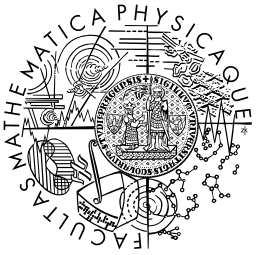


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Monika Bošániová

Rubikova kocka

Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Vladan Majerech, Dr.

Studijní program: Informatika

Studijní obor: Programování a softwarové systémy

Praha 2023

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Chcela by som poďakovať Mgr. Vladanovi Majerechovi, Dr. za odborné vedenie práce a rady, ktoré mi pomohli túto prácu doviesť do úspešného konca. Ďakujem mojej rodine, ktorá ma podporovala počas celého štúdia. Jiřímu Pelcovi, ktorý mi odovzdal nespočetne mnoho rád a informácií ohľadom práce v prostredí Unity. Anne Yaghobovej a Natálii Potočekovej za pomoc pri štylizácii. A v neposlednom rade Davidovi Nápravníkovi, bez ktorého podpory by to ani nešlo.

Název práce: Rubikova kocka

Autor: Monika Bošániová

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Vladan Majerech, Dr., Katedra teoretické informatiky a matematické logiky

Abstrakt: Táto práca je vytvorená za účelom zjednodušiť pohľad začiatočníkom na výučbu skladania Rubikovej kocky. Prechádza rôznymi pohľadmi, ako vyriešiť tento hlavolam. Zameriava sa na popis implementácie samotného aplikovaného postupu skladania. Približuje výzor prostredia aplikácie a interaktivitu rôznych elementov naprieč jednotlivými úsekmi a zahŕňa popis všetkých aplikačných komponentov. Poskytuje náhľad do spracovania vyučovacej časti a analyzuje jej efektivitu v porovnaní s existujúcimi riešeniami. Obsahuje užívateľskú dokumentáciu a návod pre pridanie vlastného algoritmu skladania v textovom formáte. Navrhuje prípadné vylepšenia do budúcnosti.

Klíčová slova: rubikova kocka riešič tutoriál

Title: Rubik's cube

Author: Monika Bošániová

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Vladan Majerech, Dr., Department of Theoretical Computer Science and Mathematical Logic

Abstract: The main goal of this thesis is to simplify the beginners' experience with learning and independently solving the Rubik's cube. We provide different perspectives on how to find the solution for this puzzle. The implementation of all used components, chosen solving process for beginners, appearance of the application environment and the interactivity of different elements are explained and described in an easily understandable way. We included insights on the teaching process and analysis of its effectiveness in comparison to similar existing solutions. User has an option of adding their own solving algorithm in text format. The text contains user documentation and suggests possible improvements for future development.

Keywords: rubik's cube solver tutorial

Obsah

Úvod

Myšlienkou tejto práce je ukázať a pomôcť užívateľom osvojiť si algoritmus skladania Rubikovej kocky. Hlavnou ideou, ktorej sme sa držali, bolo vytvoriť prostredie, ktoré by napĺňalo základy jednoduchosti, intuície, efektívnosti a praktickosti.

Už od mala sme boli fascinovaní skladaním Rubikovej kocky. Slúžila nám nie len na krátenie voľného času, ale aj ako pomôcka pri cvičení vizuálnej predstavivosti. Myslíme si, že v dnešnej dobe chýba ľuďom a hlavne deťom nejaký spôsob, ako by mohli premýšľať v 3D priestore a zároveň pri tom trénovať a vymýšľať rôzne algoritmy, či postupy. Preto sme sa rozhodli vytvoriť tento softvérový projekt, ktorým sme chceli priblížiť užívateľom prácu s Rubikovou kockou a naučiť ich jednoduché základné riešenie tohto hlavolamu.

V súčasnosti existuje viacero webových prostredí, ktoré ukážu sekvenciu ťahov, ktorá vráti Rubikovu kocku do pôvodného stavu. Napriek tomu, máloktorý ponúka možnosť sa to pri tom procese aj naučiť. Kvalitné myšlienky a nápady z týchto projektov sme aplikovali aj do toho nášho.

Jednotlivé kapitoly prevedú čitateľa vývojom prostredia aplikácie. Odôvodníme naše rozhodnutia z hľadiska implementácie jednotlivých komponentov. Do najmenších detailov opíšeme prostriedky, ktorými napĺňame naše vopred stanovené ciele. Nakoľko sa jedná o aplikáciu zameranú pre začiatočníkov, vysvetlíme princípy dosiahnutia efektívnosti osvojenia si riešenia hlavolamu. Dopodrobna popíšeme implementáciu prostredia. Užívateľovi poskytneme dokumentáciu, v ktorej nájde všetky podstatné informácie. Opíšeme vzhľad a funkcie interaktívnych prvkov. Vysvetlíme použité algoritmy zakomponovaných riešičov.

Nakoniec zhodnotíme celkovú prácu so softvérom, jeho otestovanie, silné stránky a plány vylepšenia do budúcnosti.

