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

PODPORA HRY SAGRADA NA MOBILNÍM TELEFONU S OS ANDROID

SUPPORT FOR SAGRADA BOARD GAME ON MOBILE PHONE WITH OS ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAN TRNĚNÝ**
AUTHOR

VEDOUCÍ PRÁCE **Doc. RNDr. PAVEL SMRŽ, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Trněný Jan**
Program: Informační technologie
Název: **Podpora hry Sagrada na mobilním telefonu s OS Android**
Support for Sagrada Game on Mobile Phone with OS Android
Kategorie: Počítačová grafika

Zadání:

1. Seznamte se s dosavadními aplikacemi pro podporu hraní deskových her a metodami pro zpracování a rozpoznávání obrazu v OS Android.
2. Shromážděte datovou sadu snímků herního plánu hry Sagrada v různých fázích hry a připravte data pro průběžné vyhodnocování kvality rozpoznávání a vyhodnocování pravidel a kontroly robustnosti.
3. Navrhněte a implementujte systém pro usnadnění hodnocení výsledků hry Sagrada pomocí rozpoznávání a zpracování na mobilním telefonu.
4. Vyhodnoťte realizované řešení v adekvátní uživatelské studii v reálném prostředí při hraní.
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- dle doporučení vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- funkční prototyp řešení

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 2019
Datum odevzdání: 28. května 2020
Datum schválení: 1. listopadu 2019

Abstrakt

Cílem této práce je vytvořit podpůrnou aplikaci k deskové hře Sagrada na mobilní zařízení s OS Android. Řešení se skládá z detekce a rozpoznání karty herního vzoru a kostek na hrací ploše za využití knihovny OpenCV. Následně je poskytnuta podpora, v průběhu a na závěr hry, pro výpočet bodů a kontrolu pravidel pro jednoho i více hráčů. Tyto funkce dovolují spravovat data více hráčů na jednom mobilním zařízení a pomáhají k rychlejší kontrole pravidel a výpočtu bodů. Funkce zároveň poskytují pomoc začínajícím hráčům k lepšímu pochopení hry.

Abstract

The aim of this thesis is to create support application for board game Sagrada on mobile devices with OS Android. The solution consist of detection and recognition of the game pattern card and dices on board using the OpenCV library. Subsequently, support is provided, in any state of the game, for points calculation and rules checking for any number of players. These functions allow management of data for multiple users on one mobile device and help with faster rules checking and points calculation. Functions also help for beginners in understanding the game.

Klíčová slova

desková hra, Sagrada, Android, mobilní aplikace, rozpoznání obrazu, OpenCV

Keywords

board game, Sagrada, Android, mobile application, image recognition, OpenCV

Citace

TRNĚNÝ, Jan. *Podpora hry Sagrada na mobilním telefonu s OS Android*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. RNDr. Pavel Smrž, Ph.D.

Podpora hry Sagrada na mobilním telefonu s OS Android

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. RNDr. Pavla Smrže, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

. . . . . .
Jan Trněný
28. května 2020

Poděkování

Rád bych poděkoval doc. RNDr. Pavlu Smržovi, Ph.D. za vedení práce, cenné rady a věcné připomínky, které mi pomohly při tvorbě této práce. Dále bych rád poděkoval Ing. Martinu Čermákovi, Ph.D za konzultace a věcné rady při tvorbě této práce.

Obsah

Kapitola 1

Úvod

Mobilní zařízení se dají v dnešní době využít téměř ve všech odvětvích. Ať už pro sportovní účely, nebo studijní, vždy nalezneme nějakou aplikaci, která využívá mobilního zařízení k usnadnění činností člověka. Velkou roli zde hraje i operační systém Android 3, jež svou dostupností a podílem na trhu [13] získal širokou komunitu vývojářů.

Jedním z možných odvětví, kterému se lze věnovat, je rozpoznání obrazu na mobilních zařízení. Ve většině případů se pro rozpoznání obrazu využívalo pouze kamery mobilního zařízení, ze kterého byla pořízená fotka zaslána na server, kde docházelo ke zpracování v konkrétní serverové aplikaci. Tento postup je používán z důvodů kompatibility se zařízeními, která nemají dostatečný výkon pro větší výpočetní postupy při zpracovávání obrazu. V posledních letech však mobilní zařízení obsahují výkonnější hardware, díky čemuž je možné využít této výpočetní síly bez nutnosti zasílání dat na server a redukovat tak potřebné prostředky pro vytvoření a běh aplikace. Rozpoznání obrazu přímo na mobilních zařízeních může být v některých případech klíčové. Za určitých okolností je detekce na zařízení vyžadována z bezpečnostních důvodů (ochrana soukromí, citlivé údaje). Dále také může zařízení být mimo dosah internetového připojení, v takovém případě je detekce přímo na zařízení vhodná. Stejným zaměřením se ubírá má práce, která se zaměřuje na rozpoznání obrazu na lokálním mobilním zařízení, bez využití externích možností (serveru). Konkrétně se práce bude věnovat rozpoznání prvků deskové hry na lokálním mobilním zařízení.

Využití rozpoznání obrazu je v rámci mobilních zařízení široké a lze jej nalézt v mnoha kategoriích, pro které vznikají aplikace na mobilní zařízení. Rozpoznání obrazu slouží za účelem získat informace z námi hledaných objektů v obraze. V mnoha případech je uživateli zjednodušena práce s vyhledáváním informací, které by na základě slovního popisu bylo obtížné vyhledávat. Využití rozpoznání obrazu lze uplatnit i na podporu deskových her. Deskové hry jsou součástí kultury člověka od nepaměti. S nástupem elektronických zařízení a zejména těch mobilních, se při vytváření nových deskových her začalo počítat i s využitím těchto zařízení a také rozpoznání obrazu. Pokud se podíváme na deskové hry, které vznikly bez podpory mobilních zařízení, lze u plno z nich nalézt dostupné aplikace, které alespoň částečně usnadňují hraní těchto her, nebo nám pomáhají se tyto hry naučit. Jednou z deskových her, která žádnou podporu mobilního zařízení nemá, je Sagrada 2.

V mnoha případech lze u deskových her vypozorovat neustále se opakující postup, který jde programově zjednodušit. Rozpoznání obrazu je v tomto případě výhodným řešením, jelikož získává informace o deskové hře. Z těmito informacemi lze posléze pracovat v rámci aplikace a poskytnout tak podporu, nebo rozšíření deskové hry pro hráče a nabídnout tak umocněný požitek z hraní.

Tato práce se věnuje vytvoření podpory pro deskovou hru Sagrada na zařízení s operačním systémem Android. Kromě vytvoření uživatelsky přívětivé aplikace, se práce zaobírá zejména systémem detekce části hrací plochy. Systém bude získávat informace ze vzorových karet a barevných kostek deskové hry. Při implementaci rozpoznávacího systému bude kladen důraz na využití pouze základních rozpoznávacích metod a volně šiřitelných knihoven, bez využití strojového učení. Výsledkem práce by měla být aplikace, která bude vypočítávat body hráče v daném stavu hry a na základě dostupných informací. Zároveň si bude hráč moci kontrolovat pravidla v průběhu hry.

Práce dále pokládá za důležité, aby aplikace byla využitelná i v prostředí bez možnosti přístupu k internetu. Z toho důvodu budou veškeré operace v rámci aplikace prováděny na daném mobilním zařízení.

V následujících kapitolách je popsána desková hra a její pravidla 2. Dále je přiblížena platforma OS Android 3. Práce popisuje některé pojmy a postupy počítačového vidění 5 a existující postupy při rozpoznání prvků deskových her 4. V rámci práce bude kromě popisu návrhu 6 a implementace 8, provedeno testování 9. Testování bude obsahovat kromě úspěšnosti rozpoznání obrazu také uživatelskou studii.

Kapitola 2

Desková hra Sagrada

Sagrada je desková hra vytvořená dvojicí Adrian Adamescu a Daryl Andrews od vydavatelství MindOK a Floodgate Games. Vyšla v roce 2017. Mohou ji hrát $2-4$ hráči od 8 let, ale pravidla umožňují hru pro 1 hráče. Hra trvá zhruba 40 minut a je dostupná i v českém jazyce.

2.1 Cíl hry

Cílem hry je poskládat vitráže chrámů a katedrál. Ty musí hráč poskládat, na kartu klíče vitráže, pomocí barevných kostek tak, aby získal body za úkolové karty. Zároveň se musí držet pravidel, které mohou v případě jejich porušení vést ke ztrátě bodů na konci hry.

2.2 Herní části

Sagrada obsahuje:

- Počítadlo kol a 4 okenní rámy.
- 12 karet, každá se vzorovou vitráží na každé straně.
- 90 kostek (po 18 v 5 barvách).
- 12 karet nářadí.
- 10 společných karet úkolů a 5 osobních karet úkolů.
- 24 žetonů řemeslníků.

2.3 Popis pravidel

Každý hráč si na počátku vylosuje jednu kartu herního klíče z balíčku, který následuje výběr karty osobního úkolu. Poté je vybrána karta společného úkolu, ta je pro všechny hráče stejná. Následně se odehraje 10 kol. V každém kole začínající hráč vytáhne z pytlíku s kostkami celkově $2n + 1$ kostek, kde n reprezentuje počet hráčů ve hře. S kostkami je následně hozeno a jsou ponechány na stole. Následuje fáze, kdy si každý z hráčů v daném pořadí vybírá 1 kostku, kterou umístí na jim zvolenou pozici ve vitráži. Při tom však musí dbát na pravidla hry:

1. První kostka musí být vložena na některé z rohových, nebo okrajových polí prázdné vzorové vitráž.
2. Kostky musí sousedit alespoň s jednou kostkou diagonálně, nebo vodorovně, svisle.
3. Položená kostka musí splňovat danou podmínku pole na vzorové kartě.
4. Kostky nesmí sousedit s hranou jiné kostky stejným číslem, či barvou.

Dále může hráč při svém tahu využít žetonů řemeslníka a zakoupit kartu nářadí, která umožňuje operaci s kostkami ve vitráži. První použití karty stojí pouze jeden žeton, následná další použití téže karty je však vždy o jeden žeton dražší. Po odehrání všech deseti kol se přejde do fáze, kdy hráči sčítají body v závislosti na úkolech, které byly losovány. Pokud hráč zjistí, že bylo porušeno pravidlo, musí přemístit, nebo odstranit kostky z vitráže. Za každou odstraněnou kostku z vitráže po konci hry je stržen bod. Vyhrává ten hráč, který nasbírá nejvíce bodů získané za splnění osobního a společného úkolu.

Obrázek 2.1: Desková hra Sagrada

Kapitola 3

OS Android

Kapitola popisuje z jakého důvodu byl vybrán operační systém Android a přibližuje možnosti, jak na této platformě vyvíjet. Dále se kapitola věnuje popisu jednotlivých komponent pro vývoj aplikací v OS Android.

3.1 Operační systém Android

Android je platforma pro mobilní zařízení, vytvořená konsorciem OHA[1], založená na linuxovém jádře. Primárně byla tato platforma zamýšlena pouze pro telefonní zařízení. V dnešní době však Android lze nalézt nejen v telefonech, ale i tabletech, chytrých hodinkách, televizích a dopravních prostředcích. Jedná se o OSS[2], díky čemuž je prodej zařízení s Android levnější a dostupnější. Tuto domněnku potvrzuje fakt, že v roce 2019 je podíl Android zařízení na trhu s chytrými telefony 87%[13]. V dnešní době je největším podporovatelem Android platformy společnost Google.

3.1.1 Vývojová prostředí

Pro Android existuje řada vývojových prostředí(dále IDE), které zjednodušují vývoj. Nejznámějším IDE je Android studio vydané společností Google. Kromě kompletní vývojové sady obsahuje i emulátor mobilního zařízení, takže není nutné mít pro testování fyzické zařízení. IDE je postaveno na vývojovém prostředí od JetBrains.

IntelliJ IDEA je vývojové prostředí od společnosti JetBrains vyvíjející software pro programátory a projektové manažery. Obsahuje veškeré potřebné složky pro kompletní vývoj aplikací na libovolná zařízení. Je primárně určen pro vývoj v jazyce Java.

Visual Studio není přímo určeno pro vývoj aplikací pro OS Android, ale Microsoft integroval do tohoto prostředí mobilní platformu Xamarin, která umožňuje vývoj nativních aplikací pro Android, iOS a Windows v jazyce C#. Stejně jako předchozí prostředí obsahuje emulátor. Hlavní myšlenkou je společný kód na pozadí aplikace, který se poté spojí s grafickým rozhraním specifické platformy.

[1] Open Handset Alliance viz. http://www.openhandsetalliance.com/

[2] Otevřený software s volně dostupným zdrojovým kódem

3.1.2 Android Native

Android nabízí možnost vyvíjet aplikace a programy i v jiných jazycích, než Java a Kotlin. Pomocí sady Native Developement Kit (dále NDK) lze použít nativní[3] kód v jazyce C/C++. NDK umožňuje vytvořit společnou logiku aplikace pro více různých systémů, podobně jako Xamarin. NDK se využívá zejména v případě, když potřebujeme na mobilním zařízení provádět náročné výpočetní úkony např. hry, fyzikální procesy, nebo požadujeme rychlou odezvu při výpočtu.

3.2 Tvorba aplikace

V předchozích sekcích bylo řečeno, ve kterých vývojových prostředích lze efektivně vyvíjet aplikace pro zařízení Android. V této sekci se obecně přiblíží, jak fungují aplikace v OS Android.

3.2.1 Aktivita

Aktivita je základní komponentou při tvorbě aplikace pro OS Android. Aktivity umožní uživateli skrze uživatelské rozhraní ovládat aplikaci a přijímat informace. Aplikace může takových aktivit obsahovat celou řadu. Aktivita by měla být navržena tak, aby umožnila uživateli soustředit se na jednu konkrétní věc [17].Uspořádání aktivit je hierarchické. Aplikace spustí hlavní aktivitu, která je nastavena jako hlavní v souboru `AndroidManifest.xml`[4]. Z této aktivity se posléze aktivují další aktivity v závislosti na návrhu aplikace.

