro ohraničení scény

Tabulka 1: Druhy fyzických těl a jejich základní rozdíly.

3.6.1 Ohraničení fyzického těla u spritu

Fyzické tělo spritů můžeme nastavit pomocí čtyř základních možností, které nám nabízí SpriteKit. Volit můžeme mezi kruhovým a obdélníkovým tělem, nebo si nadefinovat body (`CGPoint`)) mnohoúhelníku, které poslouží jako ohraničení fyzického těla. Jako poslední možnost máme alfa masku, u které je dobré vědět, že pro textury obsahující spoustu hran bude výpočetně náročná, a tudíž bude mít negativní dopad na plynulost hry. Pro přesné nastavení masky je důležité, aby textura měla alfa kanál neboli kanál udávající průhlednost pixelu. Díky alfa kanálu může SpriteKit najít hrany textury a aplikovat na ni alfa masku.

Za účelem optimalizace výkonu je vhodné se zaměřit na základní tvary, jako jsou kruh či obdélník, případně si vytvořit mnohoúhelník pro přesnější ohraničení textury.

3.6.2 Pokročilé nastavení fyziky

Rozdílné interakce mezi uzly dosáhneme pomocí bitové masky a delegáta fyzického kontaktu (`SKPhysicsContactDelegate`). Bitová maska je zjednodušeně jedinečná identifikace uzlu, případně skupiny uzlů. Pomocí identifikátoru bitové masky můžeme například definovat různé typy akce, která se provede při kolizi mezi uzly. Jako jednoduchý příklad může sloužit kolize hráče s NPC, jež způsobí konec hry, kolize NPC s NPC ovšem nevyvolá žádnou akci, pouze se od sebe odrazí.

4 GameplayKit

Vytváření her běžně vyžaduje algoritmy k řešení základních problémů, jako jsou například navigace NPC v herním světě, rozhodovací stromy a autonomní pohyb NPC, které tvoří základ běžných herních mechanik.

GameplayKit je framework obsahující tyto základní herní mechaniky a lze ho použít k přidání nové hratelnosti do hry založené na SpriteKitu [13]. GameplayKit poskytuje sadu nástrojů a algoritmů, které vývojářům pomáhají vytvářet dynamický herní zážitek.

Nedílnou součástí GameplayKitu je Entity Component System, obsahující entity, komponenty a systémy. Entity jsou objekty ve hře mající vlastnosti a chování. Komponenty jsou části dat připojené k entitám a definují jejich chování. Systémy jsou logikou řídící vzájemnou interakci entit. Popisu Entity Component System se podrobněji věnuje následující podkapitola 4.1.

Vývojář může použít platformu GameplayKit k vytvoření agentů AI. To hře umožňuje mít postavy NPC, které mohou interagovat s hráčem a prostředím na základě jejich definovaných cílů a chování.

GameplayKit umožňuje vytvoření stavového stroje. Jedná se o soubor pravidel, která definují, jak bude hra přecházet z jednoho stavu do druhého. To nabízí možnosti, aby hra měla více úrovní nebo stavů, jimiž může hráč postupovat.

4.1 Entity Component System

Entity Component System (ECS) je návrhový vzor používaný při vývoji her umožňující oddělit herní logiku od samotných herních prvků neboli objektů [15] [16]. Jedná se o způsob organizace herních objektů a jejich dat do dílčích částí, které mohou být opakovaně použity a kombinovány různými způsoby.

Ve svém jádru se ECS skládá ze tří hlavních částí: entit, komponent a systémů. Entity jsou základním stavebním kamenem ECS. Jde jednoduše o identifikátory, které lze použít k odkazu na herní objekty. Komponenty jsou data přidružená

k entitám. Obvykle se jedná o datové struktury obsahující informace o entitě, jako jsou například její životy, rychlost nebo stav. Systém modifikuje entity skrze aktualizace jejich komponent a provádí herní logiku.

Oddělením herní logiky od herních objektů mohou vývojáři snadno vytvářet nové entity a komponenty, aniž by museli přepisovat existující kód, což usnadňuje přidávání nových funkcí a provádění změn.

ECS usnadňuje ladění a optimalizaci hry, a to díky modulárnímu systému, jehož pomocí mohou vývojáři snadno identifikovat a opravit problémy, aniž by museli prohledávat značnou část kódu.

4.1.1 Propagace dat v Entity Component System

Jakmile jsou komponenty přidány do entity, lze manipulovat s entitou i jejími daty prostřednictvím systémů. Tok dat ECS můžeme rozdělit do následujících kroků:

- systém: naslouchá vnějším událostem a posílá aktualizace komponentám,
- komponenta: naslouchá systémovým událostem a poté aktualizuje svůj stav,
- entita: získává chování prostřednictvím změn stavů jejich komponent.

Jestliže hráč narazí svou postavou do postavy NPC, pak na to NPC může reagovat. Systém fyzického kontaktu detekuje kolizi hráče s NPC a aktualizuje komponentu zpětné vazby entitě NPC. Komponenta zpětné vazby připraví text, jako reakci na kolizi. Systém vykreslování připravený text převezme a vykreslí jej.

4.2 Agenti, cíle a chování

Agenti, cíle a chování jsou tři nástroje, které framework GamePlayKit poskytuje pro vytváření pohybové „inteligence“ NPC.

Agenti (`GKAgent`) jsou objekty ve hře, které se mohou pohybovat a interagovat s prostředím. Mají vlastnosti, jako jsou poloha, rotace, rychlost, a mohou mít také cíle a chování s nimi spojené. `GKAgent` je komponenta, proto se dá používat v architektuře Entity Component System.

Cíle (`GKGoal`) jsou úkoly, které se agenti pokouší splnit. Ty mohou být jednoduché, jako je dosažení určité pozice, nebo složitější, například vyhýbání se překážkám nebo nalezení nejkratší cesty k objektu.

Chování (`GKBehavior`) je kolekce cílů agenta ovlivňující jeho pohyb v herním světě. Cíle mohou mít různou váhu, podle níž se určuje jejich priorita. Využití cílů s váhou je například, když je potřeba změnit chování NPC na základě vzdálenosti od hráče.Cíle s kratší vzdáleností mají nastavenu vyšší váhu, a tedy i vyšší prioritu, proto jsou důležitější oproti cílům reagujícím na delší vzdálenost hráče vůči NPC. Skrze kombinování více cílů lze simulovat komplexní chování, které vypadá přirozeně, například pohyb skupiny agentů.

