VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MOBILNÍ APLIKACE "PEJSEK ZÁCHRANÁŘ" – ZRANĚNÝ CYKLISTA

MOBILE APPLICATION "PARAMEDIC DOGGY" – INJURED CYCLIST

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ADAM RICHTER**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MICHAL VLNAS**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Richter Adam**
Program: Informační technologie
Název: **Mobilní aplikace "Pejsek záchranář" - Zraněný cyklista**
Mobile Application "Paramedic doggy" - Injured Cyclist
Kategorie: Uživatelská rozhraní

Zadání:

1. Seznamte se s prostředím pro vývoj multimediálních mobilních aplikací (Unity)
2. Prostudujte problematiku první pomoci po nehodě cyklisty
3. Navrhněte rozhraní a strukturu vzdělávací aplikace pro děti na výše uvedené téma
4. Aplikaci implementujte
5. Proveďte měření a uživatelskou studii hodnotící implementované výsledky
6. Vytvořte video reprezentující výsledky vaší práce

Literatura:

- Dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Vlnas Michal, Ing.**
Konzultant: Zemčík Pavel, prof. Dr. Ing., UPGM FIT VUT
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Táto práca sa zaoberá tvorbou mobilnej aplikácie s edukačným účelom pre deti vo veku 6 až 10 rokov s kreslenou tematikou. Edukačný účel je zameraný na prvú pomoc v situácii kedy je dieťa nutné pomôcť zranenej osobe. Aplikácia je riešená formou edukačného kvízu s možnosťami A), B) a C), ktorý sa prelína s minihrami a animáciami maskota aplikácie. Z dôvodu zamerania na deti vo vekovej kategórii, ktoré sa učia plynule čítať, tak aplikácia obsahuje v nastaveniach možnosť čítania textu hlasovou formou. Práca obsahuje 7 častí vrátane úvodu a záveru. Druhá časť obsahuje všeobecný popis vývoja mobilných aplikácií. Tretia časť bližšie popisuje framework Unity, ktorého výber bol odôvodnený v štvrtej časti práce. Piata a šiesta časť popisujú návrh a implementáciu výslednej aplikácie Pejsek Záchranář.

Abstract

This work deal with creation of mobile application with educational purposes for children in age between 6 to 10 years with cartoon theme. Educational purpose is focused on first aid in situation where child is forced to help injured person. Application is solved in the form of educational quiz with options A), B) and C), which overlaps with minigames and animations of application mascot. Because of the focus on children in age category, who are learning how to read fluently the application contains in settings option text to speech. Work contains 7 parts including introduction and conclusion. Second part contains general description of the mobile application development. Third part describes in more detail framework Unity, which choice is justified in the part four. Fifth and sixth part describe design and implementation of the resulting application Pejsek Záchranář.

Kľúčové slová

mobilná aplikácia, framework Unity, vzdelávanie, prvá pomoc, C#, skripty, operačné systémy, iOS, Android, animácie

Keywords

mobile application, framework Unity, education, first aid, C#, scripts, operating systems, iOS, Android, animations

Citácia

RICHTER, Adam. *Mobilní aplikace "Pejsek záchranář" – Zraněný cyklista.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Vlnas

Mobilní aplikace "Pejsek záchranář" – Zraněný cyklista

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Michala Vlnasa. Ďalšie informácie mi poskytla Zdravotnícka záchranná služba Juhomoravského kraju. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Adam Richter
10. mája 2021

Poďakovanie

Chcem sa poďakovať svojmu školiteľovi Ing. Michalovi Vlnasovi za cenné rady a odbornú pomoc, ktoré mi poskytol pri písaní bakalárskej práce.

Obsah

Kapitola 1

Úvod

Táto práca popisuje vytvorenie edukačnej aplikácie pre mobilné zariadenia, ktorej úlohou bude ukázať dieťaťu formou kvízu ako sa má zachovať v situácii, kde sa nachádza zranená osoba a je jediné, ktoré mu môže poskytnúť pomoc. Aplikácia je orientovaná na šírku zariadenia a je vytvorená v prostredí Unity, ktoré uľahčuje vývojárom tvorbu aplikácie. Aplikácia sa zameriava na mobilné zariadenia s operačným systémom Android (Samsung, Xiaomi, . . .) a iOS (iPhone). Mobilná aplikácia je sústredená na deti približne vo veku 6-10 rokov kedy sú už schopné čítať.

Tvorba edukačných aplikácií vzniká za účelom naučiť ľudí nové poznatky a oboznámiť ich ako sa majú zachovať v konkrétnej situácii, v ktorej sa môžu v budúcnosti ocitnúť. Zameranie takýchto aplikácií je veľmi rozsiahle, kde môže byť cieľom naučiť ľudí nový jazyk, zdokonaliť ich vedomosti o histórii, prírode, ako poskytnúť prvú pomoc, uľahčenie zapamätania si všetkých pravidiel pri kurze v autoškole a simulácia rôznych situácií, v ktorých sa môže človek vyskytnúť. Väčšina edukačných aplikácií sa snaží do výuky zakomponovať prvky rôznych minihier alebo videí aby si udržali pozornosť používateľa čo najdlhšie a zároveň aby sa spôsob výuky odlišoval od stereotypu, ktorým sa vyučuje na školách.

V tejto práci sa na tvorbu aplikácie použil framework Unity, programovací jazyk C# a Adobe Photoshop 2020. Unity je multiplatformový herný engine vytvorený spoločnosťou Unity Technologies, ktorý rapidne uľahčuje tvorbu herných aplikácií, ktoré majú byť kompatibilné s viacerými platformami naraz. Aplikácia je spracovaná pomocou skriptov v jazyku C#, s ktorým vie Unity spolupracovať. Pre grafické prevedenie aplikácie bol využitý Adobe Photoshop, kde sa aplikácia zameriava na kreslený štýl aby viacej priťahoval detskú pozornosť. Cieľom práce bolo naštudovať tvorbu mobilných aplikácií pre platformy iOS a Android, prácu s frameworkom Unity a vývoj grafického používateľského rozhrania. Vytvorená aplikácia je zhotovená za účelom ukázať deťom ako pomôcť zranenému človeku v núdzi.

Kapitola súčasný stav tvorby mobilných aplikácií zahrňuje dôvody a spôsoby tvorby mobilných aplikácií pre rôzne platformy, rôznorodosť použiteľných frameworkov, pre aké platformy sa dané aplikácie vyvíjajú, k čomu slúži používateľské grafické rozhranie a existujúce aplikácie, ktoré sú zamerané na rovnakú tému. V kapitole Unity je bližšie rozpísaný princíp ako daný framework funguje a ako sa v ňom už vytvára konkrétna aplikácia. Kapitola 4 popisuje porovnanie niektorých použiteľných frameworkov pre túto prácu a technické parametre danej mobilnej aplikácie. Návrh v kapitole 5 popisuje spôsob akým sa navrhovala funkcionalita, grafické rozhranie a čo by mali obsahovať nastavenia tejto aplikácie. Kapitola 6 popisuje ako pracuje back-end aplikácie, riešenie tvorby animácií a ako bolo do aplikácie Pejsek Záchranář zakomponované grafické používateľské rozhranie.

Kapitola 2

Súčasný stav tvorby mobilných aplikácií

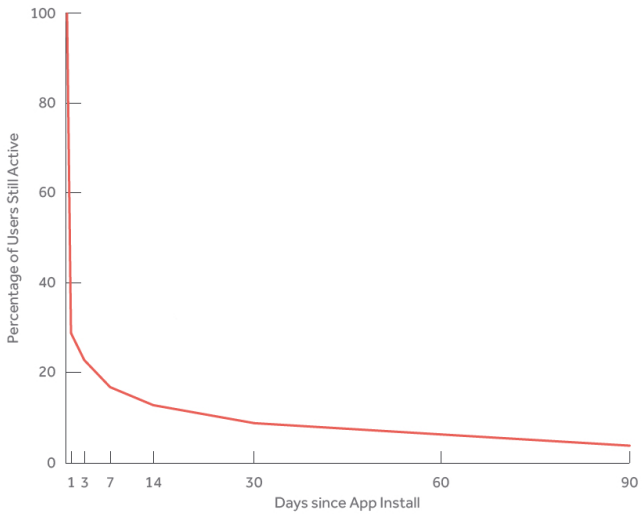
Táto kapitola opisuje techniky akými sa vytvárajú mobilné aplikácie, k čomu slúžia čo treba brať do úvahy pri vývoji, ako vykonať grafické spracovanie aplikácie a podobné aplikácie, ktoré riešia danú problematiku.

2.1 Aplikácie pre edukačné účely

Elektronické vzdelávanie je forma dištančného vzdelávania. Táto forma vzdelávania existuje už mnoho rokov a má kladný prínos do vzdelávacieho procesu. K vzdelávaniu využíva všetky možnosti, ktoré sú schopné informačné technológie poskytnúť v súčasnej dobe. Tento typ vzdelávania umožňuje ľuďom sa vzdelávať z rôznych častí krajiny, vzdelávať sa tempom akým sami uznajú za vhodné a hlavne pri vzdelávaní využiť interaktívne prvky, ktoré pomáhajú udržať pozornosť používateľa.

Pokiaľ používateľa nezaujme vzdelávací obsah danej aplikácie, výsledky zapamätania si vedomostí budú nedostatočné [13]. Mobilné technológie disponujú mnohými metódami na tvorbu interaktivity pri vzdelávaní, ktorú používatelia pri používaní aplikácie očakávajú, pretože sa s nimi stretávajú takmer vo všetkom, čo má spojitosť s mobilnými aplikáciami. Interaktivita je vlastne spôsob, ktorým sa snažia podobne zamerané aplikácie konkurovať medzi sebou. Táto vlastnosť aplikácií robí obsah zaujímavejším a hlavne efektívnejším pri memorovaní informácií. Používatelia sú väčšinou zaneprázdnení a okrem vzdelávania na nich pôsobí mnoho iných javov, ktoré môžu narušovať ich pozornosť pri vzdelávaní sa. Šanca, že používateľ prestane používať danú aplikáciu skorej ako ju využije na maximum je v niektorých prípadoch až 90% viď Obr. 2.1. Ale ak je vzdelávací obsah správne spracovaný pre vekovú kategóriu, na ktorú je cielený, tak potenciál využitia danej aplikácie je oveľa väčší. Spracovaním sa myslí predovšetkým interaktivita, ktorá priťahuje pozornosť používateľov a motivuje ich k pokračovaniu používania mobilnej aplikácie z čoho vyplýva, že šanca na opustenie aplikácie je menšia, čim je lepšie interaktivita do aplikácie zakomponovaná.

Ďalší z dôvodov prečo využiť mobilné aplikácie je, že používateľ môže dané skúsenosti nadobudnúť v bezpečnom prostredí domova. Príležitosť ukázať používateľovi konkrétne situácie, v akých sa môže v budúcnosti ocitnúť mu poskytuje možnosť byť pripravený na danú situáciu vopred. Takto vyskúšané postupy pomôžu používateľovi sa správne rozhodnúť v budúcnosti pokiaľ by daná situácia nastala.



Obr. 2.1: Graf využitia aplikácií z dlhodobého hľadiska

Cieľom týchto aplikácií je, aby si používateľ nadobudnuté informácie pamätal z dlhodobého hľadiska [13]. Spôsob akým funguje dlhodobá pamäť zaisťuje, že sme schopný využiť dané informácie, riešenia situácií, vzorce pri podobnej situácii v budúcnosti. Čím je interaktivita aplikácie vyššia, tým viac podporuje, že dané informácie sa uložia do dlhodobej pamäte človeka. Interaktivitu aplikácií [13] zabezpečujeme pomocou animácií, minihier, videí, zvukov, kvízov, grafickým spracovaním aplikácie podľa zamerania na vekovú kategóriu a taktiež podľa zamerania tematiky.

2.2 Platformy

Platformy sú vlastne operačné systémy, ktoré ovládajú zdroje zariadenia a umožňujú používateľom prístup k týmto zdrojom cez určité špecifické rozhranie [8]. Zjednodušene sa jedná o systém, ktorý komunikuje medzi hardvérom zariadenia a požiadavkami používateľa. Takýto systém má za úlohu spracovať vstupy od používateľa a odpovedá na ne spravovaním úloh a interných zdrojov zariadenia ako služby pre používateľa, ktoré od zariadenia vyžaduje. Operačné systémy [8] vykonávajú základné úlohy akými sú napríklad alokovanie pamäte, správa súborov, spravovanie vstupných a výstupných zariadení, určenie priority pokynov, ktoré sa majú vykonať. Operačné systémy môžme nájsť vo všetkých zariadeniach, ktoré obsahujú integrované obvody. Každý systém je rozdielny a jeho komplexnosť závisí od popisu práce, ktorú má vykonávať. Operačné systémy, ktoré sa vyvíjajú pre mobilné zariadenia obsahujú funkcie počítačov, ktoré sa prepájajú s funkciami ako je napríklad dotykový displej, fotoaparát, videokamera alebo možnosť telefonovania. Medzi najrozšírenejšie operačné systémy pre mobilné telefóny patrí iOS a Android. Operačné systémy ako je Symbian alebo Windows Phone 7 sú na úpadku a postupne ich vytláčajú viacej obľúbené systémy čim je práve iOS a Android. Tieto systémy sú v neustálom vývoji aby zabezpečovali bezpečnosť a zároveň zlepšovali ich výkon, funkcie a prístupnosť viacerým používateľom čo

vyžaduje časté aktualizácie softvéru. Dobrý operačný systém by mal v dnešnej dobe spĺňať minimálne tieto požiadavky [8]:

1. intuitívne ovládanie
2. zachovanie kompatibility so starými programami
3. multitasking
4. práca so sieťou
5. podpora iných operačných systémov
6. prenosnosť na iné hardware-ové systémy
7. bezpečnosť používateľa
8. cena/výkon

iOS

iOS [10] je mobilný operačný systém, ktorý bol vytvorený spoločnosťou Apple Inc, ktorú založil Steve Jobs a Steve Wozniak v roku 1976. Tento systém bol vyvinutý špeciálne pre ich vlastné potreby, takže ho môžme vidieť iba v zariadeniach viď Obr. 2.2, ktoré firma poskytuje na trhu. Tento systém ovláda zariadenia rôznych verzií ako je iPhone, iPod alebo iPad. Firma sa veľmi rýchlo preslávila zariadením iPod Touch. iOS sa prvý krát objavil v roku 2007 kedy bol pomenovaný ako iPhone OS a následne v roku 2010 bol premenovaný na iOS, tak ako ho poznáme aj dnes. Používateľské rozhranie tohto systému je založené na používaní multi-touch gest, ktorými sú pinch, reverse pinch, swipe a touch. Systém si svoju obľúbenosť vydobil najmä svojou optimalizovanosťou, prestížnosťou, bezpečnosťou a jednoduchosťou ovládania. Systém je neustále vo vývoji aby mohol konkurovať konkurencii a ako sa postupne vyvíjajú nové technológie, tak sa aj systém aktualizuje a prináša nové funkcie pre používateľov.

Architektúra

iOS je zjednodušená verzia operačného systému Mac OS, ktorý sa používa v počítačoch tejto firmy. Jedná sa o systém typu UNIX, ktorý je vyvíjaný pre mobilné zariadenia, tak obsahuje kontrolu ovládania dotykového displeja. Architektúra systému [10] sa delí na 4 vrstvy viď Obr. 2.3, ktorými sú:

1. Cocoa touch layer
2. Media layer
3. Core services layer
4. Core OS layer

Obr. 2.2: Operačný systém iOS

Cocoa Touch Layer

Cocoa touch [10] je vývojové prostredie aplikácie pre tvorbu programov, ktoré majú bežať na systéme iOS. Medzi hlavné vlastnosti ktoré poskytuje patrí **Multitasking**, **Zdieľanie**, **Rozpoznávanie gest**, **Push notifikácie** a **Ochrana dát**.

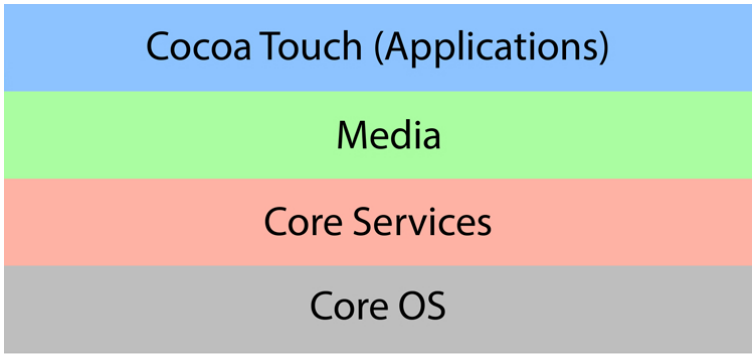
Multitasking umožňuje rýchle znovu spustenie aplikácie a aby mohli aplikácie na pozadí aktualizovať svoj obsah bez nutnosti ich znovu otvárať. Bez tejto možnosti by sa každá aplikácia vypla pri každej zmene okna na zariadení čo by viedlo k opätovnému spúšťaniu danej aplikácie. Zdieľanie poskytuje používateľovi možnosť zdieľať ich obsah s ostatnými zariadeniami, pomocou funkcie AirDrop, ktorá využíva technológiu Bluetooth. Rozpoznávanie gest obsahuje základné gestá pre ovládanie mobilného zariadenia, ale aj možnosť vytvorenia vlastných gest, ktoré môžu pomôcť ovládať zariadenie aj hendikepovaným používateľom. Push notifikácie umožňuje používateľa upozorniť ohľadom nových informácií, ktoré sa týkajú konkrétnej aplikácie bez toho aby bola daná aplikácia spustená. Ochrana dát umožňuje ukladať súkromné dáta zašifrovaním. Takéto dáta sa automaticky ukladajú do pamäte v zašifrovanej podobe aby ich nebolo možné prečítať.