Aktivita má předem daný životní cyklus, který je zobrazen na obrázku 3.1. Ten je určený sadou metod, které programátor může modifikovat. Cyklus má tři fáze[17]:

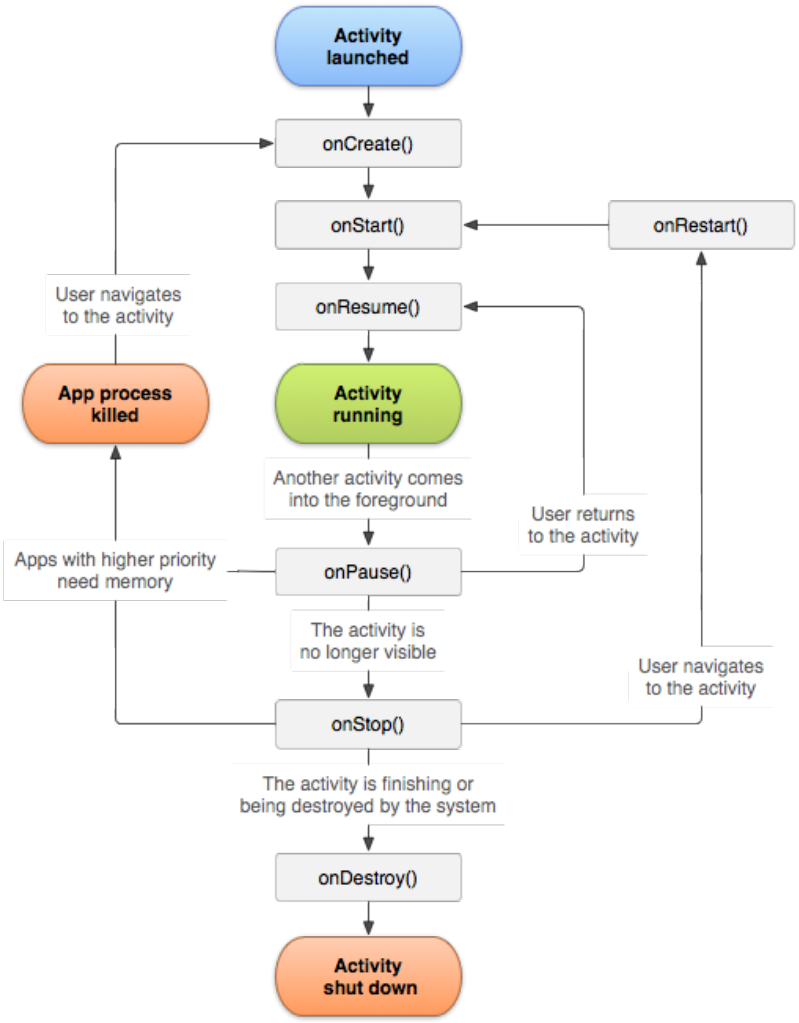
- **Aktivita na popředí** - Aktivita, která je právě zobrazena na displeji a uživatel s ní může interagovat.
- **Pozastavená aktivita** - Aktivita je stále v paměti, ale je překrytá například dialogovým oknem. Uživatel k ní nemá přístup. Při nedostatku paměti může systém aktivitu v tomto stavu odstranit.
- **Zastavená aktivita** - Aktivita je kompletně překrytá jinou aktivitou, ale nachází se v oblasti Back Stacku, lze se tedy k ní vrátit. V případě nedostatku paměti jsou primárně odstraněny aktivity v zastaveném stavu.

Při implementaci funkcionality aplikace se využívá několik metod z obrázku 3.1. V těch definujeme, co má aplikace v jednotlivých fázích životního cyklu aktivity dělat. Níže je popsáno několik metod, které ovlivňují interakci uživatele s aplikací, popřípadě práci s daty.

- `onCreate()`: Metoda je volána při vytváření aktivity. Obsahuje informace o uživatelském prostředí a připravuje potřebná data pro vytvoření aktivity.
- `onResume()`: Volána při přechodu aktivity do popředí. Zde započíná komunikace uživatele s aktivitou.
- `onPause()`: Aktivita je pozastavena při přechodu do pozadí, ale je stále viditelná. Volání této metody je zaručené, pokud má být aktivita ukončena.

[3] Kód přeložený pro specifický procesor
[4] Soubor obsahující klíčové informace o aplikaci pro sestavovací nástroj Androidu a pro OS.

- `onDestroy()`: Metoda volání při ukončení života aktivity. Volání této metody není zaručené.



Obrázek 3.1: Životní cyklus aktivity v OS Android. Převzato z: https://developer-.android.com/guide/components/activities/activity-lifecycle

3.2.2 Fragment

Aktivity umožňují interakci uživatele s aplikací viz. 3.2.1. Fragment reprezentuje chování, nebo část uživatelského rozhraní aktivity. Několik fragmentů lze kombinovat do jedné aktivity a vytvářet tak znovupoužitelné komponenty ve více aktivitách. O fragmentu lze uvažovat jako o modulární části aktivity[8]. Každý fragment je závislý na aktivitě, která jej vlastní. Pokud je aktivita ukončena, nebo pozastavena, je fragment ve stejném stavu jako rodičovská aktivita. Fragmenty, stejně jako aktivity, podléhají životnímu cyklu.

3.2.3 Zdroje

Zdroje jsou přidané soubory a statický obsah, který v kódu aplikace obsahuje. Jedná se především o definice rozhraní grafických komponentů, textových řetězců, obrázků a animačních instrukcí. Zdroje jsou takto odděleny od samotného kódu a přistupuje se k nim na základě jedinečného identifikátoru, který je každému zdroji přidělen.

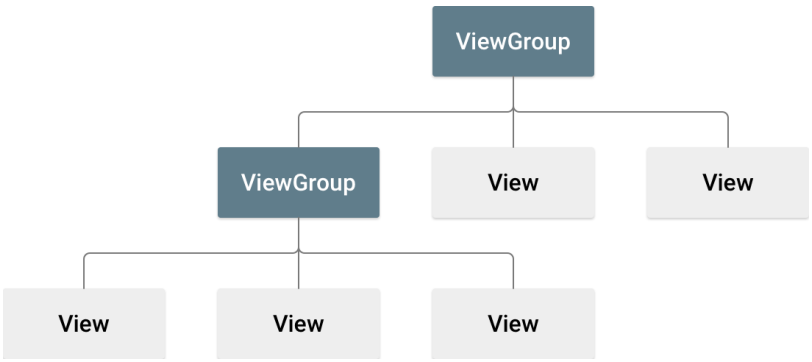
Oddělenost zdrojů od kódu umožňuje jednoduché změny definic grafických komponent, nebo přidání několika druhů textů v závislosti na jazyce daného zařízení. Zdroje také umožňují definice stejných uživatelských komponent pro různé velikosti a rozlišení jednotlivých zařízení.

3.2.4 Tvorba uživatelského prostředí

Uživatelské prostředí se vytváří v OS Android pomocí rozložení (Layout). Jednotlivé prvky v rozložení jsou složeny z hierarchie pohledů (Views), které patří do skupin pohledů (ViewGroup), jak je ukázané na obrázku 3.2 . Rozložení lze deklarovat několika způsoby[8]:

- **Deklarace UI prvků v XML** - Android disponuje XML definicemi prvků UI a jejich vlastností. Tyto prvky jsou skládány do View tříd a podtříd. Android studio 3.1.1 obsahuje Layout Editor pro snadné vkládání komponent.
- **Instanciace UI prvků za běhu** - Jednotlivé prvky nebo celá rozložení lze vytvořit programově při spuštění aplikace.

Využití XML reprezentace rozložení umožňuje separovat kód chování aplikace od jeho grafické reprezentace a lze pomocí XML definicí jednoduše měnit vlastnosti prvků rozložení.



Obrázek 3.2: Hierarchie prostředí. Převzato z: https://developer.android.com/guide/topics/ui/declaring-layout

Kapitola 4

Podpora deskových her

Pokud přemýšlíme nad podporou pro deskovou hru, musíme vytyčit prvky, kterými se lze zaobírat. Velkou roli v této úvaze hraje způsob, jakým chceme hru vylepšit. Ve výsledku chceme hru podpořit rozšířením o virtuální grafické prvky přes rozšířenou realitu, nebo v rámci aplikace. Také však můžeme detekovat herní prvky za účelem vyhodnocení herního stavu, kontrolu pravidel, nebo jako nápovědu. Následující sekce se věnují podpoře deskových her obecně, herním prvkům a existujícím řešením pro podporu deskových her.

4.1 Obecná podpora deskových her

Budeme-li uvažovat o souhrnu podpory pro deskové hry, můžeme rozdělit podporu na jednotlivé prvky (karty, kostky, hrací plochy) dané deskové hry. Z těchto prvků poté mohou být poskládány větší podpůrné celky (kontroly pravidel, vyhodnocení bodů, virtuální hráč). V základu se ve většině případu jedná o rozpoznání herních objektů v obraze a získání potřebných informací.

4.1.1 Podpora karet a kostek

Pokud hledáme počet herních karet na stole, hledáme v obrazu jednotlivé karty, které následně spočítáme. Zároveň z těchto karet můžeme získat informace, které lze použít zejména při kontrole pravidel dané hry, nebo při jejím vyhodnocení. Některé karty jsou specifické svým obsahem.

Stejným způsobem nelze získávat informace z karet *Kanasty* a z karet *Bang!*. Svým obsahem jsou tyto hry velmi odlišné. V prvním případě hledáme konkrétní vzory, které lze porovnávat s referenčním vzorem. Na druhou stranu u *Bang!* hledáme například název karty pomocí rozpoznání textu. Název karty poté hledáme v databázi aplikace, která obsahuje přesné informace o kartě v kontextu deskové hry.

Obdobně jako u karet, se velmi často v deskových hrách setkáváme s kostkami. Z kostek se ve většině případů získává informace o jejich barvě a symbolu na horní straně kostky. Velmi často je symbol tvořen kruhy tvořící číslici, nebo přímo číslicí. V takovém případě lze i u barevných kostek rozeznat číslici na kostce. Občas můžeme narazit na kostky, jejich symboly je potřebné rozeznat na základě porovnání se vzorem. V obou případech však ve většině případů požadujeme informaci spojenou se symbolem.

4.1.2 Podpora herních ploch

S hrací plochou se setkáváme v nepřeberném množství deskových her. Na hrací ploše můžeme vytvářet projekce 3D objektů, nebo získávat informace z konkrétních pozic plochy. Z typů hracích ploch uvažujme dva rozšířené typy.

Hry *Šachy*, *Scrabble*, *Mlýny* aj. mají společné rysy hracích ploch. Jedná se o plochy dodržující určitý vzor, který pokrývá celou hrací desku. Jednotlivá pole určují, kam lze položit herní prvky (figurky, kameny). Při rozpoznání těchto hracích ploch a jejich polí se můžeme držet podobného postupu rozpoznání hracích ploch na základě vyhledávání daného vzoru skrze dostupné metody pro zpracování obrazu. Pro příklad z šachovnice lze získat údaje o jednotlivých polích na základě nalezení černých a bílých čtverců, nebo detekování hran mezi jednotlivými poli, či nalezení linií jejichž spoje určují místa pro položení kamenů.

Dalším možným typem herní plochy je herní deska, která vytváří „cestu“ deskovou hrou. V takovém případě hledáme jednoznačně rozeznatelná pole v rámci hledané desky. Tyto unikátní vzory v rámci herního plánu můžeme porovnat se vzorovým obrazem pole, nebo na základě specifických vlastností hledaného objektu (barva, tvar, text).

V obou typech lze po nalezení polí určit o jaký typ pole se jedná a získat z něj potřebnou informace na základě vlastností daného pole. Informace mohou posloužit k bodovému vyhodnocení, kontrole pravidel, nápovědě a jiných.

4.1.3 Ostatní herní prvky

V předchozích sekcích byly zmíněny herní prvky, které se vyskytují v nepřeberném množství deskových her. Existují velmi specifické prvky, které mohou být unikátní pro danou hru. V takovém případě je zapotřebí vymyslet postup pro jejich detekci a získání informace. Jedná-li se například o figurku na herním plánu, bude se zřejmě nacházet na políčku hry [6]. Popřípadě lze objekt nalézt na základě porovnávání s referenčním objektem.

4.1.4 Souhrn podpory

Pokud by hypoteticky existovala knihovna pro univerzální podporu deskových her, zřejmě by vycházela z předchozích podpor a následně by byla doplněna o obecný postup při detekování specifických objektů na základě existujících rozpoznávacích metod v obraze 4.3.

Z informací získaných na základě jednotlivých herních prvků může být vytvořen systém, který vytváří komplexní podpory dané deskové hry. V případě hry *Scrabble* lze získat informace hrací plochy a jednotlivých destiček s písmenky. Při spojení destiček získáme informace o slovech. Z takto nalezených informací může být vytvořen systém pro výpočet bodů za slovo, nebo kontrolní systém existence daného slova.

V souhrnu tedy lze říci, že kromě detekce herních prvků lze mluvit o pomyslném systému, kterému dodáme informace o hledaných objektech na základě porovnávání, nebo hledání vlastností objektů. Dále soubor pravidel a postupů jak z takovými informacemi v rámci deskové hry zacházet.

4.2 Rozšířená realita

Virtuální objekty pomáhají rozšířit hrací desku o animované prvky. Ty slouží primárně pro hlubší ponoření hráčů do hry a lepší vyobrazení daných prvků, které by v podobě reálné figurky nepůsobily věrohodně. V deskových hrách trend rozšířené reality (dále AR) není moc

zastoupený. Pro některé hry se v posledních letech vytváří řešení, která rozšiřují deskové hry o nějaké AR prvky.

4.2.1 AR přístupy

Při použití AR rozlišujeme dva hlavní přístupy: *Markerbased* a *Markerless*. U Markerbased přístupu kamera snadno v obraze detekuje místo, kam umístit projekci AR modelu [23]. Toto místo je označenou nějakou jednoznačnou značkou, kterou zařízení rychle rozezná. Tento přístup používá například platforma ARTable [1].

Značky pro *Markerbased* přístup mohou být různorodé. Pro velmi rychlé a jednoduché rozpoznání značky v obraze lze použít např. QR kód. Pokud však implementujeme podporu pro hru, která již existuje a nepočítala s možností rozšířené reality, můžeme v oblasti, do které promítáme AR objekt, najít specifickou oblast, která zde bude vždy a podle ní orientovat AR objekty. V kartách lze definovat jednoznačné prvky, podle nichž vykreslíme objekty ve scéně. Příkladem můžeme uvést bojovou hru, která na základě karet herních postav vytváří projekce těchto postav a simuluje jejich boj. V kartách se rozpoznají zájmové body definující typ postavy a detekují se hrany karty. V tomto prostoru je poté vytvořena 3D projekce konkrétní karetní postavičky [3]. Obdobný přístup používá i hra Unlock. Přináší prvky Únikové místnosti do podoby karet a mobilní aplikace. Karty mají značky, které určují operace. Skrze příslušnou aplikaci v telefonu můžeme na kartách získávat indície potřebné pro únik z místnosti. Značky lze také spojovat přiložením dvou karet k sobě a najít tak nové možnosti jak uniknout z místnosti [22].

Obrázek 4.1: Rozšířená realita hry Unlock. Převzato z webu https://www.engadget.com/-2019-11-29-best-board-games-companion-app-guide.html

Na druhou stranu, *Markerless* přístup v obraze značky neobsahuje. Hodí se zejména pro malá přenosná zařízení, která nevyžadují specifickou techniku pro přesné zobrazování AR modelů. Zároveň, je tento přístup odkázán pouze na kvalitu přenosných zařízení (např. mobilní telefon), z nichž používá kromě kamery i senzory pro určení polohy a směru. Na základě těchto informací a definování místa, kde zobrazovat modely, lze poté ve snímaném obraze modelovat AR objekty [23].

[1] https://github.com/robofit/arcor

4.3 Rozpoznání herních prvků

V předchozí sekci 4.2 jsme rozebírali podporu deskových her za použití rozšířené reality. V této sekci se budeme věnovat jednotlivým prvkům deskových her, pro které existují řešení v rámci základních metod rozpoznání obrazu a strojového učení. Jednotlivé metody a řešení slouží pro získání informací z konkrétních herních prvků.