5 Úvod do hry

Výsledek této práce představuje 2D hra typu „adventure“[7] pro operační systémy iOS a iPadOS. Cíl hry spočívá v procházení příběhem pomocí interakcí s NPC a řešení hádanek v jednotlivých úrovních. Hráč kromě základního popisu ovládání joysticku a menu neobdrží žádný návod, jak postupovat dále. Veškeré poznatky pro přecházení do dalších úrovní získává hráč interagováním s NPC a nasloucháním příběhu. Příběh předává hráči úkoly a je veden formou rozhovoru mezi hlavní postavou hry a NPC. Práce také obsahuje vedlejší rozhovory ovlivňující hratelnost hry, pokud si je hráč vyslechne. Hra je dynamická a reaguje na činnosti hráče i jeho interakci s prostředím.

Postup se automaticky ukládá při dokončení úrovně a při novém spuštění je možné začít od posledního uloženého místa. Hra obsahuje menu, výběr úrovně, nastavení ovládání i doby zobrazení textu a nápovědu.. Rozhovory s NPC a herní design se postupem do nižších úrovní mění směrem k negativnímu tónu. Interakce s NPC se stávají méně informativní a občas až nepřívětivé k hlavní postavě. Úroveň ze hry lze vidět na obrázku 2.

Obrázek 2: Rozhovor s NPC v první úrovni hry.

5.1 Příběh

Základem hry je vymyšlený příběh se dvěma možnými konci odehrávající se v nespecifikovaném podvodním světě, kde je hráči zadán hlavní úkol: nalezení tajné pralátky pro dobro světa a sjednocení života. Postupem hry se příběh rozvíjí a nabízí nové informace pro zodpovězení otázek, například kam má hlavní postava jít, k čemu pralátka opravdu slouží, a zda je jeho výprava vůbec pro dobro světa.

[7] Adventure (dobrodružství) je žánr videoher, ve kterém hráč prozkoumává svět za pomoci interaktivního příběhu a řešení hádanek.

Příběh se zaměřuje na bystrost hráče a jeho zamyšlení se nad informacemi, jež mu sdělují NPC-postavy, proto může příběh končit dvěma způsoby. První konec nastane, pokud si hráč v hlavě poskládá puzzle z informací, které se v průběhu dozvěděl, nebo také díky rychlé analýze „kryptického“ příběhu v poslední úrovni. Monolog NPC má slova uspořádána naopak, proto se postava nazývá Reverso a je několikrát zmiňována v příbězích. Přečtením monologu z opačné strany získá hráč informace pro ukončení úrovně prvním způsobem. Druhou možností ukončení příběhu je následování hlavního úkolu, jenž byl hráči zadán na začátku hry.

Hra obsahuje i vedlejší rozhovory doplňující hlavní příběh a přinášející užitečné informace o jednotlivých úrovních. Některé vedlejší rozhovory lze aktivovat pouze při splnění určitých podmínek.

5.2 Interakce

Interakce s NPC je základní mechanismus, jehož pomocí může hráč získat důležité informace pro postup do dalších úrovní, či nápovědu k řešení aktuální hádanky. Interakce se aktivují pomocí kolize s NPC, nebo pokud se hráč nachází v jeho blízkosti, kdy se poté spustí například vyprávění příběhu. Určité herní postavy mohou také hráči napovědět, na co si dát pozor v následující nebo aktuální úrovni, jak rychleji postoupit hrou, a varovat před blížícím se nepřítelem.

Reakce může být i opačná, kdy NPC reaguje na rozhodnutí provedené hráčem. Například při nesplnění zadaného úkolu může být NPC nepřátelské a neposkytne hráči indicie pro postup do další úrovně. Hra směřuje hráče k tomu, aby vyhledával interakci s herními postavami, jelikož je poté odměňován informacemi pro jednodušší postup úrovněmi.

Rozhovory s NPC mohou také rozšířit hratelnost hry, například přidáním rozhodovacího dialogu pro hráče v následující úrovni.

5.3 Druhy NPC

Ve hře se můžeme setkat se čtyřmi druhy NPC, dva z nich jsou přátelské a dva nepřátelské. Všechny herní postavy jsou agenti, kteří mají cíle a chování podrobněji popsány v kapitole 6.2. Příběhové NPC má jako hlavní cíl sdělit hráči podstatné informace, a také mu může zadávat úkoly. Další NPC je typu průzkumník, jenž má za úkol brouzdat po mapě a sloužit jako výzvědná jednotka. Průzkumník může varovat hráče, že se poblíž nachází nepřátelské NPC, a také oznamuje menší nápovědy ohledně úrovně, výjimky představují pozdní úrovně hry, kde průzkumník působí primárně pasivně agresivně.

Nepřátelskými postavami jsou hledač a kamikadze. Obě NPC pasivně brouzdají po mapě, a pokud se hráč nachází v jejich dosahu, útočí. Hledač má za cíl narazit do hráče, který po kolizi zemře. NPC následuje hlavní postavu až do jejího zastavení. Stejný cíl má i kamikadze, jenomže namísto následování se katapultuje a použije přímý útok vystřelení sebe na hráče. Poté se nějakou dobu regeneruje, než může použít útok znova. Nepřátelská NPC mají nastavenu nej-

vyšší prioritu pro vyhýbání se přátelským NPC. Tuto funkcionalitu může hráč využít a použít ji jako strategii pro odehnání nepřátel.

5.4 Úrovně

Celkově je ve hře deset úrovní a úvod pro seznámení s ovládáním. Úrovně hry se člení na dva typy: příběhově interakční a úkolové. Příběhová, jak vyplývá z názvu, je úroveň určená pro vyprávění hlavního a vedlejšího příběhu. Příběhy také poskytují informace pro zdolání následující úrovně a některé obsahují nápovědu k poslední fázi hry.

Úkolové obsahují logické prvky a očekávají aplikování informací získaných v příběhové úrovni. Typy úloh jsou „projdi bludištěm“, „najdi a přines“, „najdi, posbírej a aplikuj“ a jejich variace. V určitých úrovních může být mírně pozměněna hratelnost, pokud hráč splnil vedlejší úkol, nebo si poslechl vedlejší příběh. Jednu úroveň lze zcela přeskočit při splnění prvního vedlejšího úkolu.

5.5 Herní design

Hra následuje příběh a postupně mění své úrovně od barevných tónů přes NPC interakci až po hudební doprovod. Herní design začíná jednoduše barvitě a pozitivně a krok za krokem přechází do tmavší, složitější až nehostinné podoby. Tento způsob jsem si vybral, jelikož by měl reprezentovat emoce hlavní postavy z nových informací a hledání dávno ztraceného prvku v nejtemnějších oblastech oceánu.

