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

ROZPOZNÁVÁNÍ OBRAZU NA MOBILNÍM TELEFONU PRO USNADNĚNÍ HRANÍ DESKOVÝCH HER

IMAGE RECOGNITION ON MOBILE PHONE TO FACILITATE PLAYING BOARD GAMES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MATEJ TUREK**
AUTHOR

VEDOUCÍ PRÁCE **doc. RNDr. PAVEL SMRŽ, Ph.D.**
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: **Turek Matej**
Program: Informační technologie
Název: **Rozpoznávání obrazu na mobilním telefonu pro usnadnění hraní deskových her**
Image Recognition on Mobile Phone to Facilitate Playing Board Games
Kategorie: Počítačová grafika

Zadání:

1. Seznamte se s metodami pro zpracování a rozpoznávání obrazu, vhodných pro aplikaci na mobilních telefonech, a dostupnými knihovnami, které takovou činnost usnadňují.
2. Analyzujte potřeby hráčů deskových her s cílem určení oblastí, u nichž by přenesení konkrétních činností na obrazovky mobilních telefonů, případně i podpora hraní, přinesly nejvíc.
3. Navrhněte a implementujte systém pro usnadnění hraní konkrétní hry (případně více her) pomocí rozpoznávání a zpracování na mobilním telefonu.
4. Vyhodnoťte realizované řešení v adekvátní uživatelské studii v reálném prostředí při hraní zvolené hry.
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- dle doporučení vedoucího

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

Vedoucí práce: **Smrž Pavel, doc. RNDr., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Cieľom tejto práce je vytvorenie aplikácie pre kartovú hru BANG!, ktorá uľahčí hranie tejto hry. Aplikácia je vytvorená pre operačný systém Android a využíva knižnicu OpenCV. Táto aplikáciu umožňuje rozpoznať jednotlivé karty hry a zobraziť o nich pravidlá, popis, a možné ťahy, pričom všetko toto prispieva k ľahšiemu pochopeniu hrania hry.

Abstract

The aim of this thesis is to create application for card game BANG! that helps play this game. Application is created for operating system Android and uses library OpenCV. This application allows to recognize each card and shows useful information such as rules, descriptions, possible moves, which helps to understand the game.

Kľúčové slová

Android, smartfón, kamera, kartové hry, spoločenské hry, rozpoznávanie obrazu, OpenCV, neurónové siete

Keywords

Android, smartphone, camera, card games, board games, image recognition, OpenCV, neural network

Citácia

TUREK, Matej. *Rozpoznávání obrazu na mobilním telefonu pro usnadnění hraní deskových her*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Pavel Smrž, Ph.D.

Rozpoznávání obrazu na mobilním telefonu pro usnadnění hraní deskových her

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána doc. RNDr. Pavela Smrža Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Matej Turek
5. mája 2021

Poďakovanie

Chcel by som úprimne poďakovať vedúcemu mojej práce pánovi doc. RNDr. Pavelovi Smržovi, Ph.D. za kozultácie a pomoc pri akýchkoľvek problémov, ktoré nastali.

Obsah

Kapitola 1

Úvod

Cieľom tejto práce je navrhnúť spôsob ako uľahčiť hranie kartovej hry hráčom podľa ich potreby a následne podľa toho vytvoriť aplikáciu na operačnom systéme Android, ktorá by tento spôsob uľahčovala. To zahŕňa rozpoznávanie kariet. Rozpoznávanie kariet je umožnené vďaka knižnice OpenCV, ktorá skúma vstup z kamery mobilného zariadenia a následne tento obraz spracováva. K projektu sú priložené obrázky kariet, ktoré slúžia ako predloha na porovnávanie. Pre referenčnú hru na spracovanie je využitá kartová hra BANG![3]. Cieľom však je, aby aplikácia bola univerzálna a rozšíriteľná, teda, aby sa dala využiť aj na iné kartové hry.

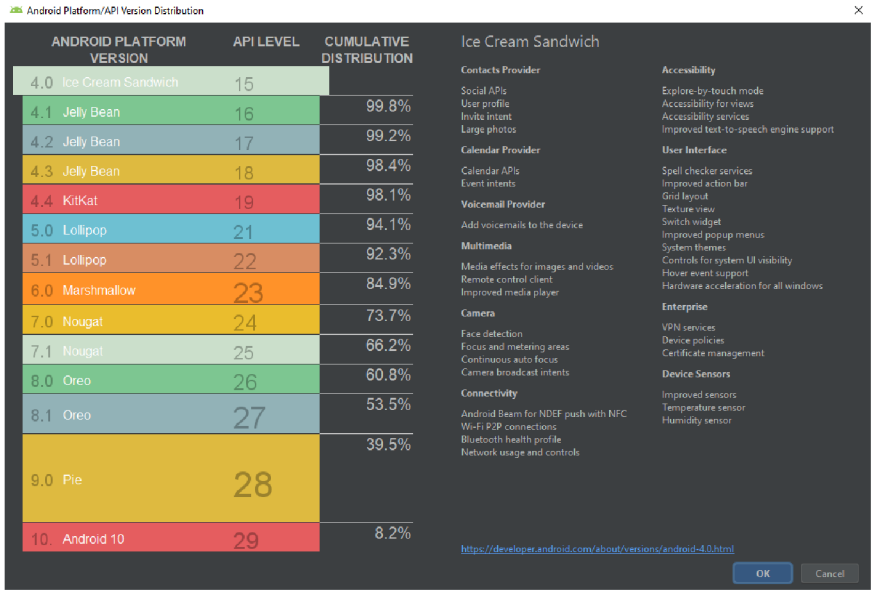
Kapitola 2

Android a spoločenské hry

Operačný systém Android [15] je open source projekt, ktorý je určený predovšetkým na mobilné zariadenia s dotykovým displejom. Obsahuje operačný systém (založený na jadre Linux), middleware, používateľské rozhranie a aplikácie. Vyvíja ho konzorcium Open Handset Alliance, ktorého cieľom je progresívny rozvoj mobilných technológií, ktoré budú mať výrazne nižšie náklady na vývoj a distribúciu a zároveň spotrebiteľom prinesú inovatívne používateľské prívetivé prostredie. Jadro operačného systému Android je navrhnuté pre prácu na rôznom hardvéri. Systém tak môže byť použitý bez ohľadu na chipset, rozlíšenie či veľkosť obrazovky.

2.1 Kompatibilita

Existuje viacero verzií OS Androidu, preto nastáva otázka minimálnej kompatibility pre tento projekt.



Obr. 2.1: Percentuálne rozloženie Android zariadení

Z tabuľky na obrázku 2.1 je priamo vidieť, že Android verzia 4.2 je podporovaná viac ako 99 percentami celkových používateľov operačného systému Android. Vo všeobecnosti platí, že čím väčšia verzia operačného systému Android je, tým aj samotné zariadenie obsahuje lepšiu hardvérovú špecifikáciu. Tá je v tomto projekte veľmi dôležitá, pretože výpočty môžu byť hardvérovo náročné. Dôležitou súčasťou však je OpenCV knižnica [1]. K jej použitiu je potrebná minimálna verzia Android 4.4, a teda SDK verzia 19. Tá je podporovaná viac ako 98 percentami celkovým počtom zariadení Android. Preto je táto verzia vybraná pre tento projekt ako minimálna verzia.

2.2 Vývojové prostredia

Existuje viacero možností výberu vývojového prostredia pre programovanie v operačnom systéme Android. Najpoužívanejším vývojovým prostredím je Android Studio[2] od spoločnosti Google. Je postavené na JetBrains. Poskytuje možnosť emulátoru, čo je vlastne virtuálne zariadenie, na ktorom sa vyvíja projekt. Tým odpadá nutnosť vlastniť a vyvíjať na fyzickom zariadení.

Ďalším vývojovým prostredím je Visual Studio[8], ktoré spolu s platformou Xamarin umožňuje vytvárať aplikácie kompatibilné pre Android, iOS a Windows. Taktiež poskytuje možnosti emulátoru.

IntelliJ IDEA[7] je taktiež vývojovým prostredím, ktoré umožňuje vývoj aplikácii na operačnom systéme Android. Je primárne určené na programovanie v jazyku Java. Umožňuje však vytvárať aplikácie pre Android, iOS a Windows.

2.3 Programovacie jazyky

Android Studio umožňuje programovať v jazykoch Java a Kotlin. Je však možné využiť Android NDK, ktoré umožňuje programovať v C, C++. Ostatné vývojové prostredia podporujú väčšinou Java, C# , C, C++.

2.4 Štruktúra kódu

2.4.1 AndroidManifest

AndroidManifest je konfiguračný súbor definovaný vo formáte XML, ktorý obsahuje nastavenia aplikácie. Obsahuje definície nasledovných funkcií:

- Názov balíka (namespace) - Android prekladač používa názov na zistenie umiestnenia entít pre preklad. Je využívaný ako jedinečný identifikátor danej aplikácie.
- Komponenty - sú to všetky aktivity, služby, broadcast receivery a content providery, ktoré majú definované základné vlastnosti ako meno Java triedy.
- Práva aplikácie - v prípade že aplikácia vyžaduje prácu s chránenými súčasťami operačného systému je užívateľ vyzvaný, aby tieto práva aplikácii udelil.
- Hardwarové a softwarové požiadavky - popisujúce, ktoré vlastnosti aplikácia vyžaduje, aby bolo jasné, ktoré zariadenie môže aplikáciu nainštalovať.