Media Layer

V tejto vrstve [10] sa nachádzajú technológie pre tvorbu 2D a 3D grafiky, zvukov animácií, efektov.

Core Services Layer

Táto vrstva zahrňuje lokalizáciu, ktorá umožňuje upravovať vzhľad a obsah aplikácie na základe polohy používateľa. Tiež využívané pri určenia polohy pomocou GPS alebo Wi-Fi [10]. Ďalej obsahujú blokové objekty, ktoré sa používajú v situáciach kedy chce programátor využiť určitý segment kódu znovu a vytváranie funkcie je v danej situácií nevýhodné.



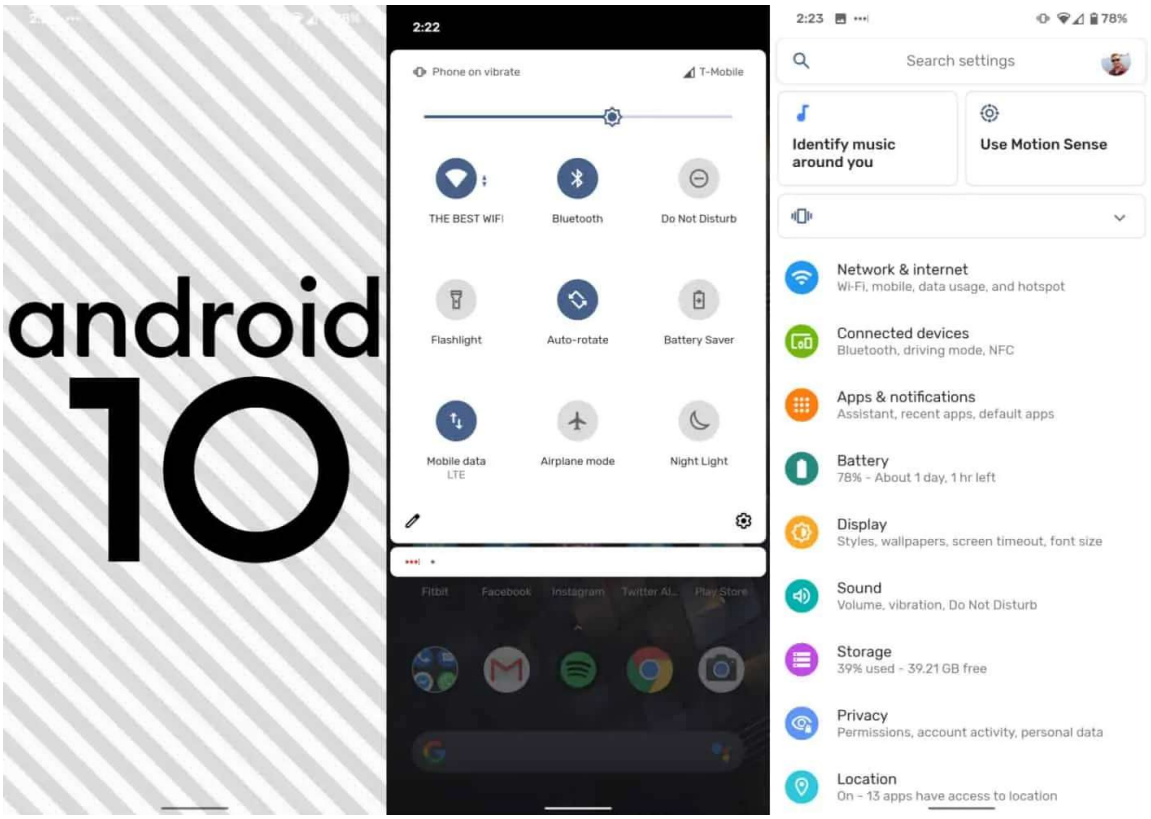
Obr. 2.3: Architektúra systému iOS

Core OS Layer

Poskytuje nízkoúrovňové služby [10] prepojené s hardvérom a sieťami. Tieto služby sú založené na Kerneli. Hlavným cieľom tejto vrstvy je zabezpečiť bezpečnosť aplikácie.

Android

Android [12] je open source operačný systém viď Obr. 2.4, ktorý na rozdiel od systému iOS vznikol pre mobilné zariadenia rôznych značiek a tým sa stal najpoužívanejším operačným systém pre mobilné zariadenia. Je založený na jadre Linux. Keďže systém využíva viacero



Obr. 2.4: Operačný systém Android

spoločností, tak jeho vývoj má na starosti Open Handset Alliance, ktorej cieľom je rozvíjať systém tak, aby mal nižšie náklady pre distribúciu a zároveň prinášal používateľom nové funkcie a zlepšené používateľské rozhranie. Jadro systému Android je upravené tak aby sa dalo využívať na rozdielnom hardvéri čo znamená, že systém je responzívny a upravuje svoj vzhľad podľa rozlíšenia obrazovky. Keďže sa jedná o open source platformu, tak každá spoločnosť, ktorá využíva vo svojich zariadeniach Android, vytvára vlastnú nadstavbu systému, aby sa líšili od konkurencie. Avšak tvorba rôznych nadstavieb pre systém Android môže zapríčiňovať väčšie zaťaženie systému, čo sa môže odraziť na potrebe zakomponovať do zariadenia silnejší hardvér.

Architektúra

Android sa z hľadiska architektúry [12] delí na 5 vrstiev, viď Obr. 2.5, ktorými sú:

1. Aplikácie
2. Application Framework Layer
3. Android RunTime Layer
4. Knižnice
5. Linux Kernel

Aplikácie

Nachádzajú sa tu všetky nainštalované aplikácie v rámci systému Android. Jedná sa o aplikácie stiahnuté používateľom a o predvolené aplikácie samotným systémom [12].

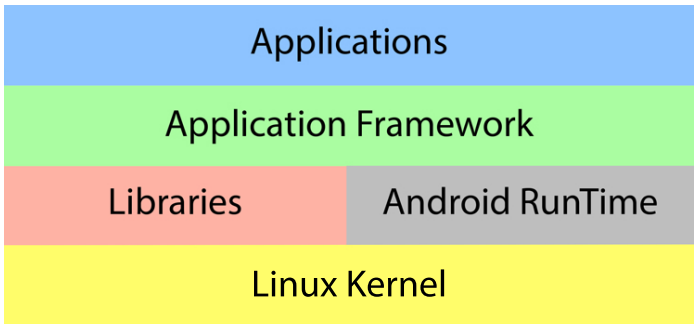
Application Framework Layer

Poskytuje veľa vysokoúrovňových služieb [12] aplikáciam formou Java class. Vývojári aplikácií pre Android používajú tieto služby pri tvorbe aplikácií. Medzi služby patrí:

Activity Manager – riadi životný cyklus aplikácií

Content Provider – povoľuje aplikáciam zdieľať dáta s inými aplikáciami

Resource Manager – povoľuje prístup k rozloženiu používateľských rozhraní, nastaveniu farieb



Obr. 2.5: Architektúra systému Android

Notification Manager – povoľuje zobrazovať na zariadení upozornenia aplikácií bez zapnutia danej aplikácie

View System – obsahuje prvky pre tvorbu používateľského rozhrania ako je napríklad tlačidlo, textové pole alebo checkbox

Android RunTime Layer

Táto vrstva poskytuje kľúčový komponent, ktorý sa nazýva Dalvik Virtual Machine [12], ktorý bol špecificky optimalizovaný pre systém Android. Tento komponent umožňuje každej aplikácii rozbehnúť žiadaný proces vo vlastnej instancii Dalvikovho komponentu.

Knižnice

Jedná sa o knižnice, ktoré sú napísané v jazyku C alebo C++ [12]. Obsahujú knižnice pre webový prehliadač, pre prístup k relačným databázam, pre vykresľovanie grafiky, pre podporu prehrávania videa, zvukov, obrazových súborov, pre vykresľovanie bitmapových a vektorových fontov.

Linux Kernel

Najnižšou vrstvou [12] je samotné jadro operačného systému, ktoré komunikuje medzi hardvérom a softvérom systému. Využíva mnoho vlastností Linuxu, ktorými je napríklad správa sietí a procesov, multitasking, zabezpečenie priority procesov, bezpečnosť systému. Hlavným dôvodom prečo sa v systéme využíva Linux je jeho kompilácia na rôznych typoch zariadení a tým pádom sa zabezpečila prenosnosť systému na iné zariadenia.

2.3 Mobilné aplikácie

Mobilné aplikácie [14] pozostávajú zo softvéru, ktorý pracuje na mobilnom zariadení a vykonáva konkrétne úlohy pre používateľa. Patria medzi rýchlo rozvíjajúce sa odvetvie informačných technológií. Jedná sa o jednoduché, lacné a stiahnuteľné aplikácie, ktoré je možné spustiť takmer na všetkých mobilných zariadeniach. Aplikácie nadobúdajú rôznych zameraní ako sú napríklad hry, cestovanie, multimédia, sociálne siete, navigácia, výučba a mnoho ďalších. Niektoré aplikácie bývajú predinštalované na mobilnom zariadení a väčšinou sa jedná o aplikácie, ktoré podporujú základné funkcie telefónov ako je prehliadač, telefón, správy, obchod pre stiahnutie aplikácií, nastavenia, kalendár. V obchode sa nachádza obrovské množstvo aplikácií, ktoré si môže používateľ dodatočne stiahnuť do mobilného zariadenia podľa jeho záujmu. Ako sa neustále zlepšuje výpočtová technika mobilných zariadení, tak to ponúka vývojárom aplikácií nové možnosti ako priniesť komplexnejšie aplikácie na mobilné zariadenia. Mobilné aplikácie sa stali súčasťou života ľudí po celom svete a niektorí si už nevedia bez nich predstaviť fungovať. Mobilné aplikácie disponujú viacerými vlastnosťami [14] ako je rýchla komunikácia, šetrenie času, nižšia spotreba energie, zábava, redukcia výdavkov.

Rýchla komunikácia pomáha udržiavať vzťahy medzi ľuďmi, ktorých od seba oddeľuje aj oceán pomocou sociálnych sietí, kde sú schopní si písať, telefonovať, posielať obrázky alebo videá. Medzi takéto aplikácie patrí v dnešnej dobe najmä Messenger a Instagram.

Šetrenie času umožňuje ľuďom riešiť ich dennú prácu za chodu, v aute alebo autobuse. Vďaka tejto vlastnosti aplikácií nemusia ľudia riešiť ich denné problémy na konkrétnom mieste ale kedykoľvek a kdekoľvek.

Nižšia spotreba energie je spôsobená tým, že veľa mobilných aplikácií poskytuje rovnakú funkcionalitu ako práca na počítači. Vďaka tomu rapidne klesá využívanie počítačov, čo vedie k nižšej spotrebe energie.

Zábava v mobilných aplikáciach je prevedená formou hier, sledovaním rôznych videí, počúvaním hudby, ktoré dokážu pocitovo urýchliť presun z bodu A do bodu B.

Redukcia výdavkov je spôsobená príchodom aplikácií, ktoré umožňujú sa skontaktovať s ľuďmi cez internet a využitie platených mobilných sietí k telefonovaniu začína byť menej efektívne.

Najväčšou výzvou mobilných aplikácií je ich limitácia a platformovosť [14]. Jedná sa o malé obrazovky čo spôsobuje neustálu konfrontáciu ako natlačiť všetky potrebné komponenty na obrazovku zariadenia tak, aby to bolo čitateľné a zrozumiteľné z pohľadu používateľa. Aplikácie musia fungovať na rozdielnych zariadeniach, ktoré sa neustále menia a vylepšujú, čo spôsobuje rozdielne veľkosti displejov a to núti vývojárov mobilných aplikácií aby ich produkty boli responzívne [14] a prispôsobovali svoj vzhľad displeju zariadenia. Mobilné zariadenia využívajú rôzne platformy (iOS, Android, ...) a tým pádom mobilné aplikácie musia byť vyvíjané tak, aby boli schopné fungovať na všetkých používaných zariadeniach. Ďalej tieto zariadenia postrádajú možnosť otvoriť všetky formáty súborov. Napriek týmto limitáciám sú mobilné aplikácie čoraz viac populárne a veľa ľudí sa snaží nahradiť počítač mobilným zariadením vďaka jeho cene, prenosnosti a využitiu v bežnom živote.

2.4 Herné enginy pre tvorbu aplikácií

Herný engine je softvér [3], ktorý umožňuje tvorbu aplikácií. Hlavným cieľom herného enginu je vytvoriť abstrakt bežných vlastností aplikácií, ktoré sa potom dajú jednoducho použiť pri tvorbe rozdielnych aplikácií. Prvý herný engine bol vytvorený už v roku 1991 pod názvom GameMaker a bol určený pre tvorbu 2D hier. Od tohto roku vzniklo mnoho herných enginov, ktoré časom zanikli alebo sa udržali medzi najlepšími a používajú sa dodnes v aktualizovanej verzii. Medzi hlavné funkcionality [3] herných enginov patria:

- Vykreslovanie 2D a 3D grafiky
- Manažovanie vstupov klávesnice, dotykového displeja ...
- Umožnenie behu viacerých procesov naraz
- Pridanie fyzikálnych vlastností objektom
- Možnosť kolízie a detekcie objektov
- Sledovanie možných udalostí každý frame aplikácie
- Multiplatformovosť
- Tvorba animácií

- Práca po sieti

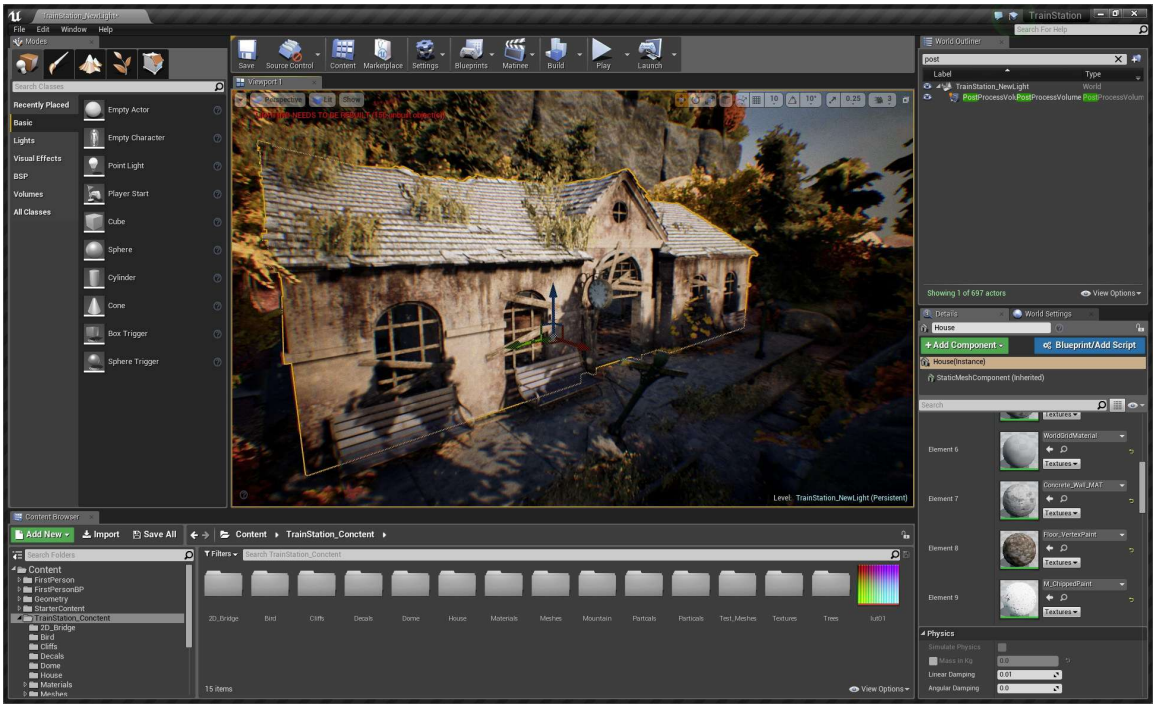
V dnešnej dobe využívajú herné enginy vysokoúrovňové jazyky ako je napríklad C#, C++, Python. Tento typ jazykov vedie k väčšej produktivite vývojárov pretože sa viac podobá na ľudský jazyk ako na jazyk, v ktorom pracujú stroje. Náročnejší preklad takýchto jazykov je už v dnešnej dobe zanedbateľný vďaka neustále vyvíjajúcemu sa hardvéru. Takisto väčšina herných enginov sa snaží aby obsahovali vlastnosť multiplatformovosti [3]. Ide o využitie jedného kódu pre tvorbu aplikácie na rôznych platformách, ktoré podporujú odlišný programovací jazyk ako je napríklad Swift pre iOS a Java pre Android. Medzi najznámejšie herné enginy v dnešnej dobe patria: Unity, Unreal Engine, CryEngine.