5.6 Menu

Pro hladší hratelnost a reakci na spuštění, pozastavení, či přerušení hry jsou zde menu. V této práci se můžeme setkat s hlavním menu, nastavením, menu pozastavení a výběru úrovně. Hlavní menu se zobrazí po každém novém spuštění hry a obsahuje tlačítka pro spuštění poslední uložené úrovně, vynulování postupu ve hře, nastavení a výběru úrovně.

Hru lze pozastavit nebo přerušit například zobrazením domovské obrazovky zařízení, či příchozím hovorem. Po návratu do hry se zobrazí menu pozastavení, které obsahuje tlačítka pro pokračování ve hře, restart úrovně, nastavení, nápovědu a možnost odejít do hlavního menu.

Ovládání hry si může hráč přizpůsobit a zvolit si dynamický, či statický joystick a dobu, jak dlouho bude na obrazovce zobrazován text. Statický joystick je umístěn v levém dolním rohu obrazovky a dynamický znamená, že se joystick zobrazí v bodě dotyku na obrazovku. Pokud si hráč v aktuální úrovni neví rady, může použít nápovědu, která je pro každou úroveň jedinečná a zobrazuje, co by měl hráč zkusit pro další postup.

6 Programátorská dokumentace

V této kapitole se čtenář postupně seznámí s používáním a implementací hlavních částí hry. Kapitola je uspořádána do pořadí, které mi po dokončení práce připadá vhodné pro začínající vývojáře her ve SpriteKitu. Budou popsány části hry jako používání scén a akcí, ovládání hlavní postavy, práce s kamerou, ECS, přerušení hry a chování NPC (agenti, cíle a chování). Zmíním také problémy, se kterými se může čtenář při vývoji her setkat, jak je řešit, a také se předvede testování hry.

6.1 Editor Scény

Základem každé části hry je scéna (`SKScene`), popsaná v kapitole 3.4, navržená ve vizuálním editoru a propojená s vlastní třídou scény, kde je definována její logika. Pro spojení je nutné v editoru pod záložkou „Ukázat inspektora vlastní třídy“ zadat název vlastní třídy. V editoru můžeme na scénu přidávat objekty, jako jsou světla, tvary, sprity, texty a další. Vývojář se nejčastěji setká s objekty spritu (`SKSpriteNode`), jež přidá na scénu v kódu, anebo v editoru. Objektům lze v editoru přidávat vlastnosti (například název uzlu, rozměry), a taktéž je možné objekt propojit s vlastní třídou.

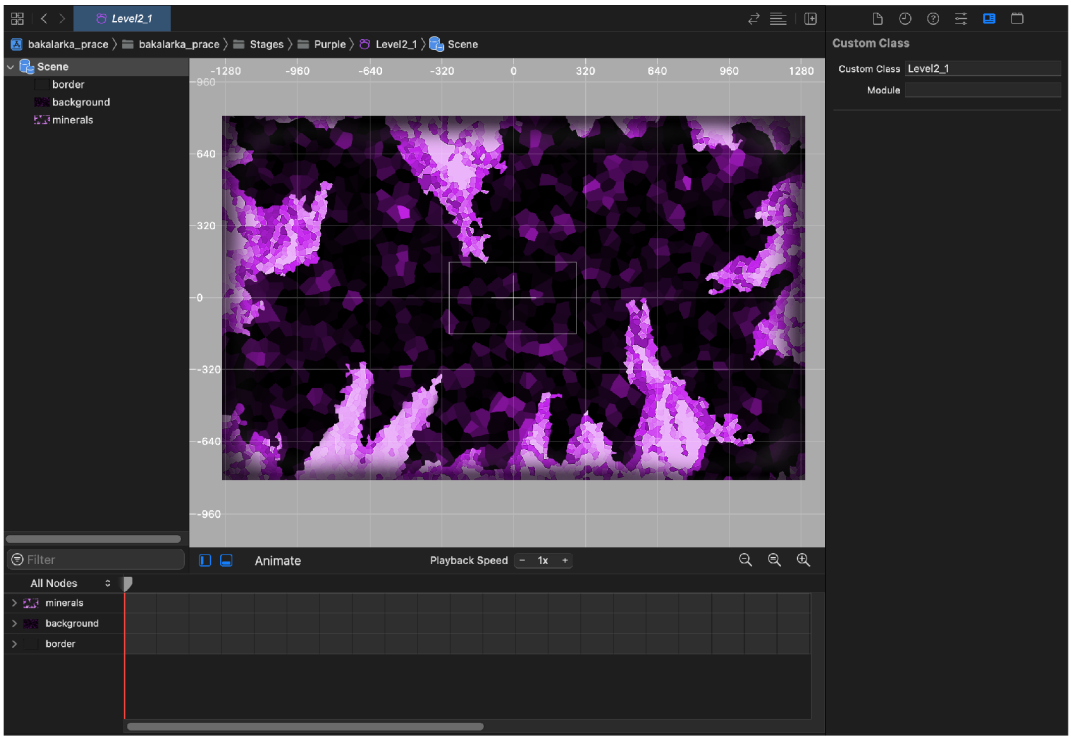
Objektům, jako jsou pozadí úrovně nebo tlačítka menu, upravuji v editoru velikost a pozici pro následnou jednodušší práci v kódu a vizuální seznámení se scénou. Aby bylo možné s tímto obsahem dále v kódu pracovat, je nutné ho přidat přes metodu `childNode`. Pro přidání nového objektu se používá metoda `addChild`. Editor scén je demonstrován na obrázku 3.

6.2 GameScene

Třída GameScene, je implementace scény (`SKScene`), která obsahuje veškerou základní funkcionalitu pro všechny úrovně. Jmenovitě jde o pohyb s hlavní postavou a kamerou, zobrazování textů na scéně, menu pozastavení, nastavení, nápovědy a ostatní funkcionalitu, například řešení oznámení o přerušení hry. Tyto funkcionality se vyskytují podrobněji v dalších kapitolách programátorské dokumentace.

Základní vlastnosti třídy `GameScene`:

`playerNode: SKSpriteNode?` – sprite uzel hlavní postavy neboli hráče
`background: SKSpriteNode?` – sprite uzel pozadí, který se používá pro pohyb kamery
`cameraFocusActions: Queue` – fronta akcí pro pohyb kamery
`storyTellingActions: Queue` – fronta akcí pro postupné zobrazování příběhu na scénu
`helpBox: HelpBox?` – zobrazování nápovědy. Nastavováno v každé úrovni zvlášť
`changingLevel: Bool` – indikování, zda se mění úroveň



Obrázek 3: Editor scén ve vývojovém prostředí XCode.