4.3.1 Hrací deska

Hrací desky patří k nejrozšířenějším prvkům deskových her. Desky slouží jako hrací mapa, která definuje prostředí hry, může hráče vést po cestě k cíli, nebo vymezuje oblasti pro pokládání figurek a jiné. Hrací desky jednotlivých her jsou různorodé a proto je komplikované vytvořit univerzální detektor hrací plochy.

Pokud vezmeme v potaz základní tvar hrací desky, ve většině případů dostaneme čtvercové, obdélníkové, nebo kruhové tvary desky. Jejich detekce je napříč více pracemi, které se zaobíraly problematikou rozpoznání hrací plochy podobná. Základem pro detekce hracích desek je hranový detektor 5.3.2. Ze získaných hran lze získat informace o tvarech jednotlivých objektů. Pokud hledáme desku ve tvaru čtverce, hledáme objekty, jejichž hrany tvoří čtverec. K získání těchto informací lze použít detektor kontur 5.3.3. Ve většině případů se při získávání oblasti hrací desky pracuje s faktem, že hrací deska je největším objektem v záběru kamery [6]. Z takto získané oblasti můžeme získávat podrobnější informace. Získání konkrétních informací z hrací desky se liší v závislosti na typu hry a tudíž typu hrací desky.

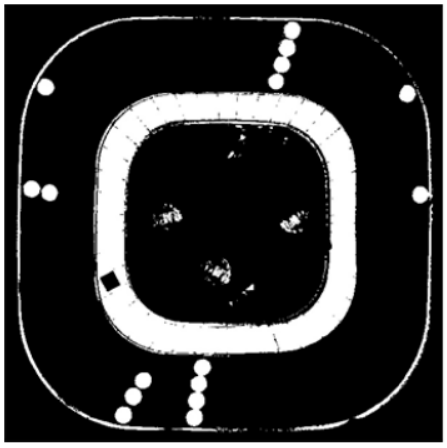
Bereme-li v potaz hrací plochy, které jsou podobné šachovnici, nebo hrací ploše hry Scrabble, snažíme se detekovat na hrací ploše jednotlivá pole. Detekce jednotlivých polí se liší od implementace. V případě hry Scrabble lze uplatnit Houghovu transformaci pro zvýraznění hlavních os (viz. 4.2a), které definují jednotlivá pole [21]. Obdobně lze nalézt pozice pro položení kamenů hry *Go*, která aplikuje Houghovu transformaci za účelem nalezení těchto pozic [18]. Dalším způsobem detekce polí, je filtrování barev těchto polí. Pokud jsou pole jednobarevná, nebo jejich většinová část je tvořena konkrétní barvou. Správná detekce jednotlivých barevných polí je závislá na správně zvoleném rozsahu jednotlivých barev a zvoleném barevném modelu 5.2.2. Příkladem můžeme vzít hru Sázky a dostihy. Zde pomocí detekce žluté barvy jsou nalezena pole na nichž závodníci umístí své figurky (viz. 4.2b) [6]. Kromě šachovnicových hracích desek existují další různé varianty, které jsou specifické svým řešením.

4.3.2 Karty

Karty jsou nedílnou součástí velké části deskových her. Přístupů k jejich detekci je několik. Kromě detekce karty v obraze, která se podobá detekci hracích desek (viz. 4.3.1), je důležité získání informací z karet. V kartě lze detekovat symboly. Pro jejich určení lze využít porovnání aktuální oblasti s referenčním obrazem. Referenční obraz je ideální fotka sloužící jako vzor. S tímto vzorem jsou posléze porovnány obrazy z konkrétní detekce. Na základě nejlepší shody se vzorem se určí typ symbolu [1].

Kromě symbolů, tvarů a barev se v kartách často nachází textová pole. Pro rozpoznání textu se v praxi používá strojové učení. Pro příklad lze uvést ML Kit for Firebase od Google. Ten, kromě rozpoznání textu, umí rozpoznat obličeje, čárové kódy, nebo identifikovat objekty v obraze. Na základě databáze slov pak detekovaný text rozdělí na slova [12].

(a) Detekce mřížky hrací desky Scrabble. (převzato z [21])



(b) Nalezení žluté dráhy pro figurky deskové hry Sázky a dostihy. (převzato z [6])

Ze subsekce víme 4.3.1, že hry mohou obsahovat hrací desky, které určují rozpoložení herních prvků na ploše (viz. Scrabble). Obdobně mohou rozpoložení udávat karty. Jestliže vyložíme karty hry Krycí jména na stůl, většinou je položíme tak, že vytvoří čtverec, nebo obdélník. Vizuálně dostáváme pomyslnou mřížku, kde každá karta je pole konkrétního řádku a sloupce. Takto rozložené karty mohou být detekovány obdobným způsobem jako hrací plochy. Na základě hranového detektoru 5.3.2, nebo detekce barevných ploch 5.2.2, mohou být jednotlivé karty detekovaný a určena pomyslná mřížka pro určení pozice [12].

4.3.3 Kostky

Při rozpoznávání kostek chceme jako hlavní informaci získat číslo na horní straně kostky. Pro zjištění kruhů na kostce se používá několik postupů.

Uvažujeme-li jednobarevnou kostku s kontrastní barvou kruhů, lze filtrovat pouze oblast těchto kruhů a následně spočítat, kolik takových kruhů na kostce existuje. Pro rozpoznání kruhů lze použít přístup s využitím aplikace Houghovy transformace kružnice [15], nebo na základě kontur 5.3.3. Při použití houghovy transformace kružnice musíme vymezit rozměry kruhů, aby detekce byla co nejpřesnější. V případě přístupu pomocí detekce kontur, porovnáváme danou konturu buď z elipsou, nebo kruhem, který ji obklopuje. Pokud jsou rozměry kontury a kruhu (popř. elipsy), který ji obklopuje, podobné, můžeme zřejmě prohlásit danou konturu za kruh čísla kostky. Po sečtení těchto kontur dostaneme výsledné číslo na kostce.

Další vlastností detekovanou v kostkách u některých her, je barva kostky. Jako příklad lze udat Sagradu, u které je barva kostky klíčovou vlastností pro správné vyhodnocení hry strojově. Barvy kostek se detekují pomocí barevných modelů 5.2.2. Na základě rozsahů jednotlivých barev určujeme, zda-li je kostka červené, či jiné barvy. Zároveň lze pomocí barevných rozsahů detekovat kostky v obraze.

4.3.4 Jiné specifické prvky hry

Kromě již dříve zmíněných prvků mohou deskové hry obsahovat další specifické a mnohdy pro danou hru ojedinělé prvky. Jejich rozpoznání vyžaduje specifické řešení pro daný případ.

V některých případech je zapotřebí detekovat pozice figurek na hrací ploše. Pokud jsme detekovali herní políčko, na kterém se figurka může nacházet, lze zjistit, zda-li se na něm figurka nachází skrze odlišnou barvu figurky na pozadí, nebo-li na políčku. Jestliže máme například červené pole, na němž se nachází figurka, lze v tomto poli detekovat hrany, nebo jinou barvu, než kterou má pole. V takovém případě lze říci, že se zde nachází figurka, nebo jiný herní objekt [6].

Kapitola 5

Počítačové vidění

Počítačové vidění je odvětví výpočetní techniky a vývoje software zabývající se vytvářením zařízení schopných získávat informace ze zachyceného obrazu [25]. V následujících sekcích budou popsány volně dostupné knihovny, které implementují základní metody potřebné pro rozpoznání obrazu, nad jejichž využitím bude uvažováno v rámci této práce. Dále tato kapitola slouží k přiblížení a vysvětlení základních metod a technik rozpoznání obrazu, jež budou použity pro vytvoření podpory deskové hry.

5.1 Volně dostupné knihovny

Mezi volně dostupné knihovny řadíme takové knihovny, které mají volně dostupný zdrojový kód. Jejich silné stránky jsou zejména počet externích modulů, díky čemuž lze použít tyto knihovny v mnoha odvětvích a při řešení rozdílných problémů při implementaci. Většina dostupných knihoven obsahuje minimálně metody pro zpracování obrazu. Liší se pouze implementací, rychlostí a programovacím jazykem. K dispozici jsou však i knihovny implementované ve více jazycích. V následujících podkapitolách jsou sepsány jedny z hlavních knihoven.

5.1.1 OpenCV a BoofCV

OpenCV je knihovna, která poskytuje široké množství metod. Kromě těch pro zpracování obrazu, umožňuje i kalibraci kamery, strojové učení a vykreslování obrazců, nebo grafů. Knihovna disponuje podrobnou dokumentací a lze se opřít i o tištěnou knihu. Výhodou této knihovny je implementace v několika programovacích jazycích (Java, C++, Python) [20][4].

Na stejném principu funguje i knihovna BoofCV. Na rozdíl od OpenCV je primárně zaměřena pouze na práci s rozpoznáním obrazu, sledování objektů a kalibraci obrazu. Dále je třeba říci, že BoofCV je implementováno pouze v jazyce Java [2].

5.1.2 Google ML Kit

K dispozici vývojářům je softwarová vývojová sada (dále SDK) ML Kit[1] na infrastruktuře Firebase[2] od Googlu. Jedná se o strojové učení, které lze využít pro detekci obličejů, textu, aj. Při tvorbě aplikace s ML kitem lze SDK využít přímo na zařízení, nebo v cloudu. Většina

[1] Dostupné z https://firebase.google.com/docs/ml-kit
[2] Dostupné z https://firebase.google.com/

metod sady je uzpůsobena obecné detekci, kterou lze aplikovat na základní rozpoznání obrazu, nebo objektů. Zároveň lze využít vlastních modelů s pomocí TensorFlow[3].

5.1.3 CraftAR

CraftAR je sada pro rozpoznání obrazu od společnosti Catchoom[4]. Jedná se o serverové řešení, ve kterém jsou data zaslána na server pomocí specifické API. Zde jsou provedeny požadované úkony. Společnost zaručuje přesnost i u obrazu v prostředí s nekvalitními podmínkami např. světelné podmínky, nekvalitní obraz [5].

5.2 Barevné modely použitelné v Sagradě

Pro potřeby Sagrady je nutné rozpoznat, zda-li slot, nebo kostka mají konkrétní barvu. Informace o barvě v obrazu se dá vyjádřit několika barevnými modely. Barevný model je matematický model popisující barvy na základě podílu jednotlivých složek, kterými mohou být vybrané základní barvy nebo jiné parametry [24]. Barevné modely lze rozdělit na dva typy podle způsobu míchání barev[16]:

1. **Aditivní míchání barev** - Složením tří základních barev při jejich maximální intenzitě získáme bílou barvu. Typické pro práci s monitory, displeji, kamerami.

2. **Subtraktivní míchání barev** - Složením tří základních barev při jejich maximální intenzitě získáme barvu černou. Typické pro práci s tiskárnami, plotry.

V následujících subsekcích budou popsány modely, které by mohli posloužit ke správné detekci barev v obraze.

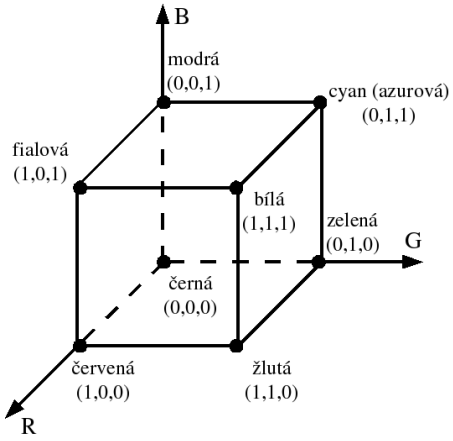
5.2.1 RGB model

RGB model je základním barevným modelem v oblasti počítačové techniky. Jedná se o aditivní typ modelu (viz. 1), jehož tři základní složky jsou **červená**(R), **zelená**(G) a **modrá**(B). Tento model si lze jednoduše reprezentovat na krychli. Na obrázku 5.1 je možné vidět, že tato reprezentace barevného modelu zobrazuje kromě RGB modelu také jeho doplňkový model CMYK. Zároveň je zde viditelný vektor mezi bílou a černou barvou. Při zpracování obrazu se RGB model používá při převádění obrazu na Grayscale (viz. 5.3.1).

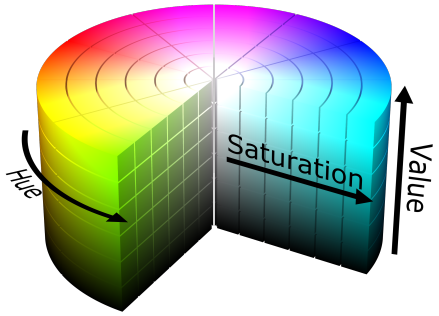
5.2.2 HSV model

HSV je uživatelsky orientovaný barevný model. To znamená, že můžeme intuitivně volit mezi jednotlivými kanály. Volit můžeme sytost (saturation), tón (hue) a jas (value). V této práci bude HSV model využit pro získání barevných masek objektů, z nichž se posléze získají sloty, popř. kostky (viz. 6.2). Masky lze získat nastavením rozsahu jednotlivých kanálů a nechat tak v obraze pouze požadovanou barvu. Rozsah jednotlivých barev můžeme vidět na následujícím obrázku 5.2.

[3]Dostupné z https://www.tensorflow.org/
[4]Dostupné z https://catchoom.com/



Obrázek 5.1: Reprezentace RGB modelu. Převzato z: https://cs.wikipedia.org/wiki/RGB



Obrázek 5.2: Rozsah barev HSV modelu. Převzato z: https://cs.wikipedia.org/wiki/HSV

5.2.3 CIE L*a*b* model

Jedná se o model, který je definován Mezinárodní komisí pro osvětlování (CIE). Barvy jsou zde vyjádřeny pomocí tří složek. Světelnost (L*) definuje hodnotou od $0 - 100$ zda-li se barva blíží bílé, nebo černé. Prvek a* definuje barvu v rozsahu mezi zelenou a červenou barvu na stupnici -128 (zelená) až 127(červená). Stejnou stupnici má i prvek b*, ten však definuje barvu v rozsahu mezi modrou a žlutou barvou. Výhoda L*a*b* modelu spočívá v přesnější definici barev tak, jak jej vnímá lidské oko.

5.3 Metody použitelné v Sagradě

Následující podkapitoly se věnují metodám, nebo postupům , které by se daly teoreticky aplikovat na podporu deskové hry Sagrada. Některé z těchto podkapitol budou v pozdější fázi vysvětleny v kontextu návrhu a implementace v podpoře pro Sagradu.

5.3.1 Grayscaling

Grayscaling je jedna z metod redukce barevného prostoru ze tříprvkového modelu na jednoprvkový model stupně šedi. Většinou se jedná o převod na 8 bitové hodnoty, tudíž redukovaný obraz může nabývat 256 úrovní šedi [16]. Pro převod se používá empirický vztah

5.1.

$$I = 0.299R + 0.587G + 0.114B \tag{5.1}$$

V této rovnici neznáme R, G, B reprezentují jednotlivé prvky RGB modelu 5.2.1, ve kterém je většina vstupních obrazů.

5.3.2 Detekce hran

Detekce hran je jednou ze základních mechanik pro zpracování obrazu. Při jejich detekci hledáme hranové body, které vznikají díky nespojitostem v normále k povrchu, hloubce, nebo například osvětlení [10]. Algoritmy pro detekci hran pracují s konvolucí filtru (zejména Sobelova operátoru) s původním obrazem, kde touto konvolucí dochází k nalezení hraničních bodů.