Unity

Tento framework [1] patrí medzi jeden z najrozšírenejších v dnešnej dobe. Je schopný vytvárať aplikácie pre širokú škálu platforiem a zároveň má veľkú komunitu ľudí, ktorý ho využívajú. Keďže tento herný engine bol použitý pre tvorbu aplikácie Pejsek Záchranár, tak je detailnejšie popísaný v kapitole 3.

Unreal Engine

Je vytvorený spoločnosťou Epic Games a bol predstavený na hre Unreal v roku 1998 čo bola hra typu FPS. Engine [9] bol zo začiatku určený len pre komerčné účely, avšak v roku 2015 s príchodom Unreal Engine 4 sa stal voľne dostupný pre širokú verejnosť s podieľom 5% zo zisku ak daný projekt zarobí viac ako 3000 dolárov za štvrťrok. Podporuje širokú škálu platforiem pre ktoré je schopný vytvárať hry viď Obr. 2.6. Programovací jazyk využívaný v tomto frameworku je kombinácia C++ a Blueprint, čo je vizuálne skriptovanie



Obr. 2.6: Unreal Engine

pomocou možnosti Drag and Drop bez priameho písania kódu. Unreal Engine je vytvorený z niekoľkých komponentov, ktoré spolupracujú pri riadení hry. Medzi hlavné komponenty [9] patrí:

- Grafický engine
- Fyzikálny engine
- Zvukový engine
- Svetlo a tiene
- Umelá Inteligencia

Grafický engine

Tento komponent [9] je zodpovedný za výstup na obrazovke zariadenia na základe informácií ohľadom scény ako sú textúry, farby, tiene, umiestnenie hľadiska. Sila grafického enginu určuje ako realisticky bude vyzerať výstup na obrazovke. Unreal Engine je schopný vytvoriť takmer fotorealistickú kvalitu obrazu vďaka jeho schopnosti optimalizovať scénu a spracovať svetlo v reálnom čase.

Fyzikálny engine

Umožňuje pridávať fyzikálne vlastnosti objektom ako je trenie, rýchlosť alebo gravitácia. Tak isto je schopný spracovať detekciu a kolíziu objektov [9]. Pre túto funkcionalitu využíva PhysX engine, ktorý bol vytvorený spoločnosťou NVIDIA. Vďaka tomuto enginu sa môže vývojár sústrediť na tvorbu hry a nemusí riešiť ako zabezpečiť interakciu medzi objektmi.

Zvukový engine

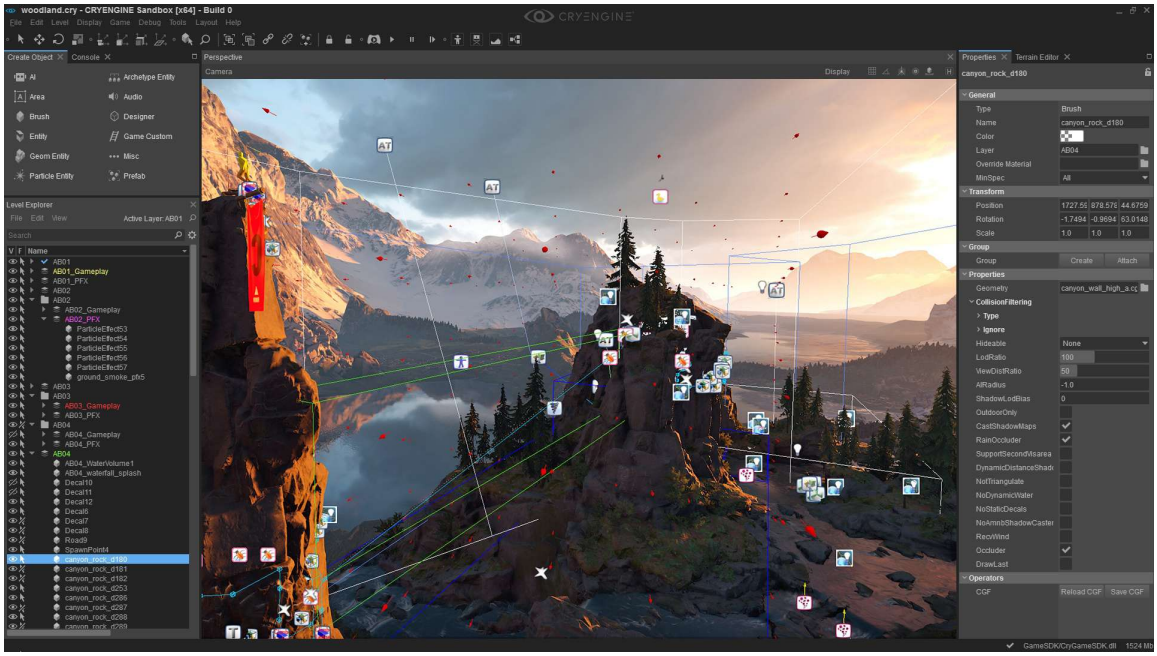
Ako už vyplýva z názvu, tak je zodpovedný za hudbu a zvuky v aplikáciách. Umožňuje pridávať rôzny typy zvukových súborov, ktoré podporujú realistickosť aplikácie.Zvuky ktoré sa spúšťajú v daný okamih po nejakej udalosti implementuje Unreal Engine pomocou funkcie Sound Cue [9].

Svetlo a tiene

Svetlo je zodpovedné za vytvorenie atmosféry scény. Správne navrhnuté tiene a svetlo v scéne dokážu rapidne ovplyvniť dojem požívateľa. Unreal Engine poskytuje 4 typy svetiel [9], ktoré sa dajú jednoducho implementovať do scény. Jedná sa o slnečné svetlo, svetlo oblohy, bodové svetlo a svetlo mierené na konkrétne miesto.

Umelá Inteligencia

Pozostáva z mnohých pravidiel, ktoré ovplyvňujú správanie objektov na scéne. Tieto pravidlá im pomáhajú napodobňovať chovanie aké by nadobudli v reálnom živote. Unreal Engine poskytuje dobrú základnú umelú inteligenciu [9], na ktorej sa dá začať vytvárať nadstavba vlastných pravidiel pre špecifické chovanie objektov.



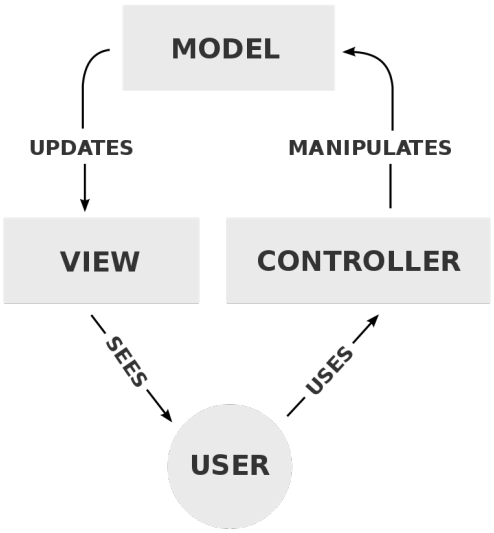
Obr. 2.7: CryEngine

CryEngine

Tento engine je známy vysokou kvalitou a vysokým rozlíšením aplikácií [6]. Bolo v ňom vytvorené mnoho AAA hier. Je vytvorený vývojármi nemeckej spoločnosti Crytek. Prvá verzia vyšla v roku 2002 a v roku 2016 vyšla jej posledná iterácia pod názvom CryEngine V. Jeho dostupnosť medzi používateľov funguje na podobný princíp ako Unreal Engine. Pokiaľ za rok na danom projekte vytvoreným v ich frameworku sa zarobí viac ako 5000 dolárov, tak je žiadaný 5% podiel na zisku. V porovnaní s dvoma predchádzajúcimi frameworkmi (Unity, UnrealEngine) má najstrmšiu krivku učenia pravdepodobne vďaka neprehľadnému editoru, menšej komunite používateľov a žiadnemu obchodu s assetmi, ktorými Unity a Unreal Engine disponujú. Je schopný vytvárať 2D aj 3D aplikácie s podporou virtuálnej reality. Tento framework je známy hlavne veľkým potenciálom pre generovanie ohromujúcej grafiky [6] prírodného prostredia viď Obr. 2.7. Zároveň jeho najväčšou nevýhodou je slabá dokumentácia a začínajúci vývojári budú mať problém využiť framework naplno. Framework podporuje programovacie jazyky ako sú C++, C# a Lua.

2.5 GUI

GUI [15] znamená grafické používateľské rozhranie. Jedná sa o typ používateľského rozhrania, ktoré umožňuje používateľovi vykonávať interakciu so zariadeniami pomocou vizuálnych objektov. Grafické rozhrania vyvinuté po roku 1970, ktoré boli vytvorené pre operačné systémy spoločnosti Apple a Microsoft udalo pomyselný štandard, ktorý sa začali snažiť implementovať všetky aplikácie. Cieľom tvorby týchto rozhraní bola neefektivita využitia textového príkazového riadku bežnými používateľmi. Základným prvkom takýchto rozhraní je, že obsahuje tlačidlá, ikony, rôzne textové polia a pomocou nich sa vykonáva interakcia s danou aplikáciou [15].



Obr. 2.8: Logika GUI

Je založené na princípe model – view – controller [15], viď Obr. 2.8. Jedná sa o určenie detekcie pomocou logických hodnôt ako je true a false (1,0), ktorý rozdeľuje vnútornú reprezentáciu informácií od spôsobu akým sú informácie poskytnuté používateľovi. Používateľ manipuluje s informáciami pomocou vizuálnych prvkov, ktoré sú nadizajnované tak, aby odpovedali dátam, s ktorými aplikácia pracuje. Vzhľad celej aplikácie môže byť kedykoľvek prerobený podla potreby, pretože grafické rozhranie je nezávislé od funkcií skrytých za používateľským rozhraním, ktoré vykonáva aplikácia. Každá aplikácia vykonáva vzhľad grafického rozhrania podľa vlastného návrhu. GUI musí byť navrhnuté tak, aby s ním dokázal pracovať aj úplný laik a tým pádom mohlo byť využité širokou škálou používateľov [15]. Grafické používateľské rozhranie je jeden z hlavných prvkov, ktoré používateľa buď zaujme alebo ho odradí od používania danej aplikácie. Preto je veľmi dôležité navrhnúť GUI tak, aby zaujalo kategóriu ľudí, pre ktorých je aplikácia cielená. Medzi výhody [15] grafického rozhrania patria:

- Jednoduchosť
- Ľudia bez väčších znalostí počítača sú schopný aplikáciu využívať
- Vytvára krajší a prehľadnejší vzhľad danej aplikácie
- GUI udržuje pozornosť používateľa

Avšak GUI má taktiež svoje nevýhody [15] medzi ktoré patrí:

- Implementácia
- Rýchlosť
- Potreba väčšej pamäte počítača
- Cena

Existuje viacero rôznych programovacích jazykov, ktoré podporujú tvorbu grafického rozhrania. Každý jazyk obsahuje svoje vlastné výhody implementácie používateľského rozhrania. Avšak Java a C# sa považujú za najlepšie možnosti [15], pretože tieto jazyky podporujú funkcionalitu GUI v prehliadači a zároveň aj v počítačových a mobilných aplikáciách.

2.6 Ostatné techniky

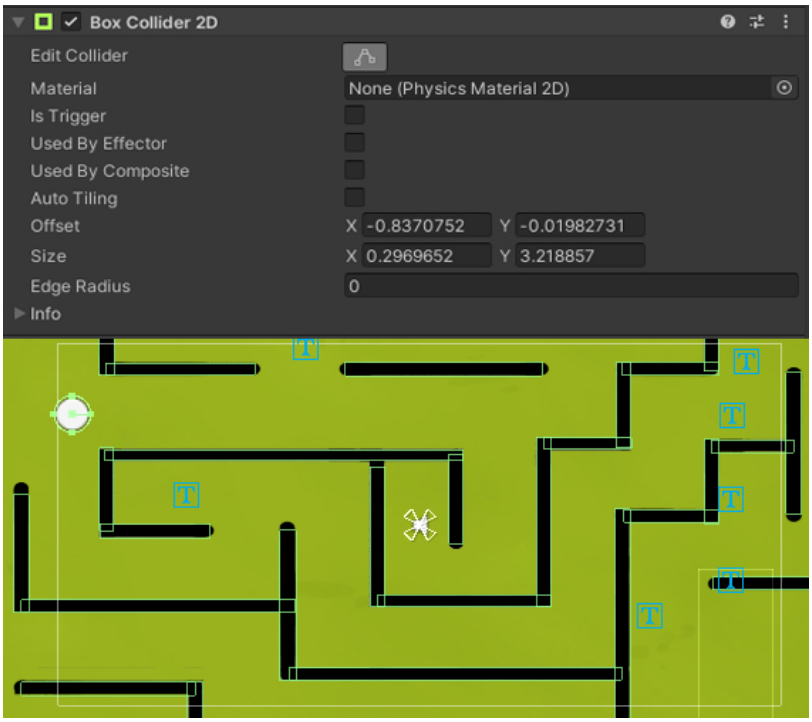
Existuje mnoho techník, ktoré sa dajú využiť pri programovaní mobilnej aplikácie. Niektorými z nich je napríklad ruská ruleta, detekcia kolízií alebo skriptovacie objekty.

Detekcia kolízií

Jedná sa o matematický problém kedy je potrebné vykonať detekciu prieseku dvoch odlišných objektov [5]. Proces kolízie v aplikáciách funguje na základe zachytenia kolízie, následného zistenia času a objektov, ktoré spôsobili kolíziu. Na túto kolíziu treba vyvinúť požadovanú akciu, ktorá ma nastať pokiaľ sa objekty stretli. Reakcia je napríklad deformácia zrazených objektov, rôzne animácie, ktoré vytvárajú efekt zrážky alebo celkového správania objektov, ktoré nadobudnú po kolízii. Algoritmy určené na detekciu kolízií sa dajú rozdeliť na dva typy [5]:

Detekčné Jedná sa o určenie detekcie pomocou logických hodnôt ako je true a false (1,0).

Určujúce Ide o určenie kolízie pomocou priesečníkov, kolíznych čiar vďaka ktorým sme schopný určiť aj konkrétnu pozíciu objektov v kolízii.



Obr. 2.9: Komponent na vytvorenie hitboxu

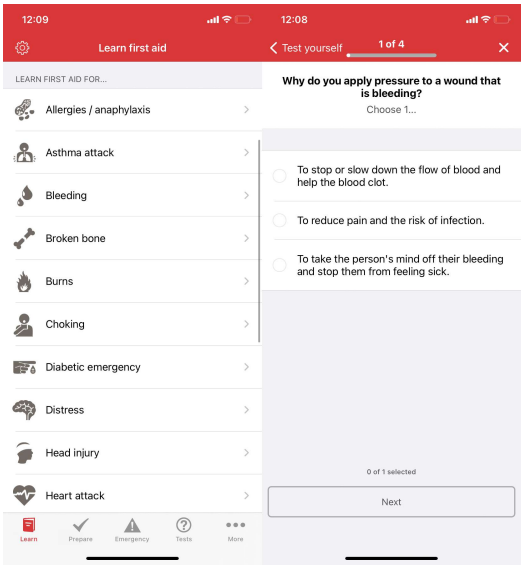
K zachyteniu kolízie sa najčastejšie využíva hitbox [5]. Jedná sa o neviditeľný tvar, ktorý nadobúda 2D aj 3D rozmery podľa zamerania hry a obklopuje daný objekt, u ktorého očakávame že sa v budúcnosti dostane do kolízie viď Obr. 2.9. Tento hitbox sa snaží vyhodnocovať v reálnom čase, či dochádza ku kolízii. Výpočet danej kolízie je celkom náročné na hardvér a tým pádom je aj hardvérom obmedzované. Čím častejšie prebiehajú výpočty kolízie, tým je väčšia šanca že sa kolízia naozaj zachytí. Avšak keďže aplikácie sú limitované hardvérom tak veľmi jednoducho môže dôjsť k tomu, že kolízia nastala medzi intervalmi vyrátania kolízie a tým pádom nebola zachytená.

Princíp algoritmu ruská ruleta

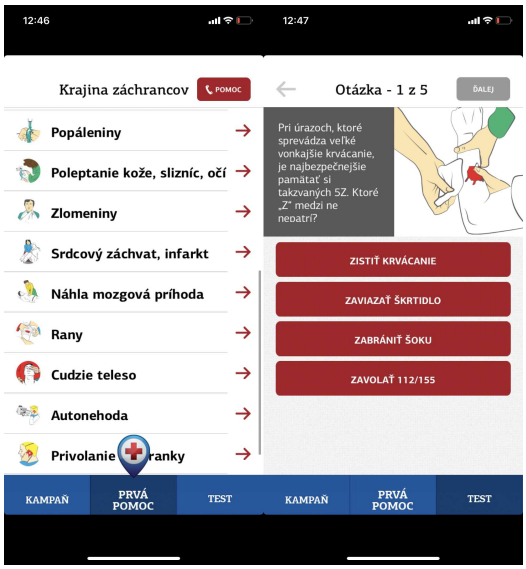
Ide o algoritmus [7] kde sa využíva aj generátor pseudonáhodných čísiel. Generátory majú rôzne podoby. Niektoré z nich sú napríklad XORShift generátor [11], Mersenne Twister generátor [17] alebo Permuted Congruential generátor [2]. Tento algoritmus nadobudol meno podľa známej ruskej hry s revolverom a jedným nábojom v komore. Cieľom tohto algoritmu je znižovať požadované kritérium v každom kroku tak, aby náhodne vygenerované číslo malo väčšiu šancu, že bude menšie ako kritérium podmienky. Čím viac iterácií algoritmu prebehne, tým je väčšia šanca, že v danej iterácii bude podmienka splnená. Algoritmus má využitie keď je potrebné do určitej miery ovplyvniť splnenie podmienky, ale zároveň je neprípustné aby nebola podmienka splnená do určitého časového kritéria.

