VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

DESKOVÁ HRA PRO APPLE TV

A GAME FOR APPLE TV

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ADAM KRAMÁR**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MARTIN HRUBÝ, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19866/2016/xkrama04

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Kramár Adam**
Obor: Informační technologie
Téma: **Desková Hra pro Apple TV**
A Game for Apple TV
Kategorie: Uživatelská rozhraní

Pokyny:

1. Prostudujte programování aplikací pro operační systémy iOS a tvOS. Prostudujte knihovny pro programování her v těchto systémech (SpriteKit apod.).
2. Navrhněte deskovou hru vhodnou pro ovládání dálkovým ovladačem Apple TV.
3. Hru implementujte.
4. Hru testujte v emulátoru Apple TV a na reálném zařízení.

Literatura:

- Dokumentace knihoven pro iOS a tvOS.

Pro udělení zápočtu za první semestr je požadováno:

1. První dva body.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Hrubý Martin, Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Táto práca pozostáva z návrhu a implementácie hry pre Apple TV a popisu použitých technológií. Výsledkom je hra napísaná v jazyku Swift, spustiteľná na zariadení Apple TV s operačným systémom tvOS. Základom hry je knižnica SpriteKit, ktorá umožňuje tvorbu jednoduchých hier pre systém tvOS. Hra je otestovaná a plne funkčná na reálom zariadení. Prednosťami hry sú nízke hardwarové nároky a jednoduchý koncept hry.

Abstract

This thesis consists of the design and implementation of the Apple TV game and a description of the technologies used. The game was programmed in Swift, executable on Apple TV with the tvOS operating system. The game is based on SpriteKit, which allows you to develop simple games for tvOS. The game is tested and fully functional on real devices. The advantages of the game are low hadrware requirements and a simple concept of the game.

Kľúčové slová

Hra, iOS, tvOS, SpriteKit, Swift, Xcode, Apple TV

Keywords

Game, iOS, tvOS, SpriteKit, Swift, Xcode, Apple TV

Citácia

KRAMÁR, Adam. *Desková Hra pro Apple TV*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Hrubý Martin.

Desková Hra pro Apple TV

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Martina Hrubého, Ph.D. Uviedol som všetky literárne zdroje a publikácie, z ktorých som čerpal.

.....
Adam Kramár
11. mája 2017

Poďakovanie

Touto cestou by som chcel poďakovať môjmu vedúcemu bakalárskej práce pánovi Ing. Martinovi Hrubému, Ph.D. za rady pri návrhu a implementácii hry a taktiež každému, kto sa podieľal na dokončovaní a testovaní.

Obsah

Kapitola 1

Úvod

V dnešnej dobe spoločnosť Apple Inc. poskytuje množstvo produktov, ako napríklad mobilné telefóny, tablety, stolové a prenosné počítače, hodinky a set-top box Apple TV. Všetky z týchto zariadení majú svoje špecifické operačné systémy, ktoré sú založené na spoločnom základe a majú odlišnosti podľa platformy. Apple TV štvrtej generácie s operačným systémom tvOS dokáže prehrávať digitálne média a slúži ako mikrokonzola. K tomuto zariadeniu je nutné pripojiť televíziu, prípadne inú obrazovku s HDMI[1] vstupom a ovládať ho pomocou diaľkového ovládača.

Moja bakalárska práca mala za cieľ vytvoriť hru pre operačný systém tvOS na zariadeni Apple TV. Informácie o programovaní týchto aplikácií som čerpal z literatúry [8] a [10]. Táto technická správa obsahuje stručný popis operačných systémov iOS a tvOS, ktoré sú v zásade rovnaké len s minimálnymi rozdielmi. Konkrétne sa zameriavam na popis knižnice `SpriteKit`, ktorá je základom pre programovanie jednoduchých hier v týchto systémoch. Následne popisujem vývojové prostredie Xcode, v ktorom je možne tvoriť aplikácie nie len pre systém tvOS, ale aj OS X, iOS a watchOS. Podrobnejšie budú spomenuté nástroje pre ladenie, rozhranie a možnosť virtualizácie tvOS. Ďalej bude spomenutý implementačný jazyk, samotné zariadenie Apple TV a niekoľko existujúcich hier, ktorými som sa inšpiroval. Po zhrnutí teoretickej časti nasleduje návrh mojej vlastnej hry, ktorý pozostáva z návrhu užívateľského rozhrania a samotnej hry. V kapitole 5 je potom spomenutá implementácia hry s bližším popisom kľúčových funkcií, ako sú kolízie a podobne. Záver práce tvorí testovanie v emulátore zariadenia vo vývojom prostredí Xcode a testovanie na reálnom zariadení Apple TV, najskôr mnou a následne s pomocou niekoľkých dobrovoľníkov.

[1] High-Definition Multimedia Interface

Kapitola 2

Programovanie aplikácií pre iOS a tvOS

Pre programovanie iOS a tvOS aplikácii je potrebné zariadenie s OS X 10.10 prípadne novšie. Toto zariadenie musí mať nainštalované vývojové prostredie Xcode, to je bežne dostupné z App Storu. Použitý programovací jazyk bol Swift vo verzii 3.1, o ktorom je viac písané v podkapitole 3.2. Výsledná aplikácia musí byť spustiteľná na zariadení Apple TV 3.3.

2.1 Operačné systémy iOS a tvOS

Táto sekcia obsahuje krátky popis operačných systémov iOS a tvOS. Následne sú spomenuté základne rozdiely a to aké výhody či nevýhody prinášajú tieto rozdiely pri tvorbe aplikácii pre Apple TV (tvOS).

Operačný systém iOS

Operačný systém iOS pôvodne známy ako iPhone OS, bol vyvinutý firmou Apple Inc. a prvý krát predstavený v roku 2007 ako operačný systém pre mobilné telefóny iPhone. Postupne bol použitý aj pre iné zariadenia vyrábané firmou Apple, ako napríklad iPod touch, iPad a iPad Mini. Po systéme Android, ide o druhý najviac rozšírený operačný systém pre mobilné telefóny. Súčasne najnovšia dostupná verzia systému je 10.3.1, ktorá bola vydaná v roku 2017.

Operačný systém tvOS

Operačný systém tvOS vytvorený firmou Apple Inc. pre štvrtú generáciu Apple TV v októbri roku 2015. tvOS je založený na operačnom systéme iOS a nejde iba o jeho orezanú verziu, ako tomu bolo pri operačných systémoch pre druhú a tretiu generáciu Apple TV. Súčasná najnovšia dostupná verzia systému je 10.2, ktorá je založená na systéme iOS 10.3 [4].

Jednou zo základných vlastností systému je možnosť pohybu v užívateľskom rozhraní pomocou diaľkového ovládača s dotykovou plochou, pomocou ktorej je možné používať viac dotykové gestá. Ďalej poskytuje možnosť sťahovania a inštalovanie aplikácií a hier, ktoré boli vytvorené špeciálne pre tvOS. Rozdiel pri používaní aplikácií oproti iOSu je v tom, že nie je možne používať samotnú obrazovku ako dotykovú plochu, ale je potrebné použiť diaľkový

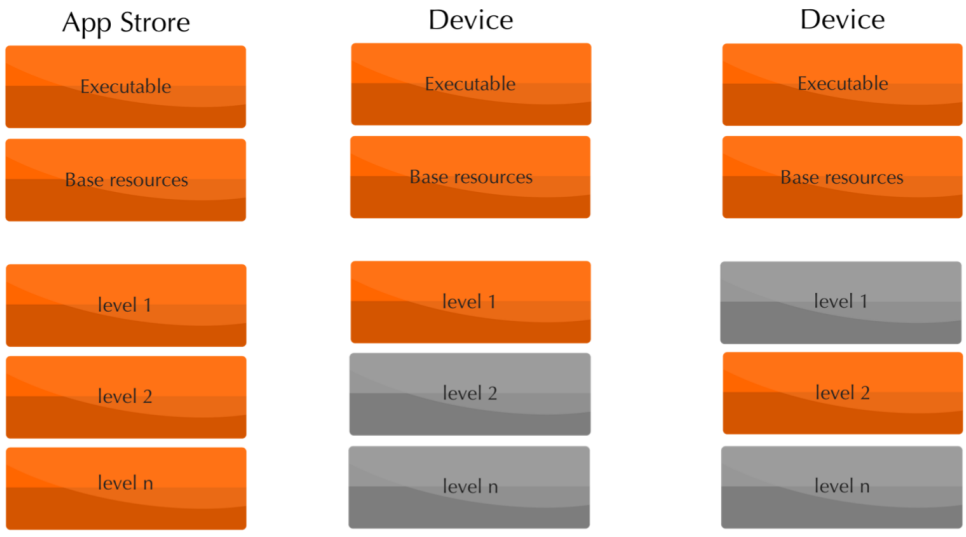
ovládač. Pomocou neho je možné sa zamerať na ľubovoľný prvok a následne potvrdiť voľbu stlačením dotykovej plochy.

Focus engine

Focus engine prináša novú metódu užívateľského vstupu pre tvOS. Táto technológia bola zavedená kvôli zmene oproti systému iOS, v ktorom užívateľ s rozhraním pracuje priamo dotykmi na obrazovke zariadenia. Televízia na takýto druh interakcie nie je optimalizovaná a je potrebné použiť diaľkový ovládač, prípadne herný ovládač. Koncept je taký, že je vždy jedna a najviac jedna položka vybratá a tá je zameraná v danom čase. Používateľ môže meniť položky v zameraní rôznymi gestami na dotykovej ploche ovládača. Zameraná položka je zvýraznená takým spôsobom, že je zväčšená a umiestnená v popredí. Z programátorského hľadiska je Focus engine jednoduchý na použitie, pretože stačí pridať do StoryBoardu niekoľko pohľadov a systém automaticky vyberie jeden, ktorý zameria. Pri užívateľskom vstupe sa nájde najbližší pohľad a zameranie prejde naň. Detailný popis funkcionalít je uvedený v literatúre [11].

Aplikácie pre tvOS

Vzhľadom na skutočnosť že tvOS je založený na systéme iOS, je možné prenášať existujúce aplikácie medzi týmito systémami len s minimálnymi úpravami. Jeden z najväčších rozdielov, ktorý ponúka tvOS je získavanie zdrojov na vyžiadanie. To znamená že pri stiahnutí aplikácie nie je potrebné sťahovať aplikáciu ako celok, ale iba potrebné časti. To môže napríklad prebiehať nasledujúcim spôsobom. Pri sťahovaní hry, ktorá sa skladá z viacerých levelov, stačí stiahnuť základne dáta a napríklad len prvý level, zvyšné časti sú asynchrónne stiahnuté za behu aplikácie, až keď sú potrebné. Vďaka tomu je stiahnutie danej hry, alebo inej aplikácie rýchlejšie a nedochádza k zahlcovaniu pamäte dátami, ktoré v danom čase nie sú potrebné.



Obr. 2.1: Znázornenie získavania zdrojov na vyžiadanie

2.2 Knižnica SpriteKit

V dokumentácii [5] je `SpriteKit` definovaný, ako grafická vizualizácia a animačná infraštruktúra, ktorá sa používa na animáciu obrázkov s ľubovoľnou textúrou. Práca s touto knižnicou spočíva v tom, že programátor určí obsah snímku a to, ako sa bude daný obsah meniť. `SpriteKit` tento obsah vykreslí efektívne s využitím grafického hardwaru. Vďaka tomuto dizajnu je `SpriteKit` vhodný pre tvorbu hier a aplikácii, ktoré požadujú flexibilitu v tom, ako sa pracuje s animáciami. `SpriteKit` používa slučku vykreslenia, kde obsah každého snímku je spracovaný predtým, ako je snímok zobrazený. Renderovanie a animácia sú uskutočňované pomocou `SKView`[1] objektu. Tento objekt vložíme do okna a následne je jeho obsah vykreslený a pretože ide o view objekt, je možné obsah kombinovať s obsahom iných view objektov.

2.2.1 Vykresľovanie scén

Jednotlivé prvky hry, teda jej obsah, sú organizované do scén a tie sú reprezentované pomocou objektu `SKScene`[2]. Scéna implementuje spracovanie obsahu pomocou logiky založenej na počte snímok za sekundu (per-frame logic). V akomkoľvek čase počas behu hry je vo view prezentovaná súčasne len jedna scéna. Pokiaľ je daná scéna prezentovaná, jej logika spracovania obsahu, aj animácie, sú vykonávané automaticky.