Základní metody `GameScene`:

`updateStoryText(with text: String, around: SKNode) -> SKLabelNode`
– metoda naplňuje frontu `storyTellingActions` příběhovým textem `text` a volá `focusOnNode` s uzlem `around`

`focusOnNode(node: SKNode, timeToFocusOn: TimeInterval)`
– metoda přidá pokyn pro přesun polohy kamery na uzel `node` do fronty `cameraFocusActions` spolu s časem `timeToFocusOn` jak dlouho má zůstat zaměřená na uzel

`waitAndRun<T>(delay: Double, function: @escaping () -> T)`
– metoda spustí funkci escaping asynchroně s prodlením delay

`pauseGame()`
– metoda pro pozastavení hry

`resumeGame()`
– metoda pro znovuspuštění hry, která byla pozastavena

`@objc applicationWillResignActive()`
– metoda Objective-C řešící událost přerušení hry (viz. Kapitola 6.9)

`@objc applicationDidBecomeActive ()`
– metoda Objective-C řešící opětovné zobrazení hry (viz. Kapitola 6.9)

6.2.1 Základní funkcionalita třídy GameScene

Při načtení scény do zobrazení se jako první zavolá metoda `didMove`, v níž se nastavuje scéna pro spuštění. V této metodě se na scénu přidávají kamera (`SKCamera`), joystick, menu pozastavení, inicializuje se simulace fyziky (`SKPhysicsWorld`) a připravují se textové objekty (`SKLabelNode`) pro pozdější použití.

Reakce na hráčův dotyk na obrazovku spravuje metoda `touchesBegan`. Zde se zjistí, čeho se hráč na scéně dotkl, a zda se má otevřít menu, nebo zobrazit joystick. Hned poté se zavolá metoda `touchesMoved`, jež reaguje na pohyb dotyku na obrazovce. Je potřeba myslet na to, že i obyčejný dotyk bez pohybu zavolá metodu `touchesMoved`. V této metodě řeším pohyb joysticku, ten více popisuji v kapitole 6.5. Nakonec se zavolá metoda `touchesEnded`, indikující konec dotyku, což je v této práci znamení pro ukončení pohybu joystickem.

Vykreslování a neustálá aktualizace herních prvků probíhají v metodě `update`. Tady je řešen pohyb kamerou a hráčem, aktualizace entit, kontrola, je-li hra pozastavena, a spravuje se postupné zobrazování příběhu na scéně. Kontakt mezi fyzickými těly spravuje metoda `didBegin`, kterou si každá úroveň implementuje samostatně. Nastavení fyziky a řešení kolizí více popisuji v kapitole 6.4.

6.3 Používání akcí

Akce (`SKAction`) se ve hře používají pro pohyb kamerou, plynulé zobrazování textu a na animace. Každý uzel (`SKNode`) může metodou `run` spustit akci. Většina akcí obsahuje nastavitelný parametr `duration`, označující dobu, po kterou se provádí. Metoda `sequence` třídy `SKAction` vytvoří z pole objektů `SKAction` jednu složenou akci. Tato složená akce je typu `SKAction`, dá se tedy spustit metodou `run`, ve které lze specifikovat, zda se spustí postupně za sebou, nebo najednou (paralelně).

Metodou `run` se spouští akce. U této metody lze specifikovat, zda se po dokončení akce spustí blok kódu, který se předává parametrem `completion`. Uživatel frameworku by očekával, že se kód provede až po skončení doby jejího trvání, nicméně opak je pravda. Akce se považuje za ukončenou v době, kdy je volána, bez ohledu na nastavený parametr `duration`, či jestli jde o akci složenou a pořád se vykonává. Tento problém lze částečně vyřešit pomocí kontroly, zda uzel má nějaké běžící akce, metodou `hasActions`. Nebo lze přesně přistoupit k akci metodou `action`, jež vrací akci, pokud je přítomna v daném uzlu, jinak vrací `nil`.

6.4 Nastavení a simulace fyziky

Pohyb hráče a NPC je řešen pomocí fyzického nastavení, díky němuž se dá simulovat plynulý pohyb. Simulace fyziky se nastavuje vlastností `SKPhysicsWorld`, kde se jí předává instance scény. Poté je možné použít delegáta fyzického kontaktu `SKPhysicsContactDelegate`, který následně při kontaktu dvou fyzických těl zavolá metodu `didBegin`. Při řešení kontaktu se nejprve v metodě zjistí z parametru `contact`, jaká dvě těla spolu kolidovala, porovnáním jejich bitmasky, separující interakce mezi jedntolivými typy fyzických těl. Pozadí úrovně je nastaveno fyzické tělo za účelem ohraničení hratelné plochy, aby hráč ani NPC nemohli z mapy odejít.

Identifikace fyzického těla je uložena ve vlastnosti `categoryBitMask`. Při kontaktu se testují vlastnosti `contactTestBitMask` a u kolizí `collisionBitMask`. Kolize se mohou zapnout i jednostranně, kdy se první tělo odrazí a to druhé zůstane bez efektu na místě. Jednostranné kolize se používají především pro ohraničení herní plochy a statické části mapy. Zmíněnou logiku používají i některá nepřátelská NPC, například v úrovni bludiště, kde mohou procházet zdí, aby se k hráči dostala rychleji a nenápadně.

Fyzickým tělům lze nastavit váhu, hustotu, rychlost a to, jestli se dá tělem pohnout skrze fyzickou simulaci. Pohyb hráče je řešen nastavením jeho vektorové rychlosti definované polohou páčky joysticku. NPC má fyzické vlastnosti nastaveny v komponentách `ContactComponent` a `MoveComponent` a jeho pohyb je definován pomocí cílů, ty jsou dále popsány v kapitole 6.10.

6.5 Joystick

Digitální joystick je prvek hry, jehož pomocí hráč ovládá hlavní postavu. Je reprezentován třídou `Joystick` a na obrazovce je zobrazen kruhem s diagonálními šipkami, jak je ukázáno na obrázku 4. SpriteKit nemá třídu pro joystick a bylo ji třeba vytvořit. Důležitou součástí joysticku je jeho páčka, která indikuje směr, kam hlavní postava směřuje. Páčkou (`Thumbstick`) se může hýbat ve vyznačeném kruhu joysticku.

Obrázek 4: Plocha digitálního joysticku.