2.4.2 Služba

Služba je komponenta bežiaca na pozadí asynchrónne. Je využívaná na časovo dlhšie operácie.

2.4.3 BroadcastReceiver

BroadcastReceiver je komponenta, ktorá počúva a vysiela na pozadí, je schopná reagovať na rôzne udalosti v systéme.

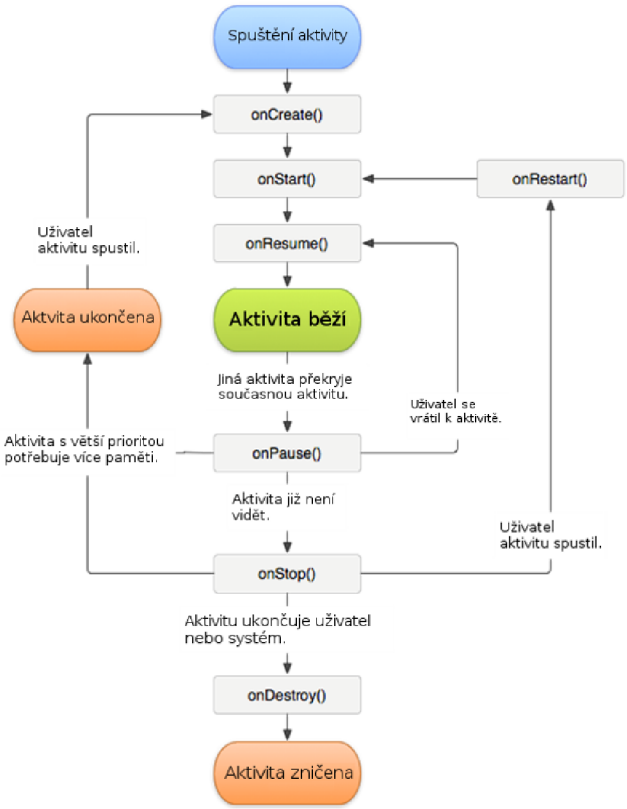
2.4.4 ContentProvider

ContentProvider je komponenta umožňujúca komunikáciu medzi rôznymi procesmi a aplikáciami.

2.4.5 Aktivita

Aktivita predstavuje najnákladnejšiu časť programu. Aktivita prepojuje užívateľské rozhranie so samotnou funkcionalitou aplikácie. Typicky je naraz spustených viacero aktivít, pričom len jedna z nich je na popredí. Aktivity môžu medzi sebou komunikovať a vymieňať si dáta.

Nasledujúci obrázok 2.2 predstavuje životný cyklus aktivity.



Obr. 2.2: Životný cyklus aktivity

Každá aktivita sa nachádza súčasne len v jednom z nasledujúcich stavov.

- Beží - Aktivita sa úspešne spustila a beží na popredí, je teda pre užívateľov viditeľná.
- Pauza - Aktivita je viditeľná, avšak je ale napríklad prekrytá inou aktivitou. (Upozornenie na prichádzajúci SMS, hovor, alebo napríklad dialóg o plnom nabití batérie pri nabíjaní). Užívateľ sa k takejto aktivite nijako nedostane, nemôže s ňou pracovať.
- Zastavená - Aktivita nie viditeľná, používateľ k nej nemá prístup, ale jej objekt ešte nebol úplne zničený. Užívateľ sa k nej bude môcť vrátiť, ak nebude zničená (napríklad nedostatkom pamäte).
- Ukončená - Aktivita je úplne mŕtva, používateľ sa k nej nemôže vrátiť.

Metódy Lifecycle

- OnCreate() - Pri spustení aktivity sa Android postará o základné veci ako vytvorenie objektu a spustenie procesu. Ďalej zavolá metódu onCreate(). V nej nadefinujeme všetko potrebné, aby sa aktivita mohla rozbehnúť, napríklad aké grafické rozhranie sa nám má zobraziť, ako sa dané grafické rozhranie bude zobrazovať (či má byť zobrazený app bar atď.), či sa majú vytvoriť globálne premenné, alebo či budeme globálne pristupovať k objektu TextView atď.
- OnStart() - Volá sa, ak aktivita bola prvýkrát spustená (po onCreate()) alebo bola aktivovaná po svojom skrytie (prichádzajúce SMS, systémový dialóg napríklad o nabitie batérie alebo iný dialóg iné aktivity), nemôže dostať užívateľský vstup.
- OnResume() - Volá sa tesne pred tým, než je aktivita posunutá do popredia (reštart, prvé spustenie alebo odpauzování), môže dostať užívateľský vstup.
- OnPause() - Volá sa pred prechodom aktivity na pozadí. Systém dostáva právomoc násilného ukončenia aktivity.
- OnStop() - Volá sa, keď sa má aktivita zastaviť, nie je viditeľná pre používateľov.
- OnDestroy() - Volá sa pred zrušením aktivity.
- OnRestart() - Ako vyplýva z predchádzajúceho diagramu, ak bola zavolaná onStop() a aktivita sa reštartuje, volá sa onRestart(), ktorý sa vykoná pred onStart().

2.4.6 Užívateľské rozhranie

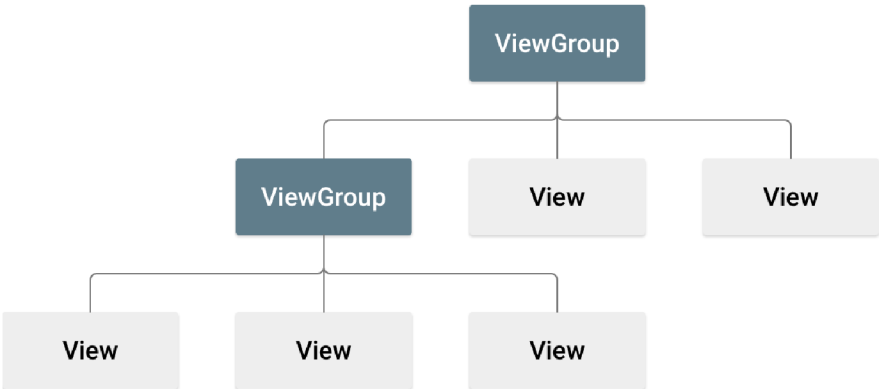
Užívateľské rozhranie je možné navrhnúť dvoma spôsobmi:

- Staticky - Definovanie XML súboru a následne pridanie do aktivity,
- Dynamicky - Vytváranie jednotlivých prvkov za behu aplikácie.

Pri obidvoch spôsoboch sa však mapovanie eventov na udalosti z užívateľského prostredia definuje v aktivite. V tomto projekte je využité statické vytváranie užívateľského rozhrania. Dôvodom je, že výsledok užívateľského rozhrania je vidieť už v samotnom dizajnéri vývojového prostredia a to aj z dizajnového hľadiska, ale aj pomáha navrhnúť lepšie rozhranie pre zariadenia s rôznymi rozlíšeniami displejov.

Štruktúra užívateľského rozhrania

Užívateľské rozhranie je definované pomocou objektov View a ViewGroup. View zvyčajne vykresľuje prvok, s ktorý užívateľ vidí a je schopný interakcie. Na rozdiel od toho ViewGroup je neviditeľný kontajner, ktorý definuje štruktúry layoutov pre View a ďalšie objekty ViewGroup. Obrázok 2.3 definuje hierarchická štruktúru definujúcu UI.



Obr. 2.3: Hierarchická štruktúra definujúca UI

Objekty View sa zvyčajne nazývajú "widgety" a môžu to byť rôzne podtriedy ako sú napríklad tlačidlo Button, textové pole TextView, atď. Objekty ViewGroup sa nazývajú "layouty" a medzi základné používané typy layoutov patria:

- Linear layout,
- Relative layout,
- Constraint layout.

2.4.7 Ukladací priestor

Existuje viacero spôsobov pre uloženie dát:

- Databáza - Android podporuje SQLite [5], ktorej knižnice sú priamo podporované.
- Zdieľané preferencie (shared preferences) - Poskytujú spôsob uloženia nastavenia dát aplikácie formou kľúč - hodnota.
- Externé a interné úložisko - Prestravuje spôsob uloženia dát priamo v pamäti telefónu.