2.2.2 Prechod medzi scénami

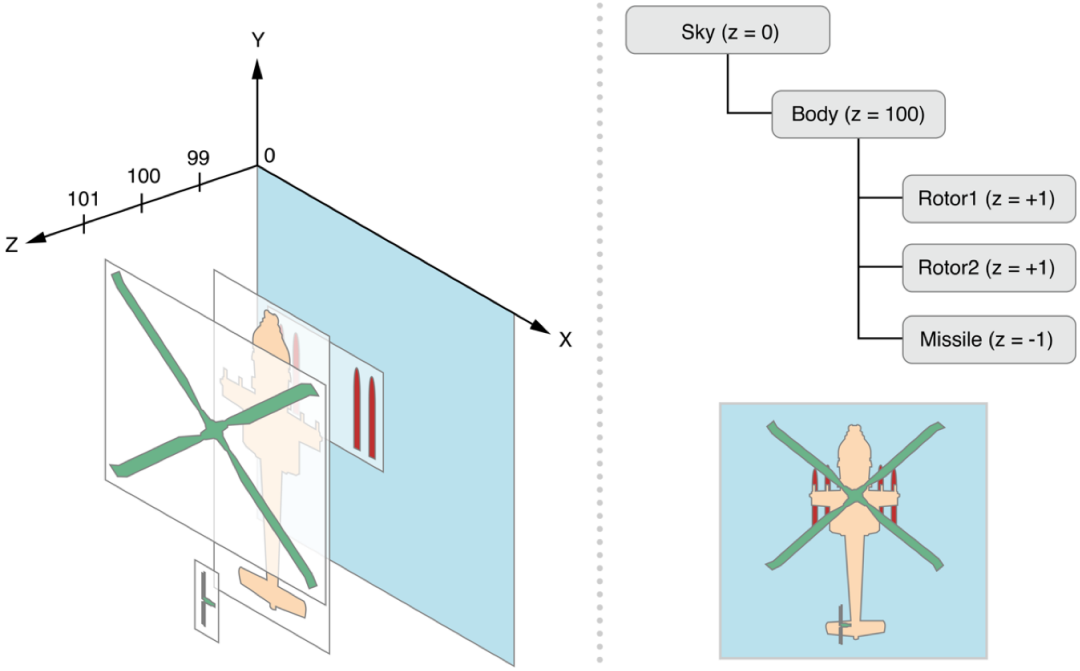
Scény je možné meniť v rámci jedného `SKView` objektu a prechod medzi nimi je možné animovať pomocou triedy `SKTransition`[3]. Scény sú v podstate bloky hry, ktoré reprezentujú častí aplikácie, ako napríklad hlavné menu, načítavanie hry, scéna v ktorej sa samotná herná časť aplikácie odohráva a prípadne scéna, kde sú zobrazené výsledky, či skóre.

Pri prezentovaní novej scény vo view, ktoré už v danom čase prezentuje scénu je možné použiť niektorý z predpripravených animovaných prechodov a to vytvára efekt spojitosti toho, čo sa odohráva na obrazovke. Typicky sa medzi scénami prechádza kvôli postupu v samotnej hre, alebo na základe užívateľského vstupu, Keď sa prechod začne, obsah atribútu, v ktorom je uložená aktuálna scéna, je zmenený na novú scénu. Následne sa vykoná samotná animácia a potom sa odstráni referencia na starú scénu.

2.2.3 Strom uzlov v scéne

Trieda `SKScene` je potomkom triedy `SKNode`[4] a pri používaní knižnice `SpriteKit` je uzol (node object) základnou stavebnou jednotkou všetkého obsahu. Objekt scény, v ktorej sa uzly vykresľujú, je označovaný ako koreňový uzol a tvorí základ stromu uzlov. Pozícia každého uzla je určená podľa koordinačného systému definovaného jeho rodičovským uzlom. Okrem toho je možné uzlu nastaviť aj iné vlastnosti, ako napríklad rotácia, či veľkosť a tie majú efekt aj na všetkých potomkov tohto uzla. Preto je možné vytvoriť komplexnejší objekt zložený z viacerých uzlov a následne s ním pracovať ako s jedným uzlom vďaka dedičnosti. Správnym zostavením štruktúry stromu uzlov je možné určovať poradie, v ktorom budú jednotlivé uzly v scéne vykreslené.

[1] SKView – https://developer.apple.com/reference/spritekit/skview
[2] SKScene – https://developer.apple.com/reference/spritekit/skscene
[3] SKTransition – https://developer.apple.com/reference/spritekit/sktransition
[4] SKNode – https://developer.apple.com/reference/spritekit/sknode



Obr. 2.2: Vykreslenie uzla [7]

2.2.4 Animácie v scéne

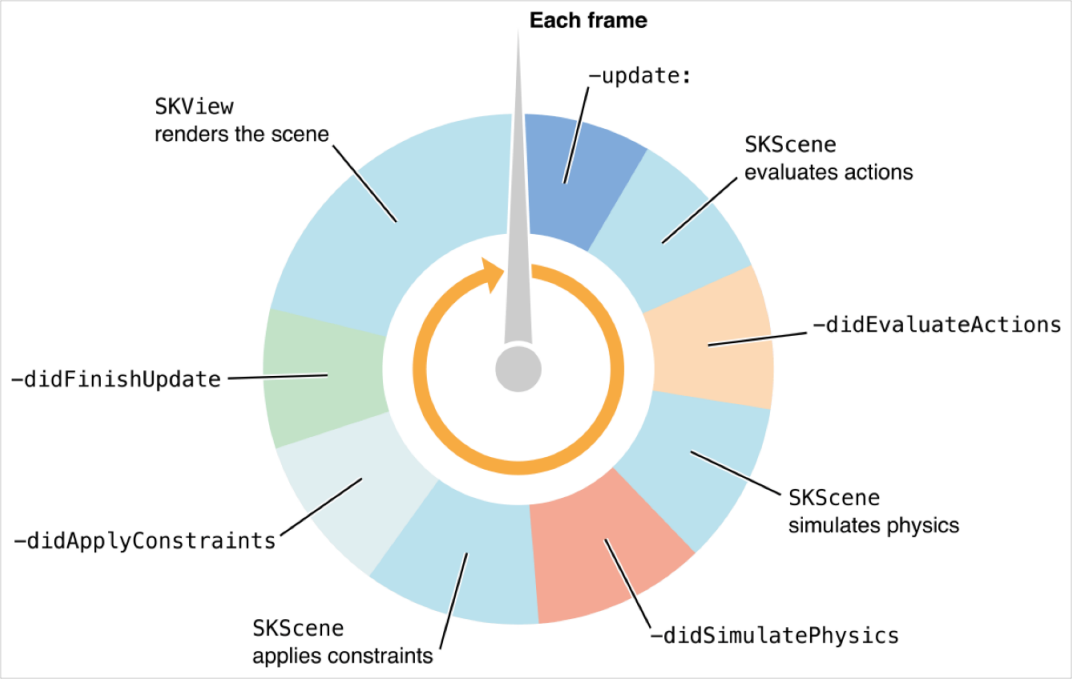
Animácie v scéne sú uskutočňované pomocou akcií a každá akcia je objekt triedy `SKAction`[5]. Pri programovaní vytvoríme akciu a pri spracovaní snímok animácie je daná akcia uskutočnená. Niektoré akcie môžu byť uskutočnené počas jednej snímky animácie, ale iné aplikujú zmeny počas niekoľkých snímok predtým, ako sú ukončené. Akcie sa využívajú na zmenu vlastností uzlov, ako napríklad zmena veľkosti, alebo zmena pozície. Okrem toho môžu zmeniť strom uzlov, prehrať zvuk, alebo spustiť blok kódu. Kombináciou viacerých akcií je možné vytvoriť rôzne efekty a to tak, že sa súčasne spustí niekoľko akcií, alebo sa vytvorí ich sekvencia. Vytvorené sekvencie, ale aj samostatné akcie, je možné automatický opakovať.

Ďalší spôsob vytvorenia animácie je pomocou spracovania v každom snímku. V tom prípade sa napríklad vykonáva daná zmena priamo pri vykreslení každého snímku a nevytvára sa žiadna akcia.

2.2.5 Životný cyklus scény

Úzko spojené s objektom scény `SKScene`, sú procesy animovania a vykresľovania scény. Tieto procesy sú aktívne iba vtedy, keď je scéna prezentovaná a vtedy scéna využíva vykresľovaciu slučku. Tá striedavo spracúva stromy uzlov v scéne a ich vykresľovanie. `SpriteKit` scénu opätovne vykreslí iba vtedy, keď sa v jej obsahu uskutoční nejaká zmena. Pri každom prechode vykresľovacím cyklom je obsah scény aktualizovaný a následne vykreslený. Nie je možne priamo zmeniť vykresľovací postup, avšak scéna obsahuje metódy a tie je možné pozmeniť a tým upraviť spracovanie obsahu. Následne na obrázku 2.3 je možné vidieť vykresľovací cyklus.

[5]SKAction – https://developer.apple.com/reference/spritekit/skaction



Obr. 2.3: Životný cyklus scény [6]

1. *update(_:)* je metóda scény volaná s doposiaľ uplynutým časom v simulácii. Je to miesto, kde je zvyčajne implementované spracovávanie vstupu, umelá inteligencia a ďalšia herná logika. Využívaná tiež pre aplikovanie zmien a spustenie akcií na uzloch.

2. Scéna spracováva akcie na uzloch, vyhľadáva bežiace akcie a aplikuje zmeny na strom uzlov. V tomto momente sa tiež spracovávajú aj akcie pridané programátorom, takto je možné docieliť začlenenie vlastného kódu do mechanizmu akcií. Nie je možné priamo kontrolovať poradie, v ktorom sa akcie vykonajú. Akcie na konkrétnych uzloch je možné vynechať tým, že sú z nich odstránené, pripadne tak, že je uzol odstránený zo stromu uzlov.

3. Po tom, ako boli spracované všetky akcie pre danú snímku, je volaná metóda *didEvaluateActions()*. Táto metóda je volaná presne jeden krát v každej snímke a to pokiaľ je scéna prezentovaná a nie je pozastavená.

4. V tomto kroku scéna simuluje fyziku na uzloch v strome, ktoré majú fyzické telo. Podrobnejší popis o tom, ako funguje simulácia fyziky v scéne, je v podkapitole 2.2.7.

5. Podobne, ako v treťom kroku, je teraz volaná metóda *didSimulatePhysics()*. Vykonáva aktualizácie špecifické pre scénu po tom, ako je dokončená simulácia fyziky.

6. Scéna aplikuje určité obmedzenia a väzby na uzly v scéne. Tieto obmedzenia a väzby slúžia pre stanovenie vzťahov medzi objektmi v scéne. Tým je možné napríklad zaistiť orientáciu určitého uzlu v scéne, bez ohľadu na to kam je presunutý.

7. Scéna volá metódu *didApplyConstraints()*, ktorá aplikuje aktualizácie väzieb a obmedzení, ak je to potrebné.

8. Ako posledná je volaná metóda *didFinishUpdate()*, ktorá je poslednou možnosťou vložiť časť kódu pred tým, ako je snímka spracovaná.

9. Scéna je vykreslená.

2.2.6 Textúrovanie objektov

Textúry reprezentované ako objekty triedy `SKTexture`[6], sú zdieľané obrázky používané pre vykreslenie uzlov do scény. Vhodné využitie textúr je napríklad pri vykreslení jedného obrázku pre viacero uzlov. Pri použití knižnice `SpriteKit` je možné použiť niekoľko zdrojov pre získanie textúr a to načítaním obrázku uloženého v balíku aplikácie, avšak je možné tvoriť textúry aj za behu aplikácie a to zahŕňa buď obrázky `Core Graphics`[7], alebo vykreslením stromu uzlov do textúry.

Celkovo je možné povedať, že `SpriteKit` zjednodušuje používanie textúr tým, že sa stará o nízkoúrovňový kód potrebný pre načítanie textúry. Pri použití veľkého množstva obrázkov, programátor môže zlepšiť výkon tým, že sa postará o časť procesu. To je napríklad možné tak, že explicitne zadáme príkaz pre načítanie textúry.

Atlas je skupina príbuzných textúr, ktoré používame spoločne. Pre predstavu takýto atlas je možné vytvoriť pre uloženie všetkých textúr potrebných pre vytvorenie animácie pohybu nejakej postavy v scéne. `SpriteKit` využíva atlas pre skvalitnenie procesu vykreslenia textúr.