Ze základních algoritmů pro detekci hran můžeme vyzdvihnout Sobelův hranový detektor. Ten využívá Sobelova operátoru ke zvýraznění hraničních pixelů v obraze. Operátor je v rozměru minimálně $3x3$ pixelů (viz. 5.3) a vždy obsahuje nulový řádek, či sloupec, který filtruje hrany. Detektor zvýrazní oblasti s vysokou frekvencí, která koresponduje s hranou[19].

1	0	−1
2	0	−2
1	0	−1

(a) S_x

1	2	1
0	0	0
−1	−2	−1

(b) S_y

Obrázek 5.3: Příklad Sobelova operátoru. Převzato z: http://statnice.dqd.cz/mgr-szz:ingra:7-gra

Cannyho hranový detektor, který lze považovat za jeden z nejpoužívanějších. Nejdříve je v grayscale obraze odstraněn šum. Poté je aplikován Sobelův operátor s daným rozměrem masky (např. $3x3$ nebo $5x5$). Poté, co je pomocí Sobelova operátoru zjištěn gradient [5], jsou odebrány body, které nejsou lokálními maximy. Takhle nalezneme přesné hrany. Na tyto hrany je posléze aplikováno dvojité prahování. První hodnota prahu určuje pixely, které jsou s jistotou určeny za hranové body, pokud mají gradient vyšší než prahová hodnota. Druhá hodnota prahu určí za hranové body takové pixely, jejichž hodnota gradientu je vyšší než hodnota druhého prahu a zároveň se nachází v okolí pixelu, který je definován jako hranový bod z první hodnoty prahu. Tímto nám vzniknou souvislé hrany[9].

5.3.3 Detekce kontur

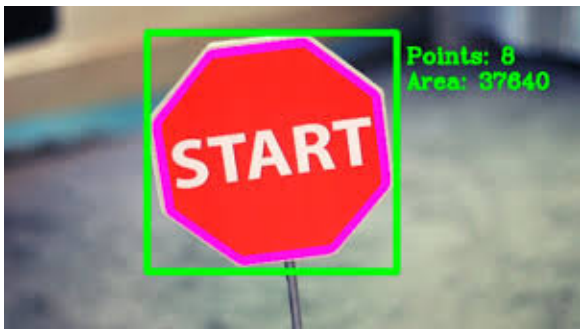
Kontury hrají velkou roli v získávání informací o objektech. Samotná detekce hran nám pouze získá hranové body, kontury na druhou stranu sloučí tyto body do křivek. Aby byly kontury získané, nemusí vzniknout pouze z hraničních bodů. Mnohdy postačí pouze obraz na který bylo použité prahování 5.3.1 a je tedy zřejmá změna jasu, textury, nebo uskupení. Tyto rozdílné detekce kontur lze rozdělit do několika kategorií, které obsahují různé metody jak získat kontury.

[5] Směr růstu. Jedná se o diferenciální operátor, jehož výsledkem je vektorové pole.

Detekce kontur přes pixely je založená na rozpoznání, zda-li právě zkoumaný pixel patří nějaké kontuře. Hlavní vlastností je hledání nespojitostí v intenzitě prahovaného obrazu. Tak lze zjistit, zda-li pixel patří k nějaké kontuře, nebo se jedná o pixel nové kontury.

Dalším postupem, jak získat kontury, je spojena s detekcí hran. Zde jsou z obrazu nejdříve získány informace o hranách, na které je posléze aplikován detektor kontur. To je docíleno shlukováním hranových bodů na základě spojitosti bodů v dané oblastí [7].

Ze získaných kontur posléze lze dostat informace o objektech. Tyto informace můžeme dále zpracovávat jako informace o velikosti, nebo konvexitě kontury. Zároveň pomocí aproximace vektoru bodů kontury lze zjistit, zda-li objekt odpovídá tvaru obdélníku, nebo trojúhelníku, či kruhu 5.4.



Obrázek 5.4: Detekce kontur a získávání informací o kontuře. Převzato z: https://www.-youtube.com/watch?v=Fchzk1lDt7Q

5.3.4 Binární morfologie

Binární morfologie je založená na matematické morfologii. Hlavní myšlenkou je získání znalostí z relace obrazu a jednoduché malé sondy (tzv. strukturní element) s jednoznačným tvarem. Pro každý pixel se ověřuje, zda-li sonda odpovídá, či neodpovídá lokálnímu tvaru v obraze [11] [14].

Eroze a dilatace

Eroze a dilatace jsou dvě základní operace binární morfologie. Obě tyto operace jsou používané jak samostatně, tak v rámci složitějších morfologických operací. Jejich účelem je zvýraznit objekty nacházející se v obraze.

Erozi lze vyjádřit jako složení dvou bodových množin s využitím rozdílů vektorů podle rovnice 5.3. Z pohledu zpracování obrazu bychom mohli říct, že pixel v binárním obraze (hodnota pixelu 1 nebo 0) má hodnotu 1 pouze v případě, že okolní pixely, v závislosti na velikosti strukturního elementu, mají také hodnotu 1 [20]. V takovém případě jsou pixely na okraji objektu nastaveny na hodnotu 0 a objekt je tak redukován 5.5c.

Dilatace pracuje opačným principem než eroze a je tím duální transformací k erozi. Jedná se o operaci, kdy se skládají body dvou množin pomocí vektorového součinu podle rovnice 5.2. Pixel v binárním obraze má hodnotu 1 v případě, že alespoň jeden pixel v okolí, opět v závislosti na velikosti strukturního elementu, má hodnotu 1 [20]. Objekt v obraze, se při dilataci zvětšuje, jak je možné vidět na obrázku 5.5b. Samostatně se dilatace používá k zaplnění malých děr v objektu a jako základ složitějších morfologií [14]. Zároveň se touto operací zvětšují objekty. Dilatace se většinou aplikuje v návaznosti na erozi, kdy objekt

je nejdříve zredukován a jsou odstraněné okolní fragmenty, které v obraze nepotřebujeme. Následuje dilatace, která opět zvětší objekty, ale nevrátí do binárního obrazu fragmenty drobných předmětů, nebo kazů v obraze.

$$X \oplus B = \{d \in E^2; d = x + b, x \in X, b \in B\}. \tag{5.2}$$

$$X \ominus B = \{d \in E^2; d + b \in X, \forall b \in B\}. \tag{5.3}$$

Otevření a uzavření

V předchozích odstavcích 5.3.4 byly popsány dvě základní binární operace. Jejich kombinací získáme dvě důležité operace ve zpracování obrazu. Ve výsledku tyto dvě operace pomáhají zvýraznit objekty a redukovat v obraze nechtěné fragmenty [14].

Při *otevření* dochází v binárním obraze k redukování, tudíž aplikaci eroze, která je následovaná dilatací. Obě tyto operace jsou prováděny se stejným strukturovaným elementem. Objekty v obraze jsou tedy nejdříve redukovány a zbaveny nepotřebných fragmentů. Tento proces je posléze následován zvýrazněním objektů do jejich původní velikosti (viz. 5.6a). Matematicky lze otevření popsat rovnicí 5.4.

Uzavření, stejně jako v případě dilatace a eroze, je duálním operátorem k morfologickému otevření. Z toho nám tedy plyne, že operace eroze a dilatace jsou zde provedeny opačným způsobem. Nejdříve je aplikována dilatace, kterou následuje eroze. Z binárního obrazu jsou tedy odstraněny drobné díry v objektech. Tyto objekty jsou posléze navráceny do původní velikosti. Matematicky lze uzavření popsat rovnicí 5.5.

$$X \circ B = (X \ominus B) \oplus B. \tag{5.4}$$

$$X \bullet B = (X \oplus B) \ominus B. \tag{5.5}$$

(a) Základní obraz pro úpravu (b) Dilatace obrazu (c) Eroze obrazu

(a) Otevření obrazu (b) Uzavření obrazu

Kapitola 6

Návrh aplikace

Jak již bylo zmíněno v kapitole 1, aplikace slouží pro vyhodnocení bodů hráče a kontrolu, zda-li byla pravidla splněna. Uživatel by měl pouze vyfotit kartu klíče, výslednou kartu s kostkami, a následně obdržet výsledek. Kromě toho bude moci v průběhu hry zjistit, zda-li splňuje jeho vitráž pravidla.

Hlavní myšlenkou této aplikace je přinést řešení založené pouze na rozpoznání obrazu přímo na zařízení bez použití serverových prostředků a také bez strojového učení, to je odůvodněné malým počtem karet klíče, jež hra obsahuje. Zároveň se při návrhu chceme vyhnout použití porovnání fotek se vzorovými fotkami, které omezují možnosti detekce případných rozšíření.

Samotný návrh se dělí na několik sekcí, které se věnují hlavním problémům dílčích detekcí jak karet klíče, tak vitráží s kostkami.

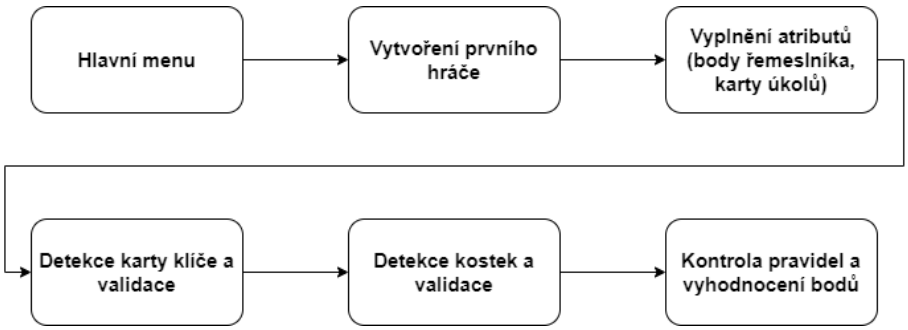
6.1 Struktura

Aplikace má uživateli sloužit jako kontrola pravidel a nástroj pro výpočet bodů. Je tudíž nutné detekovat klíč a kostky, bez nichž nelze posoudit pravidla a vypočítat body. Aplikace bude tedy postavena tak, aby uživatel zadával postupně všechny potřebné informace, ze kterých poté program může vyhodnotit výsledky.

Z hlavní nabídky se uživatel dostane do herní části. Zde lze vytvářet nové hráče a spravovat jejich data potřebná k vyhodnocení bodů a kontrole pravidel. Hra tudíž umožňuje správu nejenom jednoho hráče, ale i více hráčů na jednom mobilním zařízení. Jelikož Sagrada obsahuje dvě karty řemeslníků, které umožňují ignorovat pravidla (možnost položit kostku s jiným číslem/barvou na pozici, kde smí být položena pouze konkrétní barva/číslo), bude při vyhodnocování hráč dotázán, zda-li takovou kartu řemeslníka použil, a případně kolikrát. Tím bude zaručeno, že vyhodnocování bodů vezme v potaz veškeré scénáře, které při hraní Sagrady mohou nastat.

Hráč také bude mít všechny informace k dispozici průběžně a bude je moct upravovat. S detekcí souvisí i možnost špatně detekovaných informací. Je tedy na místě do aplikace přidat možnost validace detekce klíče a kostek. Ta probíhá po vyfocení karty a provedení detekce. Uživateli budou grafickou formou zobrazeny informace o klíči, popř. rozmístění kostek. V případě špatné detekce, bude možnost opětovného vyfocení, nebo ručního vložení informací o kartě klíče, popř. kostek na hrací ploše. Tyto možnosti budou k dispozici zejména z důvodů situací, kdy hráč fotí pod špatným úhlem, popř. se špatnými světelnými podmínkami např. odlesk zdroje světla na kartě, rozmazaná fotografie, a jiné.

Jednou z dalších možností bude ověření pravidel v průběhu hry. Jelikož vyhodnocení pravidel je založeno na informacích z klíče a kostek, bude jedno, zda-li je hra u konce, nebo ve svém průběhu.



Obrázek 6.1: Postup použití aplikace

6.2 Detekce karty klíče

Jednou z možností jak předat aplikaci informaci o typu klíče, by bylo vytvořit pole klíče přímo v aplikaci a následně skrze seznam v uživatelském rozhraní nastavit kterou vitráž hráč používá. Tento proces je však nepraktický. I když hra obsahuje pouze 24 vitráží, je možné, že se objeví rozšíření pro hru, která přinesou nové vitráže. Ty bude však nutné opět implementovat do aplikace. Z praktického hlediska dává větší smysl najít postup, při kterém lze získat informace z libovolné vitráže stejným způsobem, bez ohledu na základní vitráž, či tu z rozšíření. Toho lze dosáhnout při použití základních metod pro rozpoznání obrazu.

6.2.1 Nalezení orientačního bodu

Aby mohly být jednotlivá pole s vlastnostmi o klíči správně identifikována, je nutné předem v obraze najít záchytný bod, který nám určí, kde se v okolí karty nacházíme. Na kartě s klíčem se můžeme orientovat pomocí několika bodů:

1. **Kruhy obtížnosti klíče** - V pravé spodní části karty se nacházejí bílé kruhy, které určují obtížnost klíče a zároveň informují hráče, kolik kamenů řemeslníka má k dispozici.

2. **Název karty klíče** - Ve spodní části karty se nachází text, který obsahuje název karty klíče.

Obě dvě výše popsané oblasti mohou určit, kde se karta nachází a zároveň se podle nich může spustit detekci slotů na specifikovanou oblast nad těmito orientačními body.

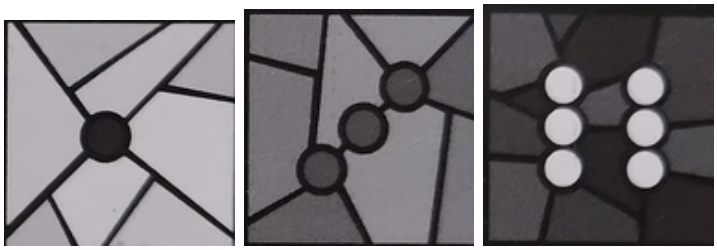
Jelikož kruhy obtížnosti klíče jsou v přímém kontrastu s černou barvou karty, lze je jednoduše detekovat a zároveň tak najít pravý krajní bod karty od nějž lze detekovat sloty karty klíče. I při této detekci však mohou nastat problémy, např. odlesk světla v okolí kruhů, čímž může dojít ke ztrátě informace, nebo hledání karty v prostředí, které může obsahovat položky podobné těmto kruhům. Odlesku světla nelze nijak zabránit, ale pro účely minimalizace prostředí bude uživateli připravena na fotoaparátu oblast, do které se musí karta vlézt. Tato oblast určuje prostor, ve kterém probíhá detekce.

Další možností orientace je název karty. Pro jeho detekci je třeba využít algoritmů pro detekci textu v obraze. Většina efektivních algoritmů však pracuje se strojovým učením, kterému se v této práci chceme vyhnout. Z tohoto důvodu není tato možnost brána v potaz.