2.4.8 SQLite

Android podporuje databázu typu SQLite. Na rozdiel od ostatných databáz založených na princípe klient-server, kde je databázový server spustený ako samostatný proces, je SQLite iba knižnica, ktorá je po prelinkovaní k aplikácii k dispozícii pomocou jednoduchého rozhrania. Každá databáza je uložená v samostatnom súbore .dbm (Database Manager), kde sa dáta ukladajú za použitia jednoduchého primárneho kľúča do rovnako veľkých blokov a využívajú sa hašovacie techniky pre rýchly prístup k dátam pri vyhľadávaní podľa kľúča. Medzi hlavné výhody SQLite patria:

- Jednoduchosť - je to odľahčená databáza, použiteľná na rôznych zariadeniach ako sú televízie, smartfóny, kamery, domáce zariadenia, atď.
- Výkon - ponúka veľmi rýchle read/write operácie, ktoré sú približne o 35% rýchlejšie ako operácie klasického file systému.
- Nevyžaduje dodatočnú inštaláciu, stačí si len stiahnuť knižnice.
- Spoľahlivosť - oproti I/O svoj zápis vykonáva pravidelne a je odolnejšia voči chybám.
- Prenosnosť - je prenosná medzi 32 a 64 bitovými zariadeniami nezávislé na endianite.
- Dostupnosť - existuje množstvo softvéru tretích strán, ktorými je pristupovaná.
- Komplexnosť a cena - poskytuje prostriedky na aktualizáciu seba a svojej štruktúry.

Medzi nevýhody patria:

- Kapacita obmedzená na 2GB
- Problémy s právami a zamykaní súborov pri zápise.

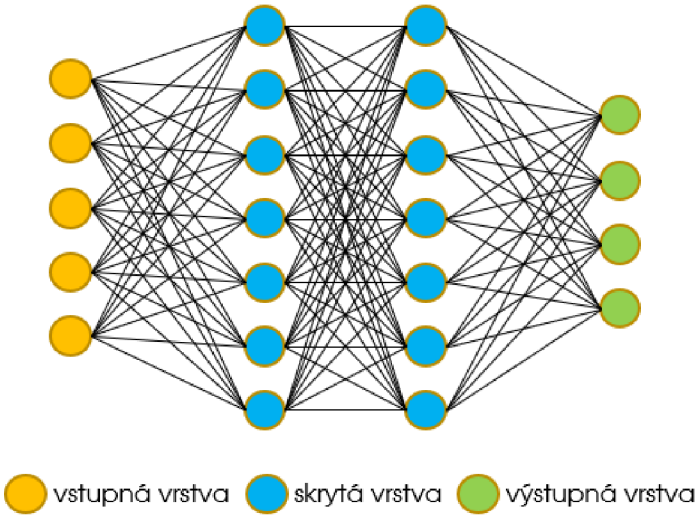
2.4.9 Gradle

Gradle je open source univerzálny nástroj na preloženie takmer každého typu programu. Má oficiálnu podporu pre jazyky Java a Kotlin, preto sa využíva aj pri preklade programov Android. Gradle skripty definujú (minimálnu) SDK kompatibilitu projektu, priložené knižnice. Pri zmene akejkoľvek v tomto súbore je potrebné tieto skripty zosynchronizovať a preložiť celý projekt.

2.4.10 Neurónové siete

Neurónové siete sú v súčasnosti považované za jeden z najlepších algoritmov strojového učenia. Vyznačujú sa veľmi vysokou presnosťou a majú pomerne širokú škálu použitia. V závislosti od použitej architektúry ich niekedy možno označiť aj ako hlboké strojové učenie. Fungujú na princípe vytvorenia modelu, ktorý následne natrénujeme na čo najväčšom množstve označených dát. Na základe týchto známych dát sa tak model naučí „predpovedať“ výsledky nových neznámych prípadov.

Architektúra neurónovej siete sa skladá z konečného počtu uzlov – neurónov. Niektoré z nich sú navzájom poprepájané, pričom prepojenia medzi neurónmi majú svoje ohodnotenia, označované ako váhy. Tieto neuróny sú potom usporiadané do vrstiev. Spravidla jednej vstupnej vrstvy, z niekoľkých skrytých vrstiev a z jednej výstupnej vrstvy. To, aké a koľko vrstiev je použitých, koľko neurónov obsahujú jednotlivé vrstvy a ako sú vrstvy poprepájané, nazývame architektúra siete. Počet skrytých vrstiev a počty neurónov v jednotlivých vrstvách označujeme ako hyperparametre siete. Príklad poprednej neurónovej siete s dvomi skrytými vrstvami je na nasledujúcom obrázku 2.4:



Obr. 2.4: Architektúra neurónových sietí

2.4.11 Spracovanie obrazu

Pre rozoznávanie obrazov je využitá knižnica OpenCV[4]. Knižnica je pridaná do projektu formou modulu. OpenCV [6] je bezplatná knižnica strojového videnia pre rôzne platformy (existujúce verzie pre GNU / Linux, Mac OS X, Windows a Android), ktorý bol pôvodne vyvinutý spoločnosťou Intel a používaný v nespočetných aplikáciách, od bezpečnostných systémov s detekciou pohybu až po aplikácie na riadenie procesov, kde sa vyžaduje rozpoznávanie objektov. Je to tak preto, lebo jeho uverejnenie je dané licenciou BSD, ktorá umožňuje jej voľné použitie na komerčné a výskumné účely za podmienok v nej uvedených. Jej základ tvorí využitie kamery, konkrétne spracovanie obrazu kamery bude rozoberané v ďalších kapitolách pri samotnej implementácii. Samotná knižnica je veľmi obsiahla a má široký záber využitia. Implementuje množstvo funkcií, ktoré sú v konečnom dôsledku využité v tomto projekte.

2.5 Spoločenské hry

Spoločenské hry sú hry, ktoré sa hrajú na nejakom hracom pláne (hracej doske). Po tomto hracom pláne sa podľa pravidiel pohybujú alebo ukladajú hracie figúrky (napr. Arimaa, DVONN). V niektorých doskových hrách sa figúrky nepohybujú, iba sa postupne kladú na hrací plán (napr. Go, Piškvorky, Scrabble).

Hracie plány (dosky), môžu byť univerzálne, spoločné pre viacero hier (napríklad šachovnica pre šach a dámu), alebo špeciálne pre jeden konkrétny druh hry (napríklad Monopoly, Človeče nehnevaj sa)

Mnohé doskové hry sú dostupné aj ako počítačové hry, v ktorých jedným z hráčov môže byť samotný počítač. Niektoré doskové hry je možné hrať aj online.

Môžeme ich rozdeliť nasledovne:

- **Kartové hry** sú hry, ktoré sa hrajú s hracími kartami.

- **Zberateľské kartové hry** je druh kartových hier, pri ktorých neexistuje pevne stanovený balíček kariet. Namiesto toho si každý hráč zostavuje vlastný balíček z dostupných kariet.
- **Doskové hry** sú hry, ktoré sa hrajú na nejakom hracom pláne (hracej doske). Po tomto hracom pláne sa podľa pravidiel pohybujú alebo ukladajú hracie figúrky . V niektorých doskových hrách sa figúrky nepohybujú, iba sa postupne kladú na hrací plán.
- **Hry na hrdinov** je druh hry, kde hráči hrajú rolu fiktívnych postáv (role-playing – hranie rol), ktoré si podľa daných pravidiel vytvoria a za ktoré v samotnej hre riadia.

Tento projekt sa zaoberá predovšetkým na kartové hry, ale je vzhľadom k univerzálnosti by bol možný využiť aj na doskové hry. Kritérium však je, aby hry obsahovali karty s rôznymi úlohami.

Kapitola 3

Návrh pri spracovaní hry Bang!

3.1 Pravidlá hry Bang!

Hra BANG! je karová hra westernového štýlu. Každý hráč má priradenú určitú rolu -

- Šerif musí zabiť všetkých banditov a odpadlíka.
- Banditi musia zabiť šerifa skorej než on zabije ich.
- Pomocníci pomáhajú šerifovi a vyhrajú, pokiaľ vyhrá šerif.
- Odpadlík sa chce stať novým šerifom - musí zostať ako posledný hráč, takže musí zabiť banditov a pomocníkov a nakoniec aj šerifa. Hra je určená pre 4 až 7 hráčov.

Hra obsahuje 103 kariet:

- 7 kariet rol
- 16 kariet postáv
- 80 herných kariet
- 7 kariet prehľadov pravidiel

Hráč má 1 kartu role, 2 karty postáv (pritom zvolenú práve len jednu a druhá slúži na ukazovanie života) a 0 až N herných kariet v ruke.

Hra je určená pre 4 až 7 hráčov, pričom kompozícia rol je nasledovná:

- 4 hráči: 1 Šerif, 1 Odpadlík, 2 Banditi
- 5 hráči: 1 Šerif, 1 Odpadlík, 2 Banditi, 1 Pomocník šerifa
- 6 hráči: 1 Šerif, 1 Odpadlík, 3 Banditi, 1 Pomocník šerifa
- 7 hráči: 1 Šerif, 1 Odpadlík, 3 Banditi, 2 Pomocníci šerifa

Hra sa hrá v smere hodinových ručičiek a začína Šerif. Samotný ťah každého hráča pozostáva z 3 krokov:

1. Potiahnutie si dvoch kariet - každý hráč na ťahu si ťahá 2 karty, pokiaľ nie sú v balíčku už žiadne, zamieša sa odpadový a tým sa vytvorí nový hrací balíček.

2. Zahranie ľubovoľného počtu kariet - hráč môže zahrať len jednu kartu BANG! (pokiaľ nemá výnimku karty pištole alebo postavy) a nemôže mať vyložené pred sebou 2 rovnaké karty s modrým okrajom.