Základní vlastnosti třídy `Joystick`:

`isDynamic: Bool` – proměnná pro definování aktuálního typu joysticku
`maxVelocity: CGFloat` – definuje maximální rychlost pohybu hráče
`isHidden: Bool` – proměnná pro kontrolu, zda je joystick viditelný na scéně

Podstatné metody třídy `Joystick`:

`maxVelocityCheck(node : SKSpriteNode)`
– kontroluje maximální rychlost uzlu `node` (hráče) a případně ji změní

`maxVelocityCheck(node : SKSpriteNode)`
– kontroluje maximální rychlost uzlu `node` (hlavní postavy) a případně ji změní

`movement(moveWith node: SKSpriteNode)`
– metoda pro pohyb s uzlem `node` (hlavní postavy) a otočení uzlu směrem kam se pohybuje

6.6 Kamera

Pohyb hlavní postavou sice vyřešil joystick, ale scéna stojí na místě a hra působí staticky. Místo pohybu scény se tu nabízí lepší možnost, a to pohyb kamerou (`SKCamera`) po scéně. Ta je nastavitelnou vlastností scény a bez jejího nastavení je zobrazena scéna celá, nebo její výseč, již lze definovat vlastnostmi `anchorPoint` a `size`. Třídu `SKCamera` rozšiřuji o metody `scaleFor`, `generateMovementAction` a `movement`. Tyto metody zajišťují hladký pohyb po scéně, možnost zaměřit se na libovolný uzel a také škálování pro různě velká zařízení.

6.7 Entity Component System

ECS, definovaný v kapitole 4.1, používám také jako předlohu pro strukturu projektu. Entity (`GKEntity`) jsou všechna NPC, hráč a pozadí s fyzickým tělem (pro hráče neprůchozí prvky). Komponenty (`GKComponent`) sloužící pro definování vlastností entit jsou popsány v kapitole 6.10. Jako systémy nepoužívám předdefinovanou třídu `GKSystem`, ale vlastní třídu `EntityManager` a také třídu atuální úrovně. Pro použití těchto systémů jsem se rozhodl na základě návodů pro práci s frameworkem GameplayKit [14]. Toto rozhodnutí je vhodné v mém případě, kdy je třeba pro každou úroveň lehce měnit funkcionalitu.

6.7.1 Spravování entit

`EntityManager` je třída pro správu entit, která je umožňuje přidat na scénu. Třída představuje zároveň systém aktualizující komponenty entit a komunikuje s třídou `MoveSettings`, zajišťující chování pro agenty, které je blíže popsáno v kapitole 6.10.

Klíčové vlastnosti třídy `EntityManager`:

`gameEntities: Set` – proměnná obsahující všechny herní entity na scéně
`player: GKEntity?` – proměnná entity hráče

Základní metody třídy `EntityManager`:

`addMsg(msgLabel: SKLabelNode)`
– metoda přidá zprávu (text) přes textový uzel `msgLabel` na scénu

`entitiesWithMoveComponent() -> [MoveComponent]`
– metoda vrací všechny entity, které se mohou pohybovat po scéně. Tato metoda se používá pro zastavení pohybu entit při pozastavení hry.

`addMsg(msgLabel: SKLabelNode)`
– metoda přidá zprávu (text) přes textový uzel `msgLabel` na scénu

`update(_ deltaTime: CFTimeInterval)`
– metoda zajišťuje aktualizaci komponenty pohybu `MoveComponent`

6.8 Implementace vyprávění příběhu

SpriteKit nabízí pouze základní funkcionalitu zobrazování textu přes třídu `SKLabelNode`, a není tak přizpůsoben pro vyprávění příběhu. Celá tato funkcionalita se musela naprogramovat.

Zobrazení příběhového textu probíhá v několika krocích:

1. vypne se možnost pohybu a skryje se tlačítko pozastavení hry,
2. zastaví se pohyb NPC,
3. text se zobrazí na obrazovce,
4. kamera se přesune na postavu vyprávějící příběh,
5. kamera zůstává na postavě do doby, než zmizí text,
6. zapne se pohyb NPC,
7. povolí se pohyb a odkryje se tlačítko pozastavení hry.

6.9 Pozastavení a přerušení hry

Hraní her může být z různých důvodů přerušeno ať už například náhlým telefonátem, nebo jenom pozastavením, kdy hráč zmáčkl tlačítko pozastavení, které je zobrazeno na obrázku. Hra s přerušením a pozastavením pracuje podobně.

Pozastavení hry je implementováno ve třídě `GameScene`, takže je dostupné pro každou úroveň a lze je spustit přes zmíněné tlačítko.

Přerušení je řízeno delegátem třídy `AppDelegate`, který oznámí objektu `GameScene` přerušení hry metodou `applicationWillResignActive`. Opětovné zobrazení hry volá metodu `applicationDidBecomeActive`. Tato oznámení zasílá objekt třídy `NSNotification` jazyka Objective-C, které třída `GameScene` zpracuje a zavolá stejnojmenné metody implementované ve třídě `GameScene`. Jelikož je třída `NSNotification` napsána v jazyku Objective-C tak metody, které volá je potřeba označit atributem `@objc`. Menu pozastavení lze vidět na obrázku 5.

Obrázek 5: Menu pozastavení hry.

6.10 Agenti, cíle a chování

Princip fungování agentů, cílů a chování je vysvětlen v kapitole 4.2. Agenti (`GKAgent`) jsou reprezentováni třídou `MoveComponent`, jež je zároveň komponentou tudíž umožňuje používat architekturu ECS. Chování (`GKBehavior`) je reprezentováno třídou `MoveSettings`. `GKBehavior` je kolekce cílů (`GKGoal`) s různými váhami, které se agenti pokouší plnit.

Při snaze o použití cíle inicializovaného parametrem `toCohereWith`, jsem narazil na chybu frameworku GameplayKit a nepodařilo se mi aktivovat jeho funkcionalitu. Následně jsem se snažil dozvědět se o chybě více, ale komunita používající GameplayKit je velice omezená. Kromě dokumentace, která zmiňuje funkcionalitu a použití, jsem víc informací nezjistil. Ostatní cíle ve hře fungují bezproblémově.

6.10.1 Třída MoveComponent

Třída `MoveComponent` zajišťuje pohyb NPC entity.

Metody definující pohyb:

`agentWillUpdate(_ agent: GKAgent)`
– metoda aktualizuje aktuální pozici agenta, což zajišťuje správné nastavení chování pomocí třídy `MoveSettings`, která nastavuje agentovi na základě vzdálenosti od agenta hráče, či jiného NPC agenta.

`update(deltaTime seconds: TimeInterval)`
– metoda nastavuje agentovi chování `behavoir`, které se zvolí na základě toho, jaké komponenty entita (agent) obsahuje a aktualizuje pohyb entity (agenta) na základě zmíněného chování. Rozdílné chování jsou například pro entity, které mají komponentu `EnemyComponent`, či `KamikazeComponent`.