2.7 Existujúce riešenia danej problematiky

Existuje zopár aplikácií, ktoré sa snažia svoj obsah zamerať na prvú pomoc. Medzi najvyužívanejšie mobilné aplikácie patrí First Aid - IFRC a Prvá pomoc pre všetkých. Avšak ani jedna aplikácia sa nezameriava na detskú vekovú kategóriu a zároveň postrádajú herné prvky čo bolo hlavným cieľom tejto práce Pejsek Záchranář.



Obr. 2.10: First Aid - IFRC



Obr. 2.11: Prvá pomoc pre všetkých

First Aid - IFRC

Ide o aplikáciu, ktorá sa snaží byť využiteľná celosvetovo vďaka možnosti voľby jazyka podľa krajiny kde sa používateľ nachádza. Aplikácia pozostáva z časti kde sa snaží používateľa naučiť čo spraviť pri konkrétnom úraze a z testov viď Obr. 2.10, kde si môže používateľ otestovať skúsenosti ktoré nadobudol z používania aplikácie. Aplikácia je dostupná pre iOS aj Android.

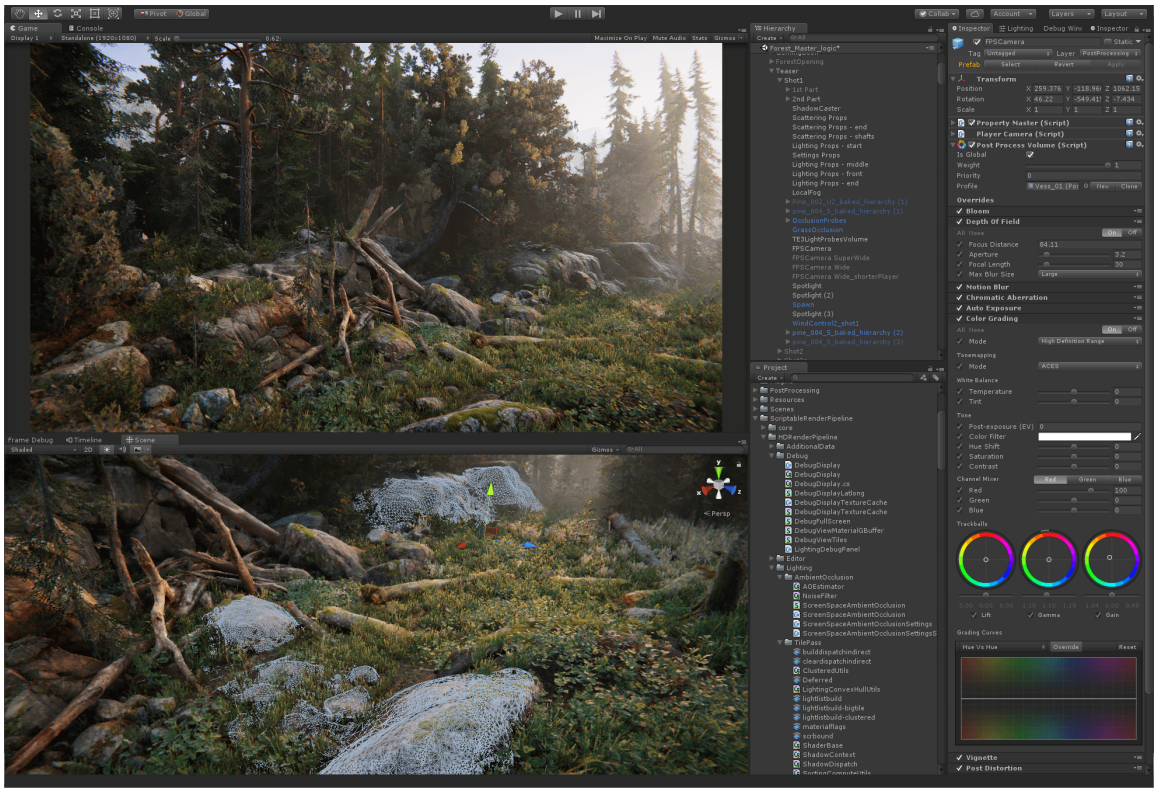
Prvá pomoc pre všetkých

Táto mobilná aplikácia funguje na podobný princíp ako spomínaná aplikácia First Aid - IFRC. Pozostáva z časti kde vysvetľujú ako sa zachovať pri konkrétom úraze a z časti kde prebieha overenie nadobudnutých skúseností. Jej výhodou je že obsahuje slovenskú lokalizáciu narozdiel od First Aid - IFRC. Naopak aplikácia nie je aktualizovaná a nie je responzívna vzhľadom k väčším displejom, ktoré sú v dnešnej dobe bežné viď Obr. 2.11.

Kapitola 3

Unity

Ide o multiplatformový herný engine [4] viď Obr. 3.1, ktorý bol vytvorený spoločnosťou Unity Technologies [1]. Najprv bol oznámený ako engine pre operačný systém OS X, ktorý vlastní spoločnosť Apple. Od tej doby bol rozšírený o podporu ďalších 25 platforiem. Na začiatku Unity podporovalo 2 jazyky [1], ktorými sú Boo a UnityScript. Oba jazyky boli počas vývoja frameworku zastaralé. Taktiež v minulosti Unity malo svoj vlastný webový prehrávač pod menom Unity Web Player [1]. Avšak s príchodom knižnice WebGL sa prestal Unity Web Player využívať a prenechal miesto novej knižnici. V Unity bolo vytvorených viacero známych hier ako je napríklad: Escape From Tarkov, Rust, Cup Head alebo Hearthstone: Heroes of Warcraft. Unity podporuje ako aj 2D, tak aj 3D projekty, ktoré



Obr. 3.1: Unity Engine

pracujú so skriptami v jazyku C# [4]. Je v ňom možné vyvíjať aj projekty vo virtuálnej realite. Tento framework je populárny najviac na mobilných zariadeniach, ale aj napriek tomu je schopný vytvárať kvalitné projekty aj na ostatné zariadenia. Aktuálne podporované platformy sú: mobilné, desktop, webové, konzolové, platforma pre virtuálnu realitu. Unity podporuje kompresiu textúr, mipmaping a nastavenia rozlíšenia podľa platformy, pre ktorú je projekt cielený [4]. Počas prvých 10. rokov existencie frameworku, bol Unity využiteľný iba po zakúpení produktu. V roku 2016 sa to zmenilo a existujú 2 možnosti. Využívanie zadarmo kde licencia umožňuje používať framework na projekty, ktoré ročne zarábajú menej ako 100 000$ alebo platiť predplatné, ktoré je vygenerované na základe zárobkov konkrétneho projektu. Framework obsahuje aj Unity Asset Store kde môžu používatelia nahrávať svoje vlastné pluginy, modely objektov, celé projekty pre komerčné ale aj bezplatné účely. Unity podporuje grafické API ako je DirectX, Metal, OpenGL a Vulkan, kde ich využitie záleží podľa platformy kam je projekt cielený [4].

3.1 Rozhranie

Okno úprav [16] je v Unity rozdelené na viacero podsekcií, ktoré sú Hierarchy, Inspector, Scene View, Game View, Assets a Animations viď Obr. 3.2.

Hierarchy

Toto okno ukazuje pozostáva zo všetkých objektov, ktoré boli vložené do scény [16]. Jedná sa aj o objekty, ktoré nie je vidno, ale stále sú prítomné v scéne. Ide napríklad o objekty so zvukom, pripravenými efektmi alebo skriptami.

Inspector

Tento komponent Unity ukazuje a umožňuje meniť všetky informácie, ktoré nadobúda konkrétny objekt. Informácie sú napríklad vlastnosti, skripty a komponenty, ktoré sú k danému objektu pripojené [16].

Assets

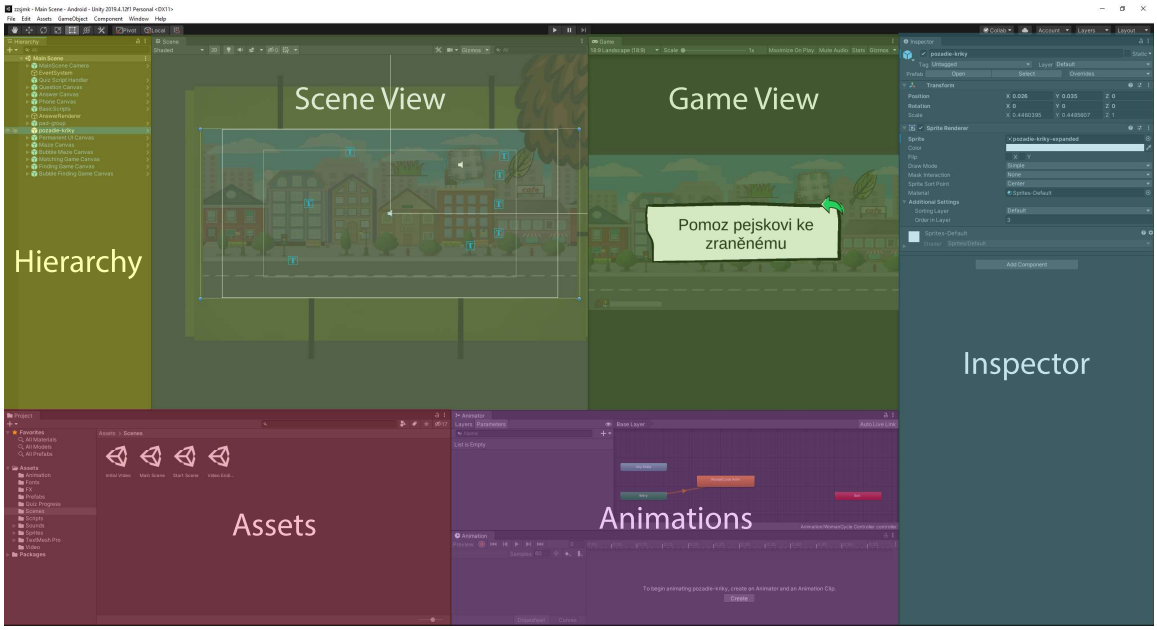
Tento panel [16] zobrazuje a zároveň umožňuje pridávať a odstraňovať všetky súbory, obrázky, animácie, zvukové efekty, skripty, textúry, ktoré sú v projekte využité.

Animations

Tu prebieha tvorba animácií objektov pomocou spritov alebo vytvorením deformácie objektu priamo v Unity. Nastavuje sa tu aj prechod [16] medzi animáciami na základe podmienok, ktoré si stanovuje používateľ.

Scene view

V tomto okno sa vytvára celkový vzhľad projektu [16]. Určuje sa pozícia objektov, dizajnuje sa prostredie, nastavuje kamera.



Obr. 3.2: Unity okno úprav

Game view

Všetko čo je vidieť v tomto okne odzrkadľuje ako bude výsledný projekt vyzerať na cielenom zariadení [16], na ktorom bude aplikácia pracovať čo uľahčuje testovanie funkcionality bez nutnosti neustále nahrávať projekt na cielené zariadenie.

3.2 Komponenty

Objekty bez pridaných komponentov nemajú žiaden význam, okrem využitia ako „zložky“ na utriedenie hierarchie [16]. Komponenty viď Obr. 3.3, ktoré pridávame na objekty sú vlastne skripty napísané v jazyku C#. Hlavnou výhodou je ich znovu použiteľnosť a tiež, že Unity už obsahuje mnohé komponenty, ktoré môžme využiť namiesto tvorby vlastných komponentov. Medzi niektoré základné komponenty Unity [1] patrí:

Rigidbody Umožňuje objektu nadobudnúť vlastnosti fyziky ako je gravitácia alebo trenie. Taktiež je potrebný pre správnu funkcionalitu kolízií.

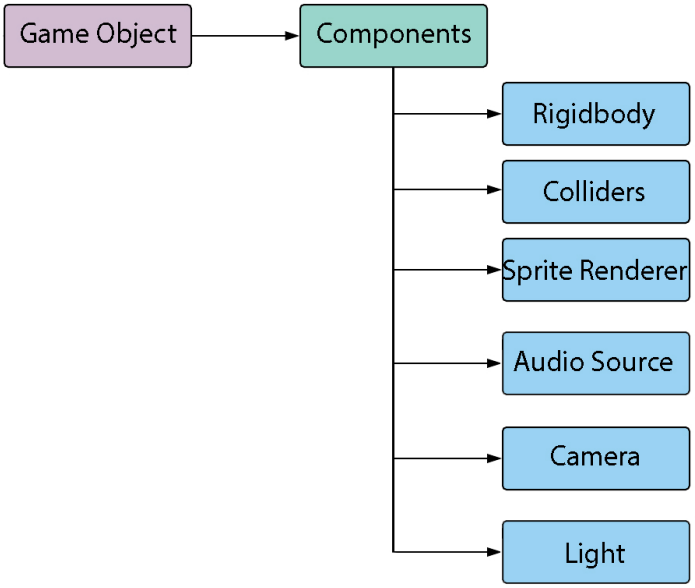
Box/Circle/Polygon Colliders Objekt získa schopnosť detekovania kolízie s iným objektom.

Sprite Renderer Objekt nadobudne konkrétny obrázok alebo farbu a tým určuje vzhľad objektu v scéne.

Audio Source V tomto komponente sa pridávajú zvukové efekty. Ide vlastne o prehrávač audia, ktorý obsahuje pár nastavení ako je napríklad hlasitosť zvuku, prehrávať zvuk v slučke alebo nastavenie priestorového zvuku pri 3D projektoch.

Camera Tento komponent určuje vlastnosti kamera ako je jej pozícia, veľkosť, ktorú má zo scény zaberať naraz alebo či sa jedná o perspektívnu alebo ortografickú kameru.

Light Umožňuje objektu vrhať tiene a simulovať svetelné lúče, ktoré dopadajú na objekt.



Obr. 3.3: Štruktúra komponentov na konkrétnom objekte

Skriptovacie objekty

Jedná sa o objekty [4], ktoré sa využívajú vo frameworku Unity a sú vytvorené na základe vzoru Flyweight. Je to trieda objektov, ktoré umožňujú uchovávať veľké množstvo dát nezávisle od skriptov. Dáta, ktoré uschovávajú nie sú nijak špecifikované a programátor si ich určuje sám podľa potreby. Ich najväčšou výhodou je, že redukujú využitie pamäte a oddeľujú dôležité dáta od samotného kódu aplikácie. Takisto tieto objekty vytvárajú jednoduchý prístup k jednoduchej úprave dát bez úpravy zdrojového kódu aplikácie.

Kapitola 4

Analýza a zhodnotenie súčasného stavu mobilných aplikácií

Cieľom tejto kapitoly je vytvoriť analýzu a zhodnotenie najpoužívanejších herných enginov v aktuálnej dobe na základe rôznych kritérií a zároveň kapitola ďalej špecifikuje cieľ a technické prvky práce.

4.1 Porovnanie dostupných frameworkov

Do porovnania herných enginov boli vybrané 3 najznámejšie, ktorými sú Unity, Unreal Engine a CryEngine. V týchto frameworkoch vyšlo mnoho známych projektov a dodnes sa v nich pracuje na mnohých, ktoré sú stále vo vývoji. Medzi kritéria, ktoré boli zobrané do úvahy patrí:

2D Z pohľadu tohto kritéria sa môže rovno vylúčiť CryEngine pretože vyniká hlavne v tvorbe 3D projektov. Unity aj Unreal Engine sú vhodné pre tvorbu 2D aplikácií avšak vďaka tomu, že Unity bolo primárne zamerané na tvorbu mobilných aplikácií, tak je vytvorenie 2D projektu oveľa intuitívnejšie.

Asset Store Ide o možnosť získania konkrétnych textúr, objektov alebo animácií zo zabudovaného obchodu, ktoré naprogramovali iný používatelia a tým uľahčiť tvorbu nových aplikácií s podobnou tématikou. Unity a Unreal Engine disponujú rozsiahlym obchodom narozdiel od veľmi malého obchodu, ktorý sa nachádza vo frameworku CryEngine.

Dostupnosť V tomto faktore ide o platformy, na ktorých sa dá spustiť konkrétny framework. Unity je schopné pracovať na systéme Linux, Mac a Windows čím je na tom lepšie ako

	Unity	Unreal Engine	CryEngine
2D	⋆⋆⋆⋆⋆	⋆⋆⋆⋆	⋆
Asset Store	⋆⋆⋆⋆	⋆⋆⋆⋆⋆	⋆⋆⋆
Dostupnosť	⋆⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆
Cena	⋆⋆⋆⋆	⋆⋆⋆⋆⋆	⋆⋆⋆
Multiplatformovosť	⋆⋆⋆⋆⋆	⋆⋆⋆⋆⋆	⋆⋆⋆
Dokumentácia	⋆⋆⋆⋆⋆	⋆⋆⋆⋆⋆	⋆⋆⋆

Tabuľka 4.1: Porovnanie jednotlivých frameworkov

konkurencia, kde Unreal Engine podporuje Mac a Windows a CryEngine zaostáva s jedinou podporou pre systém Windows.