3. Zahodenie nadbytočných kariet - hráč nesmie mať na konci svojho ťahu viac kariet ako je počet jeho životov. Nadbytočné karty odhadzuje do odpadového balíčka.

Koniec hry nastáva pokiaľ je šerif mŕtvy alebo pokiaľ sú mŕtvi banditi a aj odpadlík.

Obr. 3.1: Kartová hra Bang!

3.2 Potreby hráčov

Pri skúmaní potreby hráčov pri hraní hry BANG! sa vychádzalo zo skutočnosti, že spočiatku je hra pre nováčikov zložitá. Medzi najčastejšie problémy nových hráčov patrí nevedomosť hracích kariet. Hracie karty ovplyvňujú karty postáv a modré hracie karty rozširujú (komplikujú) hru. Pýtať sa ostatných hráčov znamená odhaliť akú kartu má, čo je nežiadúce. Samotné rozpoznávanie kariet tejto aplikácie má cieľ uľahčiť pochopenie jednotlivých kariet z balíčka, pričom ponúka odkazy na ďalšie karty, ktoré súvisia s práve vybranou kartou. Napríklad karta “BANG“ úzko súvisí s kartou “Vedľa“. Preto je potrebné pri rozpoznaní jednej karty uviesť aj odkaz na tú druhú, pretože môže nastať situácia, kedy hráč jednu kartu má v ruke, ale tú druhú nie. Pokiaľ by sa chcel pozrieť ako karta vyzerá, stačí mu získať si túto jednu.

Klasickým problémom je aj skutočnosť, že len jeden hráč vlastní túto hru a teda aj k samotným pravidlám a ďalší minimálne traja nemajú prístup k týmto pravidlám. Aplikácia taktiež ponúka prehľad všetkých kariet hry a jej pravidlá. Tým pádom umožňuje naštudovať si hru už pred samotným začiatkom.

Preto aplikácia musí umožňovať rozpoznávať karty, ale aj umožňovať prechádzať si karty v bez rozpoznávania. Ku každej karte a pri samotnej hre musia byť údaje o všetkých dôležitých náležitostiach.

3.3 Rozpoznávanie kariet

Pre rozoznávanie je potrebné určiť vzorové karty, ktoré budú slúžiť ako referenčné karty pre porovnávanie s aktuálne snímaným obrazom. Referenčné karty boli získané zo samotného balíčka odfotením a ďalším spracovaním. Následne boli uložené medzi do “assets“ aplikácie. Karty bang sú špecifické tým, že každá karta v ľavom dolnom rohu obsahuje identifikátor karty (viz. obr. 3.2).

Obr. 3.2: Karta Bang!

Tým pádom je možnosť kartu orezať tak, aby obsahovala len identifikátor karty v ľavom dolnom rohu. Avšak toto sa pri samotnej implementácii ukázalo ako neefektívny spôsob. Nastával problém, že orezaním sa získa veľmi malý priestor na rozpoznanie a výsledok nebol dostačujúci.

3.4 Návrh algoritmu

Pri návrhu algoritmu sa uvažovalo o viacerých spôsoboch:

- Tesseract,
- Porovnávanie histogramov,
- Porovnávanie pomocou predlôh,
- Porovnávanie pomocou vlastností obrazu.

3.4.1 Tesseract

Tesseract [14] je OCR (Optional character recognition) engine [13], ktorý dokáže rozoznať viac ako 100 jazykov a dá sa naučiť ďalšie. V prípade hry BANG! by sa jednalo o font "Perdido" pre nadpisy a "Palatino Linotype" pre popis kariet.

Tvoje vzdálenost od všech hráčů je větší o 1

Obr. 3.3: Príklady písma hry BANG!

Tento návrh by síce fungoval, avšak nemusel by byť dostatočne univerzálny pre prípadné rozšírenia alebo v prípade inej hry. Problémom by bolo a zvolené písmo, pretože každá hra používa iné.

3.4.2 Histogramy

Porovnávanie histogramov[10] je veľmi jednoduchá a rýchla metóda. Základná myšlienka je algoritmu pozostáva v porovnávaní farieb. Napríklad lesy obsahujú veľa zelenej farby, tvár veľa ružovej, atď. Pokiaľ teda porovnávame 2 lesy z histogramu zistíme, že obsahujú veľa podobností, pretože obidva obsahujú veľa zelenej. Nevýhoda však je, že je až príliš jednoduchý - napríklad žltý banán sa bude podobať na pláž, pretože obidve sú žlté.

OpenCV využíva metódu compareHist().

Využíva sa metrika "Chi-Square". Uvažujme histogram H1 a histogram H2. Metrika d(H1, H2) je definovaná nasledovne:

$$d(H_1, H_2) = \sum_I \frac{H_1(I) - H_2(I)}{H_1(I)} \tag{3.1}$$

3.4.3 Predlohy

Porovnávanie predlôh[11] je lepší algoritmus, ktorý priamo porovnáva obrázky pomocou matíc, ktorých hodnoty indikujú pravdepodobnosť, že objekt je v strede daného pixela. Nevýhodou však je, že vracia výsledky pokiaľ sú objekty identické, rovnakej veľkosti a orientácie.

OpenCV využíva metódu matchTemplate().

3.4.4 Vlastnosti obrazu

Považuje sa za najefektívnejší algoritmus pri vyhľadávaní obrazu. Z obrazu sa extrahujú vlastnosti, ktoré garantujú, že rovnaké vlastnosti sa vyberú aj z druhého porovnávaného obrazu (predlohy) aj v prípade rotácie alebo iného rozlíšenia. Nevýhodou tohoto algoritmu je, že je pomalý.

Konkrétne sa využíva algoritmus Oriented FAST and rotated BRIEF (ORB) recognition. Ten vychádza rýchlej (FAST) detekcii kľúčových bodov (keypointov) a krátkom,

stručnom (BRIEF) popise. Keypointy sú to priestorové umiestnenia alebo body v obraze, ktoré definujú, čo je zaujímavé alebo čo vyniká na obraze. Detekcia záujmových bodov je v skutočnosti podmnožinou detekcie blobov, ktorej cieľom je nájsť na obrázku zaujímavé oblasti alebo priestorové oblasti.

V prvom kroku používa FAST na nájdenie kľúčových bodov, potom použije Harrisovu mierku na nájdenie najlepších N bodov medzi nimi. Taktiež používa pyramídu na vytváranie funkcií s viacerými mierkami. Akshika Wijesundara[16] navrhol aplikáciu na spájanie keypointov:

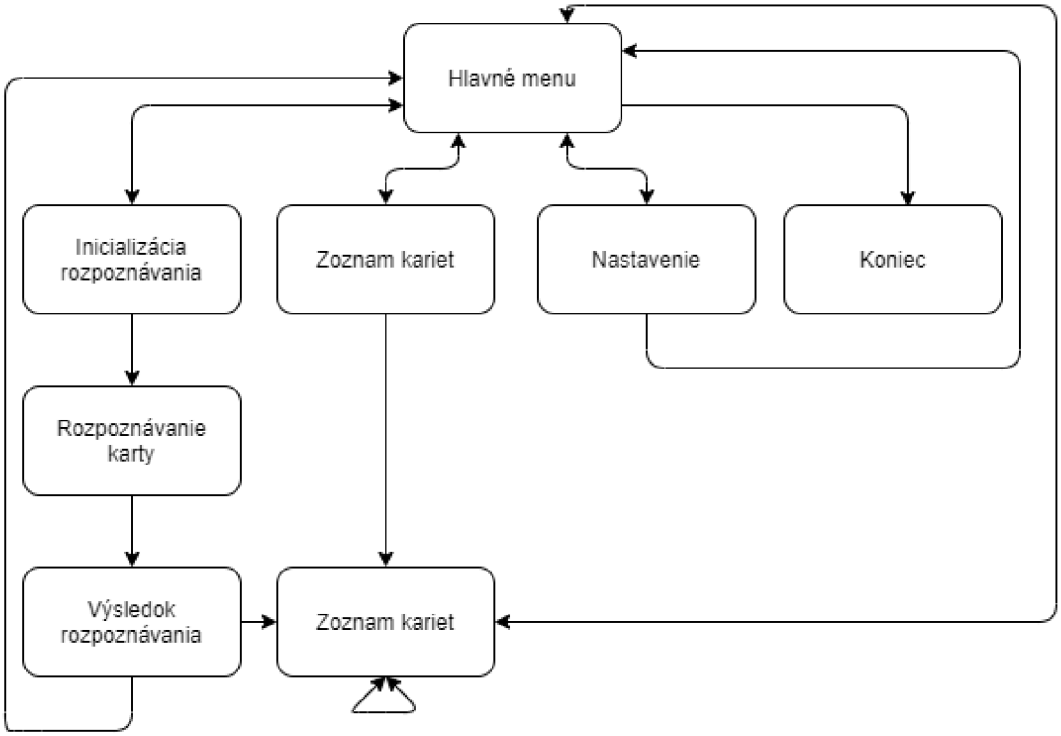
Ten sa ukázal byť ako efektívny návrh, avšak bol rozšírený o porovnávanie histogramov, čo sa ukázalo byť ako lepší návrh a výpočtovo rýchlejší.