6.10.2 Třída MoveSettings

Třídě `MoveSettings` má implementovanou pouze vícero typů inicializační metody. Tato třída se totiž používá pouze pro definování chování. Toto chování pak definuje pohyb agenta ve funkci update třídy `MoveComponent`. Chování obsahuje cíle s různou váhou, které mění například na základě vzdálenosti agenta od hráče.

6.11 Ukládání dat

Řešení perzistentního uložení aktuální úrovně a dalších je velice jednoduché. Nabízí se třída `UserDefaults`, umožňující uložení základních typů jako `bool`, `string`, či `double`. Ukládání dat probíhá metodou `set` a načítání dat metodou se shodným názvem jako základní typ, který chceme vrátit.

6.12 Testování

Při testování hry jsem zjistil, že ne všichni hráči čtou stejnou rychlostí a ne každému vyhovuje statický joystick v levém dolním rohu obrazovky. K řešení zmíněných problémů je určeno nastavení, kde uživatel může měnit dobu, po kterou se mu zobrazuje text na obrazovce, a vybrat si mezi statickým a dynamickým joystickem. Dynamický znamená, že se joystick zobrazí v bodě dotyku na obrazovku. Například na iPadu se hra lépe ovládá dynamickým joystickem.

Testování odhalilo rozdíly mezi simulátorem a fyzickým zařízením, jako například různé velikosti displeje. Tento problém jsem částečně vyřešil, a pokud nastanou problémy se simulátorem, stačí kód odkomentovat. Tudíž se vývojář nemůže spoléhat pouze na simulátor. Tento problém je při vývoji této hry těžko řešitelný, jelikož vyvíjím na macbooku z roku 2014, ten však nepodporuje nejnovější verzi operačního systému macOS, a tudíž ani novější verzi XCodu. Starší verze XCodu totiž nepodporuje iOS verze vyšší nežli 15.2, zatímco mé fyzické zařízení je iOS verze 16. Řešením nakonec bylo nahrání hry prostřednictvím online služby TestFlight od společnosti Apple umožňující distribuovat iOS-aplikace

pro testování a do obchodu aplikací AppStore. Jediným omezením této služby je, že vývojář musí čekat na schválení aplikace, což může trvat několik dní.

7 Uživatelská příručka

Uživatelská příručka přibližuje čtenáři, jak ovládat a používat vytvořenou hru. Za účelem seznámení se s obsahem a příběhem hry může čtenář použít kapitolu 5. Příručka obeznámí uživatele s doporučeným postupem, aby hrou mohl postupovat hladce. Cílem hry je postupné procházení úrovní za doprovodu příběhu, který uživatele hrou naviguje. Výběr úrovní lze manuálně povolit snímkem obrazovky (screenshot).

7.1 Hlavní menu

Úvodní obrazovka zobrazuje hlavní menu. Nachází se v něm tlačítko „Play Game“ pro spuštění hry od poslední uložené úrovně, tlačítko „Reset Progress“ pro vynulování herního postupu a tlačítko „Level Select“ pro výběr úrovně.). Výběr úrovně je při spuštění nové hry zakázán. Povolit lze snímkem obrazovky (screenshot), nebo dokončením poslední úrovně ve hře. V pravém horním rohu se nachází tlačítko s ikonou ozubeného kolečka, které označuje nastavení. Hlavní menu je ukázáno na obrázku 6.

Obrázek 6: Hlavní menu hry zobrazené na zařízení iPhone X.

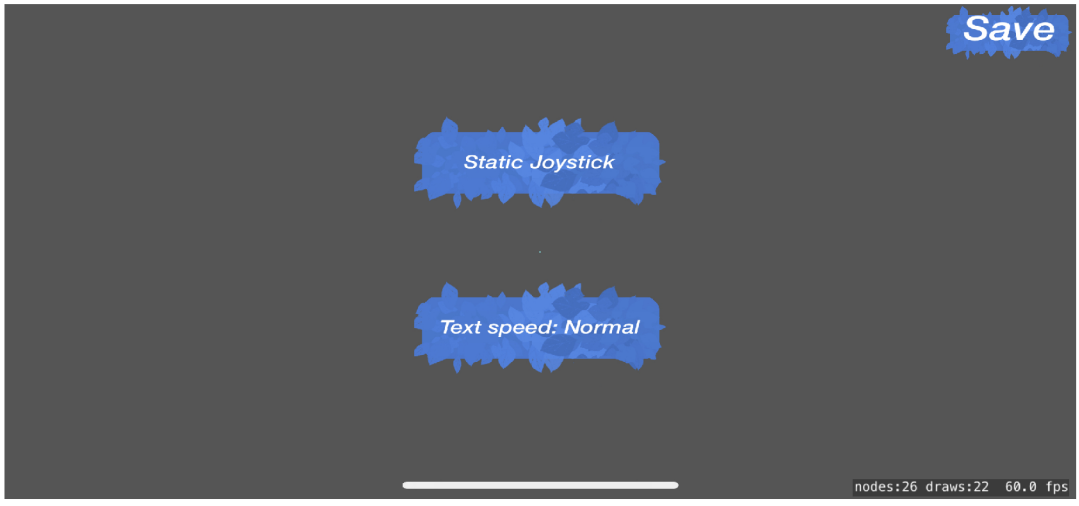
7.2 Typy úrovní

Úrovně jsou dvojího typu: příběhové a úkolové. V příběhové úrovni má hráč za úkol najít herní postavy (NPC), které vypráví příběh, od nich se dozvědět co nejvíce informací a poté přejít do následující úrovně. Krok pro postup do další

úrovně, jímž bývá většinou náraz hráče do vyprávějící postavy, popíše ta postava, která vypráví hlavní příběh. Úkolovými úrovněmi má hráč projít na základě informací, jež zjistil v úrovni příběhové. Doplňující informace hráči poskytují NPC nacházející se v dané úrovni. Aktuální úkoly nejsou nikde na obrazovce napsány, jak bývá u většiny her zvykem, ale hra definuje úkoly v příběhovém textu neboli v rozhovorech s NPC, které by měl hráč číst. Dobrým příkladem této funkcionality je hra Stray[8], kde jsou hráči úkoly taktéž předávány při dialozích s NPC.