1. Teória

Rubikova kocka je 3D mechanický hlavolam vytvorený v roku 1974 maďarským architektom a sochárom *Ernőm Rubikom*. Svoj najväčší úspech zažila v 70. a 80. rokoch 20. storočia. Vtedy sa jej predali milióny kusov. Dnes už síce nie je na vrchole svojej slávy, no viacerým ľuďom ostala v povedomí natoľko, aby nad ňou strávili aspoň kúsok svojho voľného času.

Úlohou užívateľa je za použitia rotácií jednotlivých strán, poskladať kocku do pôvodného stavu. Pre priemerného človeka je tento proces obzvlášť zložitý, ak nepozná žiaden zo známych algoritmov.

1.1 Model Rubikovej kocky

Klasická varianta tohto hlavolamu pozostáva z 26 malých kociek poskladaných do kocky o rozmeroch 3x3x3. Každá z jej šiestich stien má pridelenú farbu, ktorá bola pôvodne označovaná samolepiacou páskou. Na kocke si môžeme povšimnúť väčšinou týchto šesť farieb: červenú, žltú, modrú, zelenú, oranžovú a bielu.

Obr. 1.1 | Model Rubikovej kocky

1.2 Terminológia

Kvôli lepšiemu zorientovaniu v celkovom modeli hlavolamu si musíme vysvetliť niekoľko pojmov. Tie budeme používať v nasledujúcich kapitolách.

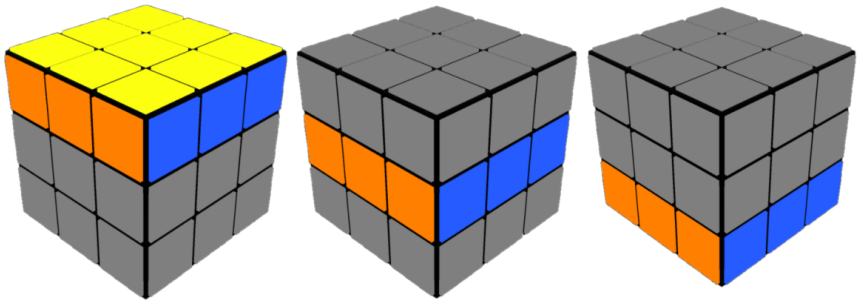
1.2.1 Typy dielikov

Dielikom rozumieme každú malú kocku, ktorá tvorí celkový hlavolam. Rozdeľujeme ich do troch skupín podľa počtu viditeľných farebných stien.

Názov	Počet dielikov na jednej stene	Počet dielikov v celom hlavolame
Stred	1	6
Hrana	2	12
Roh	3	8

Tabuľka 1.1 | Tabuľka s jednotlivými typmi dielikov hlavolamu

Klasická Rubikova kocka v rozmeroch 3x3 obsahuje tri **vrstvy** (hornú, strednú a spodnú). Ich vizualizáciu si môžeme všimnúť *na obrázku 1.2*.



Obr. 1.2 | Vizualizácia vrstiev

Pri hlavolamoch inej úrovne sa počet vrstiev mení, no je rovný rozmerom kocky. Zároveň vždy existuje iba jedna horná a spodná vrstva. Všetky ostatné medzi nimi sa považujú za stredové.

Pri zorientovaní sa v samotnom modeli majú jednotlivé steny prívlastok z hľadiska pohľadu užívateľa. Definujeme prednú, zadnú, ľavú, pravú, hornú a spodnú stranu. Musíme však brať do úvahy otočenie celkovým hlavolamom. Ak takáto situácia nastane, pomenovanie stien sa obnoví v závislosti na pozorovateľovi. To znamená, že po otočení celej kocky sa prednou stranou stáva stena otočená priamo na užívateľa.

Naša aplikácia vopred ráta s rozložením stien opísanom *v nasledujúcej tabuľke 1.2*. Upozorňujeme však, že sa jedná iba o začiatočnú rotáciu. V prípade, že užívateľ otočí kompletným modelom, ich závislosť na farbe sa mení.

Pozícia	Farba
Predná	Oranžová
Zadná	Červená
Pravá	Modrá
Ľavá	Zelená
Horná	Žltá
Dolná	Biela

Tabuľka 1.2 | Nami priradené farby k stenám kocky

1.2.2 Rotácie strán

Pri manipulácii s Rubikovou kockou sú povolené iba rotácie, ktoré nenarušia jeho tvar. Ak sa užívateľ rozhodne riešiť hlavolam inak, je jeho pokus neplatný. Akékoľvek nepovolené narábanie s modelom, môže viesť k jeho neriešiteľnosti.

Existuje viacero spôsobov, ako môžeme dané otočenia pomenovať. V celom programe používame jednu z klasických notácií, ktorá pozostáva z viacerých prvkov. Tie sú opísané v tabuľke nižšie. Upozorňujeme, že pre správnu definíciu rotácie, majú všetky elementy dobrovoľné definovanie. Okrem druhého, a to definovanie strany. Poradie prvkov je rovnako určené *tabuľkou 1.3* nižšie, a to zhora nadol.