3.5 Užívateľské rozhranie

Súčasťou návrhu užívateľského rozhrania sú tieto prvky:

- definovanie požadovaných funkcií pre jednotlivé komponenty,
- analýza užívateľskej skupiny,
- návrh grafického dizajnu užívateľského rozhrania,
- znovu použiteľnosť aplikácie,
- rozloženie prvkov aplikácie,
- tvorba modelu resp. prototypu aplikácie,
- užívateľské testovanie.

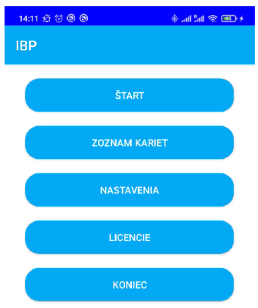
Užívateľské rozhranie je dostupné v troch jazykoch - angličtina, čeština a slovenčina. Prispôsobuje sa automaticky podľa regiónu v smartfóne. V aplikácii sú preklady uložené formou XML pre každý jazyk a v layoutoch sú texty uložené len formou odkazov na tieto preklady. Užívateľské rozhranie je definované pomocou XML layoutov uložených v res/layout. Pre každú aktivitu je vytvorený jeden layout s preddefinovanými rozloženia elementov, ktorý sa ďalej v aktivite použije. Týmto spôsobom bolo zabezpečené správne rozloženie užívateľského rozhrania a to aj naprieč rôznymi zariadeniami s rôznymi rozlíšeniami. Ďalej sa v zložke res nachádzajú aj ikonky (drawable) vygenerované v Android Studiu, animácie pri prechode aktivít (anim), spomínané preklady a farby (values) aplikácie. Pre intuitívne ovládanie aplikácie bolo užívateľské rozhranie navrhnuté takto:



Obr. 3.4: Diagram užívateľského rozhrania

3.5.1 Hlavné menu

Hlavné menu poskytuje základnú správu aktivít a umožňuje sa intuitívne pohybovať v aplikácii. Je prvým oknom pri spustení aplikácie a inicializuje aplikáciu.

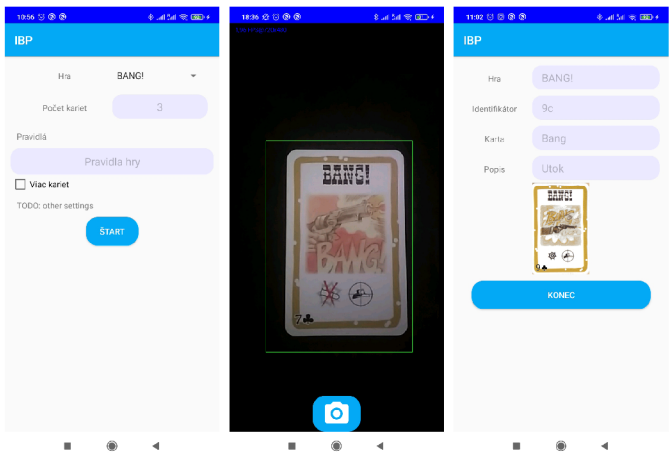


Obr. 3.5: Hlavné menu

3.5.2 Rozpoznávanie kariet

Zahŕňa 3 kroky:

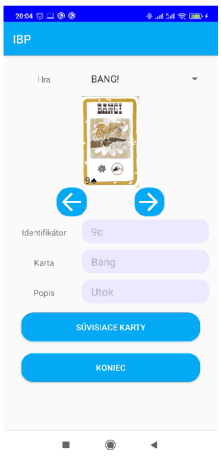
- Začína najprv inicializáciou, teda vybranie o akú hru sa jedná, o množstvo kariet (1 alebo viacej), popis (pravidlá hry).
- Pokračuje v samotnom rozpoznávaní za využitia kamery.
- Posledný krok je aktivita, ktorá zobrazuje výsledok rozpoznávacieho algoritmu.



Obr. 3.6: Inicializácia, rozpoznávanie kariet, výsledok

3.5.3 Zoznam kariet

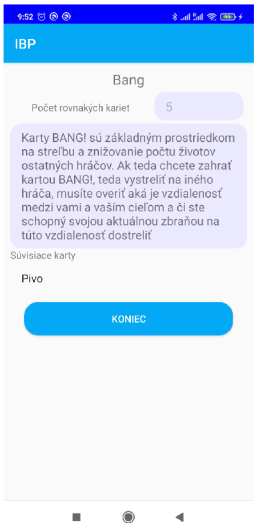
Jednoduchý zoznam hier a kariet, kde sa zobrazujú základné informácie o daných položkách. Tieto informácie sa zobrazujú podľa stĺpcov databáze, čím sa ponúka možnosť na upravovanie daných kariet a ich vlastností.



Obr. 3.7: Zoznam hier a kariet

3.5.4 Využitie karty

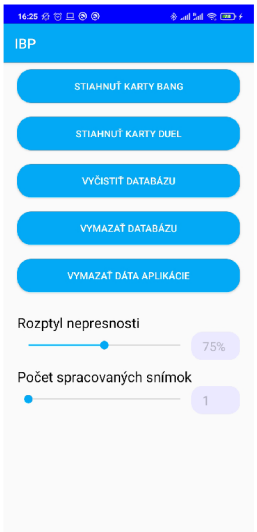
Využitie karty obsahuje detailné informácie o konkrétnej karte, jej využitie a odkazy na karty, ktoré s touto kartou súvisia.



Obr. 3.8: Použitie karty

3.5.5 Nastavania

Obsahuje nastavenia pre správu databázy SQLite, kde poskytuje možnosti naplniť databázu a vymazanie aktuálnej. Taktiež poskytuje možnosť vymazania dáta aplikácie (napríklad zdieľané preferencie). Poskytuje aj možnosť upravovať vlastnosti algoritmu, ktoré budú popísané v ďalších kapitolách.



Obr. 3.9: Nastavenia

3.5.6 Kalibrácia

V pôvodnom návrhu bola súčasťou aplikácie kalibrácia, ku ktorej sa pristupovali z menu. Účel toho bol prispôsobovať (kalibrovať) počet kľúčových bodov pre každú kartu. Veľmi sa to podobalo samotnému rozpoznávaniu kariet, kde sa však vopred vybrala karta, ktorú užívateľ chcel kalibrovať a následne pomocou kamery sa zmeral počet kľúčových bodov a táto hodnota sa uložila do databázy. Dôvodom odstránenia tejto funkcie je, že algoritmus je postavený na základe toho, že by mal nájsť kartu, ktorá obsahuje najviac kľúčových bodov medzi kartami kamery a predlohovými kartami. Vzhľadom k tomu, že hľadáme maximum, nie je potrebné zisťovať a porovnávať túto hodnotu.

3.6 Licencie

Vzhľadom k tomu, že OpenCV spadá pod Apache licenciu, je potrebné túto informáciu uviesť aj do programu. Je dostupná priamo z hlavného menu, kde sa po kliknutí zobrazí text s Apache licenciou.

3.7 Úložisko práce

Pre ukladanie zdrojového kódu práce bol vytvorený privátny repozitár na platforme GitHub s. K Android Studiu je nainštalovaný plugin, ktorý veľmi uľahčuje prenositeľnosť práce na rôznych zariadeniach. Navyše zaisťuje archiváciu zdrojových kódov v prípade straty - napríklad pokazený disk.

Kapitola 4

Implementácia programu

Ako sa spomínalo v prvej kapitole, pri programovaní bolo využité vývojové prostredie Android Studio a programovací jazyk Java. Základ pre rozpoznávanie obrazu tvorí knižnica OpenCV, ktorá je pridaná do projektu formou modulu. Táto knižnica berie ako vstup obraz z kamery a porovnáva každý snímok obrazovky.

Pri prvom spustení aplikácie je dôležité vykonať kontroly práv pre kameru. Bez týchto povolení nie je možné korektne pokračovať v správnom fungovaní aplikácie a jej funkcionalita je značne obmedzená. Bez nich neposkytuje možnosť rozpoznávať karty, avšak udržuje si svoju databázu kariet s ich všetkými popismi. Jediným povolením je povolenie na kameru, spolu s prednou kamerou a autofokusom, ktoré zabezpečuje základnú prácu s touto aplikáciou.

4.1 Spracovanie predlohy

Pôvodným cieľom bolo odfotiť predlohy pomocou fotoaparátu na smartfóne. Avšak OpenCV využíva vlastné knižnice pre kameru ako štandardná aplikácia v operačnom systéme Android. Navyše fotografie boli príliš tmavé a upravovať jas nemalo dostačujúci efekt ako to je vidieť na obrázku 4.1, kde v ľavej časti je originálna karta, v pravej časti je odfotená. Ako najvhodnejšie sa ukázalo skenovať karty na skeneri. Ten karty skvele nesvetlil a obrázky boli vo vysokej kvalite. Zostáva ich upraviť.

Obr. 4.1: Rozdiel v kamere a skeneri

Pre orezávanie obrázkov a prácu s nimi je implementovaná pomocná trieda ImageHelper. Táto trieda získava obrázky z miesta ich uloženia a prevádza ich do formy Bitmapy, ktorá je ďalej spracovávaná. V pôvodnom návrhu bolo súčasťou tejto triedy aj prevod Bitmapy do odtiene šedej, avšak toto je implementované v OpenCV ako funkcia, takže bola využívaná tá.