7.3 Ovládání hry

Pro ovládání hry v úrovních se používá digitální joystick, který je při prvním spuštění hry nastaven v levém dolním rohu. Typ joysticku lze změnit v nastavení, do něhož se lze dostat přes hlavní menu, nebo menu pozastavení. Uživatel si může vybrat mezi zmíněným statickým joystickem a dynamickým, jenž se zobrazí v místě dotyku na obrazovku. Dynamický joystick se velice hodí pro iPad, kde je statický joystick při velikosti obrazovky zařízení velice nepraktický a nepohodlný. Další zpřístupnění pro uživatele představuje nastavení jeho rychlosti čtení. Je možné si vybrat rychlost pomalou (slow), střední (normal) a rychlou (fast). Podle zmíněného nastavení se bude analogicky měnit doba, po kterou zůstane text na obrazovce. Toto nastavení zpříjemní hráči požitek ze hry s tím, že může všechny rozhovory NPC sledovat rychlostí, která je mu příjemná. Ukázka menu nastavení a toho, jak se do něj dostat, je demonstrována na obrázku 7.

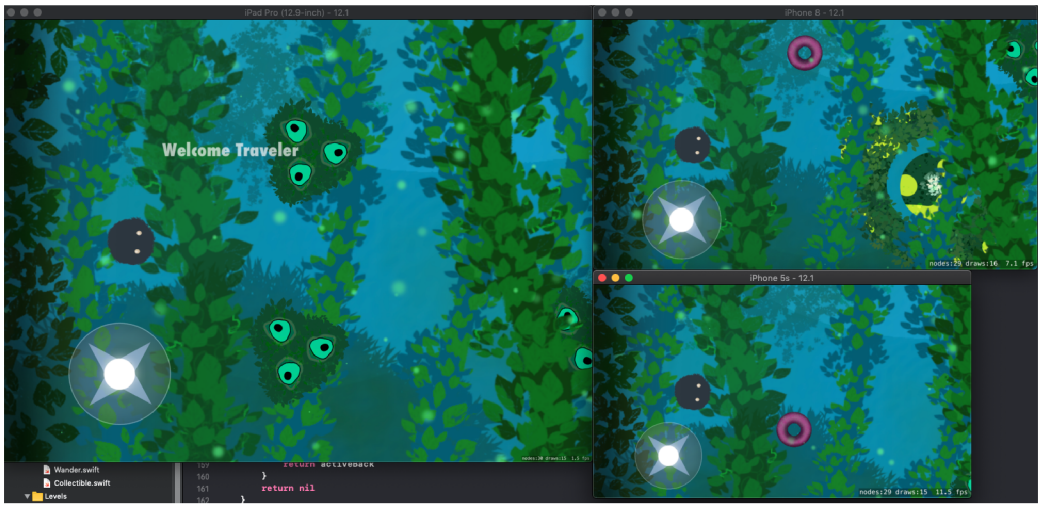


Obrázek 7: Menu nastavení kde si je hráč schopen přizpůsobit ovládání hry.

[8]Hratelnost videohry Stray: https://en.wikipedia.org/wiki/Stray_(video_game)#Gameplay

7.4 Podporovaná zařízení

Podpora zařízení je rozšířena pro všechna zařízení běžící na operačním systému iOS a iPadOS, jejichž verze je 12.0 a vyšší. Podporu zařízení, testovanou při vývoji hry, lze vidět na obrázku 8. Vývoj a ladění hry probíhaly na simulátoru iPhone 8 (iOS verze 15.2) a fyzickém zařízení iPhone X (iOS verze 16). Hra je dimenzována primárně pro zmíněná zařízení a vypadá na nich přesně tak, jak byl její design zamýšlen. Na ostatních zařízeních může díky svému rozličnému rozpoložení hra vypadat trochu jinak, avšak je možné ji hrát na všech zmíněných zařízeních bez většího dopadu na hratelnost.



Obrázek 8: Testování hry na různých zařízeních.

7.5 Instalace hry

Hru lze nainstalovat dvojím způsobem a doporučuji způsob první v následující kapitole, který je jednodušší a vyžaduje pouze zařízení kde hra bude nainstalována a spuštěna.

7.5.1 Instalace pomocí aplikace TestFlight

Uživateli potřebuje pouze zařízení, na které chce hru nainstalovat. Podporovaná zařízení jsou zmíněna v předchozí kapitole.

Kroky pro instalaci a spuštění hry jsou následující:

1. jako první si musí uživatel stáhnout aplikaci TestFlight z obchodu aplikací AppStore,

2. následně si na stejném zařízení otevře odkaz https://testflight.apple.com/join/BsVs4s9H a stisknutím tlačítka „Otevřít“, nebo tlačítka „Start Testing“ se otevře aplikace TestFlight, kde se uživateli zobrazí

pozvánka pro testování hry; **poznámka:** některým zařízením se po kliknutí na odkaz ihned otevře aplikace TestFlight a je možné ihned pokračovat dalším krokem,

3. na pozvánce je potřeba stisknout tlačítko „Přijmout“ a poté je hra připravena pro stažení. Následně se hra nainstaluje stlačením tlačítka „Nainstalovat“,
4. po dokončení předchozího kroku je hra nainstalovaná na zařízení a lze ji spustit a používat stejným způsobem jako ostatní aplikace.

7.5.2 Instalace přes vývojové prostředí XCode

Druhá možnost instalace se nabízí přes vývojové prostředí XCode. Jelikož hra podporuje iOS verze 12.0 a, vyšší, teoreticky dostačující se jeví XCode verze 10.0.0 (nebo vyšší). NNicméně kroky pro instalaci se mohou lišit, protože tento návod vychází z XCodu verze 13.2.1. Pro instalaci hry doporučuji XCode verze 13.2.1, která podporuje systémy iOS a iPadOS verze 15.2.

Kroky pro instalaci a spuštění hry jsou tyto:

1. spustíme aplikaci XCode,
2. na úvodní obrazovce vybereme nový projekt tlačítkem „Open a project or file“,
3. vybereme složku „bakalarska_prace“, která je obsažena v příloze práce ve složce „src“,
4. nahoře v menu pro aplikaci XCode v záložce „Product“ vybereme záložku „Destination“, kde si zvolíme ze simulátoru, na kterém chceme hru spustit,
5. poté si v záložce „Product“ vybereme možnost „Run“, která hru na zařízení nainstaluje a spustí.

Pokud chceme spustit hru na fyzickém zařízení jsou kroky následující:

1. první potřebujeme vývojářský účet společnosti Apple,
2. ten připojíme přes záložku „XCode“ v horním menu kde klikneme na záložku „Preferences“; tenhle krok můžeme udělat pomocí klávesové zkratky „command“ a klávesy čárky (cmd+“,“),
3. dále okně zvolíme záložku „Accounts“ a ve spodní části okna klikneme na tlačítko „+“ a ve vyskakovacím okně označíme možnost „Apple ID“ a klikneme na tlačítko „continue“,
4. vyplníme se údaje vývojářského účtu
5. připojíme fyzické zařízení kabelem k počítači,
6. poté postupujeme stejnými kroky jako pro spuštění hry na simulátoru s tím rozdílem že v kroku 4 vybereme fyzické zařízení.