2.2.7 Simulácia fyziky

Napriek tomu, že je možné kontrolovať presnú pozíciu každého uzla v scéne, je niekedy požadované, aby uzly medzi sebou mali kolízie. Kolízia dvoch, či viacerých uzlov, môže ovplyvniť rýchlosť, či smer pohybu týchto uzlov. Toto je možné zabezpečiť pomocou vytvorenia fyzického tela (`SKPhysicsBody`[8]) a následného spojenia tohto tela s uzlom. Tela sú charakterizované pomocou tvaru, veľkosti prípadne iných fyzikálnych vlastností. Scéna definuje globálnu charakteristiku pre simuláciu fyziky v objekte `SKPhysicsWorld`[9]. Keď sú fyzické tela vložené do scény, scéna na nich začne simulovať fyziku. Niektoré sily, ako napríklad gravitácia a trenie sú použité automaticky. Ďalšie sily pôsobiace na vybranú skupiny fyzických tiel je možné aplikovať pomocou rôznych polí. Tieto polia sú objekty typu `SKFieldNode`[10]. Je taktiež možné ovplyvňovať jeden vybraný uzol, konkrétne jeho fyzické telo tým, že mu je priamo upravená rýchlosť, prípadne naň pôsobíme silou. Po tom, ako je simulácia kolízie dokončená, sú pozície a rotácie príslušných uzlov upravené.

[6] SKTexture – https://developer.apple.com/reference/spritekit/sktexture
[7] Core Graphics – https://developer.apple.com/reference/coregraphics
[8] SKPhysicsBody – https://developer.apple.com/reference/spritekit/skphysicsbody
[9] SKPhysicsWorld – https://developer.apple.com/reference/spritekit/skphysicsworld
[10] SKFieldNode – https://developer.apple.com/reference/spritekit/skfieldnode

Kapitola 3

Použité technológie

V nasledujúcej kapitole bude popísané vývojové prostredie, jeho rozhranie a nástroje pre ladenie aplikácie. Ďalej bude popísaný zvolený programovací jazyk, zariadenie Apple TV a nakoniec niekoľko existujúcich hier, ktorými som sa inšpiroval pri návrhu aplikácie.

3.1 Vývojové prostredie Xcode

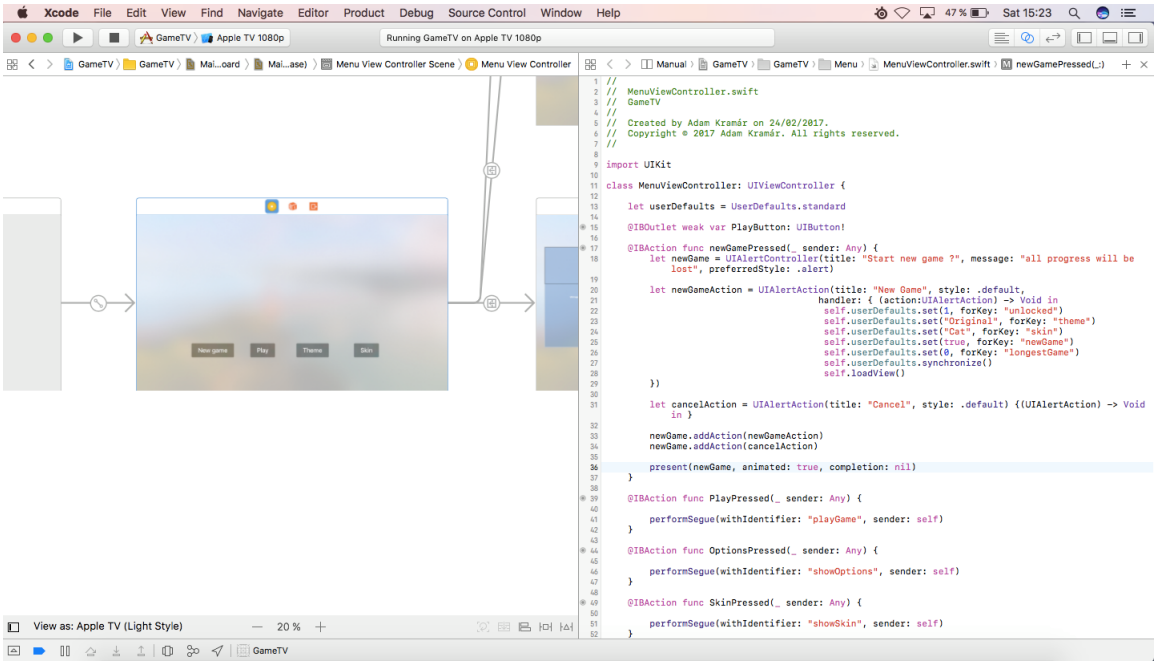
Xcode je vývojové prostredie spoločnosti Apple Inc., ktoré je voľne dostupné v App Store pre systém OS X. Obsahuje profesionálne vývojárske prostriedky na vývoj aplikácií pre operačné systémy OS X, iOS, watchOS a tvOS. Zoznam Xcodom podporovaných jazykov je Java, Python, Ruby, Rez, AppleScript, C, C++, Objective-C, Objective-C++ a Swift.

Xcode má zabudovaný emulátor zariadení, ako napríklad iPhone, iPad, Apple watch a Apple TV. Emulátor umožňuje rýchle testovanie a vytváranie prototypov aplikácie počas vývoja. Beží a správa sa ako štandardná Mac aplikácia počas simulovania prostredia, niektorého z vyššie spomenutých zariadení.

3.1.1 Rozhranie vývojového prostredia

Rozhranie integruje okrem možnosti písania kódu, aj návrh užívateľského rozhrania, ladenie a testovanie v emulátore. Bez ohľadu na použitý programovací jazyk, Xcode kontroluje syntax už počas písania a navrhuje možnosť automatického dopísania aktuálne písaného kódu, napríklad doplnenie názvu metódy.

Dôležitou súčasťou je aj InterfaceBuilder, ktorý umožňuje jednoduchý návrh užívateľského rozhrania pomocou skladania okien, pohľadov (view), prípadne iných elementov. Prechody medzi scénami a teda tok aplikácie, je možné stanoviť v StoryBoarde, z ktorého je možné prepojiť elementy s kódom pri použití prvku Assistant editor.



Obr. 3.1: Ukážka Assistant editoru

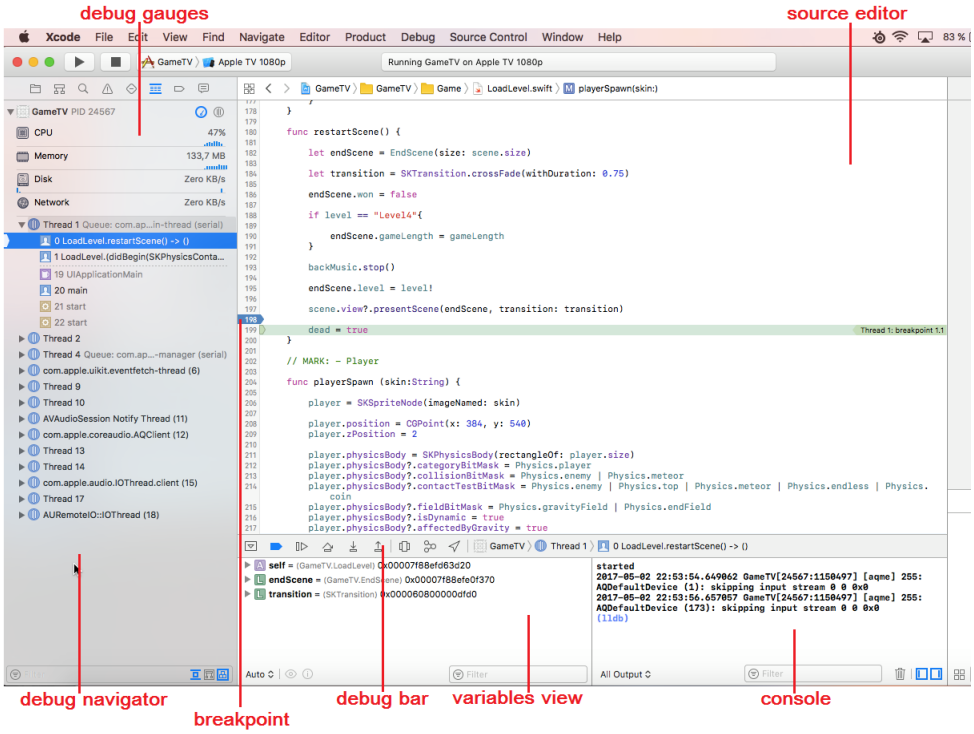
Po spustení aplikácie v emulátore, alebo na reálnom zariadení pripojenom k počítaču, Xcode okamžite spustí ladenie aplikácie. Nástroje na ladenie aplikácie poskytujú možnosť sledovať procesy, prípadne jeden proces a taktiež ukladanie záznamov rozhrania a profilovanie. Tieto nástroje poskytujú možnosť nájdenia chýb a miest v kóde, v ktorých aplikácia padá.

3.1.2 Debugger a ladenie programu

Všetky nástroje pre odhaľovanie chýb (debuggovanie) a ladenie aplikácie v Xcode sú integrované v hlavnom okne vývojového prostredia. Xcode debugger používa služby a funkcie poskytované podkladovým debuggerom LLDB[1] a ten je súčasťou LLVM[2] prekladača. LLDB je úzko prepojený s prekladačom a to poskytuje veľkú škálu možnosti so zachovaním jednoduchosti používania. Podľa manuálu [2] sa proces odhaľovania chýb delí na týchto päť častí:

1. Nájdenie samotného problému.
2. Lokalizovanie problému v kóde aplikácie.
3. Preskúmanie dátových štruktúr a riadenia toku bežiaceho kódu.
4. Aplikovanie zmien na kód, aby sa funkcionalita upravila do požadovaného stavu.
5. Spustenie aplikácie pre potvrdenie úspešného vyriešenia problému.

[1] LLDB – https://developer.apple.com/library/content/documentation/IDEs/Conceptual/gdb_to_lldb_transition_guide/document/Introduction.html
[2] LLVM – http://llvm.org/l



Obr. 3.2: Ukážka nástrojov pre ladenie

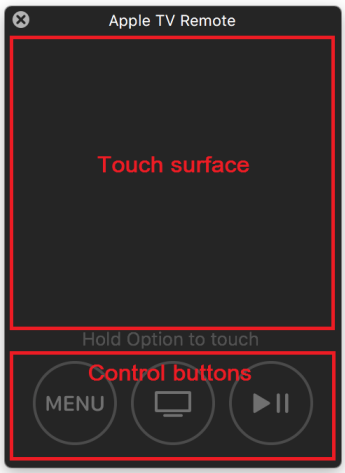
Nástroje poskytované Xcodom však nie sú špecificky určené pre spomenutú postupnosť. Niektoré z nástrojov sú napríklad zamerané na odhaľovanie problému (meradlá vyťaženia hardwaru), tie monitorujú bežiacu aplikáciu a výstup z nich je histogram, bežiaci v čase zľava doprava. Xcode celkovo poskytuje sedem meradiel a to: CPU, Pamäť, Energia, Disk, Sieť, GPU a iCloud. Tie sú zobrazované podľa platformy a typu aplikácie. Debug navigátor sa skladá z dvoch častí, prvá sú vyššie spomenuté meradlá vyťaženia. Druhá časť tvorí okno procesov, ktoré zobrazuje riadenie toku kódu. Iné nástroje slúžia napríklad na odhalenie lokality chyby v kóde (breakpoint). Breakpoint je mechanizmus pre pozastavenie aplikácie na vopred určenom mieste v kóde a preskúmanie riadenia toku aplikácie. Ďalším veľmi dôležitým nástrojom je konzola, ktorá slúži pre komunikáciu s aplikáciou. Počas toho, ako aplikácia beží, môže čítať zo štandardného vstupu (stdin) a vypisovať výstup na štandardný výstup (stdout) a na štandardný chybový výstup (stderr). V spodnej časti hlavného okna sa nachádza okrem konzoly aj ladiaca lišta (debug bar) a okno s premennými (variables view). Ladiaca lišta sa zobrazuje po začatí ladenia a obsahuje tlačidla pre otvorenie a zatvorenie celého spodného bloku ladiacich nástrojov, aktiváciu breakpointov, pozastavenie behu programu a krokovanie kódu. V okne variables view je možne skúmať stav premenných v dobe, keď je aplikácia pozastavená. Podľa príslušného označenia je možné rozlišovať medzi:

- **L** — Lokálna premenná.
- **A** — Argument.
- **S** — Statická premenná.
- **V** — Globálna premenná.
- **R** — Register.

- **V** — Inštančná premenná.
- **E** — Výraz.