6.2.2 Detekce pole

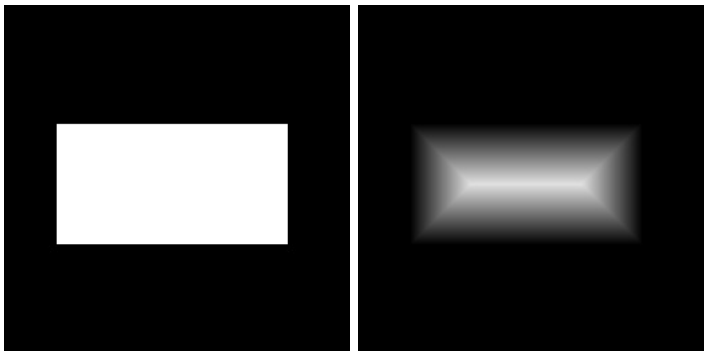
Jednotlivá pole klíče jsou tvořena buď barevnými čtverci (červená, modrá, zelená, žlutá a fialová), nebo číslicemi $1 - 6$. Pro detekci barevných slotů využijeme HSV barevný model 5.2.2. Tím nám vzniknou barevné masky jednotlivých barev, ze kterých poté vybereme jenom kontury splňující vlastnosti slotu 8.1.3.

Číselný slot je v kartě vyobrazen jako malá vitráž. Základní detekce černé a bílé barvy zde bude fungovat pouze pro číslice $6, 5, 4$, kde kontury bílých kruhů jsou jasně spočitatelné jednoduchým způsobem. Pro tato čísla bude postačující základní prahování pro oddělení bílé barvy od tmavších odstínů. Číslice $3, 2, 1$ jsou ve slotech, která mají šedý, až černý odstín. Musí projít komplikovanější detekcí, jelikož kruhy určující číslici navazují na rám menších sklíček vitráže viz. 6.2.



Obrázek 6.2: Tři typy slotu číslice

Pro detekci těchto číslic lze použít distanční transformaci. Čím více mají pixely ve svém okolí pixely v daném rozsahu prahu, tím je větší pravděpodobnost, že se nenacházejí na okraji objektu, ale v jeho středu. Jsou tudíž zvýrazněny zejména středy objektů v daném číselném rozsahu viz 6.3. Je třeba podotknout, že zejména v OpenCV algoritmus pracuje pouze s jednokanálovým zdrojem, zejména s grayscale obrazem, nebo jedním kanálem barevných modelů.



Obrázek 6.3: Aplikace distanční transformace na jednoduchý objekt

6.3 Detekce kostek

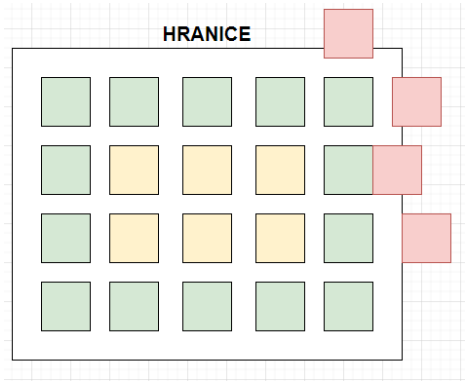
Kostky lze detekovat přes barvy stejným způsobem jako sloty barvy na kartě klíče 6.2. Po nalezení pozice pak lze pomocí bílé barvy detekovat počet bílých kruhů a tím pádem získat informaci o čísle na kostce. Zároveň je nutné určit pozici kostky, resp. její řádek a sloupec. Můžeme přes rozpoznání černé barvy najít oblast, ve které se nachází kostky. Nebo předem určíme hráči oblast v rozhraní kamery, kam se musí kostky vlézt.

Při nalezení oblasti pomocí filtru černé barvy a zjištění, zda-li se zde nachází kostky, lze přistoupit přes některý z barevných modelů. Např. u modelu HSV použijeme V-kanál, u kterého určíme číselný rozsah pro získání masky. Problém u tohoto způsobu jsou světelné podmínky. Při nich může u vyšší světelnosti docházet k odlesku světla od tmavých barev hrací vitráže a detekce oblasti nebude přesná. Na obrázku 6.4a lze vidět, že černé pozadí kolem kostek je přesvícené. Pokud nastavíme prahovou hodnotu tak, aby i toto pozadí bylo detekováno, může se stát, že oblast kostek bude příliš veliká a pozice kostek nepřesně vypočítána.

Druhou možností je využít podobné omezení, které se použije při detekci orientačního bodu na kartě klíče 6.2.1. Opět bude uživatel fotit vitráž tak, aby kostky ležely v ohraničené oblasti, na kterou se aplikuje detekce.

Za pomocí této hranice lze zjistit pozice kostek. Obvodové kostky jsou takové, které nemohou mít vertikálně, či horizontálně souseda, a to alespoň v jednom ze směrů. To lze vypočítat podle pozice pixelů, kde nemůžeme vedle kostky přidat další stejně velkou kostku, jelikož bychom přesáhli danou hranici v pixelech (viz. 6.4b), kde zelené obdélníky jsou obvodové kostky. Červené obdélníky představují sloty, které již nepatří do vymezené oblasti a tudíž se zde nic nenachází.

(a) Odlesk světla na hracím rámu



(b) Nalezení krajních kostek

Následně se kostkám přiřadí číslo řádku a sloupce. Pokud jsou některé sloty nevyplněné, dopočítá se rozdíl mezi kostkou, které má být aktuálně přiřazeno číslo a následující kostkou, nebo hranicí. Jestliže rozdíl je větší než velikost slotu, předpokládá se, že se zde nachází volný slot, do kterého nebyla vložena kostka. Podle tohoto rozdílu se poté doplní počet prázdných slotů, kterým není přiřazena pozice, ale lze díky těmto slotům přiřadit správné číslo kostkám (viz. 8.2.3).

6.4 Vyhodnocení

Hráči musí, pro vyhodnocení hry, kontrolovat pravidlo po pravidle pro každou kostku na hrací kartě. Tento proces je velmi zdlouhavý a může vést na chyby z nepozornosti. Aplikace vyhodnocuje pravidla na základě informací ze vzorové karty, tudíž o pozicích v poli 4×5 a pozicích kostek na hrací kartě. Po kontrole veškerých pravidel přiděluje body, nebo informuje o pozicích, které porušují pravidla. Pokud jsou veškerá pravidla splněna, lze přiřadit k hráči výsledný počet bodů a uzavřít jeho kartu.

Aby bylo možné přidělit body, je nutné předat aplikaci informace o kartě osobního úkolu a společného úkolu. Tyto informace budou aplikaci sděleny přes seznamy, ze kterých hráč vybere název příslušného úkolu. Program poté na základě vybraných karet vybere, které úkoly má vykonat a podle implementací těchto úkolů vypočítá body.

Při kontrole pravidel může dojít k jejich porušení. Uživatel by měl být informován o pravidle, jež bylo porušeno a o pozici výskytu tohoto narušení. Pro tyto potřeby může posloužit systém, který bude uživateli zprostředkovávat informace o jeho kartě a kostkách.

6.5 Zadávání informací o kartách

I když aplikace primárně slouží pro rozpoznání herní vitráže, mohou světelné podmínky znehodnotit obraz a detekce karty klíče, nebo detekce kostek. Za tímto účelem bude aplikace obsahovat aktivitu, kde bude možné vyplnit informace o klíči, nebo kostkách ručně. Pro jednotlivá pole bude možné, v závislosti na typu (kostka, klíč), vložit informaci o barvě, nebo čísle, případně obojím. V případě, že detekce proběhne částečně dobře a jenom některé položky budou nepřesně detekovány (například políčko s informací o čísle 6 není detekováno kvůli odrazu zdroje světla), může uživatel pouze pozměnit informaci na konkrétní položce.

Zadávací systém zároveň slouží jako informační systém, který uživateli poskytuje graficky přehledné znázornění jednotlivých polí karty klíče a kostek ve vitráži. V případě, že budou porušena některá pravidla Sagrady, bude si moct uživatel, u konkrétních polí, zjistit porušené podmínky hry.

Kapitola 7

Uživatelská rozhraní

Tato práce se věnuje podpoře Sagrady na mobilních zařízeních OS Android. Je tedy na místě, aby se práce, kromě detekce a vyhodnocení pravidel deskové hry Sagrada, věnovala i aplikaci, na které se veškeré úkony budou provádět. Následující sekce se zaměřuje na uživatelskou studii a návrh jednotlivých komponent grafického rozhraní aplikace.

7.1 Uživatelská studie

Desková hra Sagrada je pro lidi ve věku od 8 let. Lze tedy předpokládat, že cílová skupina hráčů bude od věku 8 let až do seniorního věku. V cílové skupině uživatelů aplikace nehraje zásadní roli jejich zaměstnání popř. vzdělání, domnívám se tak, jelikož alespoň jednu deskovou hru jsme hráli každý, ať už v mládí, nebo v dospělosti. Na druhou stranu uživatel musí být vlastníkem zařízení s OS Android.

7.1.1 Příklady použití

- Skupina kamarádů sedí v čajovně a hraje deskovou Sagrada. Za čas strávený v čajovně chtějí odehrát co nejvíce her. Z tohoto důvodu použijí aplikaci pro rychlejší vyhodnocení výsledků.
- Rodina při nedělním odpoledni zasedne ke stolu a hrají Sagradu. Děti používají aplikaci pro kontrolu pravidel, zároveň vyhodnocují body svým rodičům.
- Pár pozve své přátele na večeři. Po ní se hraje desková hra Sagrada. Hráči nechtějí zdlouhavě vypočítávat pravidla a používají aplikaci.
- Hráč si není jistý, zda-li má své kostky po pár tazích správně umístěné. Aby nepřišel o mnoho bodů zkontroluje pravidla pomocí aplikace.

7.1.2 Situace užití

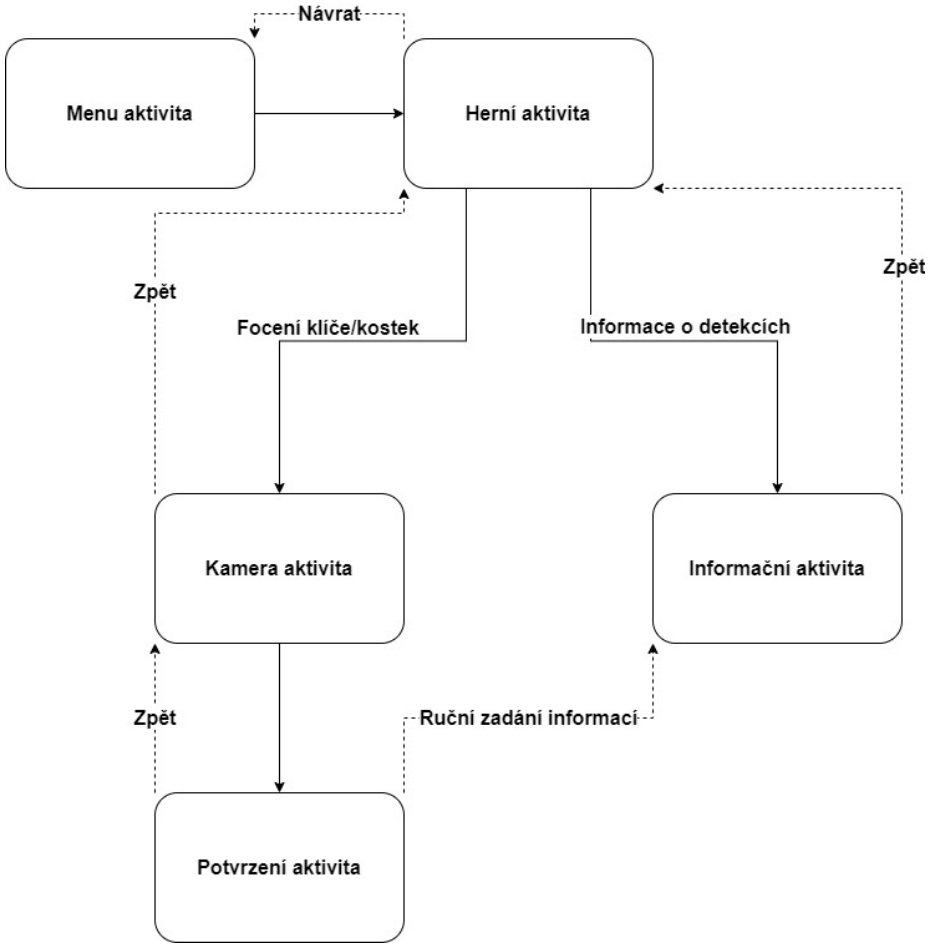
- Detekce herního klíče
- Detekce kostek na hrací kartě
- Vyhodnocení správnosti pravidel
- Správa jednoho nebo více hráčů v rámci jednoho mobilního zařízení

- Výběr osobního a společného úkolu pro hráče
- Výpočet bodů v závislosti na herním klíči, kostkách a kartách úkolů
- Výběr vítěze po výpočtu bodů všech hráčů

7.2 Návrh uživatelského rozhraní

Uživatel aplikace pro podporu Sagrady musí kromě focení a validace detekcí vyplňit doplňující informace pro přesné vyhodnocení bodů. Jedná se zejména o vyplnění karty osobního a společného úkolu a počet zbývajících kamenů řemeslníka. Při návhru aplikace tedy bude kladen i důraz na minimalizaci operací, které hráč musí v aplikaci vykonat.

Pomocí aktivit, které byly popsány v 3.2.1 lze rozdělit návrh rozhraní na jednotlivé aktivity. V rámci každé aktivity je omezené množství operací. Přechod mezi aktivitami je jednoduchý a jasný, navrženo tak, aby se nevytvářely komplikované a složité návaznosti mezi více aktivitami. Jak je možné vidět z obrázku 7.1, z menu aktivity se přechází do hlavní herní aktivity. Aktivity zároveň umožňují návrat do předchozí aktivity v případě, že uživatel přešel do aktivity, do které nechtěl.



Obrázek 7.1: Diagram aktivit

7.2.1 Herní aktivita

Herní aktivita spravuje veškeré hráče. Je možné zde upravovat veškeré informace, které hráč musí vyplnit, aby bylo možné správně vyhodnotit body hry Sagrada. Hlavním prvek této aktivity je kruhová informační oblast s počtem bodů. Tato oblast informuje uživatele, zda-li může přistoupit k vyhodnocení nebo ne. Červená výplň značí, že uživatel nemá vyplněné veškeré informace, nebo kontrola pravidel neproběhla správně (viz. 7.2a). V případě správně vyplněných dat se ukazatel obarví do zelené (viz. 7.2b) a body jsou automaticky vypočítány. Zároveň je na obrázcích 7.2a a 7.2b viditelné, kde se nacházejí informace o kartě osobního úkolu, společného úkolu a počtu kamenů řemeslníků. U těchto informací je zároveň možné upravovat hodnoty jednoduchým stisknutím grafických tlačítek značících změnu informací. Hráč kromě toho může pomocí tlačítka s foťákem detekovat kostky, nebo klíčovou kartu. Z informačního tlačítka poté přejít do aktivity s informacemi o těchto detekcích.