Obrázky sú uložené vo forme assets projekte, čo uľahčuje prístup k nim. Pre každú kartu je pridelený práve jeden obrázok a vytvorený záznam v SQLite databáze. Tieto záznamy je možné nahrať priamo v nastaveniach aplikácie. Obrázky sú uložené vo formáte JPG a každý obrázok zaberá približne 20KB a s pomerom strán približne 2:1.

Referenčné karty sú upravené tak, že sú orezané (hnedé a modré) okraje karty a odstránený identifikátor karty z ľavého spodného rohu. Okraje sú odrezané kvôli náročnosti a lepšiemu porovnávaniu. Keďže každá karta má tieto okraje rovnaké, je zbytočné ich zohľadňovať. Identifikátor karty je odstránený z dôvodu, že viaceré karty obsahovali rovnaký identifikátor. Napríklad existovala karta pivo, ktorá mala identifikátor 9 srdce a 9 srdce mala aj karta Mustang. Ďalším dôvodom na odstránenie identifikátoru je fakt, že existuje viacero rovnakých kariet, ktoré sa líšia len identifikátorom - napríklad kariet bang je v balíčku viacero a je zbytočné pre každú vytvárať záznam v databáze, keďže funkcia je rovnaká.

Predlohy kariet pri porovnávaní sú prevedené spôsobom uvedeným na obrázku 4.2:

Obr. 4.2: Orezanie kariet

4.2 Rotácia kamery

Vzhľadom k tomu, že OpenCV kamera neumožňuje kameru na výšku, ale iba na šírku, bolo potrebné si upraviť funkciu knižnice. V triede CameraBridgeViewBase.java bola upravená funkcia deliverAndDrawFrame() a pridaná funkcionalita ktorá prehadzuje šírku a výšku kamery. Toto bolo nutné upraviť priamo vo funkciách OpenCV, kvôli výpočtovej náročnosti. Pokiaľ by sa to upravovalo už v algoritme, znamenalo by to väčšiu náročnosť. Navyše pokiaľ by sa to upravovalo v algoritme, vrchná lišta systému Android by sa zobrazovala nesprávne na boku, čo by vizuálne mohlo vadiť. Následne mohla byť upravená, resp. zamknutá rotácia kamery priamo v AndroidManifest.xml, kde je definovaná formou aktivity.

Obr. 4.3: Rotácia kamery

4.3 Neuronové siete

OpenCV knižnica je vybavená funkciami na získanie blobov z obrázku, resp. framu z kamery. Ako vstup do funkcie je blob v štruktúre "Mat"(vstupný obrázok) a ako výsledok sú štyri body ohraničujúce rozpoznané objekty. Vďaka týmto bodom je možné vstupný obraz orezať tak, aby bola viditeľná len karta. Tieto štyri body sú zakreslené a na vstupnom obrázku je tak viditeľný obdĺžnik reprezentujúci vstup. Pre inicializáciu neurónových sietí[12] bolo potrebné načítať 2 konfiguračné súbory:

- MobileNetSSD_deploy.prototxt, ktorý reprezentuje samotný konfiguračný súbor.
- MobileNetSSD_deploy.caffemodel, reprezentujúci váhy konfiguračného súboru.

4.4 Algoritmus porovnania

Celý algoritmus je implementovaný v samostatnej aktivite s názvom RecognizeActivity.java, kde sa pomocou štruktúry "Bundle" predáva aktivite identifikátor hry. Prvým krokom je inicializácia OpenCV. V prípade, že by táto knižnica chýbala, nastane chyba a funkcie nebudú dostupné. Taktiež sa inicializuje kamera a neurónové siete.

Najdôležitejšou súčasťou tejto aktivity je funkcia onCameraFrame(). Táto funkcia je volaná pri každom snímku kamery. Vstupom do funkcie je snímok z kamery vo forme objektu CvCameraViewFrame, a výstupom je obrázok vo forme objektu Mat. Tento obrázok je možné upraviť a vrátiť iný (orezaný, upravený alebo v ňom zakresliť body). V tejto funkcii je volaná hlavná časť programu pre každý snímok.

Pre uľahčenie je však zabezpečené, aby sa algoritmus vykonával až po vyvolaní z užívateľské vstupu. Preto je zavedený princíp, že po vyvolaní tlačidla aplikácia spracováva a vyhodnocuje snímok. V pôvodnom návrhu bol daný časový interval, po ktorý spracovával každý snímok. Táto implementácia bola nevhodná, vzhľadom k tomu, že hra môže obsahovať niekoľko desiatok až stoviek kariet a čas vyhodnotenia dramaticky stúpa, preto sa mohlo stať, že za daný interval sa nemuseli vyhodnotiť všetky karty, prípadne by sa niektoré vyhodnocovali viackrát a niektoré len raz. Preto to bolo nahradené spracovaním jedného snímku, s možným rozšírením na spracovanie viacerých snímkov. Vstupom pre samotný algoritmus (ktorý je teda volaný z funckie onCameraFrame()) sú teda 2 Bitmapy. Jedna je získaná z uložených predlôh, druhá je výsledkom výstupu kamery smartfónu.

V algoritme sa kvôli vstupu funkcií pre OpenCV sa pracuje so štruktúrami typu bitmapa a typu Mat. Ďalej sú prevedené na odtieň šedej, čím sú redukované rôzne svetelné odchýlky.

Tu sa využívajú 3 štruktúry:

- FeatureDetector - detekuje vlastnosti,
- DescriptorExtractor - vypočítava deskriptory,
- DescriptorMatcher - porovnáva deskriptory.

Pri porovnávaní deskriptorov je využitý algoritmus "BRUTEFORCE_HAMMING" [9]

Výsledkom je teda počet kľúčových bodov, ktoré sa našli medzi predlohou a aktuálnym obrazom kamery. V optimálnom prípade je výsledok s najväčším počtom kľúčových bodov správnym výsledkom rozpoznávania.

4.5 Optimalizácia algoritmu

Tento algoritmus je veľmi výpočtovo náročný. Má lineárnu zložitosť, pretože obraz je potrebné porovnať s každou kartou v databáze dostupných kariet danej hry. Bez optimalizácie bol algoritmus veľmi pomalý. Dalo sa to pozorovať na snímkach za sekundu (angl. FPS - frames per second) kamery.

Ako prvou optimalizáciou bola čo najväčšia inicializácia ešte pred spustením samotného algoritmu. To napríklad obsahovalo získanie si všetkých predlôh kariet ešte pred spustením samotného algoritmu pri inicializácii aktivity v metóde "onCreate", získanie ostatných dát.

Nasledujúcim zrýchlením bolo úpravou OpenCV funkcie pre rotáciu na výšku spomínanú vyššie v kapitole 4.2.

Ďalším zrýchlením algoritmu bolo prevedenie časti funkcionality do asynchrónnej funkcie. To sa ukázalo ako najefektívnejšia optimalizácia, avšak výrazne skomplikovala funkčnosť. Problémom boli nielen dlhé čakacie doby na pozadí aplikácie, ktoré spôsobovali to, že

niektoré výsledky neprišli v časovom intervale a boli vyhodnotené, resp. neboli vyhodnotené vôbec.

Posledným bolo ohraničenie hodnôt histogramu na adekvátnu hodnotu. Pokiaľ bola hodnota histogramu príliš veľká, znamenalo to, že karty sa príliš líšili a nepristupovalo sa k druhému kroku algoritmu. Táto hodnota bola získaná experimentálne. Táto optimalizácia zredukovala počet spracovaných kariet, avšak zvýšila počet vykonávaných funkcií a algoritmus sa spomalil. Preto bola odstránená.

4.6 Vyhodnocovanie

Pre vyhodnocovanie je pre všetky karty vytvorená pomocná štruktúra typu hašovacia mapa. V tejto mape sú uložené informácie typu kľúč - hodnota. Kľúč predstavu identifikátor karty a hodnota predstavuje počet kľúčových bodov. následne sa vyhľadáva karta s najväčším počtom kľúčových bodov podľa kariet. Tu však vznikajú rôzne nepresnosti týkajúce sa obrázkov. Aj keď boli karty zdigitalizované pomocou skenera, niektoré obrázky majú lepšiu kvalitu ako druhé. Preto je zavedená ďalšia premenná hodnota - koeficient karty (v štruktúre databázy je pomenovaný ako "multiplier"). Táto hodnota je konštanta, typická pre každú kartu iná. Je to desatinné číslo určujúce akou konštantou sa má počet kľúčových bodov vynásobiť.

Karta s najväčším počtom kľúčových bodov (vynásobená koeficientom) je výsledkom vyhľadávania.