3.1.3 Virtualizácia tvOS

Manuál [3], popisuje prácu s virtualizovaným systémom nasledovne.Pri používaní Apple TV štvrtej generácie používateľ pracuje so zariadením nepriamo pomocou diaľkového ovládača. Interakcia je založená na modely zamerania sa na prvok užívateľského rozhrania (focus model). Pri emulácií zariadenia je možne odsimulovať väčšinu akcií, ktoré užívateľ môže na zariadení vykonať. Emulátor taktiež umožňuje virtualizáciu diaľkového ovládača, čím poskytuje reálnejšie informácie o tom, ako bude aplikácia fungovať na skutočnom zariadení. Pomocou gest na ovládači systém rozhoduje o tom, aké akcie sa uskutočnia v užívateľskom rozhraní. Virtualizovaný ovládač má dve hlavné oblasti a to dotykovú plochu pre navigáciu a výber. V spodnej časti ovládača sú tri tlačidla pre spúšťanie akcií. Pre použitie virtuálneho ovládača je potrebné držať tlačidlo alt (Option) a prejsť kurzorom myši po dotykovej ploche ovládača.



Obr. 3.3: Vyrtuálny ovládač pre Apple TV

Celkovo emulátor umožňuje pracovať s tvOS tromi spôsobmi:

- **Navigácia pomocou klávesnice**
- **Virtuálny ovládač**
- **Skutočný ovládač pripojený pomocou technológie bluetooth**

Pre použitie fyzického ovládača sú dve možnosti, buď použiť Apple TV Remote, alebo herný ovládač. Je potrebné spárovať ovládač s počítačom pomocou bluetooth a pokiaľ je ovládač kompatibilný s Apple TV, emulátor ho dokáže identifikovať.

3.2 Programovacie jazyky

Od roku 1986 bol používaný jazyk Objective-C pre vývoj aplikácií bežiacich na zariadeniach spoločnosti Apple. Ide o objektovo orientovaný jazyk, ktorý je rozšírením jazyka C a tiež

využíva zasielanie správ jazyka Smaltalk. V roku 2014 bola predstavená alternatíva a to jazyk Swift[3].

Swift je viac účelový programovací jazyk pre iOS, macOS, watchOS a tvOS aplikácie a je založený na jazykoch C a Objective-C. Swift si osvojuje zaužívané programovacie vzory a pridáva niekoľko moderných vlastností, vďaka ktorým je programovanie ľahšie a flexibilnejšie. Napríklad nahrádza Smaltalkové zasielanie správ bodkovou notáciou, nevyžaduje použitie stredníku (;) na konci riadka a nie sú potrebné hlavičkové súbory. Je dizajnovaný pre prácu s Cocoa a Cocoa Touch frameworkom, budovaný LLVM (Low Level Virtual Machine) prekladačom a je možné s ním pracovať vo vývojovom prostredí Xcode od verzie 6.

Pri tvorbe aplikácií pre zariadenia od Apple je Swift alternatívou k jazyku Objective-C, ale preto že si osvojuje moderný koncept programovacích jazykov, má jednoduchšiu syntax. Cieľom jazyka Swift je vytvoriť, podľa možnosti, čo najlepší jazyk pre tvorbu mobilných aplikácii a systémové programovanie. Popularita Swiftu oproti jazyku Objective-C sa zvyšuje a má potenciál ho časom nahradiť. Syntaxou, jednoduchosťou a efektivitou je podobný skriptovacím jazykom.

3.3 Apple TV

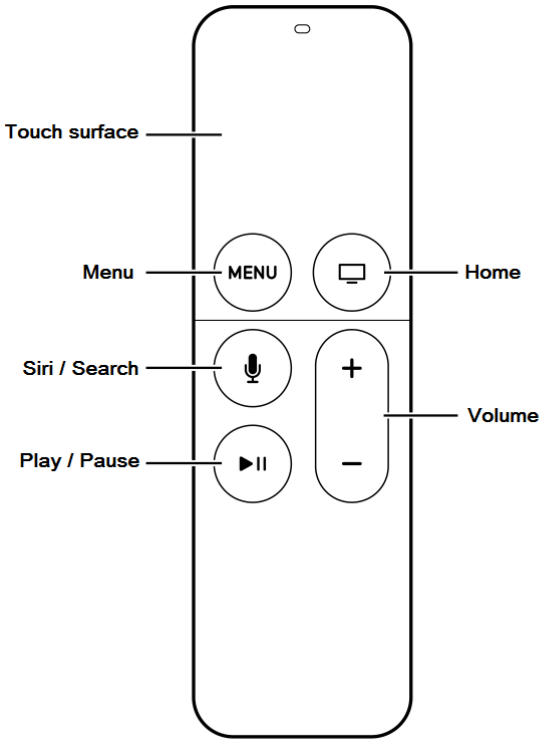
Apple TV je prehrávač digitálnych médií a mikrokonzola vytvorená spoločnosťou Apple Inc. Ide o malé zariadenie schopné primať dáta z rôznych zdrojov a prenášať ich do televízie. Apple TV je HDMI-kompatibilné zariadenie, čiže pre umožnenie prenosu do televízie musí byť zariadenie pripojené cez HDMI (High-Definition Multimedia Interface) kábel. Apple TV neobsahuje žiadne integrované ovládanie, preto musí byť použitý diaľkový ovládač, prípadne zariadenie s operačným systémom iOS a aplikáciou Apple TV Remote, ktorá je dostupná z App Storu. Uvedené informácie som čerpal z manuálu [1].

Hardware:

- 64-bit A8 procesor
- 32 GB alebo 64 GB pamäte
- 2 GB RAM
- 10/100 Mbps Ethernet
- WiFi 802.11a/b/g/n/ac

Odkedy Apple TV využíva tvOS, používatelia môžu hrať hry, používať aplikácie sociálnych sieti a pozerať filmy. Toto prináša nové možnosti pre vývoj aplikácií. Konkrétne sa pozrime na možnosti vývoja hier. Pri návrhu hry je treba zobrať do úvahy, že jediná možnosť ovládania je prostredníctvom diaľkového ovládača, alebo zariadenia s aplikáciou Apple TV Remote. Ovládač sa skladá z dotykovej plochy, na ktorej je možne zadávať rôzne gestá a pod ňou sa nachádza šesť tlačidiel s rôznymi funkciami. Okrem toho je v ovládači zabudovaný gyroskop a akcelerometer, čo určite otvára nové možnosti pri tvorbe aplikácií.

[3]Swift – https://en.wikipedia.org/wiki/Swift_(programming_language)



Obr. 3.4: Schéma Apple TV Remote

3.4 Existujúce hry

Medzi súčasne najpopulárnejšie hry pre Apple TV podľa článku [9] patria zložitejšie univerzálne hry vytvorené pomocou knižnice `Metal`[4], ktorá poskytuje prístup ku GPU a tým umožňuje vytvoriť komplexné hry s náročnejšou grafikou. Je však možné vidieť aj zastúpenie jednoduchších hier, vytvorených pomocou knižnice `SpriteKit`, s ktorou budem pracovať pri tvorení mojej hry. Následne sa bližšie pozrieme na niektoré z hier, ktoré som pri návrhu analyzoval.

3.4.1 Jetpack Joyride

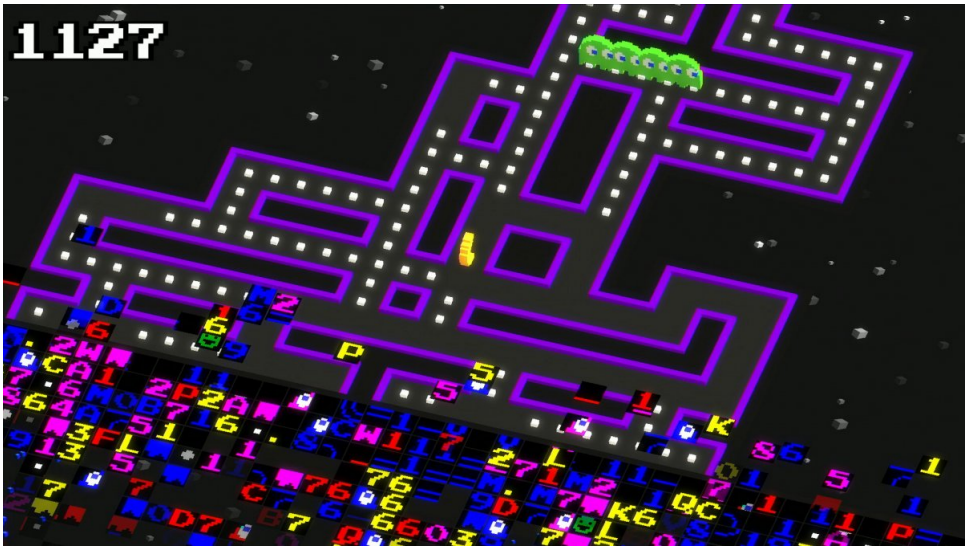
Jetpack Joyride je hra v štýle lietania v nekonečnom levely a uhýbania sa prekážkam ozvláštnená o zbieranie mincí, rôznych bonusov a vylepšení. Koncept hry je jednoduchý, hra sa dá rýchlo pochopiť a tým vie zaujať väčšie množstvo hráčov. Táto aplikácia mi slúži, ako inšpirácia pri návrhu konceptu ovládania, ktorý v skratke pozostáva z automatického pohybu scény a uhýbaniu sa prekážkam pohybom na y-ovej osi (pohyb k hornému, alebo dolnému okraju scény). Ďalej som do návrhu zahrnul nutnosť zbierania mincí, pre možnosť postupu v hre.

[4]Metal – https://developer.apple.com/reference/metal

Obr. 3.5: Screenshot z hry Jetpack Joyride

3.4.2 Pac-Man 256

Pac-Man je klasická arkádová hra, ktorú poznáme už od roku 1980. Verzia hry pre zariadenia spoločnosti Apple sa v princípe zhoduje s pôvodnou hrou. Princíp je taký, že Pac-Man pri pohybe po mape pojedá bodky a vyhýba sa duchom. Hra je rozšírená o modernejšiu grafiku a rôzne elementy ako je zbieranie objektov s náhodným výskytom na mape. To spôsobuje akcie, ktoré môžu zásadne ovplyvniť priebeh hry. Zaujímavá vlastnosť tejto hry, ktorou som sa taktiež inšpiroval, je prínos určitej miery náhodnosti do hry. Konkrétne som návrh rozšíril o náhodne generované skupiny prekážok.



Obr. 3.6: Hra Pac-Man 256 [12] — Záber z hry

Kapitola 4

Návrh aplikácie

Táto kapitola obsahuje popis mnou navrhnutého riešenia doplnený o návrh užívateľského rozhrania a náčrt scény na obrázku 4.2. Pri navrhovaní aplikácie som sa snažil zachovať jednoduchosť užívateľského rozhrania a zároveň dať budúcim používateľom možnosť si hru prispôsobiť.

4.1 Návrh vlastného riešenia

Pri návrhu koncepcie hry som vychádzal z poznatkov zistených pri analyzovaní už existujúcich hier. Základnou úlohou hráča je navigovať svoju postavu mapou tak, aby sa vyhol všetkým prekážkam a zároveň pozbieral čo najviac mincí. Hráč bude pozorovať scénu z pohľadu tretej osoby a bude mať k dispozícii aktívny pohyb nahor. Ďalej pre postup môže využívať pôsobenie gravitácie a iných polí. Prechod hrou bude rozdelený na dve hlavné časti a to tak, že prvá časť sa bude skladať zo série levelov. Levely budú na seba závislé v tom zmysle, že ich bude treba postupne odomykať. V druhej časti hry bude jeden nekonečný level, v ktorom pôjde o získanie čo najväčšieho skóre.