Cena Každý framework je dostupný zadarmo do doby kým zárobok z projektu neprekročí konkrétnu hranicu. U Unreal Engine je hranica 1 000 000$, Unity 100 000$ a CryEngine 5000$. Po prekročení tejto hranice sa určité percento zo zárobku presúva spoločnosti, ktorá vlastní framework.

Multiplatformovosť V dnešnej dobe sa postupne každý framework snaží nadobudnúť túto vlastnosť aby bol schopný vytvárať projekty pre všetky typy systémov. Frameworkom Unity a Unreal Engine sa to úspešne podarilo a podporujú všetky používané platformy. CryEngine trochu zaostáva a jeho podpora mobilných platforiem sa momentálne nachádza v beta verzii.

Dokumentácia Čím lepšou dokumentáciou framework disponuje, tým jednoduchšie sa používateľom s ním pracuje. Dokumentácia pri Unity a Unreal Engine je na vysokej úrovni na rozdiel od CryEngine, ktorý ju má celkom chabú a ťažko vyhľadateľnú, čo spôsobuje začiatočníkom veľké problémy.

Celé porovnanie je založené na mojom názore a informáciach, ktoré som vyhľadal, čo môže spôsobiť, že porovnanie nie je úplne objektívne. Na základe výslednej tabuľky 4.1 som sa rozhodol využiť framework Unity pre aplikáciu Pejsek Záchranář.

4.2 Cieľ realizačnej časti práce

Cieľom práce je vytvorenie mobilnej aplikácie Pejsek Záchranář určenú deťom vo veku 6 až 10 rokov kedy už postupne nadobúdajú schopnosť čítať. Aplikácia má ukázať deťom ako sa zachovať v prípade, že sa ocitnú blízko zranenej osoby, ktorá potrebuje prvú pomoc. Aplikácia je prelínaná hernými prvkami, kvízom a animáciami aby si udržala detskú pozornosť a pomohla deťom memorovať jednotlivé kroky postupu.

4.3 Technické parametre práce

Pre vývoj mobilnej aplikácie bol zvolený framework Unity, ktorý využíva pre tvorbu skriptov programovací jazyk C#. Aplikácia sa delí na 5 scén, ktorými sú: hlavné menu, úvodné video do deju, kvíz s prvkami animácie a hier, finálne video, finálna obrazovka s linkami na sesterské aplikácie. Kvíz je robený formou otázok s možnosťami A), B), C). Tieto otázky sa prelínajú s otázkami formou hry. Medzi hry patrí telefón kedy musí dieťa vyťukať správne číslo a zavolať záchranke, bludisko cez ktoré sa potrebuje maskot aplikácie dostať ku zranenému, otázka s výberom správnej pomôcky, ktorou sa dá pomôcť zranenému a samotné vyčkávanie sanitky a upozorneniu kde sa nachádza zranený. Aplikácia podporuje responzívnosť a je schopná sa prispôsobovať rozdielnym veľkostiam mobilných obrazoviek. Mobilná aplikácia podporuje dva najrozšírenejšie operačné systémy, ktorými sú iOS a Android.

Kapitola 5

Návrh mobilnej aplikácie

V tejto kapitole sa popisuje návrh aplikácie, ktorý vznikal pred samotnou implementáciou. Obsahuje bližší popis a mock-up s obrázkami všetkých scén a nastavení aplikácie.

5.1 Mobilná výuková aplikácia

Počiatočným návrhom bolo vytvoriť mobilnú aplikáciu zameranú na edukáciu v odvetví prvá pomoc. Táto aplikácia je navrhnutá na základe požiadavok, ktoré žiadala Zdravotnícka záchranná služba Juhomoravského kraju. Mobilná aplikácia je zameraná na deti vo veku 6 až 10 rokov, kedy sú už schopné čítať. Keďže aplikácia je cielená na deti, tak obsahuje kreslenú grafiku, ktorá je pre detí viac prívetivá. Aplikácia je navrhnutá tak, aby bola orientovaná na šírku mobilných zariadení, čo umožňuje viac priestoru pre tento typ aplikácie. Edukačná časť aplikácie pozostáva z kvízu, na ktorý sa dá odpovedať možnosťami A), B) a C). Tieto kvízové otázky sa prelínajú s hrami aby si aplikácia udržala pozornosť dieťaťa. Kvízové otázky popisujú ako sa zachovať v situácii zraneného cyklistu, ktorý spadol a poškodil si koleno. Prostredie úrazu je zasadené do detského parku s preliezkami, kde najčastejšie dochádza k úrazu. Aby dieťa nebolo vtiahnuté priamo do situácie „vyrieš otázky a pomôž zranenému“, tak úrazu predchádza kreslená animácia cyklistu, ktorý narazí do odpadkového košu a zraní sa. Celý kvíz bude sprevádzaný maskotom, ktorého navrhol maliar zo záchrannej služby a bude stvárňovať postavu, ktorá sa snaží zranenému pomôcť. Maskot pozostáva z pár vytvorených animácií a je stvárnený ako Pejsek Záchranář. Do kvízu sú zakomponované 4 rozdielne minihry, ktoré sú bližšie popísané v sekcii 5.2 v časti Kvíz s hernými prvkami. Po úspešnom dokončení kvízu nastane ďalšia animácia príchodu sanitky k zranenému a jej následného odjazdu so zraneným. Po zapnutí aplikácie sa zobrazí hlavné menu, kde budú možnosti spustiť aplikáciu, otvoriť nastavenia a bude obsahovať okno s licenciami, ktoré je nutné uviesť z dôvodov využitia voľne dostupných obrázkov s licenciami pre tvorbu mobilnej aplikácie. Poslednou scénou v aplikácii bude scéna s odkazmi na sesterské aplikácie a s možnosťou sa vrátiť naspäť do hlavného menu. Taktiež má obsahovať logo zdravotníckej záchrannej služby Juhomoravského kraju v hlavnom menu a na poslednej scéne s odkazmi na ďalšie aplikácie. Mobilná aplikácia má byť ozvučená a taktiež má obsahovať možnosť čítania textu kvízových otázok a odpovedí zvukovou formou kvôli deťom, ktoré ešte nie sú schopné veľmi dobre čítať. Aplikácia má byť multiplatformová a je zameraná na 2 najväčšie mobilné platformy, ktorými sú operačné systémy iOS a Android. Keďže oba systémy obsahujú množstvo mobilných zariadení s rozdielnymi veľkosťami obrazoviek,

tak aplikácia musí byť responzívna aby bola schopná sa prispôsobovať rôznym veľkostiam mobilných zariadení.

5.2 Používateľské rozhranie a prvky scén

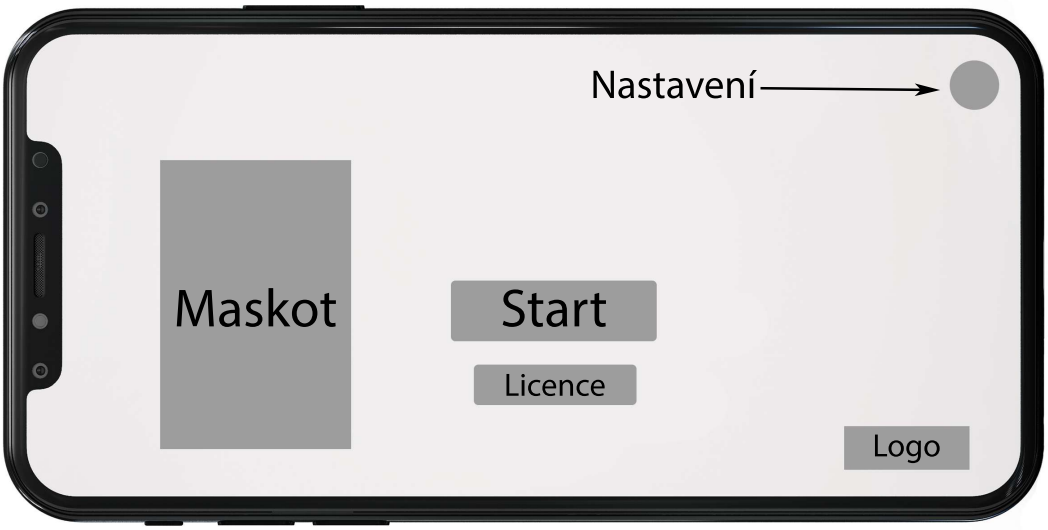
Celá aplikácia Pejsek Záchranář je rozdelená do piatich scén, ktoré sú bližšie rozpísané a znázornené v následujúcich podsekciách. Medzi scény, ktoré sa odohrávajú po sebe v tomto poradí patrí hlavné menu, úvodná animácia, kvíz s hernými prvkami, koncová animácia a okno s odkazmi.

Hlavné menu

Prvá scéna, ktorá sa spustí po zapnutí aplikácie. V strede obrazovky sa bude nachádzať väčšie tlačidlo s názvom „Start“, ktoré bude spúšťať celý obsah mobilnej aplikácie. Pod tlačidlom štart sa bude nachádzať o niečo menšie tlačidlo s názvom „Licence“. Po stlačení tlačidla „Licence“ sa otvorí okno kde sa budú nachádzať odkazy na všetok použitý licencovaný materiál. V pravom dolnom rohu sa bude nachádzať logo zdravotníckej záchrannej služby Juhomoravského kraju. Hlavné menu bude taktiež obsahovať tlačidlo k prístupu k oknu s nastaveniami, ktoré sa bude nachádzať v pravom hornom rohu aplikácie viď Obr. 5.1. Možnosti nastavení sú bližšie popísané v sekcii Možnosti aplikácie. Do ľavej časti od tlačidla „Start“ sa vzhľadom na pozadie hlavného menu vhodne umiestni postava psíka Defíka, čo je maskot celej aplikácie. Pozadie by malo znázorňovať prostredie detského parku, v ktorom dôjde už ku konkrétnemu úrazu.

Úvodná animácia

Animácia pozostáva z udalosti, ktorá vedie k úrazu cyklistu. Pozadie je znázornené ako park s chodníkom a lavičkami na sedenie, kde sa nachádzajú aj odpadkové koše. Cez obrazovku



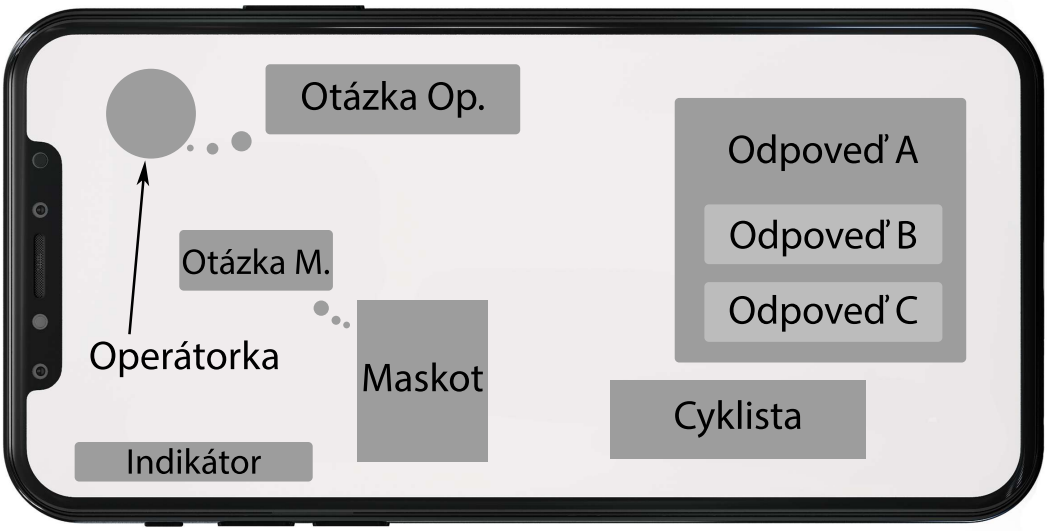
Obr. 5.1: Ukážka rozloženia komponentov v main menu

sa zľava doprava preženú dvaja cyklisti, kde jeden prejde bez úrazu, ale druhý, ktorý pôjde, narazí do vyššie zmieneného odpadkového košu, ktorý sa následne prevrhne a spadne spolu s cyklistom na zem. Po páde cyklistu má cyklista roztrhnuté gate a zakrvavené koleno. V úvodnej animácii je znázornený efekt nárazu aby upútal pozornosť, čo následne prepína animáciu do následujúcej scény s názvom Kvíz s hernými prvkami.

Kvíz s hernými prvkami

Jedná sa o samotný kvíz, ktorý nastane po úraze cyklistu. V ľavej časti obrazovky sa zobrazí maskot aplikácie, ktorý bude podľa odpovedí na otázky naznačovať, či sa jedná o správnu alebo nesprávnu odpoveď. Na pravej strane obrazovky sa bude vyskytovať okno s kvízovými otázkami typu A), B) a C). Používateľ zvolí odpoveď následujúcim ťuknutím na danú možnosť kvízu. Maskot aplikácie bude mať nad hlavou textovú bublinu, v ktorej sa bude textová otázka postupne zobrazovať písmeno po písmene. Nie všetky otázky sú položené z pohľadu psíka záchranára, ale aj z pohľadu operátorky zdravotníckej záchrannej služby. Pokiaľ sa jedná o otázky operátorky, tak sa jej okno s textovou bublinou zobrazí v ľavom hornom rohu a bublina maskota zmizne. V ľavom dolnom rohu bude po celú dobu zobrazený indikátor postupu kvízom viď Obr 5.2. Po každej správne zodpovedanej otázke a dokončenej hre sa indikátor o určitú časť naplní, čo značí postup kvízom. Otázky sa prelínajú spôsobom kvízová otázka, otázka formou hry, kvízová otázka a znovu otázka formou hry. Aplikácia obsahuje štyri minihry medzi ktoré patrí: Vytáčanie čísla, Bludisko, Vybratie správnej pomôcky pre ošetrenie zraneného, Vyčkávanie sanitky a upozornenie na seba po jej príchode.

Vytáčanie čísla Na pravej strane obrazovky sa namiesto okna s kvízom zobrazí mobilný telefón, na ktorom sa bude vyťukávať správne číslo záchrannej služby. Telefón bude mať vzhľad dotykového telefónu a na mieste kde sa zobrazuje volané číslo sa bude po zadaní



Obr. 5.2: Ukážka rozloženia komponentov v kvíze

čísla zobrazovať či sa jedná o správne alebo nesprávne číslo. Po vytočení správneho čísla sa kvíz posunie k ďalšej otázke.

Bludisko Cieľom tejto hry bude dostať psíka záchranára ku zranenému. Na mobilnom zariadení sa zobrazí bludisko na celú obrazovku kde zranený je umiestnený v pravom dolnom rohu a psík záchranár naopak v ľavom hornom rohu. Bludiskom je nutné prejsť bez dotyku stien bludiska inak sa pozícia psíka vráti naspäť na začiatok. Po úspešnom prejdení bludiskom ku zranenému sa hra prepne naspäť do okna s kvízovými otázkami.

Vybratie správnej pomôcky V tejto minihre ide o ošetrenie kolena zraneného. Na obrazovke sa nachádza zranený, psík záchranár a okno s pomôckami na ošetrenie kolena. Používateľ má na výber z troch pomôcok, ktorými mu môže koleno ošetriť. Možnosti sú znázornené ikonami. Potiahnutím ikony na zranené koleno sa vykoná následná animácia podľa ktorej sa dá vydedukovať, či používateľ zvolil správnu pomôcku na ošetrenie.

Vyčkávanie sanitky Kvíz sa nahradí cestou, na ktorej je veľká premávka. Autá prechádzajú cez obrazovku zariadenia v oboch smeroch. Po určitej dobe sa medzi autami preženie sanitka. Používateľ má za úlohu na sanitku kliknúť skorej ako prejde cez obrazovku mobilného zariadenia. Pokiaľ sanitka prejde cez obrazovku, tak sa nič nedeje a vozidlá ďalej vytvárajú premávku. Po náhodne zvolenej dobe sa sanitka znovu preženie cez obrazovku. Tento cyklus sa opakuje do doby kým používateľ na ňu neklikne a tým sa presunie k ďalšej kvízovej otázke.