Definícia prvku	Popis
Prirodzené číslo	Počet otáčaných vrstiev ALEBO Poradie jednej vrstvy (závisí za zadaní písmena 'w')
[F, B, R, L, U, D]	Jeden znak označujúci rotáciu podľa danej steny
Písmeno 'w'	Pohyb vonkajšej a súsediacej vnútornej vrstvy spoločne (možnosť viacerých, ak je zadaná prvá položka)
Apostrof (')	Rotácia sa bude vykonávať v protismere hodinových ručičiek
Číslica 2	Uhol rotácie bude 180°, inak 90°

Tabuľka 1.3 | Zoznam prvkov pre zadanie rotácie

V *tabuľke1.4* nižšie sme spísali príklady rôznych rotácií aj s opisom.

Príklad rotácie	Popis
F	Otočenie prednej steny o 90°
Fw	Otočenie prednej steny a vrstvy vedľa nej o 90°
3Uw'2	Otočenie troch vrstiev spoločne s hornou o -180°
2B2	Otočenie iba druhej vrstvy od zadnej steny o 180°

Tabuľka 1.4 | Príklady rotácií

Týmto spôsobom definovania je možné zadať ľubovoľnú konfiguráciu vrstiev a uhlov.

Musíme však spomenúť, že existujú aj notácie, ktoré slúžia ako skratky pre zápis rotácií. Tie sme stručne spísali *do nasledujúcej tabuľky 1.5*.

Rotácia	Popis
x	Otočenie celého hlavolamu v smere rotácie R
y	Otočenie celého hlavolamu v smere rotácie U
z	Otočenie celého hlavolamu v smere rotácie F
M	Otočenie všetkých stredových vrstiev v smere L
E	Otočenie všetkých stredových vrstiev v smere D
S	Otočenie všetkých stredových vrstiev v smere F

Tabuľka 1.5 | Ďalšie možné notácie rotácií

2. Analýza a požiadavky

V tejto kapitole sa zameriame na rôzne spôsoby riešenia daných problémov, s ktorými sme sa pri implementácii stretli. Rovnako nahliadneme na ich výhody a nevýhody. Porovnáme ich efektivitu a vhodnosť z hľadiska cieľov projektu.

2.1 Riešič

Jednou z najväčších otázok bola implementácie riešiča Rubikovej kocky. Jeho úlohou je napomáhať užívateľovi pri skladaní hlavolamu či kontrolovať validitu užívateľom napísaného postupu. Zvážili sme preto vhodnosť viacerých algoritmov a prístupov, ktoré sme spísali do nasledujúcich podkapitol.

2.1.1 Výber postupu

Existuje veľa spôsobov, ako zložiť Rubikovu kocku. Od úplného začiatočníckeho princípu až po profesionálny. Keď užívateľ drží po prvýkrát tento hlavolam v rukách, väčšinou sa zasekne po dokončení jednej zo strán.

Hlavnou prioritou bolo vytvoriť riešič, ktorý by postupoval sekvenčne po jednotlivých fázach a uľahčil užívateľovi zapamätanie si algoritmických postupov, pomocou ktorých dokážeme poskladať hlavolam do pôvodnej podoby.

Po preštudovaní viacerých rôznych metód, sme sa zhodli pre použitie postupu spísaného na oficiálnej stránke *Rubiks*[1]. Hlavným dôvodom tohto rozhodnutia bola jednoduchosť spísaného riešenia.

Dôvodom prečo sme nepoužili metódy *CFOP*[2], *Petrus*[3] či *Roux*[4] je ich predpoklad, že užívateľ má aspoň nejaké elementárne znalosti skladania a vie sa orientovať na modeli Rubikovej kocky.

Navyše naša aplikácia bude podporovať navrhnutie vlastného riešiča užívateľom. To znamená, že náš program nebude fixovaný iba na jeden postup, ale užívateľ by si veľmi jednoducho vedel napísať aj vlastný postup.

2.1.2 Výpočet riešenia

Pri hľadaní vhodného algoritmu, ktorý by dokázal poskladať zadanú Rubikovu kocku do cieľového stavu za pomoci užívateľom definovaných rotácií, sme sa rozhodovali medzi tromi prístupmi – *Dijkstrov prehľadávajúci algoritmus*, *Iterative deepening* a *A**.

Z hľadiska edukatívnosti potrebujeme, aby vybraný algoritmus pracoval v *malých medzikrokoch*, ktoré sú pre užívateľa jednoduchšie pochopiteľné. Nakoľko všetky postupy skladania pre začiatočníkov sú postavené na viacerých

[1] `https://assets.ctfassets.net/r3qu44etwf9a/6kAQCoLmbXXu29TTuArrk1/404118e1f9bfb6f9997157a284bbc572/Rubiks_Solution-Guide_3x3.pdf`

[2] `https://en.wikipedia.org/wiki/CFOP_method`

[3] `https://speedsolving.fandom.com/wiki/Petrus_Method`

[4] `https://getgocube.com/learn/intro-to-the-roux-method/`

sekvenciách rotácií (kombá), ktoré je nutné si zapamätať, vyžadujeme, aby náš zvolený spôsob dokázal s nimi efektívne pracovať a zakomponovávať ich vo svojom výstupe. V ideálnom prípade by ich mal *uprednostňovať* pred jednotlivými rotáciami. Navyše očakávame, že algoritmus nájde riešenie (ak existuje) v rozumnom čase.