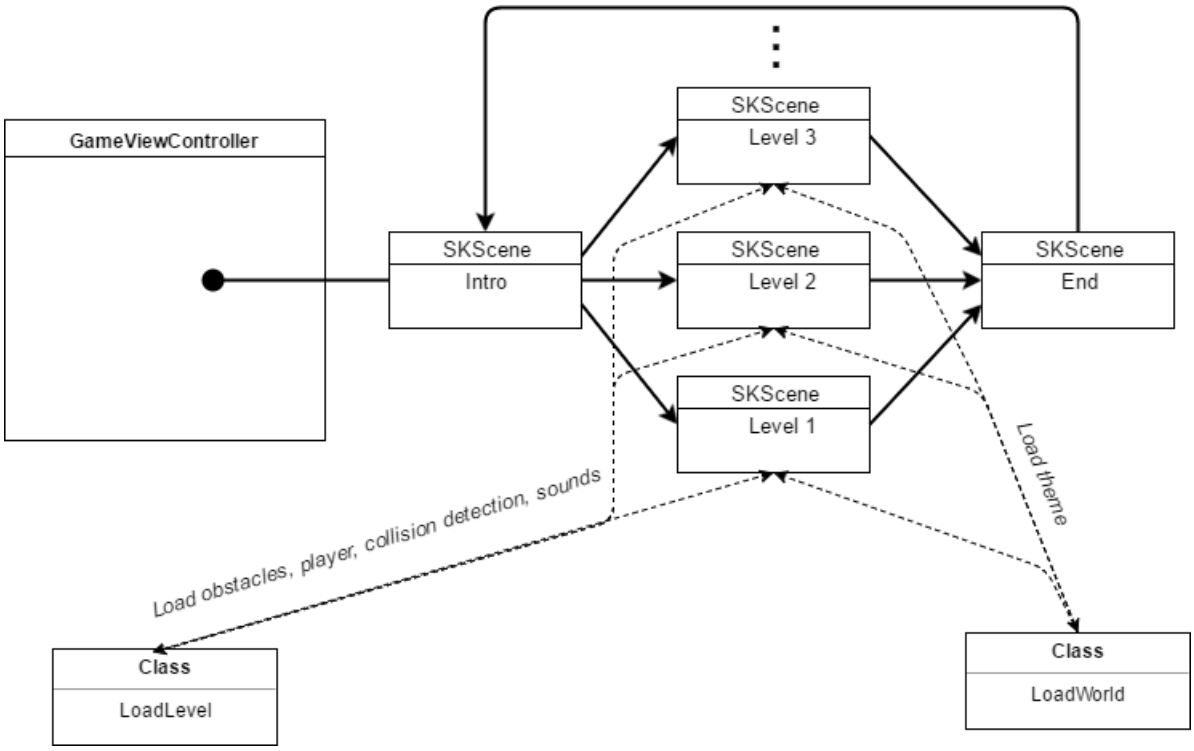
Jedno zariadenie Apple TV na rozdiel od mobilných telefónov a tabletov zvyčajne používa viac používateľov a preto by mal mať každý z hráčov možnosť prispôsobiť si hru podľa svojich preferencii a to tak, že si vyberie jednu z ponúkaných tém. To bude mať vplyv na farbu pozadia, tvar a farbu prekážok a zvuky ktoré budú použité.

Všetky témy, z ktorých bude možné vyberať, som navrhoval tak, aby pri hre bolo možné ľahko rozlíšiť čo je prekážka, ktorej je potrebné sa vyhnúť a naopak, ktoré miesta sú bezpečné. Hráč bude mať vždy iba jeden život, teda pri kolízii s prekážkou pokus prejsť úroveň končí a bude nutné začať celý level od začiatku. Úroveň končí tým, že sa hráč vyhne všetkým prekážkam. Nie všetky prekážky bude možné vidieť priamo, ale budú reprezentované pomocou rôznych silových polí (magnetické, gravitačné). Okrem toho sa v scéne budú objavovať aj mince, ktoré bude naopak potrebné pozbierať. Pre odomknutie nového levelu bude okrem prejdenia na koniec súčasného levelu potrebné aj pozbierať dostatočný počet mincí, inak sa level neodomkne. Zmena nastáva pri dosiahnutí posledného levelu, v ktorom sa žiadne mince nenachádzajú. Level je nekonečný a prekážky sa v ňom náhodne generujú, takže pri každom pokuse je level iný. Cieľom je vydržať čo najdlhšie bez narazenia do akejkoľvek prekážky, pričom skóre sa počíta podľa sekúnd strávených pri danom pokuse. Scéna sa postupne zrýchľuje, čo ma za následok zvýšenie náročnosti.

4.1.1 Štruktúra hry

Navrhol som rozdelenie hry medzi niekoľko scén, z nich sú dve spoločné pre všetky úrovne a to `Intro` a `End`. V scéne `Intro` bude zobrazený náhľad zvoleného levelu, témy a základné informácie o danom levely, ako napríklad potrebný počet mincí pre otvorenie ďalšej úrovne. Do `End` scény sa bude dať dostať vždy po úspešnom, alebo neúspešnom ukončení pokusu prejsť úroveň. Okrem týchto scén bude mať každý level svoju vlastnú scénu, v ktorej je definovaný priebeh. Takýmto spôsobom je možne jednoducho vytvárať nové úrovne a pridávať ich k existujúcim. Menší rozdiel oproti tomuto systému sa bude nachádzať pri zostavovaní nekonečného levelu, kde bude vytvorených niekoľko skupín prekážok a tie budú náhodne skladané po spustení úrovne. Všetky scény búdu prezentované vo ViewControllery[1] a bude ich možné meniť pomocou prechodov.

Ďalšou kľúčovou časťou budú triedy pre načítanie zvolenej témy, kontrolovanie fyzického kontaktu, vytvorenie prekážok a akcií, vykreslenie objektov do scény a prehrávanie zvukov. Všetky tieto požadované funkcie bude vhodné rozdeliť do viacerých tried pre zachovanie prehľadnosti. Počiatočný návrh počíta s rozdelením minimálne do dvoch tried. Scény budú s týmito triedami komunikovať a tým sa zabezpečia jednotné podmienky pre všetky úrovne v hre.



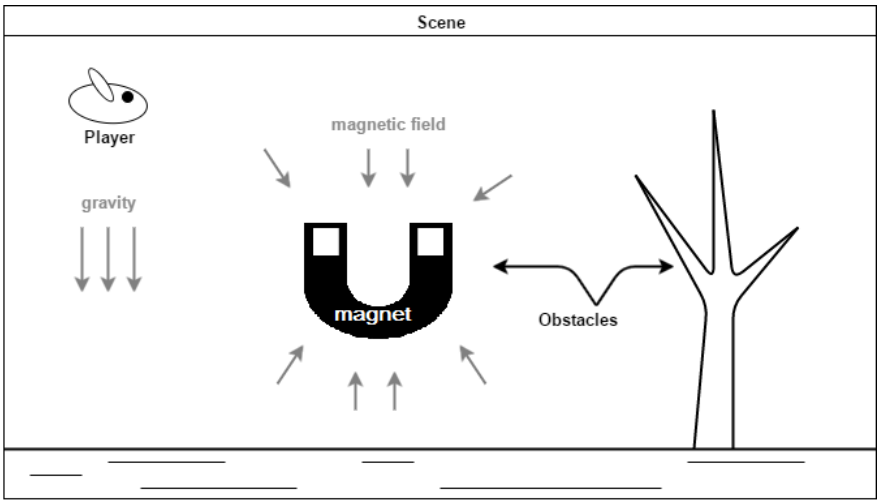
Obr. 4.1: Štruktúra návrhu scén a tried

4.1.2 Návrh scény

Po stanovení štruktúry som si vytvoril náčrt scény so základnými prvkami, ako je hráč, niekoľko prekážok a smery pôsobenia silových polí. Hráč bude mať možnosť aktívneho pohybu len smerom k hornému okraju obrazovky, scéna sa bude automaticky pohybovať smerom

[1] UIViewController – https://developer.apple.com/reference/uikit/uiviewcontroller

od pravého okraju obrazovky k ľavému okraju. Scény sa pohybujú plynulým tempom, s výnimkou nekonečného levelu, v ktorom sa rýchlosť stupňuje s uplynutým časom. Samotný pohyb v scéne bude realizovaný pohybom prekážok určitou rýchlosťou smerom k ľavému okraju scény. Hráč sa po x-ovej osi pohybovať nebude, s výnimkou pohybu spôsobeného silovým polom.

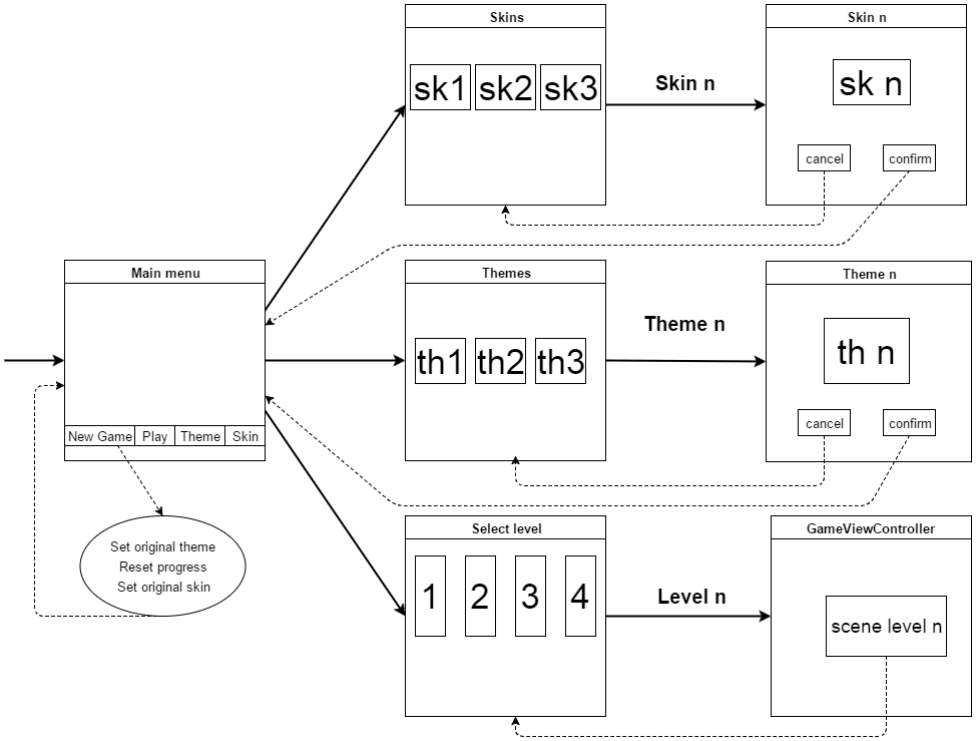


Obr. 4.2: Náčrt scény

4.2 Návrh užívateľského rozhrania

Aplikácia okrem hernej časti bude obsahovať aj menu zložené s niekoľkých ViewControllerov, jedným bude úvodná obrazovka, alebo hlavné menu, z ktorého sa bude možné dostať do všetkých ostatných častí. Každá z nich sa skladá z dvoch častí a to z nadhľadu možností, z ktorého bude možné sa dostať do na detailný pohľad, výberom niektorej z položiek. Detailný pohľad bude poskytovať náhľad zvolenej položky, možnosť potvrdenia výberu a možnosť zrušenia výberu. Pomocou týchto výberov bude možné meniť prvky hry, ako sú téma hry a vzhľad hráča.

Ďalší prvok menu bude výber levelu, počas ktorého, po zvolení možnosti z nadhľadu, nebude nasledovať detailný pohľad. Namiesto toho sa aplikácia prepne do hernej časti, kde bude prezentovaná prvá scéna.



Obr. 4.3: Návrh štruktúry menu

Kapitola 5

Implementácia

Pre implementáciu mojej hry som sa rozhodol použiť jazyk Swift, ktorý je popísaný v kapitole 3.2. Vývoj bol realizovaný na zariadení Macbook Air a bolo použité vývojové prostredie Xcode. V nasledujúcej kapitole budem popisovať implementačné detaily jednotlivých častí programu. Pre začiatok bude spomenutý postup pri vývoji užívateľského rozhrania. Následne popíšem postup zvolený pri implementovaní samotnej hry.

5.1 Užívateľské rozhranie

Hlavnou úlohou pri tvorbe grafického užívateľského rozhrania bolo vytvoriť ho čo najjednoduchšie, aby užívateľ trávil v menu čo najmenej času. Základná obrazovka, čiže hlavné menu, bolo spracované, tak aby obsahovalo minimálny počet prvkov, konkrétne štyri. Funkcionalita prvého s názvom `New Game` je vytvorenie novej hry a to znamená, že sa zmaže všetok doposiaľ nahratý postup a nastaví sa základná téma. Druhé tlačidlo s názvom `Play` používateľa presmeruje do menu v ktorom si môže vybrať level. Toto menu je riešené pomocou `UICollectionViewController`[1], kde každá bunka obsahuje odkaz na jeden z dostupných levelov. Po výbere jednej z možnosti sa spustí hra vo zvolenej úrovni. Posledné dva prvky pre výber témy a skinu hráča sú podobné, preto ich popíšem spoločne. Po výbere jednej z týchto dvoch možnosti sa zobrazí taktiež `UICollectionViewController` s ponukou možností a po zvolení jednej z možností sa zobrazí detailný náhľad tejto položky. Z detailného náhľadu je možné, buď potvrdiť vybranú položku a vrátiť sa automaticky do hlavného menu, alebo vyber zrušiť a vrátiť sa do predchádzajúceho ViewControlleru. Zmeny vykonané výberom témy, či skinu, zostávajú nezmenené aj po vypnutí a zapnutí aplikácie. Viac o spôsobe ukladania tých volieb a rovnako o ukladaní postupu v hre je popísané v podkapitole 5.2.4. Keďže ide o aplikáciu pre tvOS, spätná navigácia v menu nie je riešená klasickým spôsobom známym z aplikácii pre systém iOS. Pokiaľ sa chce užívateľ vrátiť na predchádzajúce okno, nerobí tak pomocou tlačidla v ľavom hornom obrazovky, ale priamo pomocou zabudovaného tlačidla na ovládači TV Remote.