Koncová animácia

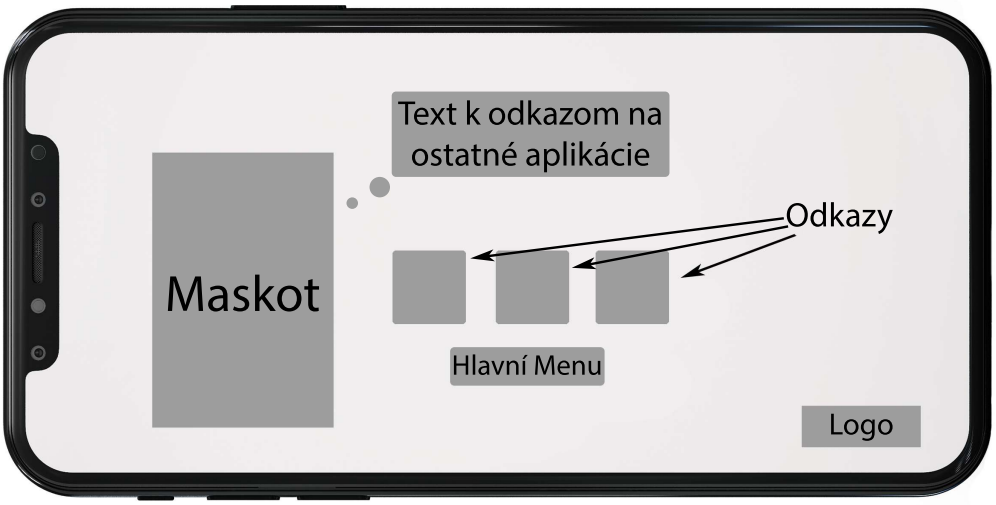
Animácia pozostáva z udalostí, ktoré nastanú po vykonaní prvej pomoci zranenému cyklistovi. Po úspešnom dokončení kvízu sa pozadie aplikácie prepne na rovnaké aké sa nachádzalo v úvodnej animácii, ktoré pozostávalo z lavičiek, chodníku a odpadkových košov. Z pravej strany príde do obrazovky mobilného zariadenia sanitka s blikajúcim majákom a zastaví sa až v úrovni zraneného. Určitú dobu postojí, naloží zraneného a znovu sa dá do pohybu a odíde pravou stranou obrazovky. Po odjazde sanitky z obrazovky mobilného telefónu sa animácia prepína do následujúcej scény s názvom Okno s odkazmi

Okno s odkazmi

Jedná sa o poslednú scénu aplikácie, kde sa budú v strede obrazovky vyskytovať tri odkazy v podobe ikony na tri ďalšie aplikácie zdravotníckej záchrannej služby Juhomoravského kraju. V pravom dolnom rohu sa bude znova nachádzať ich logo a zároveň bude pod odkazmi na zvyšné aplikácie tlačidlo, ktorým sa bude dať vrátiť naspäť do hlavného menu viď Obr 5.3. Názov tlačidla bude „Hlavní Menu“. Pozadie tejto scény bude totožné s pozadím scény hlavného menu.

5.3 Možnosti aplikácie

Po kliknutí na nastavenia sa do popredia hlavného menu rozbalí okno s nastaveniami. V hornej časti okna bude nápis „Nastavení“ a v pravom hornom rohu bude tlačidlo určené na zavretie tohto okna viď Obr. 5.4. Mobilná aplikácia poskytuje dve možnosti nastavenia chodu aplikácie. Medzi funkcie v nastaveniach patrí možnosť ovládať zvukové efekty a zapnúť text čítanou formou. Pri prvom spustení aplikácie sú zvuky zapnuté a text čítanou formou je vypnutý.



Obr. 5.3: Ukážka rozloženia komponentov v poslednej scéne

Zvukové efekty Medzi zvuky patria všetky zvukové efekty, ktoré sa nachádzajú v aplikácii ohľadom animácií, grafického používateľského rozhrania a minihier. Ide o možnosť zapnúť a vypnúť zvukové efekty v aplikácii.

Text čítanou formou Jedná sa o celý text kvízu vrátane otázok a odpovedí, ktorý má okrem písomnej aj formu hlasovej nahrávky. Po zapnutí tohto nastavenia sa daný text prehrá po prvom kliknutí na zvolenú odpoveď. Druhým kliknutím na rovnakú odpoveď používateľ dáva najavo, že ju chce zvoliť ako správnu odpoveď na kvízovú otázku.

Obr. 5.4: Ukážka rozloženia komponentov v nastaveniach

Kapitola 6

Implementácia aplikácie Pejsek záchranář

V tejto kapitole je napísaná podrobná implementácia aplikácie Pejsek Záchranář vrátane back-endu, tvorby vzhľadu aplikácie, využitých techník a hodnotenia aplikácie.

6.1 Back-end aplikácie

Funkcionalita aplikácie je vytvorená z 22. skriptov, ktoré ovládajú rôzne scény aplikácie. Následne bude logika aplikácie rozpísaná postupne podľa toho ako nasledujú scény v aplikácii.

Hlavné menu

Táto scéna využíva 7 skriptov medzi ktoré paria: BasicScriptHolder, SceneLoader, SettingsWindow, LicenseWindow, UISounds, OptionsController a PlayerPrefsController.

BasicScriptHolder je najjednoduchší skript, ktorý má na starosti zabezpečovať orientáciu aplikácie na šírku zariadenia. Keďže je potrebné orientáciu udržovať vo všetkých scénach, tak je aplikovaný do každej jednej scény.

SceneLoader obsahuje základné funkcie, pomocou ktorých ovláda prechod k ďalšej scéne alebo návrat do hlavného menu. Princíp funguje tak, že každá scéna má svoje unikátne identifikačné číslo podľa, ktorého vie SceneLoader umožniť pohyb medzi scénami.

SettingsWindow a LicenseWindow fungujú na rovnaký princíp, kde ich úlohou je zabezpečiť otvorenie a zatvorenie okna s nastaveniami a licenciami na základe používateľského požiadavku.

UISounds má na starosti zabezpečiť zvukové efekty všetkých tlačidiel grafického používateľského rozhrania, ktoré sa nachádzajú v rámci aplikácie. Tieto tlačidlá majú všetky rovnaký zvukový efekt.

OptionsController ovláda funkcionalitu nastavení. Zapisuje informácie o tom či sa vyplo alebo zaplo konkrétne nastavenie pred spustením kvízu. Zároveň nastavuje základné nastavenia, ktoré majú byť nastavené pri spustení aplikácie, ktoré sú zapnutý zvuk a vypnuté čítanie textu.

PlayerPrefsController obsahuje funkcie pomocou ktorých OptionsController zapisuje a udržiava informácie. Tieto informácie sa zapisujú do súboru, ktorý je určený podľa operačného systému, na ktorom beží aplikácia. Takýto súbor je veľmi jednoducho hacknutelný, ale keďže sa jedná len o zapnutie a vypnutie konkrétnych zvukov, tak je úplne jedno či

ich niekto zmení z hry alebo nasilu priamo zo súboru. PlayerPrefsController je v podstate „middleman“ medzi súborom s informáciami a skriptami, ktoré potrebujú prístup k týmto informáciám.

Úvodná animácia

Keďže sa jedná len o scénu s jednoduchou animáciou tak jej funkcionalita bola zachytená do dvoch skriptov, ktoré sú Cyclist a Collision.

Cyclist skript má za úlohu simulovať jazdu cyklistov určitou rýchlosťou zľava doprava. Pohyb vytvára pomocou funkcie update, ktorá pri každom aktualizovaní scény posúva cyklistu o určitú časť po x-ovej ose. Akonáhle cyklista opustí obrazovku mobilného zariadenia tak je odstránený aby zbytočne nezaťažoval hardvér mobilného zariadenia. Zároveň cyklisti obsahujú animáciu bicyklovania, ktoré sú bližšie popísané v sekcii 6.3.

Collision detekuje situáciu, v ktorej cyklista má naraziť do odpadkového koša. Vo chvíli keď táto situácia nastane, tak tento skript spustí efekt nárazu, prepne odpadkový kôš do ležatej polohy a zmení objekt cyklistu na objekt zraneného cyklistu, ktorý leží na zemi. Pár sekúnd po zrážke prepína scénu na scénu s kvízovými otázkami.

Kvíz s hernými prvkami

Ide o hlavnú scénu celej aplikácie. Táto scéna pozostáva z 10. skriptov, medzi ktoré patria: QuizScriptHandler, MazeControls, FinishLine, MatchingGame, CloudMover, TrafficJam, XorShiftRandom, CarsSpawned, AmbulanceClicked, ProgressBar. Logika kvízových otázok a odpovedí je vyriešená pomocou skriptovacích objektov v skripte QuizConfig, do ktorých sa ukladajú potrebné informácie.

Skriptovacie Objekty Pre správny chod kvízu, je pre každú otázku vytvorený jeden skriptovací objekt. Obsahujú viacero informácií, ktoré sú potrebné pre funkčnosť kvízu. Do týchto objektov sú zapísané tieto informácie: kvízová otázka, možnosti odpovedí k danej otázke, počet správnych odpovedí, označenie správnych odpovedí, prepojenie na následujúcu otázku, číslo otázky, informácia či sa jedná o otázku položenú operátorom alebo maskotom, prepojenia na konkrétne zvukové nahrávky otázok a odpovedí. Zároveň na každý skriptovací objekt sú v skripte QuizConfig vedené funkcie, ktoré sú schopné vytiahnuť konkrétnu informáciu, ktorú môže chod aplikácie v danej chvíli vyžadovať.

QuizScriptHandler je hlavný skript, ktorý manažuje chod celej scény s kvízom a jeho hernými prvkami. Jeho základnou úlohou je vložiť do vopred nachystaného grafického rozhrania otázky a odpovede na základe informácií, ktoré obsahujú skriptovacie objekty. Následne po zvolení správnej odpovede vymieňa obsah textových polí za nové kvízové otázky a odpovede a aktualizuje informácie ohľadom novej otázky kvízu. Taktiež ovláda spúšťanie animácií maskota viď sekcia 6.3 na základe odpovedí, ktoré zvolil používateľ. Má na starosti vykonať efekt písania otázok a odpovedí písmeno po písmene na čo využíva funkciu coroutine. Táto funkcia umožňuje pozastaviť časť kódu na dobu dokiaľ nie je splnená nejaká konkrétna podmienka. Medzi takúto podmienku môže patriť napríklad čas, ktorý musí uplynúť pred vykonaním zvyšku kódu v coroutine. Ďalej kontroluje kedy má počas kvízu nastať hra a automaticky prepína scénu do scény s minihrou. Prvú minihru má ešte na starosti QuizScriptHanlder. Ide o minihru telefón. Skript čaká na vstupy od používateľa, ktorý vyťukáva číslo na telefóne. Postupne ako sa číslo vyťukáva tak si ho skript zapamätáva a zároveň na displej mobilu vypisuje zadané číslo. Po vytočení zadaného čísla skript skontroluje či sa jedná o správnu variáciu čísla. Pokiaľ je číslo správne spustí efekt vytáčania a následne prepne scénu na ďalšiu kvízovú otázku. Ak sa jedná o nesprávne číslo, tak o tom informuje

používateľa a znovu ho nechá skúsiť zavolať na správne číslo. Ďalšou minihrou, ktorú spúšťa je bludisko. Bludisko je ovládané dvomi skriptami, ktorými sú MazeControls a FinishLine.

MazeControls umožňuje používateľovi pohybovať s guľôčkou v rámci bludiska na základe toho, že guľôčka nasleduje pohyb prstu po obrazovke mobilného zariadenia. Pokiaľ guľôčka narazí do steny bludiska, tak sa jej pozícia automaticky reštartuje na štartovnú pozíciu.

FinishLine sa spustí po úspešnom prejdení bludiska do cieľa. Skript detekuje či sa guľôčka nachádza v cieli a pokiaľ áno, tak scénu prepne naspäť do kvízovej časti a znovu prenechá vedenie skriptu QuizScriptHandler.

Treťou minihrou je vybranie správnej pomôcky na ošetrenie kolena. Túto minihru ovláda iba jeden skript s názvom MatchingGame.

MatchingGame vybranie pomôcky detekuje pomocou funkcií *OnMouseUp()*, ktorá premiestňuje pomôcku po obrazovke zároveň s prstom používateľa a *OnMouseDown()*, ktorá reštartuje pozíciu pomôcky. Pri priložení nesprávnej pomôcky na koleno zraneného cyklistu sa vykoná animácia naznačujúca nesprávny výber a nechá používateľa vybrať inú pomôcku. Po pretiahnutí správnej pomôcky na koleno sa vykoná animácia maskota, ktorá naznačuje správny výber, prepne scénu naspäť do kvízu a prenechá znovu ovládanie skriptu QuizScriptHandler.

Poslednou minihrou je vyčkávanie sanitky. Pozostáva z piatich skriptov, ktorými sú: CloudMover, TrafficJam, XorShiftRandom, CarsSpawned a AmbulanceClicked.

CloudMover má za úlohu simulovať pohyb oblakov. Neviditeľným objektom v scéne „cloudSpawner“ je určené miesto, kde sa má oblak vytvoriť a následne prechádza obrazovkou mobilného zariadenia. Po opustení obrazovky sa objekt oblaku vymaže a následne sa vytvorí nový objekt na určenom mieste pre tvorbu oblakov. Takýmto spôsobom sa oblaky pohybujú do nekonečna. Každý typ oblaku má svoju rýchlosť pohybu a miesto pre vytvorenie jeho kópie čím sa zaisťuje náhodnosť pohybu oblakov.

TrafficJam využíva princíp algoritmu ruská ruleta pre zvýšenie šance vytvorenia sanitky do premávky áut viď sekciu 6.4. Zároveň využíva dva neviditeľné objekty „spawnery“, na ktoré vytvára nové objekty vozidiel, ktoré vytvárajú premávku. Každý „spawner“ ovláda jeden pruh cestnej premávky. Čas medzi vytváraním nových vozidiel je náhodne vybraný z konkrétneho intervalu, ktorý bol prispôsobený rýchlosti pohybu vozidiel. Pokiaľ sa v scéne už vyskytuje sanitka, tak skript zabraňuje vytvoreniu ďalšej sanitky dokiaľ sa objekt predchádzajúcej sanitky neodstráni zo scény.

XorShiftRandom je skript, ktorý generuje náhodné čísla pomocou posuvných registrov. Je využitý v princípe algoritmu ruskej rulety viď sekciu 6.4.

CarsSpawned rieši orientáciu vozidla po jeho vytvorení. Keďže sa jedná o premávku a vozidlá jazdia v oboch smeroch, tak je nutné vozidlo aj správne otočiť. Otáčanie vozidla funguje na základe informácie, že každé novo vytvorené vozidlo je nasmerované na cestný pruh smerom doľava. Vďaka tomuto sa objekt s vozidlom iba otočí o 180 stupňov podľa potreby. Niektoré vozidlá obsahujú nápisy, ktorým je treba zabrániť v otočeniu s vozidlom a umiestniť ich na správne miesto. Pokiaľ by sa nápis otáčal s vozidlom vytváral by sa zrkadlový efekt. Tento skript pracuje s informáciou či otáčané vozidlo obsahuje nápis a pokiaľ áno, tak prekryje jeho nápis druhou variáciu nápisu pre otočené vozidlá. Každý objekt vozidla, ktorý sa dostane mimo obrazovku mobilného zariadenia je automaticky odstránený zo scény aby zbytočne nezaťažoval hardvér mobilného zariadenia.

AmbulanceClicked je jednoduchý skript, ktorý vyčkáva kedy používateľ ťukne na vozidlo sanitky. Po kliknutí na sanitku odstráni všetky naklonované objekty zo scény a vráti ovládanie naspäť skriptu QuizScriptHandler.

Zakaždým keď QuizScriptHandler požaduje novú kvízovú otázku a odpovede zároveň zavolá skript ProgressBar.

ProgressBar vykoná aktualizáciu indikátora progresu v rámci aplikácie a zvýši jeho hodnotu o 1 / počet všetkých otázok kvízu.

Po dokončení poslednej minihry dokončí skript QuizScriptHandler zvyšok kvízu a vo chvíli kedy sa zodpovedá posledná otázka, tak prepne kvíz na predposlednú scénu pod názvom Koncová animácia.

Koncová animácia

Táto scéna pozostáva z jedného skriptu pod názvom EndingVideo, ktorý riadi beh celej animácie.

EndingVideo funguje na podobný princíp ako skript Cyclist. Určuje rýchlosť pohybu sanitky cez obrazovku mobilného zariadenia pomocou funkcie Update(). Na pozícii so zraneným cyklistom je umiestnený neviditeľný objekt, ktorý detekuje kolíziu. Vo chvíli keď sa ho sanitka dotkne tak skript zastaví sanitku na pár sekúnd a za túto dobu sa premiestni zranený cyklista do sanitky. Následne sanitka odchádza z obrazovky telefónu. Po jej odchode sa objekt sanitky vymaže a skript prepne scénu na poslednú s názvom Okno s odkazmi.

Okno s odkazmi

Je poslednou a finálnou scénou aplikácie. Má nad ňou kontrolu iba jeden skript s názvom EndingScene.

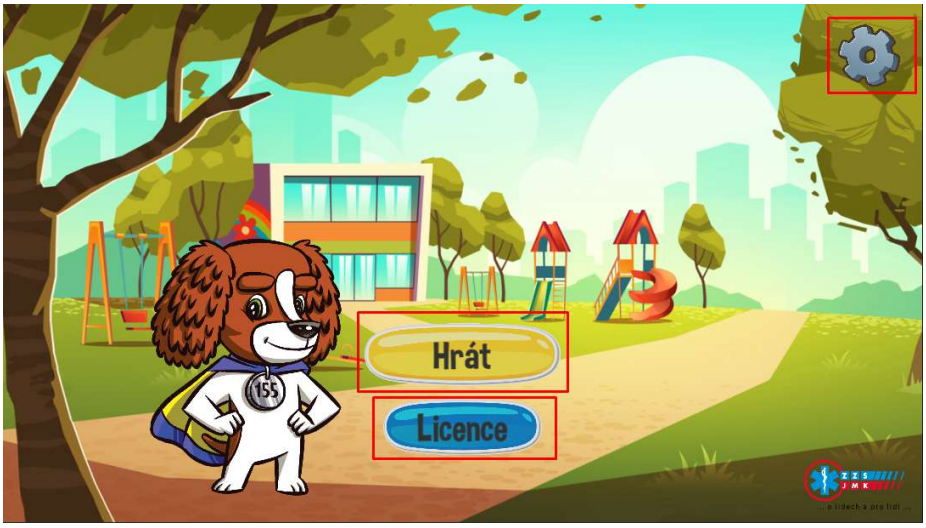
EndingScene umožňuje používateľovi sa vrátiť na začiatok aplikácie alebo sa prekliknúť na konkrétnu aplikáciu do obchodu so zvyšnými troma aplikáciami určenými na prvú pomoc pri úraze, ktoré vlastní zdravotnícka záchranná služba Juhomoravského kraju. Zároveň pri aktivácii tejto scény skript spúšťa postupné vypísanie textu maskota na záver aplikácie.