(a) Aktivita hráče bez vyplněných informací

(b) Aktivita hráče v situaci, kdy lze vyhodnotit body

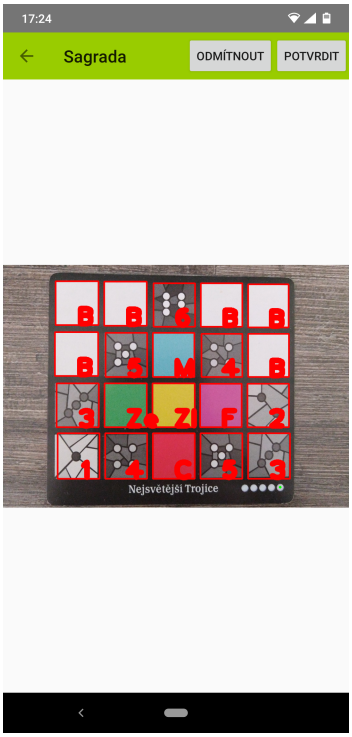
Herní aktivita kromě informací o hráči zároveň demonstruje správu jednotlivých hráčů. Při porovnání obrázků 7.2a a 7.2b lze rozpoznat počet hráčů účastnících se hry. Obrázek 7.2a zobrazuje hru, ve které se nachází pouze jeden hráč, naopak 7.2b zobrazuje hru s více hráči. Zde je patrné, že každý hráč má svoji vlastní záložku a orientace v hráčích je přehledná.

7.2.2 Aktivity rozpoznávání obrazu

Pro potřebu detekce karty klíče a kostek je navržena aktivita detekce. Jejím hlavním účelem je poskytnout uživateli rozhraní kamery s vymezenou oblastí (viz. 7.3a), nad kterou bude spuštěna detekce. Kromě vyfocení musí být schopen validovat tyto detekce. Kamera po vyfocení zašle obrázek aktivitě s náhledem na fotku, ve které se zobrazí obraz s informacemi o detekovaných položkách ve vyfoceném obraze. V tomto případě poté hráč má možnost

jednoduše určit, zda-li je kostka/klíč správně detekován. Tuto validaci provádí uživatel aplikace skrze dvě tlačítka („Odmítnout“ a „Potvrdit“) v horním panelu aktivity. To je demonstrované na obrázku 7.3b. Pokud uživatel validuje detekci, je navrácen zpět do herní aktivity. V případně že detekce není validní, může opět obraz vyfotit, nebo ručně zadat informace.

(a) Aktivita s rozhraním kamery



(b) Aktivita s aplikovanou detekcí na kartu klíče

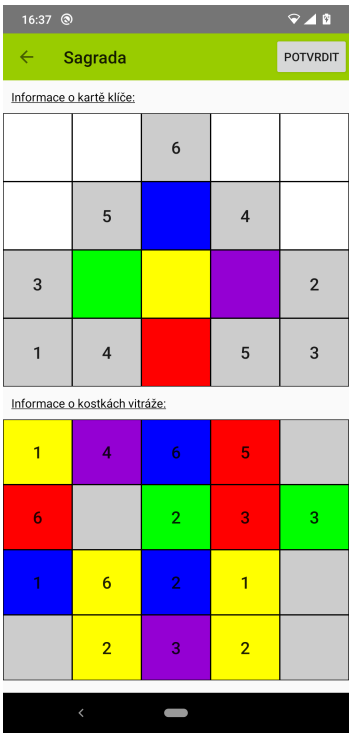
7.2.3 Informační aktivita

Uživatel musí mít k dispozici informace o detekovaném klíči a kostkách. Tato aktivita neslouží pouze pro zobrazení detekovaných informací, ale zároveň jako aktivita pro ruční zadávání informací. Je-li například jedno políčko herního klíče špatně detekováno, může uživatel v informačním aktivitě vyplnit přesnou informaci v daném políčku. Dále zde může uživatel zjistit informace o porušených pravidlech. Je-li na konkrétním poli porušeno pravidlo hry, je pole graficky zvýrazněno a dlouhým podržením políčka se zobrazí informace o porušeném pravidle. Informační aktivita graficky zobrazuje dvě karty.

První karta graficky znázorňuje rozložení polí herního klíče (viz. horní polovina obrázku 7.4. Jednotlivá pole mají buď konkrétní barvu pole karty klíče, nebo číslici, jež určuje jaké číslo kostky zde může být položeno. V případě, že nebyla karta klíče detekována, jsou zobrazena pouze šedá pole. Uživatel s těmito poli může operovat. Po kliknutí na pole se zobrazí nabídka, ve které si vybere, zda-li má být pole barvou, nebo číslem. Po výběru je poli tato hodnota přiřazena a změní se grafická prezentace konkrétního políčka.

Druhá karta reprezentuje detekované kostky (viz. spodní polovina obrázku 7.4. Políčka karty reprezentují kostky ve vitráži, tudíž každé políčko nese informaci o barvě a čísle kostky. V případě, že na dané pozici není žádná kostka, je pole pouze šedé. Pokud uživatel

ručně doplňuje kostky do karty, zobrazí se menu s barvami, u kterých zároveň určuje číslo kostky. V této kartě se zároveň zobrazují informace o porušení pravidel. Porušuje-li kostka pravidlo hry, je v kartě zvýrazněna a dlouhým podržením příslušného políčka se zobrazí informace o pravidle, které bylo porušeno.



Obrázek 7.4: Informační aktivita s vyplněnými údaji

Kapitola 8

Implementace

Pro účely implementace jsem zvolil prostředí Android Studio (viz. 3.1.1), které obsahuje nezbytné nástroje pro překlad a krokování. Dále má rozšíření emulátoru Android zařízení, přes který lze aplikaci testovat na více verzích Android OS. Využil jsem také podpory Android Native C/C++. A to za účelem implementace metod pro rozpoznání obrazu. Knihovna OpenCV, která je v práci hlavním zdrojem metod, podporuje kromě jazyků Python a C/C++ i jazyk Java. Ten však neobsahuje implementace některých algoritmů. Pro předejití problémům, jsem zvolil jazyk C/C++.

Následující podkapitoly popisují implementace hlavních tříd. Každá z těchto tříd odpovídá jedné části ze schématu aplikace (viz. 6.1).

8.1 Detekce klíče

Detekce klíče je implementována ve třídě **PatternAnalyzer**. Dělí se do dvou hlavních metod. Metoda `CreatePatternGrid()` zahrnuje funkce pro nalezení kontrolního bodu a slotů klíče. Výstupem této metody je vektor částečných obrazů na základě detekovaných slotů (viz. 8.1.3). Metoda `GetCardPattern()` z těchto částečných obrazů získává informace o typu slotu, zda-li se jedná o číslo, nebo barvu, jak je popsáno v sekci 6.2. Jednotlivé subsekce popisují kroky celkového rozpoznání karty klíče.

8.1.1 Zpracování vstupní fotografie

Vstupní pořízená fotografie karty klíče je nejdříve zpracována funkcí `PrepGrayImg()`. Zde dochází k převedení obrazu na odstíny šedé skrze grayscaling (viz. 5.3.1. Obrazu je nejprve vyrovnán histogram intenzity pixelů, aby posléze mohlo proběhnout rozostření obrazu gaussovským šumem. Z takto připraveného obrazu lze posléze detekovat jednotlivé klíčové prvky, ze kterých posléze zjistíme potřebné informace.

8.1.2 Kontrolní bod

Z upraveného obrazu lze najít kontrolní bod. Jeho detekce je ve funkci `DetectControlPoint()`, která v šedém obraze hledá kruhové kontury. Konkrétně v obraze hledáme kruhy určující obtížnost karty klíče, jak je popsáno v sekci 6.2. Z prahovaného obrazu se detekují hrany skrze Cannyho hranový detektor (viz. 5.3.2). Poté, co získáme hrany lze jednoduše nalézt kontury, pro jejichž nalezení poslouží funkce, z knihovny OpenCV, `findContours()`. Z objevených kontur posléze filtrujeme pouze takové kontury, jejichž vlastnosti se blíží vlast-

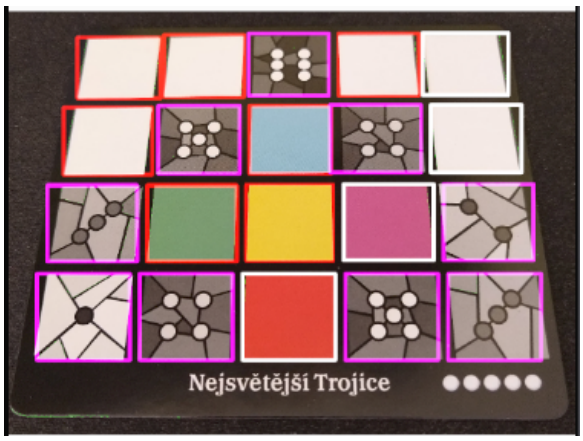
nosti kruhových objektů. Jestliže nějaký objekt projde tímto filtrem, je vložen do vektoru kruhů. Z vektoru kruhů je posléze vybrán kruhový objekt, který se v obraze nachází nejblíže pravému dolnímu okraji.

8.1.3 Nalezení slotů klíče

Pomocí funkce `ApplyColorMasks()` jsou nalezeny barevné sloty karty. Funkce z masek barev detekuje kontury obdélníkového tvaru. Masky jsou vytvářeny z rozsahů jednotlivých barev (červená, modrá, zelená, fialová, žlutá a bílá) přes jejich rozsahy definované HSV modelem (viz. 5.2). Po nalezení těchto slotů přijde na řadu nalezení slotů, které nebyly detekovány. Jedná se zejména o sloty číslic $1 - 6$.

Z prohledání všech karet klíče je patrné, že alespoň jeden slot na řádku nebude obsahovat slot s číslicí. Za tohoto předpokladu lze posléze dohledávat nedetekované sloty. Z detekovaných barevných slotů lze rekonstruovat řádky karty klíče. Funkce `RectPatternGrid()` tyto řádky rekonstruuje. Nejprve z detekovaných slotů určí referenční sloty funkce `CompleteRefRects()`. Pro každý řádek existuje jeden referenční slot. Tímto slotem se může stát slot, který je na daném řádku nejblíže kontrolnímu bodu. Rozměry referenčního slotu poslouží jako rozměry pro nedetekované sloty. Nedetekované sloty dohledává funkce `FindBlankRects()`.

Od referenčního slotu se iteruje ve dvou směrech. Nejprve vpravo od referenčního slotu, poté opačným směrem. V obou směrech se přidávají nové sloty s menším odskokem, který nahrazuje černou mezeru mezi sloty na kartě klíče. Ve směru vpravo je přidávání slotů zastaveno ve chvíli, když nový slot překročí x hodnotu, kterou má kontrolní bod. Při přidávání slotů vlevo od referenčního slotu, je přidávání slotů zastaveno ve chvíli, když počet slotů je roven požadovanému počtu slotů na řádku. V našem případě se tedy přidávání bodů zastaví, pokud slotů na řádku je 5. Prvotní detekce slouží pro určení referenčních slotů. Ty předně určují průměrnou velikost slotu a pokud jsou vedle sebe detekovány dva sloty, tak také jejich vzdálenost. Dále je také ve většině případů detekován alespoň jeden slot na každém řádku. Od těchto slotů je následně detekován zbytek řady v obou směrech na řádku. Vytváření nových slotů končí, pokud je dosaženo maximum počtu slotů na řádek.



Obrázek 8.1: Detekce slotů vzorové karty

8.1.4 Určení typu slotu

Z detekovaných slotů je poté třeba získat potřebné vzory. Tuto funkci plní metoda `GetCardPattern()`. Pro každý slot určí, zda-li se jedná o barvu, číslo od $6-5$ nebo od $3-1$ v tomto pořadí. Jelikož slot s barvou je určený jednoznačně na základě HSV modelu, není pravděpodobné, že by se v této oblasti obrazu nacházel slot jiné barvy, nebo slot s číslicí. Z tohoto důvodu je nejprve zjištěno, zda-li slot obsahuje barvu.

Detekci číslic je nutné dělit na dvě části. Funkce `DetectWhiteBlobNumber()` se prahováním a morfologickým otevřením (viz. 5.3.4) upraví obraz, aby se zvýraznily bílé kruhy v obraze. Ve výsledném binárním obraze zůstanou pouze oblasti, které reprezentují kruhy, popř. drobné dlaždice na pozadí políčka karty. Pro získání kruhů je nutné detekovat jejich kontury. Detekcí hran získáme hranové body za použití Cannyho detektoru 5.3.2. Z hranových bodů jsou následně získány kontury. Jestliže po aproximaci kontury obsahuje kontura více jak 8 bodů, je s vysokou pravděpodobností kruhového tvaru. I zde se však mohou vyskytovat objekty, které nejsou kruhy. Dochází tak zejména v případě, že políčko karty je dobře osvětleno a některé objekty pozadí se dostatečně přibližují prahovým hodnotám. Pokud objekty splňují podmínky počtu bodů kontury, proložíme takovýmito body elipsu. Je-li oblast kontury velká alespoň jako 8/9 velikosti oblasti elipsy a kontura nemá žádnou rodičovskou konturu, je pravděpodobné, že se jedná o kruhovou konturu. Takováto kontura se inkrementuje v čítači kontur. Velikost oblasti kontury je přidána do vektoru velikostí a tato velikost je přičtena k sumě velikosti pravděpodobných kontur kruhů. Po skončení procházení všech kontur v obraze, je vytvořena průměrná velikost oblasti kontur. Jelikož kruhy jsou stejných rozměrů a nevhodných objektů v obraze je méně, bude se průměrná velikost oblasti přibližovat k hodnotám kruhů. Z této průměrné hodnoty jsou tedy vybrány pouze ty oblasti, které splňují podmínku 8.1, kde *tolerance* představuje hodnotu po kterou ještě konturu tolerujeme jako pravděpodobný kruh. V našem případě pracujeme s $tolerance = 20 pixel$. Počet hodnot splňujících podmínku je brán jako počet kruhů na políčku karty. Je-li tento počet větší než 6 a menší jak 4, nejedná se zřejmě o námi hledané číslice a vracíme neúspěch. Je zřejmé, že se bude jednat zřejmě o číslice nižších hodnot.

Ve funkci `DetectLowerNumber()` je rozpoznáván obraz komplikovanějšího vzoru číslice. Obraz těchto vzorů je tvořen mozaikou vitráže, která je rozdělena pomocí černých linií a kruhů, této problematice se věnovala sekce 6.2. Po prahování a distanční transformaci obrazu v metodě `LowerNumMask()` získáme masku, ze které by mělo být možné detekovat číslice skrze metodu `DetectqLowerNumberContour()`. Tyto funkce jsou upraveny pro potřeby různých světelných podmínek. Zejména funkce `LowerNumMask()` vytváří masku na základě dvou rozdílných parametrů při prahování. Buď funkce prahuje základním prahováním, jak je definované v OpenCV skrze parametr `THRESH_BINARY`, nebo se prahuje skrze inverzní prahování `THRESH_BINARY_INV`. V druhém případě se jedná o prahování pro zjištění číslice 1, která oproti číslicím $2-3$ je nejblíže černé barvě. Zbývající číslice mají barvu do světlých odstínů šedé.