Po dlhšom rozmýšľaní a zvažovaní sme si vybrali implementáciu *Iterative deepeningu*, ktorá je tou najjednoduchšou zo spomínaných troch metód. Vytvorili sme pravidlo, v ktorom platí, že ak užívateľ zadefinuje sekvenciu ťahov, tak je jej váha rovná ľubovoľnej inej rotácii. Čo spôsobilo uprednostnenie komb.

Aby algoritmus dobehol v rozumnom čase a vydal riešenie, boli sme nútení obmedziť hĺbku vytváraného stromu. Tým pádom riešič vytvorený užívateľom, musí byť navrhnutý efektívne, čo znamená, že zadané podciele musia byť v najhoršom prípade splnené v ohraničenom rozsahu.

S odstupom času vnímame výhody Dijstrovho a A* algoritmu, ktoré by dokázali nájsť riešenie rýchlejšie a my by sme nemuseli klásť podmienky na užívateľa. Na druhú stranu siahli by sme po ich implementácii, až pri zakomponovaní váhovej heuristiky, ktorá by určovala dôležitosť jednotlivých rotácií.

V momentálnom prípade pre vyriešenie daného problému bol Iterative Deepening dostatočne silný.

2.2 Model kocky

Počas plánovania 3D modelu Rubikovej kocky, sme kládli dôraz na *univerzalitu modelu*. Chceli sme zobraziť viac ako len klasický model 3x3x3, preto sme museli pracovať s myšlienkou rozšíriteľnosti našej aplikácie o ďalšie hlavolami.

Rozhodli sme sa implementovať kocky s rozmermi dva, tri a štyri. Pričom pridanie ďalších nových rozmerov by v budúcnosti *nemalo byť prekážkou*. To znamenalo prispôsobiť generovanie modelov, výpočty rotácií, analýzu pohybov a pod.

2.2.1 Vytvorenie modelu

Pri vytváraní modelu sme sa zamýšľali nad dvomi možnými postupmi riešenia.

Prvým bolo generovanie modelu *počas behu programu* (tzv. za runtime). Užívateľ by bol schopný zadať ľubovoľnú hodnotu, ktorou by určil rozmery modelu. Silnou výhodou by bola možnosť používateľa zvoliť model veľkostne prispôsobený jeho potrebám. Tu nastáva problém pri väčších hlavolamoch, ktoré by museli byť zmenšené a prispôsobené obrazovke užívateľa, čo by spôsobilo menšiu reakčnú plochu dielikov a nezmestiace sa zrkadlá modelu.

Navyše, ak by sme užívateľa neobmedzili, bolo by mu povolené vygenerovanie Rubikovej kocky nad jeho hardvérové možnosti. V takejto situácii by v najhoršom prípade mohlo nastať aj nútené zastavenie aplikácie.

Ďalšou z nevýhod je strávený čas pri výpočtoch generovania. Samotný model upravujeme vždy pred spustením vybraného levelu. Zmeny sa týkajú jeho prefarbenia a prispôsobenia požiadavkám, ktoré sa v každej časti menia. To by znamenalo čakanie na vygenerovanie modelu a jeho následnú úpravu, čo by

spôsobilo zbytočné spomalenie.

Druhou možnosťou bolo vytvorenie modelov s *vopred nami vybranými rozmermi*. Týmto rozhodnutím by sme získali *kontrolu* nad rozhodnutiami užívateľa a navyše sme ich mohli rovno používať a pracovať s nimi, vďaka ich predgenerovaniu.

2.2.2 Manipulácia s 3D modelom

Už od prvého momentu bolo jednou z našich priorít vytvorenie 3D modelu, ktorý sa bude *čo najviac podobať tomu reálnemu*. To však prinieslo niekoľko problémov, z hľadiska jeho funkcionalít a možných spôsobov manipulácie s ním. Snažili sme sa nájsť ideálne riešenie, ktoré by uľahčilo užívateľovi prácu s našou virtuálnou Rubikovou kockou.

Ovládanie otáčania stien

Pri výbere spôsobu vyvolania rotácií na modeli sme zvažovali viacero prístupov.

Tým prvým boli *tlačidlá*, ktoré by reprezentovali jednotlivé ťahy kockou. To by znamenalo vytvorenie tlačidla pre každý z nich. Toto riešenie by spôsobilo, že užívateľ by sa jednoducho stratil medzi ich množstvom.

Vedeli by sme ich vizuálne rozmiestniť po aplikácii, farebne oddeliť, no nič by nezmenilo fakt, že týmto prístupom, by sme nenaplnili jeden z hlavných cieľov, v ktorom sa snažíme vytvoriť čo najviac reálny model Rubikovej kocky.