Samotná tvorba rozhrania prebiehala v Storyboarde. Je to grafický editor, v ktorom je možné zostaviť všetky ViewControllery a následne v nich prvky, ako sú tlačidla, obrázky a podobne. V Storyboarde je možne vytvoriť aj prechody medzi oknami, ktoré sa volajú Segue. Do stromu okien som pridal aj `Navigation Controller`[2], ktorý zjednodušuje pri-

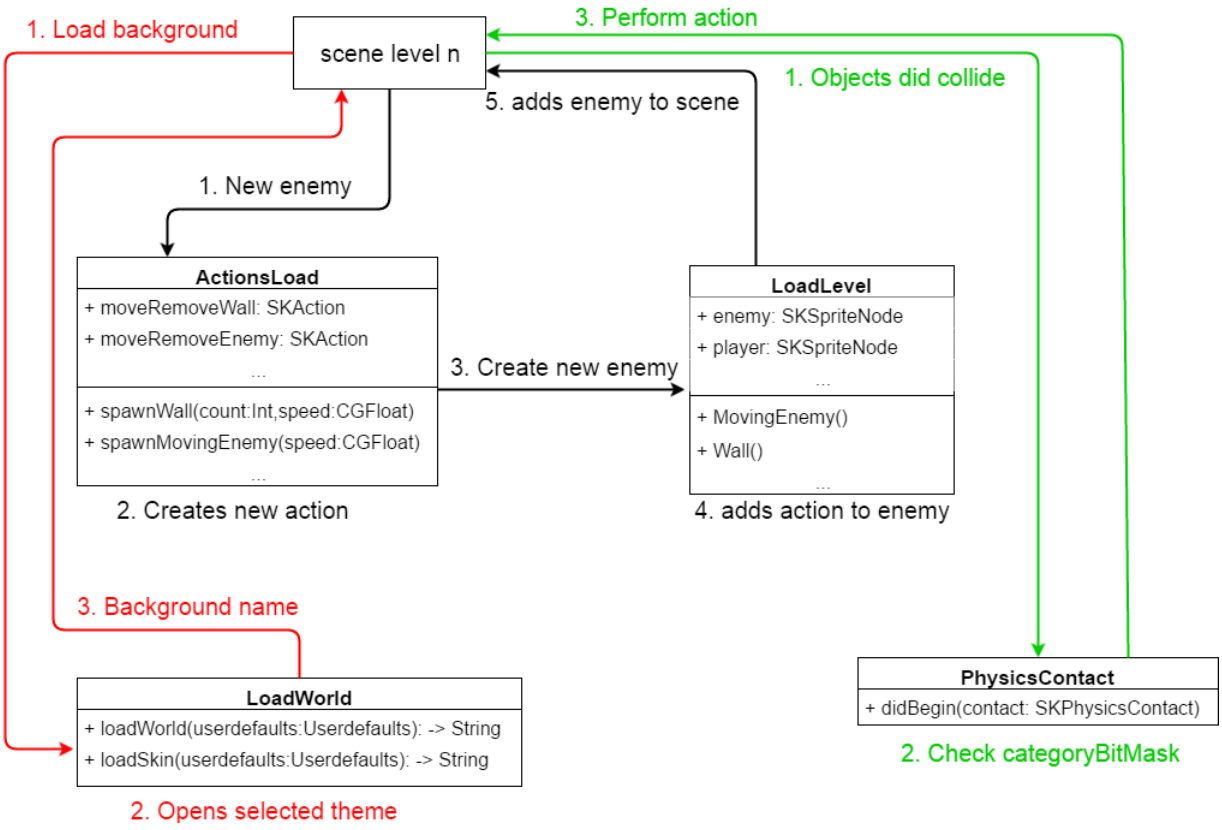
[1] UICollectionViewController – https://developer.apple.com/reference/uikit/uicollectionviewcontroller

[2] Navigation Controller – https://developer.apple.com/library/content/documentation/WindowsViews/Conceptual/ViewControllerCatalog/Chapters/NavigationControllers.html

dávanie nadpisov a taktiež z dôvodu, že umožňuje automatické skoky z vnorených prvkov na koreňové okno, bez nutnosti vytvárania nového prechodu.

5.2 Implementácia hry

Po vytvorení menu so všetkými nastaveniami som začal tvoriť samotnú hru. Ako prvý som si vytvoril návrh a ten som postupne upravoval a dolaďoval popri tvorení hry. Konečné riešenie spočívalo v rozdelení každého levelu do troch scén a rozdelenie všetkých funkčných časti, ako sú detekcie kolízii, pridávanie uzlov, vytváranie akcii, načítanie pozadia podľa témy a prehrávanie zvukov medzi štyri triedy. Všetky triedy a ďalšie postupy budú popísane v nasledujúcich sekciách.



Obr. 5.1: Ukážka rozdelenia funkčnosti do tried

5.2.1 Pridávanie objektov do scény

Pre pridanie objektov do scény slúži trieda `LoadLevel` a čiastočne aj trieda `ActionsLoad`. Pomocou týchto dvoch funkcií sú pridávané do jednotlivých scén všetky objekty, čiže samotný hráč, prekážky, mince a objekt, pomocou ktorého je ukončovaný level. Pri nekonečnom levely je tento objekt nahradený objektom pre opakovane pridanie náhodnej skupiny prekážok do scény.

Trieda `ActionsLoad` obsahuje metódy pre vytváranie všetkých akcii, ktoré sú neskôr priraďované uzlom. Väčšina z nich predstavuje akcie typu moveRemoveSomething, čo znamená, že daný uzol sa presunie cez celu dĺžku scény a následne je odstránený. Po tom, ako

je pripravená akcia, vytvorí sa potrebný uzol (môže ich byť aj viac rovnakého druhu) a je mu daná akcia priradená. Uzlu sa priraďuje okrem akcie aj počiatočná pozícia, veľkosť a fyzické telo. Vďaka fyzickému telu je možné aby uzly mali medzi sebou kolízie, o čom podrobnejšie píšem v podkapitole 5.2.2. Po tom, ako je uzol vytvorený a sú mu priradené všetky vlastnosti a potrebné akcie, je uzol pridaný do scény.

Trieda `LoadLevel` obsahuje metódy pre vytvorenie novej hry, pre reštartovanie bežiacej hry a pre načítanie obrázkov podľa zvolenej témy. Okrem toho sa stará o pridávanie efektov v podobe rôznych emiterov a prehrávanie zvukov. Spustenie novej hry spočíva v tom, že sa zo scény odoberú všetky uzly a všetky akcie a následne sa znovu načíta celý level.

Pomocná trieda `LoadWorld` slúži na zistenie názvu pozadia podľa vybratej témy, metódy pre načítanie samotného pozadia sa nachádzajú v triede `LoadLevel`. Obdobná situácia platí pre názov zvolenej textúry postavy hráča.

5.2.2 Kolízie objektov v scéne

Uzlom, s ktorými chceme pracovať ako s fyzickými objektmi, musíme vytvoriť a prideliť fyzické telo. Fyzické telo je objekt typu `SKPhysicsBody` a priraďuje sa uzlom do vlastnosti `physicsBody`. Keď scéna spracúva snímku, aplikuje vplyv fyziky na telo pripojené k uzlu. Tieto vplyvy zahrňujú pôsobenie gravitácie, trenia a kolízie s inými objektmi. Na fyzické telo môžu byť aplikované aj iné sily prostredníctvom silových polí, prípadne priamo impulzy.

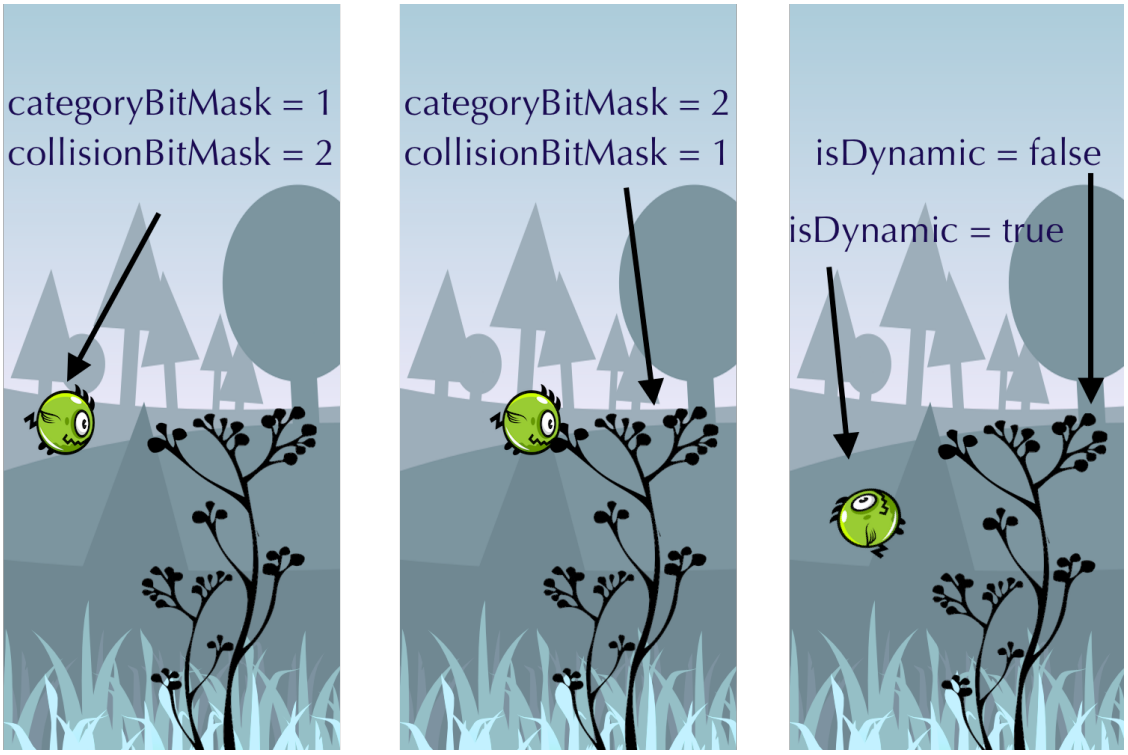
Po tom, ako je uzlu pridelené fyzické telo, je potrebné mu nastaviť niekoľko vlastností. Ako prvé je potrebné nastaviť kategóriu objektu a to sa robí pomocou vlastnosti `categoryBitMask`. Do rovnakej kategóriu môže patriť ľubovoľné množstvo rozdielnych uzlov. Úzko prepojená s kategóriou je vlastnosť `collisionBitMask` a jedná sa o masku definujúcu, ktorá kategória fyzických tiel môže mať kolíziu s týmto fyzickým telom. Keď sa dve tela dostanú do vzájomného kontaktu, môže nastať kolízia. V tom prípade sa porovná maska kategórie jedného z tiel s maskou kolíznej kategórie druhého tela a uskutoční sa logická operácia AND. Pokiaľ je výsledok operácie nenulový, telá sú ovplyvnené kolíziou. Nie vždy je však žiaduce, aby sa objekty pri kontakte fyzicky zrazili a preto existuje ďalšia vlastnosť `contactTestBitMask`. To je maska ktorá definuje, ktoré kategórie tiel spôsobujú oznámenia o križovaní sa s daným telom. Pokiaľ dve tela zdieľajú rovnaké miesto v danom čase, podobne ako pri kolízii, sa porovnávajú bitové masky kategórie a kontaktu logickou operáciou AND. V prípade nenulového výsledku je vytvorený objekt `SKPhysicsContact`[3] a je predaný delegátovi.