Detekce nižších číslic tak bere v potaz i možnost, že za horších světelných podmínek nebude při první detekci nalezena číslice, v takovém případě se pak opětuje detekce, ale jsou změněny hodnoty pro prahování a nastavení filtru kontur.

$$contourArea() \leq average\_area\_size + tolerance \tag{8.1}$$

8.2 Detekce hracích kostek

Pozice kostek na hrací vitráži je implementovaná ve třídě **DiceAnalyzer**. Tato třída obsahuje metody potřebné k získání informací o kostkách a jejich pozic na hrací ploše vitráže. V rámci detekce kostek se pracuje zejména s detekcí barvy na základě HSV modelu 5.2 a hledání kontur kruhových objektů. Výstupem třídy je pole obsahující položky s informacemi o kostkách.

Kostka je definována strukturou `Dice_s`. Struktura obsahuje informace o čísle, řádku, sloupci, barvě a zároveň nese informaci o obdélníků, který definuje oblast v obraze, kde se kostka nachází. Od začátku detekce kostek se tak pracuje s vektorem obsahujícím strukturu `Dice_s`. Položky jsou postupně doplňovány o informace, které jsou aktuální funkcí detekovány. Veškerá detekce kostek probíhá v rámci metody `DetectDices()`.

8.2.1 Ohraničení oblasti kostek

Pro přesné určení pozice kostky na hrací vitráži, je nezbytné definovat oblast, která vymezuje kostky. I přes určení oblasti ve které bude probíhat detekce, je možné, že uživatel vyfotí větší oblast, než tu, která vymezuje kostky. Jako horní a dolní hranice jsou pro oblast kostek nastaveny okraje vyfoceného obrázku. Dále stačí pouze nalézt levou vertikální hranici.

Metoda `getXLeftBound()` slouží pro nalezení levé vertikální hranice. Nejdříve vytvoří masku černé barvy. Po upravení této masky morfologickým otevřením 5.3.4, se hledá největší kontura v obraze. Obdélník, který tuto konturu obklopuje, zpravidla obklopuje veškeré kostky na herní vitráži, tudíž tohoto obdélníku získáme *x-hodnotu* horního nejlevějšího pixelu. Tento pixel se stane naší vertikální hranicí.

8.2.2 Detekce kostek

Detekce oblastí, kde se mohou kostky nacházet, probíhá v metodě `DetectDiceSlots()`. Zde se v obraze převedeném do HSV modelu vytvoří masky barev, které mohou kostky mít. Masky jsou poté spojeny do jednoho obrazu. Tento obraz podstoupí úpravu skrze gaussovský šum a morfologické otevření. Následně je zmenšen na rozměr $512x512$ a prahován. Z výsledného obrazu jsou získány kontury objektů, které budou považovány za kostky.

Kostky je následně potřeba seřadit. Řazení probíhá v metodě `SortSlots()`. Kostky jsou nejdříve seřazeny podle jejich *y-hodnoty*. Jelikož kostek na řádku může být nanejvýš 5, jsou seřazené kostky přiřazovány do vektoru řádku. Při vkládání se zjišťuje několik informací o možné kostce:

1. kostka následuje bezprostředně za poslední kostkou ve vektoru řádku,
2. kostka má vyšší *y-hodnotu*, než kostky na řádku,
3. kostka je poslední kostkou vektoru všech detekovaných kostek.

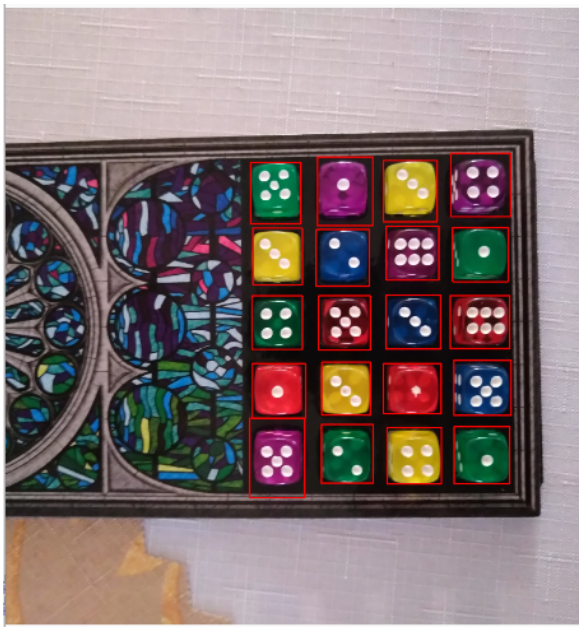
V případě 1 je kostka přidána do vektoru řádku a přistupuje se k přiřazení nové kostky do řádku. Nastane-li případ 2, jsou kostky ve vektoru řádku seřazeny a doplněny o prázdné pozice. Ty je potřebné doplnit z důvodu přesného určení řádku a sloupce kostek. Doplnění probíhá v metodě `AddBlankSlots`. Každá kostka ve vektoru řádku je zkoumána, zda-li se nachází na levém okraji oblasti kostek, nebo je-li v bezprostřední blízkosti s kostkou vlevo od ni. Pokud ani jedna podmínka není splněna, dopočítává se vzdálenost kostky od kraje oblasti, popř. od nejbližší kostky vlevo. Tato vzdálenost následně určuje, kolik kostek by

se do tohoto intervalu mohlo vejít. Počet takovýchto prázdných slotů je posléze přidán do vektoru, aby byly vyplněny prázdné oblasti. Po doplnění prázdných pozic je vektor řádku vložen do výstupního vektoru.

Splňuje-li kostka podmínku 3, vyčerpaly jsme veškeré detekované objekty. Kostka je zařazena do vektoru řádku, který je poté vložen do výstupního vektoru.

8.2.3 Přiřazení pozic

Z výstupního vektoru, o kterém je psáno v předchozí sekci 8.2.2, přiřadí metoda `DetectDices()` kostce řádek, sloupec a číslo. Nejprve u kostky určí, zda-li se jedná o kostku skrze metodu `FindDice()`. Tato metoda nejprve určí barevný typ kostky. Opět, přes rozsahy jednotlivých barev pro kostky, získá masky binární obrazy masek těchto barev. Pro každou masku spočítá počet nenulových pixelů funkcí `countNonZero()`. Následně je vybrána ta barva, jejíž počet nenulových pixelů je největší a tato barva je přiřazena ke kostce. Následně je třeba určit, zda-li se na kostce nachází číslice. V metodě `IsNumber()` je skrze kontury bílých kruhů spočítá číslici, kterou kostka obsahuje. Pokud není zjištěno číslo, považuje se tato kostka pouze za slot a nebude vrácena jako kostka. V opačném případě je kostce přiřazeno číslo a následně i řádek a sloupec. Zároveň je při doplnění těchto informací zvýrazněna oblast kostky ve výstupním obrázku, jak je možné vidět na obrázku 8.2.



Obrázek 8.2: Detekce kostek při plném využití vitráže

8.3 Vyhodnocení pravidel a výpočet bodů

V předchozích sekcích byla popsána implementace detekce karty klíče a kostek ve vitráži hry Sagrada. Informace získané z těchto detekcí lze použít pro vyhodnocovací systém a systému kontroly pravidel deskové hry. Při implementaci těchto systémů jsem myslel na možné rozšíření systémů. Oba systémy jsou tak navrženy, aby při rozšíření o nová pravidla a karty úkolů stačilo pouze přidání funkce, která zaobaluje pravidlo, nebo kartu úkolu.

8.3.1 Výběr karet úkolů a výpočet bodů

Každý hráč dostává na začátku hry kartu osobního úkolu. Tato karta určuje, jak má kostky ve své vitráži pokládat, aby získal co nejvíce osobních bodů. Kromě takto získaných bodů, lze získat body za kartu společného úkolu, kterou všichni hráči mají stejnou. Aplikace pro vyhodnocení těchto bodů obsahuje systém, který za hráče na základě informací z detekcí karty klíče a herních kostek vypočítává body. Celý systém vyhodnocení bodů je implementován v třídě `GameBoard`.

Při vyhodnocení bodů pro konkrétního hráče jsou instanci třídy `GameBoard` předány pole slotů z karty klíče a pole kostek z vitráže. Přes metodu `Evaluation()` se následně body vypočítají. Metodě je nutné předat informace o *osobním úkolu*, *společném úkolu* a *počtu zbylých bodů řemeslníka*. Následně je vytvořena nová třída úkolu.

Výčtové třídy `CQ_TYPES` pro společný úkol a `PQ_TYPES` pro osobní úkol definují typy jednotlivých karet pro jednoduší přiřazování výpočetních funkcí systému a zároveň pro reprezentaci úkolů v grafickém rozhraní.

Třída úkolu `Quest` přiřazuje výpočetní funkce pro úkoly a zároveň provádí výpočet bodů úkolů. Po vytvoření instance třídy je nutné zavolat metody `SetPersonalCalculator()` a `SetCommonCalculator()`. Tyto metody nastavují atributy třídy úkolu `personalQCalculator` a `commonQCalculator`. Atributy jsou typu `IQuestCalculatea` jedná se o rozhraní, které deklaruje funkci `Calculate()`. Do atributů je potom vložena definice funkce pro konkrétní typ úkolu. Poté, co jsou nastaveny oba typy úkolů, může dojít k výpočtu bodů v rámci metody `RunEvaluation()`. Tato metoda vypočte body volané skrze dříve zmíněné atributy a přičte k nim počet zbylých bodů řemeslníka. Výsledná hodnota je následně vrácena instanci třídy `GameBoard`.

8.3.2 Kontrola pravidel

Ještě předtím, než je možné vypočítat body hráče, je nutné zkontrolovat, zda-li byla splněna veškerá pravidla hry. Vyhodnocení pravidel je implementováno ve třídě `RuleHandler`. Třídě je nutné předat detekované sloty karty klíče a detekované kostky vitráže. Kromě toho je nutné předat také informace o modifikačních kartách řemeslníka.

Karty řemeslníka umožňují dodatečné tahy ve hře. Příkladem můžeme udat, že hráč využije kartu řemeslníka, která mu umožní vzít kostky z nabídky a zaměnit si jejich číslo podle vlastní potřeby. Existují však dvě karty řemeslníka, které umožňují porušit pravidlo. Jedná se o karty *Brusný papír* a *Rydlo na eglomisé*. Tyto karty dovolují hráči položit kostku jiné barvy, resp. čísla na políčko karty klíče jiného čísla, resp. barvy. Aby bylo možné zjistit, zda-li bylo pravidlo opravdu porušeno, nebo zda-li hráč použil jednu z těchto dvou karet, je v rámci uživatelského rozhraní implementován způsob, kterým hráč zadá systému informace o těchto kartách a zda-li je použil. Systém pro kontrolu pravidel poté tyto informace zahrne při kontrole.

Třída `RuleHandler` vyhodnocuje body v metodě `CheckRules()`. Zde se iterativně pro každou kostku prochází jednotlivá pravidla hry, kde každé pravidlo má vlastní funkci. Pokud je některé z pravidel porušeno, systém ke konkrétní kostce přiřadí informaci o pravidle, které bylo porušeno skrze výčtovou třídu `RULE_ERR`. Hodnota tohoto typu je posléze použita v informační aktivitě grafického rozhraní, kde je uživatel informován o typu chyby, kterou musí opravit. Systém zároveň při porušení pravidla ukončuje vyhodnocování a vrací *false*.

8.4 Externí balíčky

Aplikace primárně řeší podporu pro hru Sagrada, z tohoto důvodu jsem v aplikaci použil několik externích balíčků, díky kterým bylo možné zaobírat se důležitými aspekty aplikace v rámci podpory pro deskovou hru.

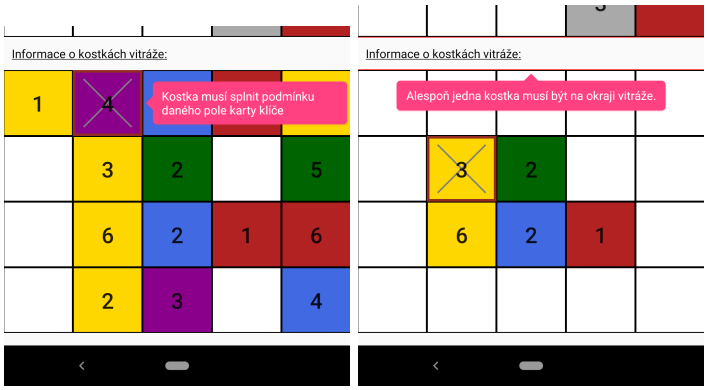
8.4.1 Rozhraní kamery

Pro využití kamery jsem původně počítal se základní funkcionalitou kamery a tak i s implementovanou kamerou přímo od Google. V pozdější fázi implementace rozhraní a testování detekcí bylo zapotřebí rozšířit kameru o některé funkcionality. Z tohoto důvodu bylo třeba implementovat vlastní rozhraní kamery, nebo najít vhodný volně dostupný balíček. Po zvážení možností, jsem se rozhodl použít volně dostupný balíček s vhodnou implementací kamery.

Z několika možností nakonec byla vybrána knihovna **CameraView**[1]. Knihovna poskytuje širokou škálu možností pro nastavení kamerového zařízení. Zároveň za programátora řeší správu povolení, jejichž implementace je obsažena v knihovně. Vývojář nemusí knihovnu použít pouze programově. Kameru lze deklarovat v rámci XML definice rozhraní grafických prvků a skrze atributy elementů nastavovat kamerové zařízení. Knihovna také podporuje výběr formátů fotky a jednoduše dostupné informace o vlastnostech fotografie.

8.4.2 Popis chyby kostky

V případě, že je porušeno pravidlo, musí být uživatel nějakým způsobem informován, jaké pravidlo porušil. Za tímto účelem jsem použil knihovny **Tooltips**[2]. Knihovna implementuje jednoduchý způsob pro zobrazení a vytváření tooltipů. Ty jsou uživateli zobrazeny u špatně položené kostky po dlouhém podržení políčka kostky (viz. 8.3a. Tooltip obsahuje informaci o porušeném pravidle, pokud uživatel opraví kostku, tooltip zmizí. V případě, že je porušeno pravidlo celé vitráže, je tooltip zobrazen u nadpisu příslušné karty v informační aktivitě, to lze vidět na obrázku 8.3b.



(a) Tooltip s porušením pravidla kostky

(b) Tooltip s porušením pravidla karty

[1] K dispozici z: https://natario1.github.io/CameraView/home
[2] Dostupné z: https://github.com/tomergoldst/tooltips

Kapitola 9

Testování

Pro zajištění kvality aplikace jsem provedl několik fází testovaní za účelem zjištění kvality rozpoznání obrazu a použitelnosti u uživatelů. Testování probíhalo na úzké skupině uživatelů. V prvotní fázi testování detekcí se využilo užší datové sady, která zahrnovala obrázky z prostředí s různým nasvícení a několika druhy podkladů. Zároveň byly využity snímky, kdy se v obraze odráží odlesky světelných zdrojů, nebo přechází světelný paprsek přes detekované segmenty. Z testování vzešlo, že pro účely přesnější detekce karty klíče a kostek, je potřeba vyhradit oblast při focení, ve které je potřeba kartu klíče, nebo kostky vitráže umístit, aby byla zaručená spolehlivá detekce.

9.1 Detekce karty klíče

Při kontrole detekce karty klíče se pracovalo jak s obrázky datové sady, tak v pozdější fázi testování za reálných podmínek. Při testování byla hodnocena detekce kontrolního bodu, jednotlivých políček karty a hodnoty v těchto polích.