Druhým riešením, ktoré sa najviac podobalo realite, bolo *ťahanie jednotlivých vrstiev myšou*. Užívateľ by si zvolil farebnú plôšku jedného z dielikov, čo by simulovalo položenie reálneho prsta na model kocky. Následne by ťahavým pohybom vo vybranom smere zrotoval vrstvu.

2.2.3 Štruktúra rotácií

Aby sme v našom programe vedeli uchovávať a pracovať s rotáciami, museli sme vymyslieť zadefinovanie, ktoré by bolo jednotné pre uchovávané dáta a zároveň spracovanie otočenia 3D modelu.

Ako sme spomínali *v kapitole Teória 1.2*, notácia rotácií sa viaže vždy k jednej zo šiestich stien. Z tohto dôvodu sme dlho pracovali s myšlienkou definovať otočenie trojicou **stena**, **vrstva** a **uhol**.

Výhodou je jednoduché a intuitívne rozparsovanie textového zápisu rotácie. Zoberme si príklad *F*, ktorého výsledkom by bola trojica – *Predná stena, prvá vrstva, uhol 90°*.

Problém nastáva, pokiaľ by sme chceli rovnako zadefinovať otočenie virtuálnym modelom. V 3D priestore pracujeme s rotáciami okolo ôs x, y a z. To by znamenalo namapovať *šesť stien* na *tri osy*, čo by viedlo k možnosti zadefinovať rotáciu dvomi rôznymi spôsobmi. Vezmime do úvahy klasický model Rubikovej kocky 3x3x3, jeho rotácia F je ekvivalentná rotácii 3B'.

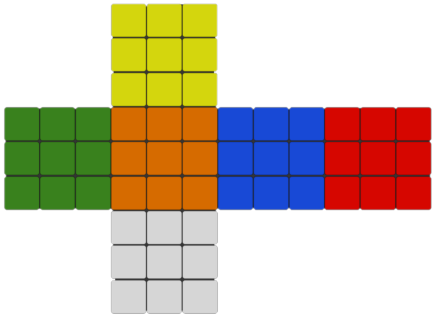
Z tohto dôvodu sme sa rozhodli pozmeniť definíciu a nahradiť každé *dve protiľahlé steny jednou osou*. Vezmime si spomínaný príklad F, ktorého ekvivalentom by bola trojica (Os z, prvá vrstva, uhol 90°).

2.2.4 Náhľad užívateľa na skryté steny

Nakoľko v aplikácii dokážeme zobraziť len tri zo šiestich strán, museli sme vymyslieť spôsob ako poskytneme užívateľovi náhľad na *kompletne celý model* v každom momente. Čo by navyše zrýchlilo jeho schopnosť orientovať sa v 3D priestore s modelom Rubikovej kocky.

Varianty 2D mapy

Jednou z možností bola implementácia 2D mapy, ktorá by zobrazovala celý 3D model. Jej vizualizáciu si môžeme pozrieť *na obrázku 2.1*.



Obr. 2.1 | Ukážka možného 2D zobrazenia modelu

Vytvorenie 2D mapy síce vyriešilo náš problém, ale všimli sme si isté nedostatky. Vnímanou nevýhodou bola strata 3D vizuálnosti a potreba nadobudnutia u užívateľov istej zručnosti v práci s ňou.

V praxi, keď sa užívateľ pokúšal nájsť všetky steny niektorého z dielikov, bol nútený stráviť netriviálne množštvo času lúštením riadku a stĺpca 2D mapy. Čo mohlo zapríčiniť, že užívateľ radšej otočil celým 3D modelom, nakoľko mu to prišlo jednoduchšie a intuitívnejšie.

Odzrkadlenie modelu

Druhým riešením bolo vytvorenie tzv. "zrkadiel- *obr. 2.2*, ktoré by odrážali vizuál zadných stien priamo pri 3D modeli.

Obr. 2.2 | Ukážka zrkadlového zobrazenia modelu

Hlavnou výhodou oproti spomínanej 2D mape je, že užívateľ priamo vidí odvrátené strany dielikov. Rýchlo a efektívne zistí, či hlavolam je poskladaný do požadovaného stavu alebo kde sa nachádza kocka, ktorú potrebuje.

2.2.5 Kontrola poskladaného modelu

Jedným z hlavných aspektov aplikácie je *vyskúšanie skladania* hlavolamu užívateľom. Z tohto dôvodu je potrebné, aby sme zistili, či daný hlavolam je už *vo finálnej fáze*.

Najväčšou otázkou pri implementácii tohto problému bolo, ako poznáme, že hlavolam je zložený, až na rotáciu komplet celej kocky.

Cieľový stav, do ktorého sa chceme dostať má fixnú podobu. Ráta s vyobrazením stien tak, ako sme ich opísali *v tabuľke 1.2*. Preto sme sa zamysleli nad viacerými možnými postupmi.

Prvým a najtriviálnejším riešením by bolo požadovať od užívateľa, aby vždy po doskladaní levelu otočil hlavolamom tak, aby korešpondoval s rozložením stien finálneho cieľa. To znamená, že skončí s oranžovou stenou vpredu, vpravo bude modrá, hore žltá atď.