Funkciu delegáta fyzického sveta v mojej aplikácii zastupuje trieda `PhysicsContact`. Delegát je volaný, keď dve fyzické tela prídu do vzájomného kontaktu a zároveň maska kategórie jedného tela sa zhoduje s maskou kolízie druhého. Trieda obsahuje jednu metódu s názvom *didBegin()* a na vstupe prijíma objekty typu `SKPhysicsContact`. Potom, na základe toho o kolíziu akých dvoch tiel ide, uskutoční požadovanú akciu.

Podobne sa pracuje aj so silovými poľami s tým rozdielom, že pre aplikovanie vplyvov pola na ľubovoľný uzol, musíme fyzickému telu tohto uzla nastaviť vlastnosť `fieldBitMask`. Je to maska definujúca, ktorými poľami môže byť fyzické telo ovplyvnené. Funguje na rovnakom princípe ako fyzický kontakt medzi uzlami.

Fyzickému telu je možne prideľovať aj iné vlastnosti, ako napríklad vplyv gravitácie, možnosť byť ovplyvnený kolíziou, možnosť rotácie a mnohé iné.

[3] SKPhysicsContact – https://developer.apple.com/reference/spritekit/skphysicscontact

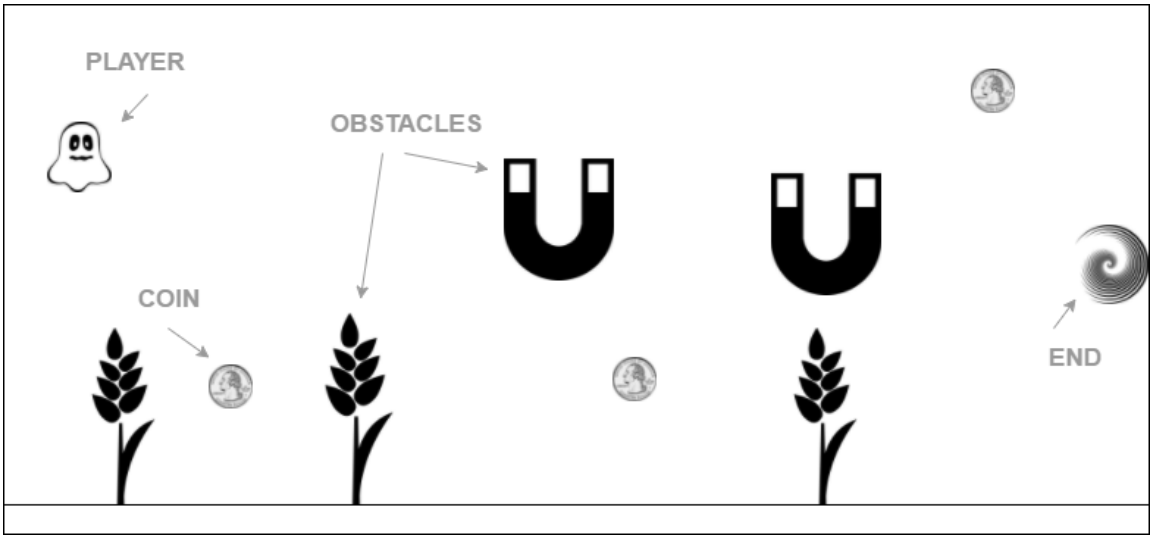


Obr. 5.2: Úkážka kolízie hráča s prekážkou

5.2.3 Štruktúra levelov

Štruktúra konečných levelov je vopred stanovená s menšou mierou náhodnosti pri niektorých prekážkach. Za vytvorenie a pridanie jedného druhu prekážky do scény je zodpovedná jedná metóda (*enemy()*, *movingEnemy()*), ktorá prípadne komunikuje s inými metódami typu moveRemove pre vytvorenie pohybovej akcie. Samotnú štruktúru, alebo podobu levelu som tvoril pomocou akcie `wait`, ktorá čaká náhodný čas s tým, že je možné jej zadať priemernú čakaciu dobu. To znamená, že po tom, ako je akcia spustená, čaká a sama sa následne ukončí. Vždy, keď je akcia spustená, vypočítava sa nový náhodný čas čakania, ktorý sa pohybuje v rozsahu polovice zadaného stredného času. Po ukončení sa vykoná cely blok príkazov, ktorý táto akcia obaľuje, napríklad pridanie novej prekážky do scény, alebo ukončenie daného levelu.

Pri poslednej, teda nekonečnej úrovni, nastáva menšia zmena. Level nie je možné zostaviť celý dopredu, ale možné pripraviť si skupiny prekážok, ktorých tvorba je založená na rovnakom princípe, ako tvorba celých prechádzajúcich levelov. Po prekonaní jednej skupiny sa náhodne vyberie ďalšia. Podľa toho, koľko takýchto skupín už hráč prekonal, sa pohyb scény a vlastne všetkých prekážok zrýchľuje.



Obr. 5.3: Ukážka štruktúry prvého levelu

5.2.4 Ukladanie dát

Trieda `NSUserDefaults`[4] poskytuje programovacie rozhranie pre prácu s nastaveniami aplikácie. Tieto nastavenia majú vplyv na to, ako sa aplikácia chová, tým pádom poskytujú určitú možnosť pre užívateľa, aby si ju prispôsobil. Napríklad umožňuje používateľovi zmenu jednotiek, v ktorých sa zobrazujú hodnoty, avšak pre moju aplikáciu bolo vhodnejšie pomocou tejto triedy umožniť používateľovi zmenu témy a vzhľadu hernej postavy. Okrem toho slúži pre ukladanie aktuálneho progresu v hre a ukladanie najvyššieho dosiahnutého skóre.

Počas behu aplikácie je možne použiť `NSUserDefaults` objekt na prečítanie základných nastavení, ktoré aplikácia používa. Tieto nastavenia sa nachádzajú v databáze. Informácie z databázy sú uložené v cache pamäti, aby nebolo potrebne pristupovať do databázy pri každom získavaní hodnoty. Metóda *synchronize()* je automaticky volaná v periodických intervaloch a tým sa udržuje synchronizácia medzi cache pamäťou a databázou. Dáta sú ukladané a následne získavané pomocou kľúčov.

[4]NSUserDefaults – https://developer.apple.com/reference/foundation/userdefaults

Kapitola 6

Testovanie

Testovanie bolo najviac opakovanou činnosťou počas celého procesu implementácie navrhnutej hry. Pri vývoji a po dokončení aplikácie bolo testovanie uskutočňované v prostredí Xcode, ktoré obsahuje zabudovaný emulátor zariadení od Apple ako je iPhone, iPad a hlavne pre účely testovania mojej aplikácie emulátor zariadenia Apple TV štvrtej generácie s operačným systémom tvOS. Po dokončení testovania v emulátore bola aplikácie otestovaná na reálnom zariadení. Nasledujúcej kapitole budem popisovať zistené skutočnosti pri testovaní.

6.1 Testovanie v emulátore

Väčšina testov počas programovania aplikácie bola uskutočnená vo vývojov prostredí Xcode na už spomínanej vstavanej aplikácii emulátor s použitím virtuálneho ovládača. Takýto spôsob testovania však nemôže nahradiť testovanie na reálnom zariadení, pretože odozva virtuálneho ovládača nie je zhodná s reálnym ovládačom. Aj napriek tomu väčšina testov bola uskutočnená v emulátore z dôvodu rýchlejšieho spustenia a taktiež možnosti používať klávesnicu pri navigácii v menu, čo podstatne šetrilo čas.

Obr. 6.1: Virtualizácia zariadenia Apple TV s ovládačom

Testované boli vždy menšie časti kódu hneď po tom, ako boli implementované, tým som sa snažil zabrániť nahromadeniu chýb. Po tom, ako bolo otestovaných niekoľko, alebo

všetky menšie časti programu, aplikácia bola testovaná ako celok, čo pomohlo odhaliť zásadné chyby, ako nesprávne implementované rozhrania tried, či nevhodne navrhnuté časti úrovni hry. Chyby v rozhraniach tried boli zložitejšie na odhalenie, pretože sa neprejavovali pri každom teste. Pri ladení bola nápomocná konzola prostredia Xcode a taktiež variables view, kde sa zobrazovali aktuálne hodnoty všetkých premenných v pamäti. Záverečné testy vo virtualizovanóm prostredí som rozdelil na dve časti a to testy užívateľského rozhrania a testy hrateľnosti. Záverom sú spomenuté aj hardwarove nároky hry, avšak ide len o orientačné údaje pretože aplikácia nebola spustená na zariadení Apple TV, ale na Macbooku.

Testovanie užívateľského rozhrania

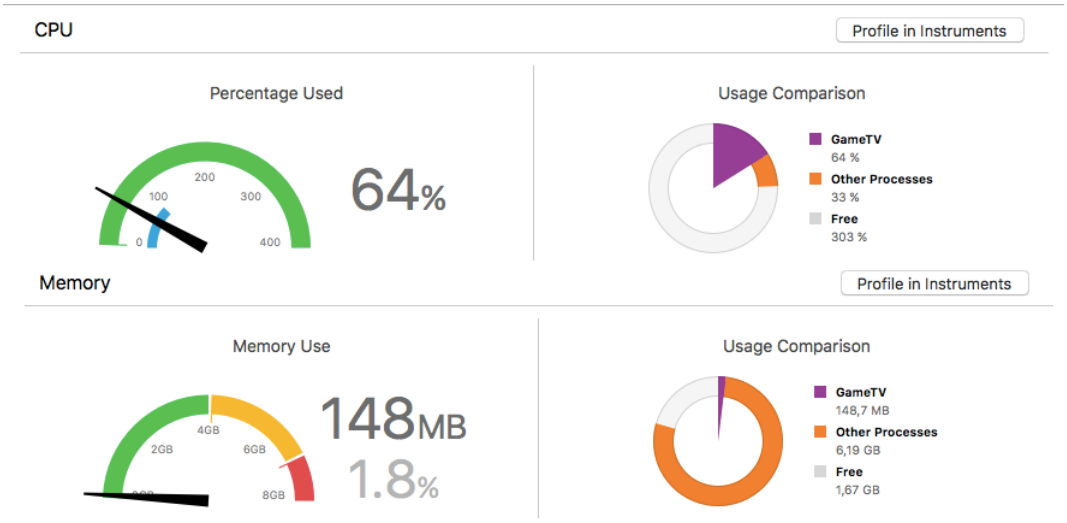
Pri testoch rozhrania som sa zameral na pohyb užívateľa v menu aplikácie. Hlavne nastavovanie témy, vzhľadu hernej postavy a následne na odomykanie levelov. Pri testovaní nastavenia témy bolo nutné, aby zmeny zostali zachované ako po spustení ľubovoľnej úrovne, po jej dokončení a rovnako po vypnutí a zapnutí aplikácie. Pre ukladanie týchto dát som použil triedu `NSUserDefaults`, ktorá je popísaná v kapitole 5.2.4. Záverečné testy užívateľského rozhrania v emulátore neodhalili žiadne zásadné chyby, ale nezanedbateľnou súčasťou výsledku budú aj testy na reálnom zariadení, ktoré sa chová podstatne inak.

Testovanie hrateľnosti

Pri testovaní hry som navrhol niekoľko spôsobov ovládania hernej postavy, ktoré som postupne obmieňal a vyberal to najvhodnejšie. Pri testovaní na virtualizovanom zariadení nebol pozorovateľný takmer žiaden rozdiel v zvolených metódach ovládania, avšak pri reálnom zariadení môže hrať odozva ovládača veľkú rolu a preto je pre účely testovania k dispozícii viacej možnosti. Okrem ovládania som sa pokúsil zamerať na nechcené situácie, ako je napríklad prípad zmiznutia hernej postavy zo zobrazovanej scény. Používateľ nie je schopný sa z takého stavu hry dostať, keďže ma možnosť aktívneho pohybu iba jedným smerom. Problém bol vyriešený pridaním neviditeľných stien na hraniciach scény, ktoré pôsobia ako nárazníky a v pripadne kontaktu s nimi je hráč odrazený späť do scény. Vzhľadom k tomu, že emulátor nedokáže prehrať aplikáciu tak kvalitne ako reálne zariadenie, môžu byť výsledky testov zavádzajúce a preto je potrebné uskutočniť celkové testovanie aplikácie na reálnom zariadení.