6.2 Vizuálna stránka a GUI aplikácie

Z dôvodu, že všetky scény využívajú podobné a aj rovnaké komponenty pre tvorbu vizuálnej stránky aplikácie, tak následný popis ich využitia nebude kategorizovaný podľa scén. Pre tvorbu pozadia jednotlivých scén boli použité volne dostupné obrázky, ktoré museli byť následne dokreslené v programe Adobe Photoshop 2020, pretože ich veľkosť nebola dostatočne veľká aby pokryla aj veľkosti obrazoviek najnovších smartfónov. Taktiež v ňom dochádzalo k úpravám animácií cyklistov, okna nastavení alebo aj k úprave polôh maskota pre vytvorenie animácií. Medzi použité komponenty grafického rozhrania patria: Camera, Button, Toggle, TextMeshPro a Text, SpriteRenderer, Canvas, Image, Slider.

Camera je v tejto aplikácii statická. Keďže nie je potrebné aby sa pohybovala z miesta na miesto počas chodu aplikácie, tak automaticky naraz zaberá celú plochu diania.

Button je základný komponent každého frameworku, ktorý umožňuje vytvárať objekty, ktoré reagujú na kliknutie. Tomuto komponentu sa dá meniť napríklad jeho vzhľad, typ farby pred a po kliknutí. Tento komponent je využitý na viacerých miesta v rámci aplikácie. Jeho hlavné využitie je v hlavnom menu kde je zakomponovaný do tlačidiel „Start“, „Licence“ a tlačidla „Nastavení“, ktoré je prevedené pomocou ikony ozubeného kolieska viď Obr. 6.1. Využíva sa aj pri samotnom kvíze, kde je zapracovaný do možností A), B) a C). Komponent predstavuje hlavný spôsob komunikácie používateľa s aplikáciou.

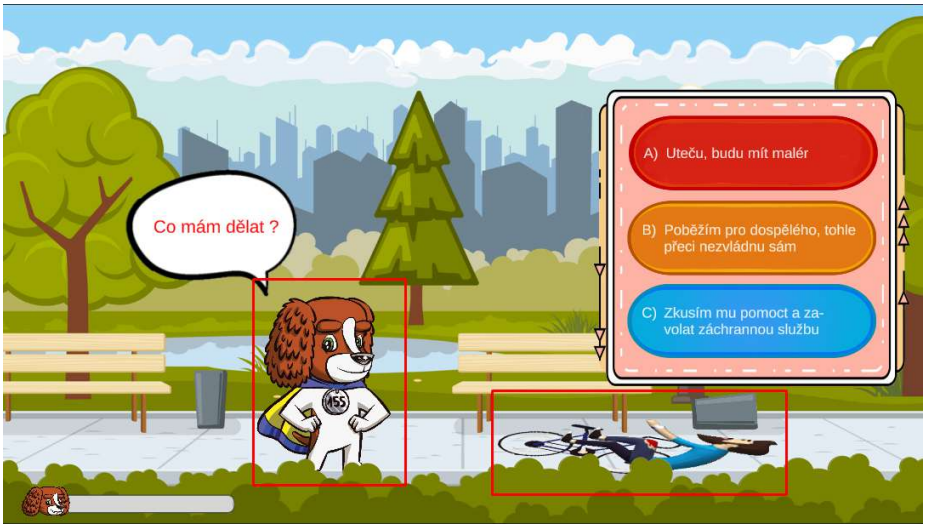


Obr. 6.1: Ukážka využitia komponentu Button

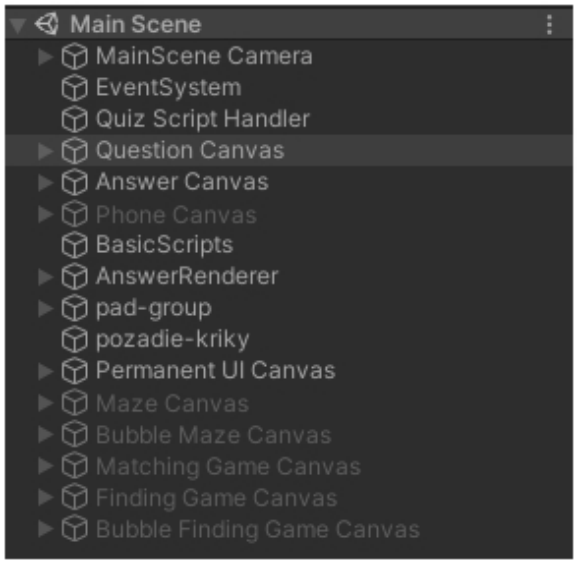
Toggle je jednoduchý komponent, ktorý berie vstup od používateľa na základe logiky true a false. V aplikácii má využitie v nastaveniach, kde rozhoduje o tom či budú konkrétne nastavenia zapnuté alebo vypnuté. Jeho vzhľad je prevedený formou klasického zaškrtávacieho poľa, ktoré sa nachádza takmer v každom online dotazníku a používatelia už sú s nimi oboznámení.

TextMeshPro a **Text** umožňujú kontrolu nad textom v aplikácii. Menia napríklad farbu, veľkosť, typ písma, obsah textu, umiestnenie v ploche. Sú využité všade kde sa v aplikácii nachádza text. Sú využité aj v komponentoch Button viď. Obr. 6.1.

SpriteRenderer umožňuje objektom pridať vzhľad. Takýto komponent je využitý pri každom pozadí celej scény alebo objekte, ktorý má mať svoju statickú pozíciu v scéne bez ohľadu na veľkosť obrazovky mobilného zariadenia. Ako ukážka sa dá použiť maskot a zranený cyklista v scéne s kvízom viď Obr. 6.2.



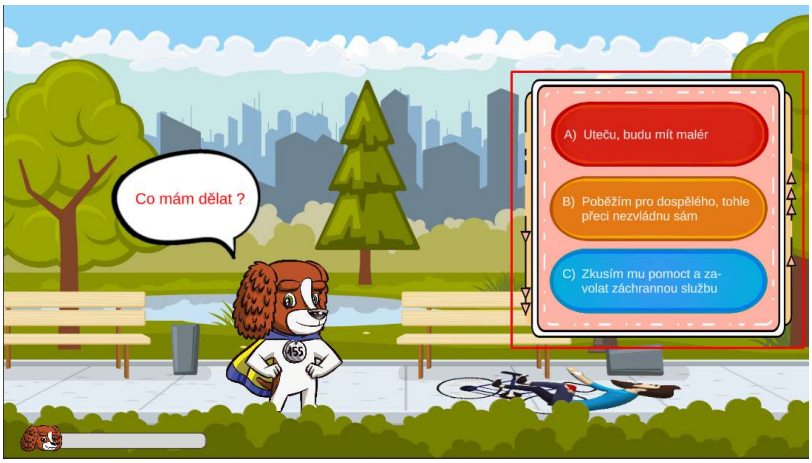
Obr. 6.2: Ukážka využitia komponentu SpriteRenderer



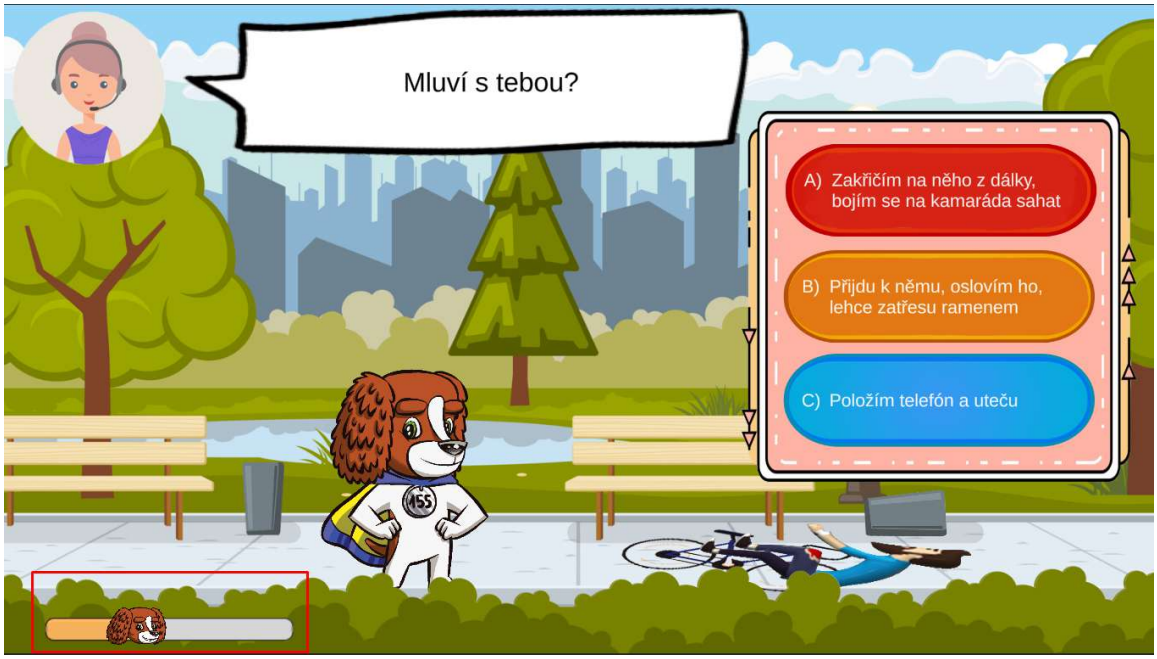
Obr. 6.3: Ukážka hierarchie obsahujúcej jednotlivé objekty

Canvas inak povedané plátno je komponent, pod ktorý sa umiestňujú grafické prvky aplikácie. Keďže v scéne s kvízom sa nachádzajú minihry, otázky od operátorky a od maskota, tak je každá zo spomenutých vecí rozdelená do vlastného plátna. Vďaka tomu sa dá jednoducho prepínať medzi jednotlivými plátnami podľa potreby. Každé plátno dostalo svoj jedinečný identifikátor pod ktorým vystupuje v back-end časti aplikácie. Štruktúra scény z obrázku 6.2 potom vyzerá následovne viď Obr. 6.3. Tmavé plátna sú zo scény kvízu vypnuté a svetlé naopak zapnuté a vytvárajú aktuálnu scénu kvízu.

Image na rozdiel od komponentu SpriteRenderer sa používa presne pri opačných situáciach. Jedná sa o objekty, ktoré musia byť prítomné na obrazovke mobilného zariadenia za každú cenu závisle na veľkosti displeju. Vďaka nemu môžu byť komponenty zakaždým rovnako vzdialené od kraju obrazovky telefónu. Takýmto spôsobom je vyriešené napríklad okno s kvízovými možnosťami viď Obr. 6.4. Okno bude na obrazovke zariadenia prítomné za každých okolností.



Obr. 6.4: Ukážka využitia komponentu Image

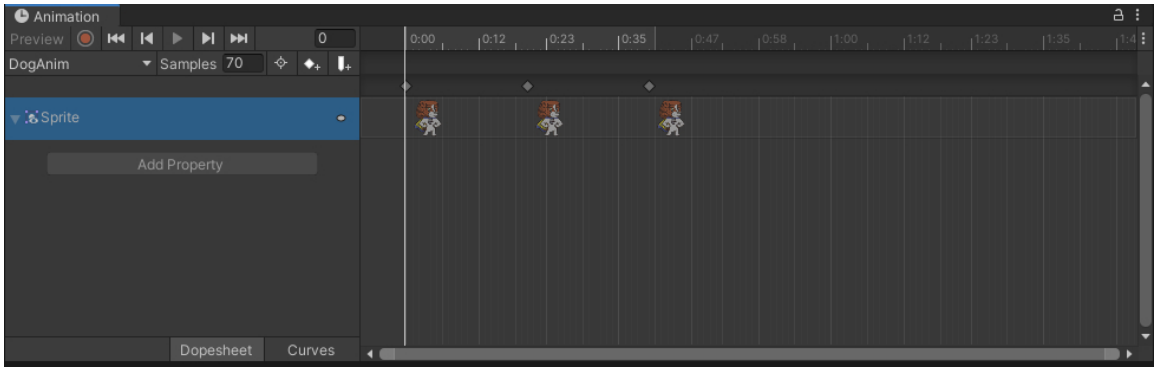


Obr. 6.5: Ukážka využitia komponentu Slider

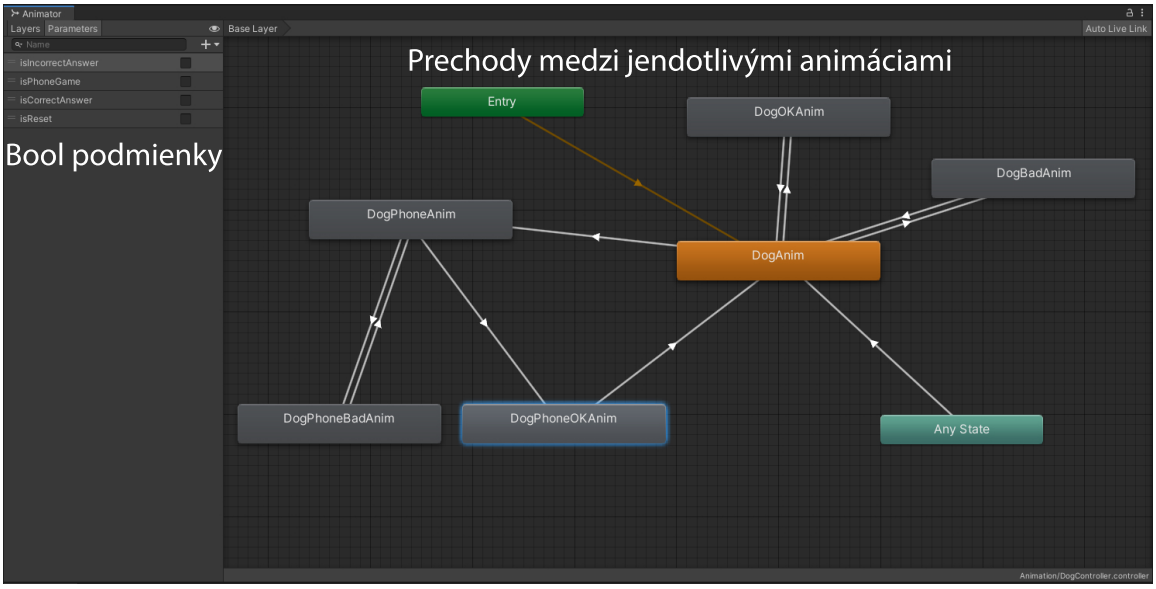
Slider je komponent ktorý sa dá presúvať pomocou rukoväte z minimálnej na maximálnu hodnotu. V aplikácii je zakomponovaný ako indikátor progresu kvízu. Bola mu odobraná možnosť posúvania rukoväte pomocou používateľa a pridaná funkcionalita kde sa mení farba posuvníku pred rukoväťou. Rukoväť bola vymenená za hlavu maskota, ktorá indikuje progres v rámci kvízu viď Obr. 6.5.

6.3 Riešenie animácií

Implementácia animácií bola vykonaná pomocou funkcionalít zahrnutých vo frameworku Unity. Animácia sa vytvára pomocou dvoch prvkov ktoré sú Animator a Animation. Najprv sa v komponente Animation viď Obr. 6.6 vytvorili jednotlivé animácie, ktoré má maskot byť schopný vykonávať. Animácie pozostávajú z jednotlivých obrázkov, ktoré zachytávajú rozdielne polohy maskota. Podľa potreby sa animáciám pridal „loop time“, ktorý umož-



Obr. 6.6: Ukážka vzhľadu a využitia komponentu Animation



Obr. 6.7: Ukážka vzhľadu a využitia komponentu Animator

nil neustále opakovanie danej animácie. Takto vytvorené animácie bolo nutné medzi sebou prepojiť na čo slúži komponent Animator viď Obr. 6.7. Keďže maskot potreboval vedieť viacero animácií ako je ukázať palec hore, zakrútiť prstom a upozorniť, vrtieť chvostom, tak sa animácie vykonávali pomocou prechodov. Ako podmienky pre prechod medzi jednotlivými animáciami boli využité podmienky typu bool, ktoré sa vyhodnocujú v back-ende aplikácie. Po vytvorení celého chodu animácie sa pripojil na konkrétny objekt s maskotom komponent Animator, do ktorého sa pridali konkrétne animácie, ktorými mal objekt disponovať.