Nakoľko sa snažíme vytvoriť užívateľsky prívetivú aplikáciu, toto riešenie neprichádzalo do úvahy. Nemohli sme od užívateľa požadovať zbytočné kroky navyše, ktoré by sme mohli dopočítať sami za neho.

Druhý možný postup bolo zvoliť prístup „hrubej sily“, ktorý spočíva jednoduchým vygenerovaním všetkých možných modelov zrotovanej kocky s rovnakou konfiguráciou. To by znamenalo presne 24 stavov, ktoré by sme po jednom porovnávali s finálnym cieľom. Pokiaľ by sme našli jednu zhodu, vedeli by sme, že sa kocky rovnajú. Nevýhodou tohto prístupu je nadmerné počítanie bez hlbšieho rozmyslenia a optimalizácie.

Ako ďalší postup sme zvažovali nájdenie stredov kocky a podľa ich polohy zrotovať model na svoje pôvodné miesto výpočtom a následné porovnanie s cieľom. Táto idea sa nám veľmi pozdávala. Nepočítali by sme zbytočne nadmerný počet výpočtov a riešenie by sme dostali rýchlo a efektívne.

Nakoniec sme sa však rozhodli pre poslednú možnosť, a to použiť algoritmus, ktorý máme naimplementovaný. Zavolaním Iterative deepening, ktorému povolíme iba rotácie celého hlavolamu - x,y,z.

Nakoľko máme povolené iba tri rotácie, s ktorými sa vďaka šesť stranovej kocke dostaneme do rozumnej hĺbky výpočtu, zvolili sme tento postup. Jeho hlavnou výhodou bolo opakované použitie už naimplementovaného algoritmu, ktorý v rozumnom čase vráti riešenie.

2.3 Vyobrazenie menu levelov

Pri zamýšľaní nad vizuálom menu s levelmi sme sa snažili nájsť riešenie, ktoré by dokázalo ukázať finálny stav Rubikovej kocky. Nakoľko užívateľ má právo spísania vlastného riešiča, museli sme túto scénu vytvoriť tak, aby bola

jednoducho modifikovaná zdrojovými súbormi.

Prvotný nápad spočíval vo vytvorení si modelu Rubikovej kocky na pozadí pri behu programu. Ten by sa prefarbil podľa užívateľsky zadaného stavu a odfotil kamerou v prostredí Unity. Snímku by sme si uložili a následne vložili do scény.

Prvou nevýhodou tohto riešenia je nákladné renderovanie a následné uchovanie obrázku, ktoré by bolo časovo a pamäťovo náročné. Navyše by užívateľ musel nastavovať pozíciu a uhol, pod ktorým by kamera zosnímala model kocky tak, aby výsledná snímka zobrazovala podstatnú časť daného levelu.

Preto sme následne prišli s riešením, ktoré prefarbí kocku počas behu aplikácie. Do scény sme umiestnili kameru, ktorá rotuje okolo modelu, a tým poskytuje užívateľovi pohľad na všetkých šesť stien.

3. Dizajn

Hlavným cieľom aplikácie bolo vytvorenie intuitívneho prostredia v programe Unity, ktoré zabezpečí rýchlejšie osvojenie algoritmu skladania Rubikovej kocky a dopraje príjemný užívateľský zážitok. V tejto kapitole si preto priblížime dizajn celej aplikácie. Chceli by sme zdôrazniť, že *všetky používané obrázky a ikony* sú *nami nakreslené a vytvorené*.

3.1 Scény

Program je tvorený jednotlivými *scénami*, kde každá má iný sémantický význam. *Na obrázku 3.1* je vizualizovaný ich tok. Jednotlivé šípky zaznamenávajú *presuny* medzi nimi.

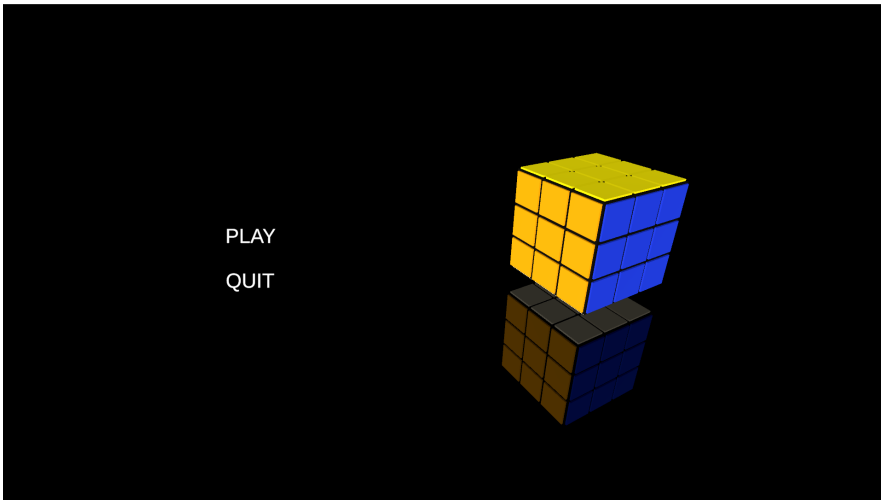
Obr. 3.1 | Tok scén aplikácie

Povšimnime si, prácu a načítanie súborov medzi prechodmi od prvej až k tretej scéne. Ich spracovanie prebieha *na pozadí* počas behu programu. Pred spustením menu s výberom veľkosti hlavolamu a riešiča musíme najprv zistiť ich výskyt. V prípade, že by užívateľ zmenil nami definované súbory, pri novom štarte aplikácie a načítaniu druhej scény sa všetky prepíšu na ich *pôvodnú verziu*.