Ďalej sa však uchovávajú aj ďalšie podobné karty, určené vzťahom:

$$K_i <= 0.75 * K_{max} \tag{4.1}$$

Tento koeficient je možné upraviť v nastaveniach, kde minimálna hodnota je 50 percent. To zabezpečuje, aby karty, ktoré sa podobajú neboli zahodené, ale aby sa tiež prikladali k výsledkom. Pre veľmi podobné karty je niekedy náročné rozlíšiť, ktorá je práve hľadaná. Preto sa ukladajú aj karty, ktorých výsledný počet nájdených kľúčových bodov je podobný výsledku karty s maximálnym počtom kľúčovým bodov. Môže to vzniknúť napríklad zlým osvetlením kariet a pokiaľ by sa táto informácia neuchovávala, správny výsledok by sa stratil. Všetky tieto výsledky sa predávajú ďalšej aktivite na zobrazenie výsledku.

4.7 Reprezentácia výsledku

Pre reprezentáciu výsledku je vytvorená vlastná aktivita, ktorá zobrazuje nasledujúce informácie:

- Obrázok a názov výslednej karty,
- Základný popis karty,
- Súvisiace karty a odkaz k nim,
- Karty, ktoré sa podobajú a môžu byť skutočným výsledok vyhľadávania a odkaz k nim.

Tieto informácie sa čerpajú z predchádzajúceho algoritmu, databáze a úložiska obrázkov. Z tohoto kroku môže ísť užívateľ naspäť na vyhľadávanie ďalšej alebo rovnakej karty alebo naspäť do menu.

4.8 Zložitosť algoritmu

Zložitosť algoritmu je lineárna, avšak rýchlo stúpa, kde najväčšia ovplyvňujúca premenná je počet kariet vynásobený počtom spracovávaných snímkov a to z toho dôvodu, že algoritmus sa musí vykonať pre každú kartu zvlášť a to pre každý snímok zvlášť.

4.9 Uloženie dát

Pre uloženie dát je využité SQLite. Obsahuje tabuľky hier, kariet a väzobnú tabuľku medzi kartami. Táto databáza bola v priebehu implementácie viackrát menená, aby zodpovedala aktuálnemu návrhu.

Tabuľka hier obsahuje nasledovné polia:

- ID - identifikátor hry,
- Name - názov hry,
- Rules - pravidlá hry,
- Folder - miesto pre uloženie obrázkov kariet,

ID je primárny kľúč tejto tabuľky hier.
Tabuľka kariet obsahuje:

- ID - identifikátor karty,
- GameID - kľúč odkazujúci na hru ku ktorej táto karta patrí,
- Name - názov karty,
- Description - Popis karty,
- Count - počet rovnakých kariet,
- Image - názov obrázku karty,
- Multiplier - koeficient na určenie počtu skutočných keypointov.

ID je primárny kľúč tejto tabuľky kariet.
Väzobná tabuľka obsahuje:

- ID1 - identifikátor prvej karty,
- ID2 - identifikátor druhej karty.

V prvom kroku bolo potrebné vytvoriť k tejto štruktúre SQL príkazy (angl. queries) na vytvorené tabuliek. Bolo to nadefinované vo funkcii onCreate(), kde boli využité len príkazy CREATE TABLE názvy tabuliek, stĺpcov a ich typov a primárnych kľúčov.

K jednotlivým tabuľkám sú vytvorené aj objekty odpovedajúce Game a Card, ktoré obsahujú implementácie pre operácie s kartami. Samotné texty sú preložené do 3 jazykoch, ale v skutočnosti sa ukladá do databázy len jeden a to podľa nastaveného jazyku operačného systému Android (teda podľa "Locale"). Sú to teda stĺpce názov hry, pravidlá hry, názov karty, popis karty. V pôvodnom návrhu bolo vytvorenie každého prekladu pre každý text

a tomu zodpovedajúce stĺpce v tabuľke, avšak kvôli nevyužiteľnosti a dátovým a časovým požiadavkám k prístupu bolo toto zjednotené len do jednej tabuľky.

V projekte je implementovaná táto databáza vo forme triedy rozširujúcej triedu SQLiteOpenHelper. Pri návrhu databáze bol využitý návrhový vzor jedináčik (singleton), kde funkcia na získanie vracia práve tú istú jednu inštanciu databáze. Tým je zabezpečené vhodné otváranie/zatváranie databáze, uľahčuje to prístup a znižuje aj prístupovú dobu. V prípade, že by to nebolo implementované týmto návrhovým vzorom, vznikala by väčšia pamäťová náročnosť, bolo by potrebné správne otváranie a uzatváranie databázy po každej operácii a pokiaľ by sa toto neustriehlo spôsobovalo by to pády aplikácie. Vzhľadom k tomu, že táto databáze nie je príliš veľká na štruktúru, tak sú len implementované pomocné funkcie na vkladanie dát, získavanie dát, rušenie tabuliek, tvorba tabuliek, aktualizácia tabuliek (pre iné verzie databáz). Jedná sa väčšinou len o jednoduché funkcie pomocou základných podmienených príkazov, jednoduchých spájaní tabuliek, prípadne radení. Tieto príkazy sú implementované pomocou SQL príkazov (angl. queries) a kurzoru, ktorý pristupuje k výsledku.

Projekt je implementovaný tak, že databázu je možné naplniť záznamami v nastaveniach aplikácie, kde sú taktiež funkcie na mazanie a správu databázy. Tieto záznamy sa však nahrávajú aj pri prvom spustení aplikácie, kde sa kontrolujú zdieľané preferencie (shared preferences). Pokiaľ sa teda aplikácia spúšťa prvýkrát je zabezpečené, aby obsahovala všetky dáta pre plnú funkčnosť aplikácie.

Databázu je možné upravovať v metóde onUpgrade(). Databáza bola vytvorená s verziou 1. Pokiaľ sa však táto verzia zmení, je možné previesť SQL príkazy na úpravu štruktúry databázy (pomocou príkazov ALTER).

4.10 Štruktúra kódu

Štruktúra kódu a zdrojových súborov bola kvôli prehľadnosti definovaná nasledovne:

- Zdrojové súbory aplikácie
  - AndroidManifest.xml,
  - Java súbory,
    - priečinok pre aktivity,
    - priečinok pre objekty,
    - databáza a ImageHelper,
  - Res súbory,
    - animácie,
    - drawable obsahujúce ikonky,
    - layout obsahujúce definované XML súbory pre užívateľské rozhranie,
    - values obsahujúce preklady, farby, štýly
  - assets obsahujúce obrázky kariet a inicializačné súbory pre neurónové siete,
  - ostatné vygenerované súbory.SQL
- OpenCV zdrojové súbory

4.11 Užívateľské rozhranie

Užívateľské rozhranie bolo implementované podľa návrhov v predošlej kapitoly. Tieto návrhy sa viackrát menili, preto sa častokrát muselo aj rozhranie upravovať. Taktiež boli nadefinované vlastné grafické prvky, animácie.

Kapitola 5

Rozšírenie programu

5.1 Ďalšia hra

Aby bola aplikácia univerzálna, je potrebné zaistiť, aby pracovala nielen s kartovou hrou BANG! ale aj s ďalšími hrami. To zahrňuje vykonať tie isté kroky ako pri zakladaní záznamu o prvej hre:

- Rozšíriť záznamy databázy o ďalšiu hru a ďalšie záznamy kariet,
- Uložiť obrázok každej karty aplikácie,
- Testovať, porovnať výsledy a reprezentovať výsledok.

5.1.1 Hra

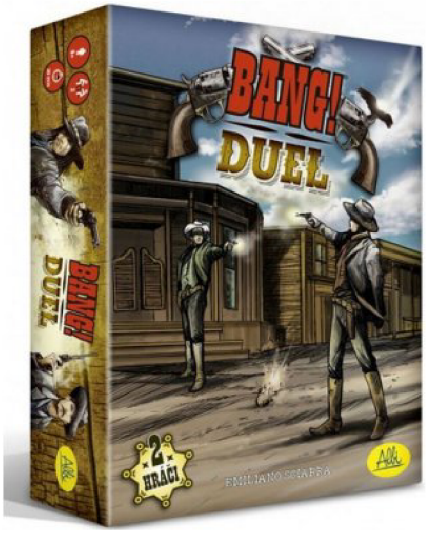
Ako druhá hra bola využitá hrá BANG! Duel. Bang! Duel je samostatná hra pre dvoch hráčov, ku ktorej nie je potrea základnú hru Bang!.

Hru hrajú dvaja hráči. Každý hráč ovláda svoju skupinu pištoľníkov. Každá skupina má dva typy postáv - aktívne postavy a postavy v zálohe. Každý hráč má svoj balíček kariet. Tieto balíčky sa ale v priebehu hry zmiešajú dokopy. V hre víťazí ten, kto ako prvý vyradí postavy svojho protivníka. Zvyčajne sa hrá so štyrmi postavami.

Hra obsahuje niekoľko typov kariet. Karty akcie majú hnedý okraj a ich efekt sa vyhodnotí okamžite. Karty vybavenia majú modrý okraj a ich použitie má dlhodobý efekt. Bang! karty majú červený okraj - nimi útočíte na súperovu aktívnu postavu, za každý ťah môžete túto kartu zahrať iba jednu.

- 80 hracích kariet,
- 24 kariet postáv,
- 4 prehľadové karty,
- 1 karta prípravy,
- 20 žetónov nábojov,
- 2 žetóny aktívnych postáv,

Pre účely testovania a znovupoužiteľnosti na inú hru, neboli využité všetky karty z balíčka, ale len obmedzený počet kariet. Tieto karty sú graficky veľmi podobné klasickej hre BANG! s tým rozdielom, že vľavo dole sa nenachádza identifikátor karty.