6.4 Ostatné prvky implementácie

V následnej časti je bližšie popísaný spôsob implementácie konkrétnych prvkov, ktoré boli využité pri tvorbe aplikácie.

Ruská ruleta

Princíp tohto algoritmu je využitý pri generovaní sanitky do scény. Okrem sanitky dochádza ku generovaniu aj ostatných typov vozidiel. V aplikácii je využitých 18 typov vozidiel vrátane sanitky. Algoritmus ruskej rulety má na starosti ovplyvniť náhodnosť generovania sanitky tak, aby sa vygenerovala v krátkom čase s určitou náhodnosťou. Logika ovplyvnenia náhodnosti začína vygenerovaním náhodného číslo pomocou náhodného XorShift generátoru čísel. Toto číslo je následne porovnané s hodnotou $q = 1/n$, kde $n =$ počet všetkých vozidiel. Pokiaľ je vygenerované číslo väčšie ako hodnota q, tak sa do scény generuje náhodne vybrané vozidlo nepočítajúc sanitku a následne sa upraví hodnota q pomocou následujúceho vzorcu 6.1:

$$q = \frac{q}{(1-q)} \tag{6.1}$$

Akonáhle dôjde k ďalšiemu generovaniu vozidla, tak sa znovu vygeneruje nové náhodné číslo a toto číslo sa už porovnáva s novonadobudnutou hodnotu q z predchádzajúceho vzorcu.

Týmto spôsobom sa zakaždým znižuje kritérium hodnoty q, čím sa zvyšuje šanca splnenia danej podmienky, že náhodne vygenerované číslo bude menšie ako hodnota q. Vo chvíli kedy je podmienka splnená, tak sa automaticky generuje vozidlo sanitky.

Detekcia kolízií

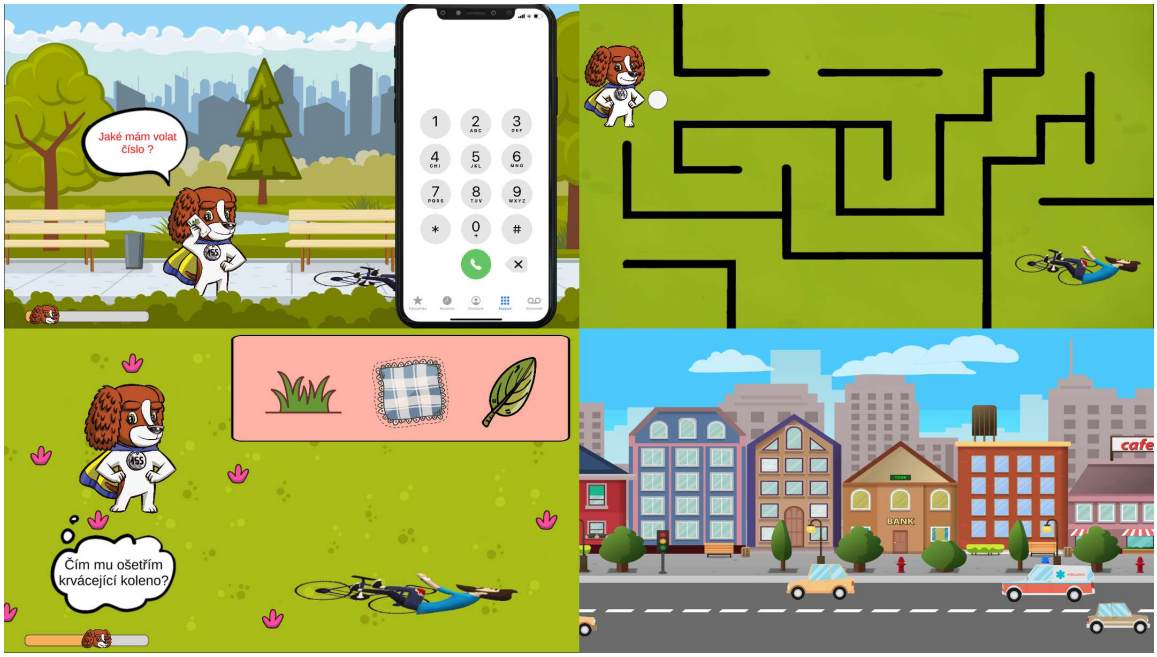
Zachytenie kolízie objektov sa v Unity vykonáva pomocou pár komponentov. Ide o komponent typu Rigidbody2D a komponent Collider2D, ktorý môže byť rôznych tvarov ako je napríklad Circle, Box, Polygon. Kolízia objektov je v minihre bludisko implementovaná tým spôsobom, že steny v bludisku majú na sebe komponent BoxCollider2D, ktorý má na starosti zachytávať kolíziu. K jeho správne funkčnosti musia steny obsahovať Rigidbody2D, ktorý objektom pridáva fyzikálne vlastnosti. Objekt bez komponentu Rigidbody2D nie je schopný detekovať kolízie. Keďže na danú minihru nemá vplývať gravitácia tak sa typ ich tela nastavil ako kinematický. Tak isto musí komponent Collider2D obsahovať aj objekt, ktorý má byť schopný prechádzať bludiskom. Objekt obsahuje rovnaké komponenty ako steny bludiska s jediným rozdielom kde sa využil komponent Circle Collider2D namiesto Box Collider2D keďže sa jedná o objekt okrúhleho tvaru. Navyše má tento objekt v komponente Circle Collider2D zapnutú možnosť „Is Trigger“. Vďaka tomuto je schopný tento objekt ohlásiť každú kolíziu do ktorej sa v bludisku dostane. V back-ende aplikácie sa takáto kolízia odchytáva pomocou funkcie *OnTriggerEnter2D()* kde sa už vykonávajú konkrétne akcie, ktoré majú nastať po kolízii objektov. Keďže pri detekovaní kolízie dochádza pri každom „frame“ aplikácie, tak pri rýchlych pohyboch objektom v bludisku nemuselo dochádzať ku kolízii, pretože objekt prešiel cez stenu bludiska v čase medzi vyhodnocovaním jednotlivých „frame-ov“. Tento problém bol vyriešený pomocou nastavenia maximálnej rýchlosti, ktorou sa môže objekt v bludisku pohybovať. Na podobný princíp detekcie kolízií fungujú aj zvyšné časti aplikácie, kde bolo nutné detekovať kolízie objektov.

Čítanie textu

Do aplikácie je zakomponované aj čítanie textových otázok a kvízov. Ide o nahrávky každej otázky a odpovede v rámci kvízu. Vo chvíli kedy sa zapne v nastaveniach možnosť čítania textu, tak sa trochu pozmení ovládanie aplikácie. Pri prvom kliknutí na jednu z daných možností odpovedí sa text prehrá zvukovou formou. Pri druhom kliknutí sa už očakáva, že používateľ zvolil danú odpoveď ako správnu. Táto možnosť čítania textu funguje na jednoduchej logike kde sa pomocou hodnoty bool vyhodnocuje či sa na otázku kliklo prvý alebo druhý krát a podľa toho sa už vykonávajú jednotlivé akcie aplikácie.

6.5 Používateľské štúdium a hodnotenie

V aplikácii sa počas testovania odhalilo pár chýb, ktoré boli následne spracované a odstránené. Jedná sa o chybu, kde indikátor progresu v rámci kvízu bol ovládateľný používateľom, kde jeho zmyslom bolo graficky zobrazovať posun v rámci kvízu. Ďalšia objavená chyba bola počas zapnutej možnosti čítania textu. Pri kliknutí na ďalšiu odpoveď sa neprečítal text automaticky, ale spustil sa až po tom ako sa dočítal text z predchádzajúcej odpovede, na ktorú bolo kliknuté. Pri minihre bludisko bolo možné pri vyššej rýchlosti „preskakovať“ steny, čo bolo následne ošetrené určením maximálnej rýchlosti kolieska po bludisku. Poslednou objavenou chybou v aplikácii boli nefungujúce zvuky v poslednej scéne s odkazmi. Počas testovania neboli odhalené žiadne ďalšie chyby.



Obr. 6.8: Ukážka minihier v aplikácii

Najviac pozitívne hodnotenou funkciou aplikácie bolo spracovanie indikátoru progresu a zároveň jednoduchosť ovládania aplikácie. Narozdiel od toho bolo najviac sťažností na minihru bludisko, kde na menších obrazovkách zariadení bol problém uchopiť koliesko, ktoré je určené na pohyb po bludisku. Tento problém bol následne vyriešený zväčšením plochy bludiska na celú obrazovku zariadenia, zväčšením neviditeľného objektu, ktorým sa dá objekt uchopiť a malým zredukovaním šírky stien bludiska čo umožnilo väčší priestor pre pohyb. Mimo vyššie spomínaného problému sa pri minihrách viď Obr. 6.8 žiaden iný problém nevyskytol.

Aplikácia bola testovaná na najstaršom možnom dostupnom zariadení, ktorým bol Samsung Galaxy A5 (2017), čím je zaručená kompatibilita na zariadeniach s podobným a lepším hardvérom. Na tomto zariadení sa nachádza systém android verzie 8.0.0. Aplikácia Pejsek Záchranář zaberala približne 150 mb pamäte RAM a s veľkosťou na disku 65,84 mb. Výsledný produkt bude časom umiestnený do obchodu oboch platforiem, ktorými sú GooglePlay a AppStore.

Kapitola 7

Záver

Cieľom práce bolo naštudovať vývoj mobilných aplikácií so zameraním na edukáciu, frameworky, ktoré sa zaoberajú tvorbou aplikácií pre mobilné zariadenia a následne samotné spracovanie aplikácie vo vybranom frameworku. Tento cieľ bol splnený.

V druhej časti sú popísané materiály, ktoré bolo nutné naštudovať pre tvorbu mobilnej aplikácie. V tretej časti je bližšie popísaný framework Unity, ktorý bol využitý pri tvorbe aplikácie Pejsek Záchranář. Framework Unity bol vybraný na základe tabuľky v štvrtej časti, ktorá bola zhotovená podľa osobne vybraných kritérií a preferencií. Priebeh návrhu mobilnej aplikácie aj s vizuálnym prevedením je popísaný v piatej časti. Šiesta časť obsahuje bližší popis implementácie danej aplikácie. Pre tvorbu tejto práce boli využité programy, ktorými sú framework Unity, Adobe Photoshop, jazyk C# a rôzne upravené obrázky, ktoré boli dostupné pod licenciami, ktoré umožňovali ich použitie na nekomerčné účely.

Výsledná aplikácia Pejsek Záchranář splňuje kritéria, ktoré boli stanovené pri návrhu aplikácie. Spĺňa svoj edukačný zámer v situácii kedy je potrebné aby dieťa poskytlo prvú pomoc. Jej grafické prevedenie spĺňa kreslenú tematiku, obsahuje samotný kvíz, do ktorého boli otázky a odpovede poskytnuté Zdravotníckou záchrannou službou Juhomoravského kraju, ktorý sa prelína s hernými prvkami v podobe štyroch minihier, ktorým účelom je si udržať detskú pozornosť. Aplikácia je zameraná na detskú vekovú kategóriu vo veku 6 až 10 rokov kvôli čomu je v aplikácii zahrnutá aj možnosť čítania textu zvukovou formou. Výsledná aplikácia je responzívna a taktiež má vlastnosť multiplatformovosti, kde sa zameriava na systémy typu iOS a Android.

Vytvorením mobilnej aplikácie Pejsek Záchranář som si zlepšil znalosti o tom ako vyzerá proces tvorby mobilných aplikácií, bližšie som spoznal program Adobe Photoshop, v ktorom boli nutné úpravy použitých obrázkov a hlavne som sa zoznámil s frameworkom Unity a jazykom C#, ktoré som musel podrobne naštudovať aby som bol schopný implementovať navrhnutú aplikáciu pre mobilné zariadenia.

Na aplikácii je nutná aj práca v blízkej budúcnosti. Ide o pridanie aplikácie do obchodov oboch operačných systémov kde sa jedná o AppStore a GooglePlay. Taktiež po nahraní všetkých sesterských aplikácií do obchodu, ktoré sú zamerané na prvú pomoc je nutné dodať do poslednej predpripravenej scény aplikácie odkazy s ich ikonami pre jednoduché prepojenie so zvyšnými sesterskými aplikáciami.

Literatúra

[1] Blackman, S. *Beginning 3D Game Development with Unity 4: All-in-one, multi-platform game development.* 2. vyd. Apress, január 2013. 808 s. ISBN 978-1-4302-4899-6.

[2] Bouillaguet, C., Martinez, F. a Sauvage, J. Practical seed-recovery for the PCG Pseudo-Random Number Generator. *IACR Transactions on Symmetric Cryptology.* September 2020, zv. 2020, č. 3, s. 175–196. DOI: 10.13154/tosc.v2020.i3.175-196.

[3] Brooks Lewis, A., Brahnam, S., Kapralos, B. a Jain C., L. *Recent Advances in Technologies for Inclusive Well-Being: From Worn to Off-body Sensing, Virtual Worlds, and Games for Serious Applications.* 1. vyd. Morgan and Claypool Publishers, február 2017. 405 s. ISBN 978-1-6084-5866-0.

[4] Dr. Edward, L. *Getting Started with Unity 5: Leverage the power of Unity 5 to create amazing 3D games.* 1. vyd. Packt Publishing, máj 2015. 184 s. ISBN 978-1-7843-9831-6.

[5] Ericson, C. *Real-Time Collision Detection.* 1. vyd. CRC Press, december 2004. 632 s. ISBN 978-1-5586-0732-3.

[6] Gundlach, S. a Martin K., M. *Mastering CryENGINE: Use CryENGINE at a professional level and master the engine's advanced features to build AAA quality games.* 1. vyd. Packt Publishing, apríl 2014. 278 s. ISBN 978-1-7835-5025-8.

[7] Hammersley, J. *Monte Carlo Methods: Monographs on Applied Probability and Statistics.* 1. vyd. Springer, Dordrecht, marec 1964. 178 s. ISBN 978-94-009-5821-0.

[8] Helal, S., Wendong, L. a Bose, R. *Mobile Platforms and Development Environments: Synthesis Lectures on Mobile and Pervasive Computing.* 1. vyd. Morgan and Claypool Publishers, február 2012. 120 s. ISBN 978-1-6084-5866-0.

[9] Lee, J. *Learning Unreal Engine Game Development: A step-by-step guide that paves the way for developing fantastic games with Unreal Engine 4.* 1. vyd. Packt Publishing, február 2016. 274 s. ISBN 978-1-7843-9815-6.

[10] Neuburg, M. *Programming iOS 14: Dive Depp into Views, View Controllers and Frameworks.* 1. vyd. O'Reilly Media, Inc., november 2020. 1256 s. ISBN 978-1-4920-9217-9.

[11] Panneton, F. a L'Ecuyer, P. On the Xorshift Random Number Generators. *ACM Trans. Model. Comput. Simul.* New York, NY, USA: Association for Computing Machinery. Október 2005, zv. 15, č. 4, s. 346–361. DOI: 10.1145/1113316.1113319. ISSN 1049-3301.

[12] Phillips, B., Stewart, C. a marsicano, K. *Android Programming: The Big Nerd Ranch Guide*. 3. vyd. Big Nerd Ranch Guides, marec 2017. 624 s. ISBN 978-0-1347-0605-4.

[13] Sarrab, M. *Mobile Learning (m-learning) Concepts, Characteristics, Methods, Components: Platforms and Frameworks*. 1. vyd. Nova Publishers, február 2015. 137 s. ISBN 978-1-6346-3252-2.

[14] Schell, R., Lee, V. a Schneider, H. *Mobile Applications: Architecture, Design, and Development*. 1. vyd. Prentice Hall, apríl 2004. 362 s. ISBN 978-0-1311-7263-0.

[15] Stone, D., Minocha, S., Woodroffe, M. a Jarret, C. *User Interface Design and Evaluation*. 1. vyd. Elsevier, apríl 2005. 704 s. ISBN 978-0-0805-2032-2.

[16] Takoordyal, K. *Beginning Unity Android Game Development: From Beginner to Pro*. 1. vyd. Apress, jún 2020. 270 s. ISBN 978-1-4842-6002-9.

[17] Tian, X. a Benkrid, K. Mersenne Twister Random Number Generation on FPGA, CPU and GPU. In: *AHS '09: Proceedings of the 2009 NASA/ESA Conference on Adaptive Hardware and Systems*. USA: IEEE Computer Society, 2009, s. 460–464. AHS '09. DOI: 10.1109/AHS.2009.11. ISBN 9780769537146. Dostupné z: https://doi.org/10.1109/AHS.2009.11.

Príloha A

Obsah priloženého média

Priložené médium sa skladá z nasledujúcej adresárovej štruktúry:

- `src` – zdrojové súbory mobilnej aplikácie Pejsek Záchranář
- `src_text` – zdrojové súbory textovej časti práce
- `executable` – obsahuje vyexportovaný súbor pre systém Android pripravený pre vloženie do mobilného zariadenia
- `zzsjmk` – kvízové otázky a odpovede poskytnuté Zdravotníckou záchrannou službou Juhomoravského kraju

Ďalej adresár obsahuje video, ktoré demonštruje priebeh aplikáciou Pejsek Záchranář, súbor readme, ktorý obsahuje spôsob akým spustiť projekt vo frameworku Unity, PDF súbor s originálnou kvalitou obrázkov a súbor s licenciami na obrázky, ktoré boli použité v rámci aplikácie.