3.1.1 Hlavné menu

Táto scéna (*obr. 3.2*) slúži ako uvítacia obrazovka do prostredia našej aplikácie. Preto sme sa vyhrali s vizuálom zobrazovaného modelu.

Spolu s Rubikovou kockou sa v menu nachádzajú dve tlačidlá, *Play* a *Quit*, ktorých funkcionality sú definované ich názvom.

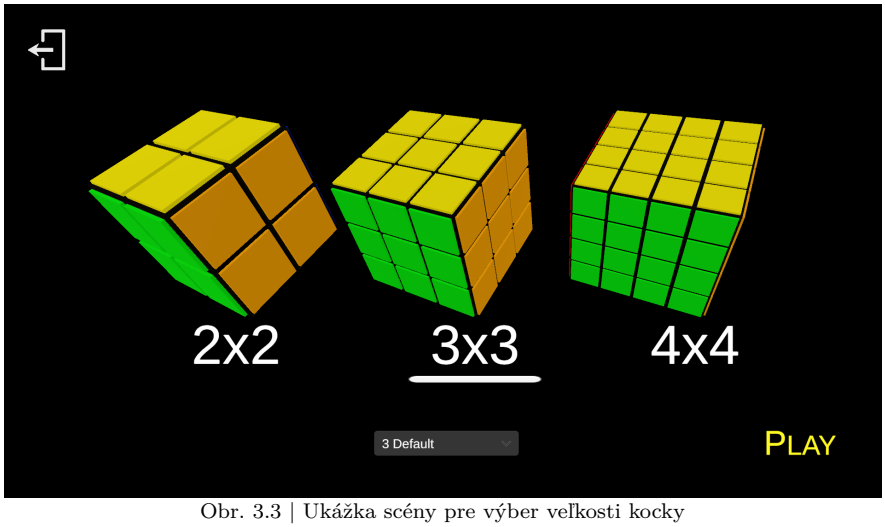


Obr. 3.2 | Ukážka hlavného menu aplikácie

3.1.2 Menu výberu hlavolamu a riešiča

Po stlačení tlačidla *Play* v hlavnom menu sa užívateľovi zobrazí *výber veľkosti* Rubikovej kocky a následný *riešič*, s ktorým bude naša aplikácia pracovať (*obr. 3.3*).

Potvrdenie voľby spustí *validáciu súborov*. V prípade chyby aplikácia vypíše presné znenie a názov súboru, v ktorom problém nastal.

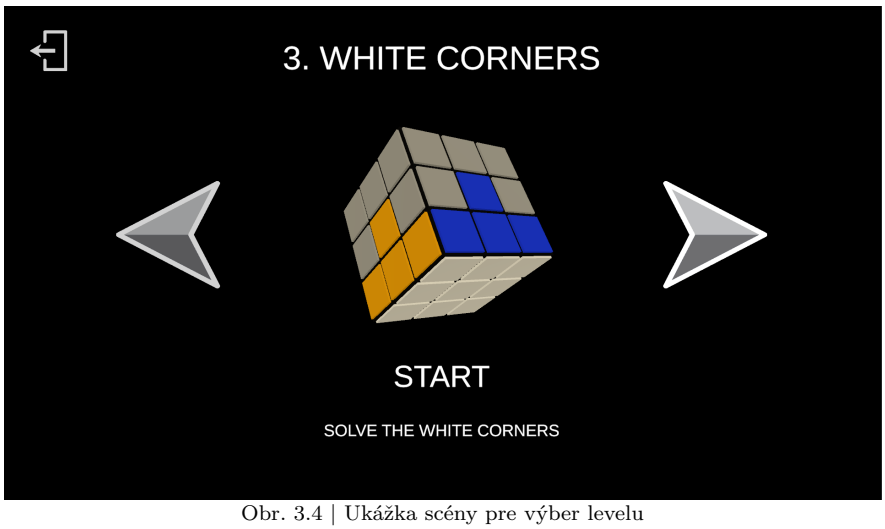


Obr. 3.3 | Ukážka scény pre výber veľkosti kocky

3.1.3 Menu s levelmi

V poradí treťou scénou je výber levelu (*obr. 3.4*). V strede obrazovky dominuje model Rubikovej kocky, ktorý je prispôsobený výberu užívateľa z predošlej scény. Záber všetkých jeho stien zabezpečuje kamera, ktorá rotuje primeranou rýchlosťou okolo.

Každý komponent v scéne je aktualizovaný z riešiacich súborov.