8 Závěr

Výsledkem práce je 2D hra typu „adventure“ pro systémy iOS a iPadOS naprogramovaná v jazyce Swift za pomocí frameworků SpriteKit a GameplayKit. Cílem hry je postupné procházení úrovní za doprovodu příběhu. Hra obsahuje 10 úrovní a dva možné konce, interaktivní prostředí, řadu NPC a příběh, který ji doprovází. Úrovně se dělí na příběhové, v nichž se vypráví hlavní a vedlejší příběh kde jsou hráči zadávány úkoly. Úkolové úrovně obsahují hádanky, dá se v nich strategizovat a jejich řešení je přiblíženo příběhovou úrovní. Klíčovým prvekem hry jsou interakce mezi hráčem a NPC, jež se mohou na základě hráčových rozhodnutí měnit. Práce staví na architektuře Entity Component System a ovládání NPC je definováno pomocí agentů, cílů a chování. Ke hře byly vytvořeny hudební doprovod, grafika postav a úrovní. Hra má implementováno nastavení umožňující hráči přizpůsobit a zpříjemnit ovládání hry.

V textové části jsem porovnal možnosti vývoje her pro platformu iOS, zkoumal jsem herní enginy i programovací jazyky. Vysvětlil jsem důvody výběru platformy SpriteKit a programovacího jazyka Swift. Tuto platformu podrobněji popisuji a uvádím její použití pro vývoj her. Dále přibližuji platformu GameplayKit která doplňuje SpriteKit o architekturu ECS a o ovládání NPC. Následně jsem popsal obsah hry a seznámil čtenáře s používáním a implementací hlavních částí hry, kde jsem zmínil i problémy při vývoji hry a jejich řešení.

Závěrem bych chtěl ještě poznamenat, že výběr frameworku SpriteKit znesnadnil práci na vývoji hry, jelikož tento framework není příliš obsáhlý a chyběla zde základní funkcionalita specifická pro tuto hru. Příště bych preferoval pokročilý herní engine, jako je například Unity, nebo Unreal Engine. I přes to jsem se s nevhodným výběrem naučil pracovat a hru dodělal do konce, což se také v praxi objevuje, a díky tomu jsem získal cenné zkušenosti do budoucna.

A Obsah přílohy

`doc/`
: Složka obsahuje text závěrečné práce ve formátu PDF, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

`src/`
: V této složce se nachází kompletní zdrojové texty hry. Dále je ve složce XCode projekt, kde je umístěna veškerá grafika a hudba pro tuto hru.

`readme.txt`
: Soubor typu txt obsahuje instrukce pro instalaci a spuštění hry této práce, včetně všech doporučených požadavků pro jeho bezproblémový provoz.

Literatura

[1] FULL SCALE. What is a Game Engine? [online]. [cit. 2023-01-04]. Dostupné z: https://fullscale.io/blog/what-is-game-engine/

[2] EPIC GAMES. Unreal Engine 5 (game engine) [online]. [cit. 2023-01-04]. Dostupné z: https://www.unrealengine.com/en-US/unreal-engine-5

[3] SALAMA, Ramiz; ELSAYED, Mohamed. A live comparison between Unity and Unreal game engines. Global Journal of Information Technology: Emerging Technologies, 2021 [online]. [cit. 2023-01-04]. 11 s. 01-07. Dostupné z: https://doi.org/10.18844/gjit.v11i1.5288

[4] STEFFEN, Itterheim. Why Apple Created Sprite Kit And What It Means For Cocos2D [online]. [cit. 2023-01-04]. Dostupné z: https://web.archive.org/web/20160310144822/http://www.learn-cocos2d.com/2013/06/apple-create-spritekit/

[5] COMPUTER HISTORY MUSEUM. The Forces of Change in the Transformation Age Making Make Software: Change the World! Designing a New Space for Learning. Core Magazine [online]. 2017, 62-65 [cit. 2022-12-19]. Dostupné z: http://s3.computerhistory.org/core/core-2017.pdf

[6] MCCRACKEN, Harry. Lyft Goes Swift: How (And Why) It Rewrote Its App From Scratch In Apple's New Language [online]. [cit. 2022-12-16]. Dostupné z: https://www.fastcompany.com/3050266/lyft-goes-swift-how-and-why-it-rewrote-its-app-from-scratch-in-apples-new-lang

[7] APPLE INC. Swift (programming language) [online]. [cit. 2022-12-13]. Dostupné z: https://www.apple.com/swift/

[8] WELLS, Garrett. The Future of iOS Development: Evaluating the Swift Programming Language [online]. Claremont McKenna College, April 28, 2015 [cit. 2022-12-16]. Dostupné z: https://scholarship.claremont.edu/cmc_theses/1179/

[9] NEKRASOV, Alex. JetBrains' AppCode IDE vs. Xcode [online]. [cit. 2023-01-04]. Dostupné z: https://betterprogramming.pub/appcode-instead-of-xcode-a12f2d2810e2

[10] APPLE INC. XCode [online]. [cit. 2023-01-04]. Dostupné z: https://developer.apple.com/documentation/xcode

[11] APPLE INC. SpriteKit (framework) [online]. [cit. 2023-01-04]. Dostupné z: https://developer.apple.com/documentation/spritekit/

[12] CORON, Tammy. Apple Game Frameworks and Technologies: Build 2D Games with SpriteKit & Swift. Pragmatic Programmers, 2021 [cit. 2023-01-04]. 504 s. ISBN: 9781680507843

[13] APPLE INC. GameplayKit (framework) [online]. [cit. 2023-01-04]. Dostupné z: https://developer.apple.com/documentation/GameplayKit

[14] ACKERMANN, Ryan. GameplayKit Tutorial: Entity-Component System, Agents, Goals, and Behaviors [online]. [cit. 2023-01-04]. Dostupné z: https://www.kodeco.com/706-GameplayKit-tutorial-entity-component-system-agents-goals-and-behaviors

[15] MURATET, Mathieu; GARBARINI, Délia. Accessibility and serious games: What about Entity- ComponentSystem software architecture? [online]. GALA, 2020. [cit. 2023-01-04]. Dostupné z: https://hal.archives-ouvertes.fr/hal-02987484/document

[16] APPLE INC. Entities and Components [online]. [cit. 2023-01-04]. Dostupné z: https://developer.apple.com/library/archive/documentation/General/Conceptual/GameplayKit_Guide/EntityComponent.html