Obr. 5.1: Bang duel

5.1.2 Výsledky ďalšej hry

Vzhľadom k tomu, že návrh aplikácie sa snažil byť čo najuniverzálnejší, výsledky odpovedali skutočným výsledkom. Tým pádom je možné považovať aplikáciu za univerzálnu v prípade, že sa doplní ďalšia hra.

5.2 Spracovávanie viacerých kariet súčasne

Pri spracovávaní viacerých kariet súčasne sa vyskytli nasledovné problémy:

5.2.1 Počet kariet

Aplikácie nevie presne rozoznať práve hľadaných počet kariet. Na snímku z kamery bolo problém rozoznať počet pokiaľ mal hráč karty v ruke a pokiaľ ich nemal, mohli vznikať aj tak problémy s rozoznaním počtu kariet

5.2.2 Neprehľadné karty v ruke

Pokiaľ mal užívateľ karty v ruke, karty sa navzájom prekrývali. To znamená že viditeľná karta bola len 1 a ďalšie (schované za ňou) boli vidieť len čiastočne. To veľmi skresľovalo výsledky a tieto karty neboli rozoznané

5.2.3 Úprava podmienok

Z týchto problémov sa vyvodili nasledujúce podmienky:

- Hráč má karty položené na stole, tak aby sa nezakrývali medzi sebou.

- Hráč určí počet kariet, ktoré hľadá vstupom pri inicializácii.

5.2.4 Výsledok rozšírenia

Pri tomto rozšírení sa prejavovali problémy, ktoré môžu nastať aj pri rozpoznávaní kariet po jednej. Avšak výsledky boli také, že karty sa objavovali vo výsledkoch vyhľadávania vždy, avšak niektoré z nich boli výsledkom podľa rovnice nepresnosti a nie hlavné hľadané karty.

Kapitola 6

Testovanie

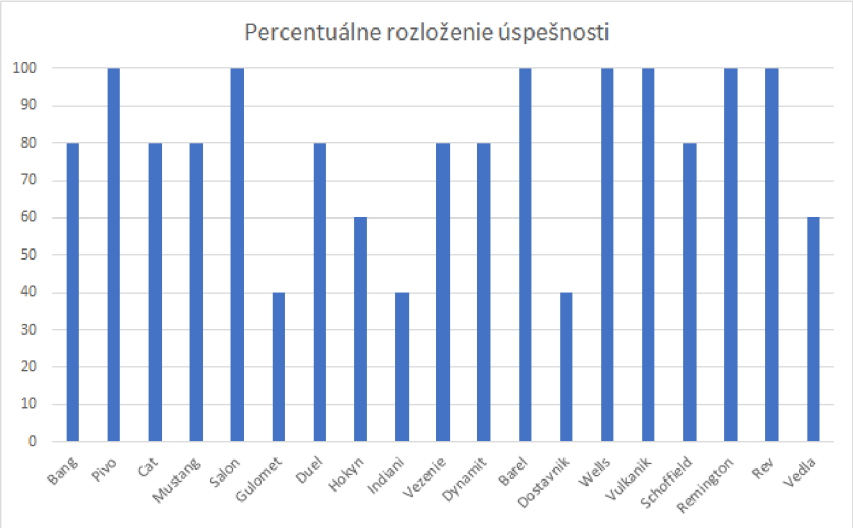
Testovanie je možné rozumieť dve rôzne sekcie:

- Testovanie funkcionality - interné testovanie vývojárom,
- Testovanie použiteľnosti - otvorené testovanie.

6.1 Testovanie funkcionality

Testovanie funkcionality pozostávalo z testovania, či hľadané karty odpovedali skutočným výsledkom vyhľadávania. Toto testovanie prebiehalo v rôznych fázach po každej zmene alebo úpravu algoritmu samotného rozpoznávania. Každá fáza bola spočiatku testovaná na menšom počte kariet a postupne tieto karty pribúdali. Spočiatku sa výsledky veľmi líšili hľadaným výsledkom. V poslednej fáze prebiehalo najväčšie testovanie, ktoré zahŕňalo všetky karty.

Výsledky testovania sú reprezentované grafom 6.1.



Obr. 6.1: Graf úspešnosti

Tu je vidieť, že priemerný výsledok je takmer 75 percent úspešnosti. Nesprávne výsledky je možné odôvodniť nepresnou určenou konštantou pre výpočet, zlou predlohou na porovnanie alebo nepresným meraním (neostrý obraz).

Avšak aj napriek tomu, že výsledky neboli vždy 100 percentné, vo výsledkoch sa vždy nachádzala hľadaná karta (s preklikom) ako možná druhá karta.

Toto testovanie taktiež obsahovalo chyby v užívateľskom rozhraní aplikácie alebo prácu s databázou, ktoré boli ihneď odstraňované, pretože často spôsobovali pády aplikácie a neumožňovali testovanie funkčnosti.

6.2 Testovanie použiteľnosti

Pri testovaní použiteľnosti bola aplikácie otestovaná na nižšom počte užívateľov, ktorí si preklikali funkcionalitu aplikácie. Z výsledkov je možné zhodnotiť, že toto testovanie odhalilo viaceré chyby alebo (grafické) vylepšenia na užívateľskom rozhraní a chyby v presnosti algoritmu.

Kapitola 7

Záver

Táto aplikácia sa preukázala ako univerzálna a ďalej možná rozšírení. Je možné s ňou pracovať pre vyhradenú skupinu ľudí, ktorí sa zoznamujú s konkrétnou kartovou hrou alebo danú hru ešte nepoznajú. Funkcionalita aplikácie sa preukázala ako použiteľná, vzhľadom k tomu, že výsledná karta sa vždy objavila vo výsledku vyhľadávania. Nevýhoda je, že nie vždy sa karta objavila ako skutočný výsledok (ale niekedy len z množiny možných výsledkov) a náročnosť na spracovávanie výsledku počas snímania.

Literatúra

[1] *The OpenCV Reference Manual*. 2.4.9.0. Itseez, April 2014.

[2] ANDROIDDEVELOPERS. *Android Studio*. 2020. Dostupné z: https://developer.android.com/studio.

[3] BANG. *BANG*. 2020. Dostupné z: http://www.bang.cz/cs/bang.html.

[4] BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. 2000.

[5] HIPP, R. D. *SQLite*. 2020. Dostupné z: https://www.sqlite.org/index.html.

[6] ITSEEZ. *Open Source Computer Vision Library* [https://github.com/itseez/opencv]. 2015.

[7] JETBRAINS. *IntelliJ Idea*. 2021. Dostupné z: https://www.jetbrains.com/idea/.

[8] MICROSOFT. *Visual Studio*. 2021. Dostupné z: https://visualstudio.microsoft.com/cs/.

[9] OPENCV. *Basics of Brute-Force Matcher*. 2021. Dostupné z: https://docs.opencv.org/master/dc/dc3/tutorial_py_matcher.html.

[10] OPENCV. *How to run deep networks on Android device*. 2021. Dostupné z: https://docs.opencv.org/master/d8/dc8/tutorial_histogram_comparison.html.

[11] OPENCV. *How to run deep networks on Android device*. 2021. Dostupné z: https://docs.opencv.org/3.4/d4/dc6/tutorial_py_template_matching.html.

[12] OPENCV. *How to run deep networks on Android device*. 2021. Dostupné z: https://docs.opencv.org/3.4/d0/d6c/tutorial_dnn_android.html.

[13] SERGVOLOSHYN. *Android OCR Application Based on Tesseract*. 2019. Dostupné z: https://www.codeproject.com/Articles/1275580/Android-OCR-Application-Based-on-Tesseract.

[14] SMITH, R. *Tesseract*. 2021. Dostupné z: https://github.com/tesseract-ocr/tesseract.

[15] TUTORIALSPOINT. *Android Studio tutoriál*. 2020. Dostupné z: https://www.tutorialspoint.com/android/android_studio.htm.

[16] WIJESUNDARA, A. *Object Recognition with OpenCV on Android*. 2016. Dostupné z: https://akshikawijesundara.medium.com/object-recognition-with-opencv-on-android-6435277ab285.

Príloha A

Program - zdrojový kód

Obsahom tejto prílohy je zdrojový kód aplikácie vo forme inštalačného balíčka APK. Tento balík zaberá viacej miesta a to z toho dôvodu, že OpenCV knižnica má obrovskú kapacitu v porovnaní so samotným kódom aplikácie.

Súčasťou aplikácie sú:

- Zdrojové kódy aplikácie
- OpenCV knižnica
- Obrázky, ikonky, preklady
- Predlohové obrázky kariet
- SQLite databáza

Príloha B

Inštalačný balík

Obsahom tejto prílohy je inštalačný balík vo formáte APK. Balík je inštalovateľný na smartfónoch s operačným systémom Android verzie 4.4 a vyššie. Tento súbor je možné vygenerovať pomocou zdrojových súborov z prílohy A a to vo vývojovom prostredí Android Studio. Balík obsahuje približne 100 MB.