Z výsledků testování karty klíče za normálních podmínek vzešlo, že při standardních světelných podmínkách nevznikají problémy s detekcí. U karet byly správně detekovány kontrolní body, jednotlivá pole a jejich vlastnosti. V ojedinělých případech docházelo k rozdílným detekcím v rámci políčka se vzorem čísla $1 - 3$, kdy políčko s číslem 1 bylo identifikováno za číslo dva, nebo políčka s čísly 1 a 2 nebyly detekovány vůbec.

Po provedení testů karty v odlišných podmínkách, se projevilo, že kvalitu detekovaného snímku nejvíce ovlivnil odlesk zdroje světla na kartě, nebo příliš tmavé světelné podmínky. Zde docházelo k zániku informací karty. Při focení karty z přílišné vzdálenosti mohl nastat problém, kdy se ve vymezené oblasti pro detekci objevily objekty, které připomínaly kontrolní bod. Dále mohla být fotka naklopená, nebo dokonce rozmazaná. Všechny tyto aspekty negativně ovlivnily detekci již při detekci kontrolního bodu, nebo v pozdějších fázích detekce.

V rámci uživatelského testování (20 pokusů na subjekt) rozpoznání karty klíče nedocházelo k větším problémům s detekcí. V případě, že subjekty použily vymezenou oblast kamery pouze pro kartu klíče, docházelo k přesným detekcím v testovacích podmínkách (místnost s umělým zdrojem světla nad stolem v odpoledních hodinách, místnost pouze s venkovním zdrojem světla za zataženého počasí, místnost s venkovním zdrojem světla za jasného počasí bez umělého zdroje světla). K chybám detekce docházelo pouze v případech, kdy subjekt vyfotil rozmazanou fotografii, nebo nevyfotil celou kartu a aplikace tak ne-

použila, pro detekci, část karty mimo vyznačenou oblast detekce. Většina chyb spojená se světelnými podmínkami se týkala problematické detekce čísel $1 - 3$ na polích karty klíče.

9.2 Detekce kostek

Kostky byly testovány na datové sadě a zejména za reálných podmínek. Při jejich detekci jsem zkoumal, zda-li jsou správně nalezeny kostky, detekovány jejich barva a číslo. Na závěr se kontrolovalo, zda-li u kostky odpovídá číslo sloupce a řádky.

Za normálních podmínek a u všech snímků datové sady proběhla detekce úspěšně, stejně tak určení jejich pozic. V případě, že se snížily světelné podmínky, byly některé kostky špatně detekovány. Jednalo se zejména o kostky, které svou barvou blížily tmavému pozadí a splynuly s prostředím. Při snížených světelných podmínkách docházelo k problému u detekce některých kostek. Jednalo se především o detekci barev kostek za velmi snížených světelných podmínek, kdy některé kostky splynuly s černým podkladem a okolím. Jelikož kostky imitují skleněnou strukturu a propouští světlo, docházelo ke vzniku velmi tmavých míst za nízkých světelných podmínek. V případě, kdy na kostky byl puštěn přímý zdroj umělého světla, nebo světlo ze slunečních paprsků v tmavší místnosti, docházelo ke ztrátě informace o čísle kostky, nebo jejímu špatnému rozpoznání.

Při provedení uživatelského testování (20 pokusů na jeden subjekt), nedocházel za stejných podmínek prostředí jako v 9.1 k výrazným problémům. Z testování vzešlo, že v některých případech nebyly kostky detekovány z důvodu odlesku světla na hrací kostce. V druhém případě nebyly některé kostky detekovány z důvodu focení kostek pod úhlem a došlo tak k zániku informace o mezeře mezi kostkami. Subjekt po takto vyfoceném snímku sám došel na způsob, jak správně vyfotit kostky ve vitráži.

9.3 Uživatelské testování

Aplikace byla v závěrečné fázi vývoje testována na úzké skupině deseti uživatelů. Cílem tohoto testování bylo odhalit chyby při používání aplikace a získat zpětnou vazbu od uživatelů na rozložení prvků v aplikaci. Uživatelské testování zároveň posloužilo pro získání dalších dat k testování rozpoznání obrazu (viz. 9.1 a 9.2).

Uživatelé byly nejprve seznámeni s pravidly deskové hry a první hra byla odehrána bez použití mobilní aplikace. Připomínek subjektů bylo několik. Jako hlavní problémy viděli zdlouhavost závěrečných výpočtů a nutnost posouvat vitráže sousedy. Soused se nejprve musel zorientovat v soupeřově vitráži. Každý subjekt testoval detekce jednotlivých prvků hry na 20 pokusech. Zároveň byly subjektu pokládány dotazy ohledně rozložení komponent grafického rozhraní a zda-li subjekt chápu k čemu slouží a jak má s aplikací zacházet.

Následující hry byly odehrány s využitím mobilní aplikace. Uživatelé se pár minut seznamovali s aplikací. Při následném používání projevili intuitivní užívání vytváření nových uživatelů a přiřazování karet k hráči. Subjekty také ihned věděli, které tlačítko použít při detekci. Důležitým testem prvků rozhraní byla schopnost subjektů vyfotit kostky a kartu klíče do obdélníkové oblasti. Při tomto testu uživatelé projevily schopnost jednoznačně pochopit, kam na kameře umístit kartu a kostky. Při detekci kostek a karty se rozpoznání obrazu podařilo jednotlivých subjektů ve většině pokusů. Nepřesné detekce byly zejména z důvodu, že uživatelé vyfotily rozmazaný obraz, nebo u karty klíče nevyfotili celou kartu, ale pouze políčka karty.

Subjekty zároveň měly několik připomínek. Stěžejní připomínkou byla potřeba implementace dialogové okna při použití zpětné šipky u informační aktivity a aktivity hráče. V těchto případech se stalo, že se uživatelé vrátili na předchozí aktivitu a informace, které vyplnili, se ztratily. Dále vyjádřili zájem o změnu vyhodnocení bodů. Subjekty se domnívaly, že by bylo vhodné, kdyby se body vypočítávaly automaticky pokaždé, když se změní některá informace hráče. Aplikace při nalezení prvního porušení pravidel končí vyhodnocení pravidel a vrací první chybnou kostku. Tento postup uživatelům přišel zmatečný a bude proto upraven. Nově bude systém kontroly chyb procházet celou vitráž a vrátí chyby všech kostek.

Z testování na subjektech vyplynula intuice používání aplikace ze strany subjektů. Při používání aplikace nedošlo k selhání aplikace. Subjekty zároveň, bez nutnosti nápovědy, věděli, jak fotit obrazy.

Kapitola 10

Omezení a rozšíření

Při vývoji aplikace byly nastaveny určité hranice, bez kterých by fungování podpory pro deskovou hru bylo méně úspěšné. Hlavním omezením aplikace je nutnost fotit karty klíče a kostky vitráže ve vyhraněné sekci vyznačené barevným obdélníkem, jak bylo řečeno v kapitole 9.

10.1 Omezení

Aplikace detekuje karty klíče s menší pravděpodobností na úspěch v případech, kdy jsou světelné podmínky velmi minimální, nebo naopak extrémní. Příkladem můžeme udat velmi temnou místnost, nebo odraz zdroje světla na kartě. Obdobné omezení se vztahuje i na kostky vitráže, které při velmi malém zdroji světla ztrácí informaci o barvě. V případě, že jsou kostky přesvícené a dochází k odleskům světla, může opět zaniknout informace o barvě kostky. Dále, pokud jsou kostky přesvícené, může dojít také k zániku informace o čísle na kostce.

V případě samotné aplikace, se řešení omezuje pouze na zařízení, u nichž je dostupná kamera mobilního zařízení. Práce nebyla testovaná v situacích, kdy mobilní zařízení neobsahovalo, nebo u něj nebyla detekována kamera. Aplikace obsahuje podporu pouze dvě jazykové lokalizace (*česká* a *anglická*).

10.2 Rozšíření

Pro další postup práce by bylo vhodné doplnit možnost využití aplikace pro zařízení, u nichž není detekováno kamerové zařízení. V tomto případě by údaje mohly být doplněny skrze informační aktivitu 7.4. Z poznámek subjektů uživatelské studie vzešlo, že by uživatelé ocenili statistiky hráčů , kteří dříve hru hráli. Možným rozšířením je tedy přidání databáze se jmény hráčů a body, které získali.

V rámci rozpoznání herního klíče a kostek vitráže, by se dalo zaměřit nad zlepšením detekce ve zhoršených podmínkách (např. odlesky karet), nebo detekcí karet a kostek v obraze bez specifické oblasti detekce. Dále lze přidat možnost automatické detekce počátečních bodů kamenů řemeslníka, jejichž počet je roven počtu kruhů obtížnosti na kartě klíče.

Kapitola 11

Závěr

Cílem práce bylo vytvořit aplikaci pro podporu deskové hry Sagrada na OS Android. Podpora byla zaměřena na usnadnění výpočtu bodů hry a vyhodnocení správnosti pravidel za použití rozpoznání obrazu dvou klíčových prvků hry - *karta klíče* a *kostky vitráže*.

Nejprve byly získány informace o deskové hře a existujících podporách pro Sagradu. Ze zjištění vyplynulo, že Sagrada neobsahuje žádnou existující podporu. Následně byly nastudovány způsoby a metody použitelné pro rozpoznání obrazu a teorie pro vytváření aplikací na platformě OS Android. Poté byly nalezeny existující řešení pro rozpoznání herních prvků různých deskových her.

Při návrhu aplikace jsem vycházel z nastudované teorie a existujících řešeních. Při implementaci byly detekce prvků hry podrobeny testování na snímcích z datové sady, která byla vytvořena, aby zachytila hru v různých fázích postupu a za různých světelných podmínek. Testování na datové sadě posloužilo ke zlepšení funkčnosti rozpoznání herních prvků. Aplikace byla průběžně ve vývoji testována také v reálných podmínkách. V závěru vývoje byla aplikace podrobena uživatelskému testování. Z testování vyplynulo, že uživatelé byly až na drobné připomínky s aplikací spokojení a intuitivně využívali jednotlivé komponenty grafického rozhraní.

Výsledkem práce je aplikace pro mobilní zařízení s OS Android, která umí rozpoznávat herní prvky deskové hry Sagrada. Ze získaných informací poté dokáže vyhodnotit body hráče a kontrolovat dodržení pravidel hry. Aplikace dále obsahuje správu více hráčů na jednom zařízení a automatické vyhodnocení bodů v případě, že jsou všechny potřebné informace dostupné. O jednotlivých funkcích, které aplikace umožňuje, informuje vytvořený plakát.

Ve vývoji aplikace by se dalo dále pokračovat. Zejména by bylo vhodné se zaměřit na zlepšení designu aplikace a pokračovat na zlepšení kvality detekce herních prvků, popř. přidat nové rozšíření detekce herních prvků.

Literatura

[1] *OpenCV Playing Card Detector*. 2017. Dostupné z: https://github.com/EdjeElectronics/OpenCV-Playing-Card-Detector.

[2] Abeles, P. *BoofCV v0.25* [online]. 2016. Dostupné z: http://boofcv.org/.

[3] Azim, M. F., Hidaya, E. W. a Rachman, A. N. *Android Battle Game Based on Augmented Reality with 2D Object Marker*. 2018. [cit. 17. března 2020]. Dostupné z: http://join.if.uinsgd.ac.id/index.php/join/article/view/v3i29/104.

[4] Bradski, G. a Kaehler, A. *Learning OpenCV*. 1. vyd. O'Reilly Media, Inc., 2008. ISBN 978-0-596-51613-0.

[5] Catchoom. *Android CraftAR Cloud Image Recognition SDK* [online]. Dostupné z: https://documentation.catchoom.com/documentation/image-recognition-sdk/android-image-recognition-sdk/.

[6] Cihlářová, D. *Uživatelské rozhraní s kamerou pro deskovou hru*. 2016. Diplomová práce. Vysoké učení technické v Brně, fakulta informačních technologií.

[7] Gong, X.-Y., Su, H., Xu, D., Zhang, Z.-T., Shen, F. et al. An Overview of Contour Detection Approaches. *International Journal of Automation and Computing*. Červen 2018, roč. 15.

[8] Google inc.. *Android Developer Guide* [online]. Dostupné z: https://developer.android.com/guide.

[9] Green, B. *Canny Edge detection tutorial*. 2002 [cit. 17. března 2020]. Dostupné z: https://dce8603d-a-62cb3a1a-s-sites.googlegroups.com/site/setiawanhadi2/1CannyEdgeDetectionTutorial.pdf.

[10] Hlaváč, V. *Hledání hran a hranových bodů*. [cit. 17. března 2020]. Dostupné z: http://people.ciirc.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/22EdgeDetectionCz.pdf.

[11] Hlaváč, V. *Matematická morfologie*. [cit. 17. března 2020]. Dostupné z: http://people.ciirc.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/71-3MatMorpholBinCz.pdf.

[12] Hurta, M. *Podpora hry Krycí jména na mobilním telefonu s OS Android*. 2019. Diplomová práce. Vysoké učení technické v Brně, fakulta informačních technologií.

[13] IDC Corporate. *Smartphone Market Share* [online]. říjen 2019 [cit. 10. ledna 2020]. Dostupné z: https://www.idc.com/promo/smartphone-market-share.

[14] Kolouchová, M. *Morfologické operace ve zpracování obrazu*. 2008. Diplomová práce. Vysoké učení technické v Brně, fakulta informačních technologií.

[15] Kolínek, D. *Podpora hraní deskové hry Mlýny mobilní aplikací*. 2019. Diplomová práce. Vysoké učení technické v Brně, fakulta informačních technologií.

[16] Kršek, P. *Základy počítačové grafiky — skripta IZG* [online].

[17] Lacko Luboslav. *Vývoj aplikací pro Android*. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.

[18] Musil, T. *Optical Game Position Recognition inthe Board Game of Go*. 2014. Diplomová práce. Charles University in Prague, Faculty of Mathematics and Physics.

[19] Muthukrishnan, R. Edge Detection Techniques For Image Segmentation. *International journal of computer science and information technology*. Prosinec 2011, roč. 3, s. 259–267.

[20] OpenCV team. *OpenCV 4.1.1 Documentation* [online]. Dostupné z: https://docs.opencv.org/4.1.1/.

[21] Staněk, J. *Rozpoznání stavu hry Scrabble*. 2013. Diplomová práce. Vysoké učení technické v Brně, fakulta informačních technologií.

[22] Summers, N. *The best board games with an app-based twist* [online]. 2019 [cit. 10. ledna 2020]. Dostupné z: https://www.engadget.com/2019/11/29/best-board-games-companion-app-guide.

[23] Urban Šimon. *Uživatelská rozhraní pro hru na hrdiny na interaktivním stole*. 2019. Diplomová práce. Vysoké učení technické v Brně, fakulta informačních technologií.

[24] Wikipedie. *Barevný model — Wikipedie: Otevřená encyklopedie*. 2019. [Online; navštíveno 19. 03. 2020]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Barevn%C3%BD_model&oldid=17328514.

[25] Wikipedie. *Počítačové vidění — Wikipedie: Otevřená encyklopedie*. 2019. [Online; navštíveno 13. 05. 2020]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Po%C4%8D%C3%ADta%C4%8Dov%C3%A9_vid%C4%9Bn%C3%AD&oldid=17575209.