Testovanie hardwarových nárokov

Aplikácia je cielená špeciálne pre zariadenie Apple TV a preto nie je potrebné brať do úvahy situáciu, kde by musela bežať na zariadení so slabším výkonom, ako tomu môže byť pri tvorbe obdobnej aplikácie pre zariadenia s operačným systémom iOS. Pri týchto zariadeniach treba brať do úvahy, že rozdiel v hardwarovom vybavení jednotlivých modelov už nie je zanedbateľný. Pomocou módu ladenia vo vývojovom prostredí Xcode je možné sledovať percentuálne vyťaženie procesora aplikáciou a taktiež využitie pamäte. Podľa grafov využitia CPU a pamäte je možné vidieť, že výsledné nároky sú relatívne nízke a samotné využitie procesora sa pohybuje v rozmedzí 60-70%. Tieto údaje zobrazujú využitie hardwaru pri testovaní v emulátore na Macbooku a graf, ktorý zachytáva vyťaženie zariadenia Apple TV je možné vidieť v podkapitole 6.2.1 na obrázku 6.3.



Obr. 6.2: Zaťaženie hardwaru pri testovaní

6.2 Testovanie na reálnom zariadení

Po dokončení testovania v emulátore som zahájil testy na reálnom zariadení a to najprv samostatne, za účelom odhalenia rozdielov oproti predošlým testom a overenia funkčnosti aplikácie. Následne bola testovaná samotná hrateľnosť s využitím Apple TV Remote a aplikacíe pre zariadenia s operačným systémom iOS, ktorá umožňuje použiť zariadenie ako dialkový ovládač.

6.2.1 Záverečné testovanie

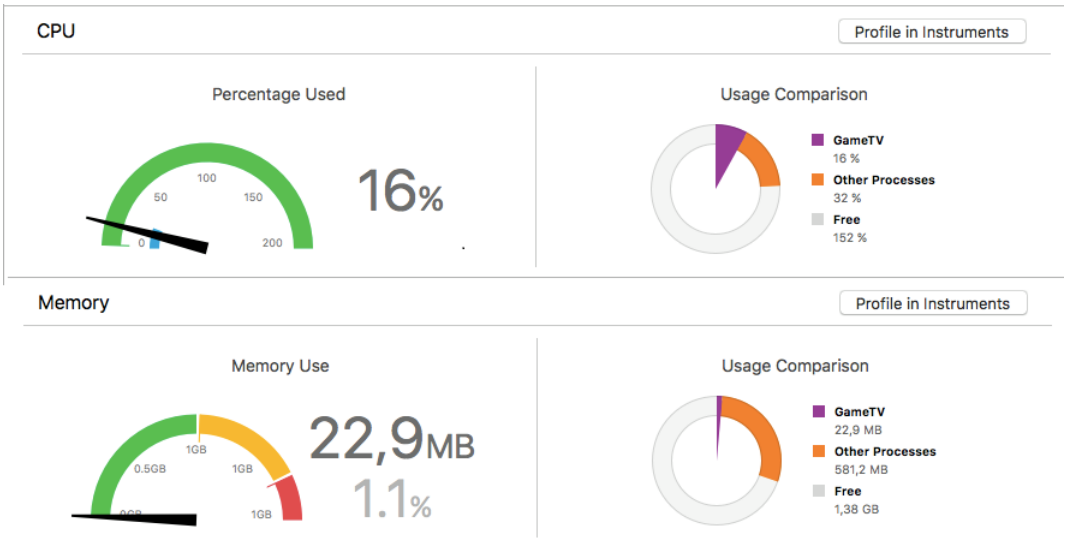
Po začatí záverečného testovania bolo okamžite odhalených niekoľko závažných aj menej závažných problémov. Medzi menej závažné problémy patrilo rozlíšenie použitých obrázkov v menu pre výber vzhľadu hernej postavy a použité pozadia boli na väčšej obrazovke značne rušivé.

Podstatné však boli problémy po začatí hrania v zvolenom levely a to konkrétne s obsahom metódy *update(_:)*. Scéna spracováva obsah pomocou per-frame logiky, takže táto metóda je volaná pri vykreslení každého snímku. Ako som už spomínal pri predchádzajúcom testovaní, hra dokáže na reálnom zariadení bežať s oveľa väčším frame-ratom, ako v emulátore. Ako príklad uvediem posúvanie pozadia proti smeru pohybu scény, ktoré bolo niekoľko násobne väčšie. Okrem toho nastal rovnaký problém pri pohybe niektorých objektov v scéne, ako napríklad pohyb prekážok, či mincí. Tento problém zasiahol celkovú hrateľnosť, ale bolo relatívne jednoduché ho odstrániť vzhľadom na to, že sa jednalo o jednu metódu. Postup bol taký, že časť kódu bola presunutá do triedy `LoadLevel`, konkrétne pohyb prekážok a mincí. Tým pádom bol vlastne zmenený spôsob pohybu týchto objektov z pohybu v každom snímku o konštantnú vzdialenosť, na pohyb pomocou akcie na konkrétnu pozíciu mimo zobrazovanej časti scény. Spolu so skrátením obsahu metódy *update(_:)* sa znížilo aj vyťaženie hardwaru. Zvyšok, čiže pohyb pozadia, bol ponechaný v metóde s menšími úpravami v počte pixelov o koľko sa pozadie v každom snímku posunie.

Po odstránení neočakávaných chýb a problémov prišiel na rad výber vhodného spôsobu ovládania hernej postavy. Pri testovaní v emulátore som navrhol spôsoby s využitím im-

pulzov, otáčania vektora gravitačnej sily a iných silových polí. Všetky navrhnuté spôsoby sa pri použití reálneho zariadenia chovali inak, ako pri prvotných testoch, hlavne spôsoby založené na intenzívnom klikaní na dotykovú plochu ovládača. Tie boli takmer úplne nepoužiteľné. Do finálnej verzie hry bol ponechaný pohyb pomocou otáčania gravitačného vektora pôsobiaceho na hráča. Tento spôsob neumožňuje rýchle zmeny smeru, ale vhodne zapadá do celkového tempa hry.

Pri využití ladiaceho nástroja sa ukázalo, že zaťaženie hardwaru pri spúšťaní aplikácie, pohybe v menu, ale aj počas hrania samotnej hry, je celkovo relatívne nízke. Využitie CPU sa počas behu aplikácie nachádza v rozmedzí 15-20% a počas tridsiatich minút súvislého testovania nikdy neprekročilo 23%. Tieto hodnoty je možné vidieť aj na nasledujúcom obrázku.



Obr. 6.3: Zaťaženie hardwaru pri testovaní na reálnom zariadeni

6.2.2 Testovanie užívateľmi

Počas testovania aplikácie inými hráčmi som sa zameral na pozorovanie toho, ako vnímajú zložitosť jednotlivých úkonov, ktoré som im zadal a následne ich celkový názor na hru. Na testovaní sa zúčastnilo celkom šesť dobrovoľníkov.

Pre začiatok testovania nedostal nikto inštrukcie, ako aplikáciu používať, len sériu úloh skladajúcu sa zo začatia novej hry, nastavenia inej ako základnej témy a spustenia ľubovoľného levelu. Vzhľadom k tomu, že sa v menu nachádzajú iba štyri tlačidla a to, že aplikácia neumožňuje pri prvom spustení začať hru bez vytvorenia novej hry, táto úloha nerobila nikomu problém. Tieto testy sa zameriavali na orientáciu užívateľa v menu a to sa prejavilo, ako dostatočne intuitívne.

Druhou a dôležitejšou časťou testovania boli testy samotnej hry. V týchto testoch mal tester zhodnotiť ovládanie, pričom sa porovnávalo ovládanie pomocou iPhonu a príslušnej aplikácie a pomocou Apple TV Remote. Väčšina zúčastnených po prvých pokusoch preferovala ovládanie pomocou iPhonu namiesto originálneho ovládača s odôvodnením, že pri danom koncepte ovládania, bola odozva na dotyk prirodzenejšia.

Kapitola 7

Záver

Cieľom mojej práce bolo preštudovať tvorbu aplikácií pre systémy iOS a tvOS, následne navrhnúť hru, ktorú bude možné spustiť na zariadení Apple TV a ovládať ju pomocou diaľkového ovládača. Ďalším krokom bolo hru implementovať a na záver ju testovať v emulátore a na reálnom zariadení.

Zdrojom informácií potrebných pre úspešne navrhnutie a implementáciu hry boli konzultácie s vedúcim bakalárskej práce, elektronické knihy, dokumentácia knižníc pre systémy iOS a tvOS a iné manuály k použitým technológiám.

Aplikácia bola tvorená vo vývojom prostredí Xcode a napísaná v jazyku Swift. Pri samotnej tvorbe bolo nutné viac krát pozmeniť návrh aplikácie, ale koncepcia sa držala pôvodného návrhu. Veľmi nápomocnou súčasťou pri tvorbe bol zabudovaný emulátor zariadení spoločnosti Apple, ako sú iPhone, iPad a pre účely mojej práce Apple TV. Pri testovaní na reálnom zariadení bolo nájdených niekoľko nedostatkov, ktoré bolo potrebné odladiť pred tým, ako bola aplikácia poskytnutá testerom pre zhodnotenie výsledku. Aplikáciu sa podarilo naimplementovať podľa mnou vypracovaného návrhu a na záver úspešne prešla testovaním. Za prednosti aplikácie by som označil jednoduchosť ovládania a celkovo jednoduchý koncept a taktiež to, že aplikácia ma počas behu veľmi nízke nároky na hardware.

Jadro aplikácie, ktoré pozostáva zo štyroch tried, poskytuje možnosť nezávislého pridávania nových úrovni, bez nutnosti väčších zásahov do už vytvoreného obsahu. Tieto triedy sa starajú o vytváranie prekážok, akcií, simuláciu fyziky a tematický vzhľad hry. Týmto spôsobom implementácie sa mi podarilo docieliť to, že aplikáciu je možné v budúcnosti jednoducho rozširovať o nový obsah. V rámci možných rozšírení do budúcna bude možné okrem pridávania nových levelov, pridávať aj nové témy, nové druhy prekážok či zmena obťažnosti už existujúcich levelov.

Ciele tejto bakalárskej práce boli splnené. Po zoznámení sa s potrebnými technológiami, som hru navrhol a následne implementoval. Výsledkom je funkčná hra, otestovaná a spustiteľná na reálnom zariadení.

Literatúra

[1] Apple Inc.: *Apple TV and tvOS*. [Online; navštívené 4. 5. 2017].
URL https://developer.apple.com/library/content/documentation/General/Conceptual/AppleTV_PG/

[2] Apple Inc.: *Debugging Tools*. [Online; navštívené 4. 5. 2017].
URL https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/debugging_with_xcode/chapters/debugging_tools.html

[3] Apple Inc.: *Interacting with tvOS*. [Online; navštívené 4. 5. 2017].
URL https://developer.apple.com/library/content/documentation/IDEs/Conceptual/iOS_Simulator_Guide/AppleTV/AppleTV.html

[4] Apple Inc.: *Preparing Your tvOS App for the App Store*. [Online; navštívené 4. 5. 2017].
URL https://developer.apple.com/tvos/

[5] Apple Inc.: *SpriteKite*. [Online; navštívené 4. 5. 2017].
URL https://developer.apple.com/reference/spritekit

[6] Apple Inc.: Frame processing in a scene. 2017, [Online; navštívené 4. 5. 2017].
URL https://developer.apple.com/reference/spritekit/skscene

[7] Apple Inc.: Parents are drawn before children. 2017, [Online; navštívené 4. 5. 2017].
URL https://developer.apple.com/reference/spritekit/sknode

[8] Neuburg, M.: *iOS 9 programming fundementals with swift*. O'Reilly Media, 2015, ISBN 978149193677.

[9] Painter, L.: 24 great games you can play on your Apple TV right now. *Macworld*, 2016: str. 2, [Online; navštívené 4. 5. 2017].
URL http://www.macworld.co.uk/feature/apple/24-best-apple-tv-games-2016-action-adventure-puzzle-scroller-gaming-minecraft-3611419/

[10] raywenderlich.com Team: *tvOS Apprentice*. Razeware LLC, 2016, ISBN 978-1942878247, 548 s.

[11] Wagner, C.: tvOS SDK: An iOS Developer's Initial Impressions. *Raywenderlich*, 2015, [Online; navštívené 4. 5. 2017].
URL https://www.raywenderlich.com/114313/tvos-initial-impressions

[12] Williams, M.: Pac-Man 256 Turns a Glitch Into Gameplay. *Usgamer*, 2015, [Online; navštívené 4. 5. 2017].
URL http://www.usgamer.net/articles/pac-man-256-turns-a-glitch-into-gameplay

Prílohy

Príloha A

Obsah priloženého CD

Priložené CD obsahuje nasledujúce položky:
`Projekt/` - projekt spustiteľný v prostredí Xcode pre operačný systém tvOS10
`Doc-src/` - zdrojové súbory k tomuto dokumentu
`Doc/` - technická správa a technická správa vo verzii pre tlač