Obr. 3.4 | Ukážka scény pre výber levelu

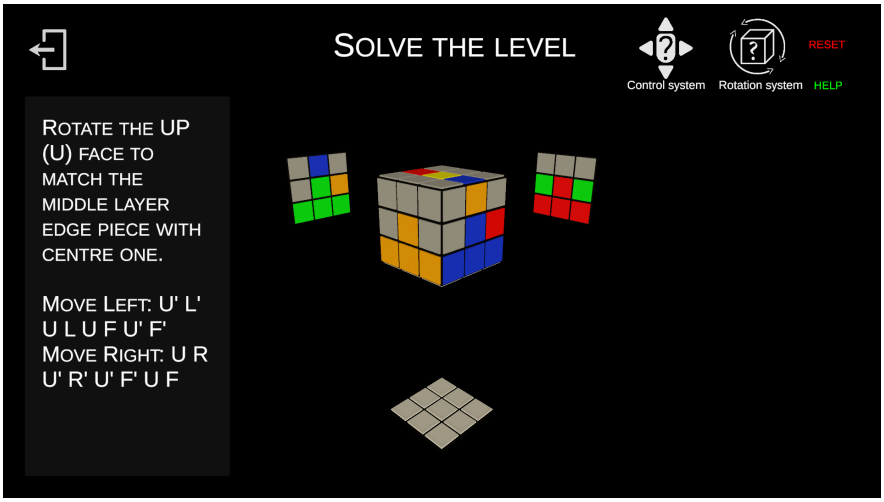
3.1.4 Level

Hlavnou scénou je samotný level, ktorý pozostáva zo štyroch edukatívnych častí. Najprv je užívateľovi poskytnutý tutoriál formou textu na ľavej strane

obrazovky (*obr. 3.5*).

Používané texty sú už zo spomínaného *návodu Rubiks*[1]. Jedninou zmenou bolo spojenie prvého a druhého kroku dokopy. Čo znamená, že v našom ukážkovom riešení požadujeme od užívateľa, aby rovno poskladal tvar *Daisy*. Rozdeľovanie tohto kroku na dve fázy, nám prišlo zbytočné.

Po splnení tutoriálu (ktorý je možné preskočiť), je užívateľ vystavený otestovaniu jeho znalostí. Počas celej doby skladania, má k dispozícii krátky súhrn algoritmov.



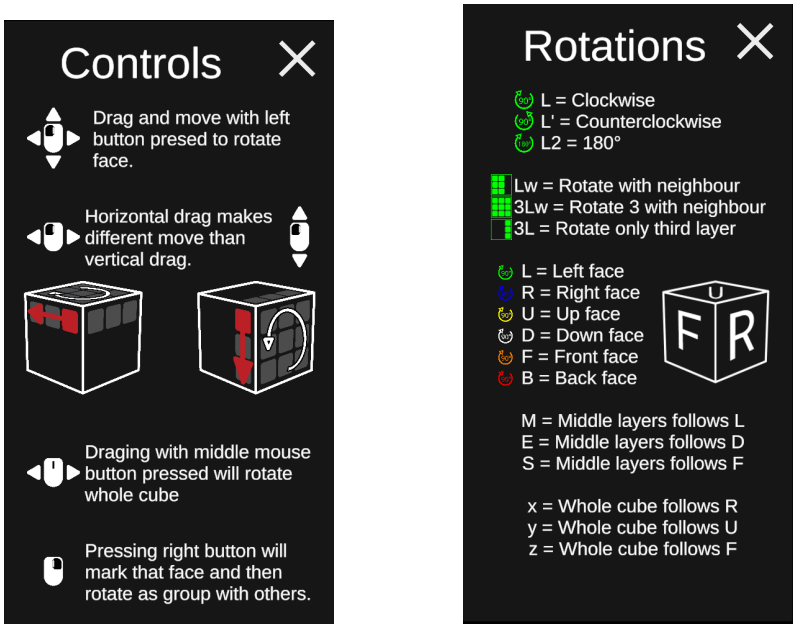
Obr. 3.5 | Ukážka scény levelu

Užívateľský návod

Nakoľko sa v aplikácii vyskytuje notácia ťahov, ktorá by nemusela byť pre užívateľa známou, pridali sme okno, ktoré je dostupné v každom leveli.

Rovnako sa tam nachádza otvárací panel, ktorý slúži ako vysvetlivka ovládania rotácií modelu. Ich vizuál je k nahliadnutiu *na obrázku3.6*.

[1] `https://assets.ctfassets.net/r3qu44etwf9a/6kAQCoLmbXXu29TTuArrk1/404118e1f9bfb6f9997157a284bbc572/Rubiks_Solution-Guide_3x3.pdf`



(a) Ukážka návodu ovládania (b) Ukážka návodu pre notáciu rotácií

Obr. 3.6 | Ukážka návodov v scéne levelu

Veľkou výhodou našej aplikácie je, že užívateľovi počas testovacej časti v leveli zakryjeme farebné plôšky, ktoré nie sú pre splnenie potrebné. Tým pádom sa hráč môže sústrediť na splnenie cieľu bez nejakých rušivých elementov.

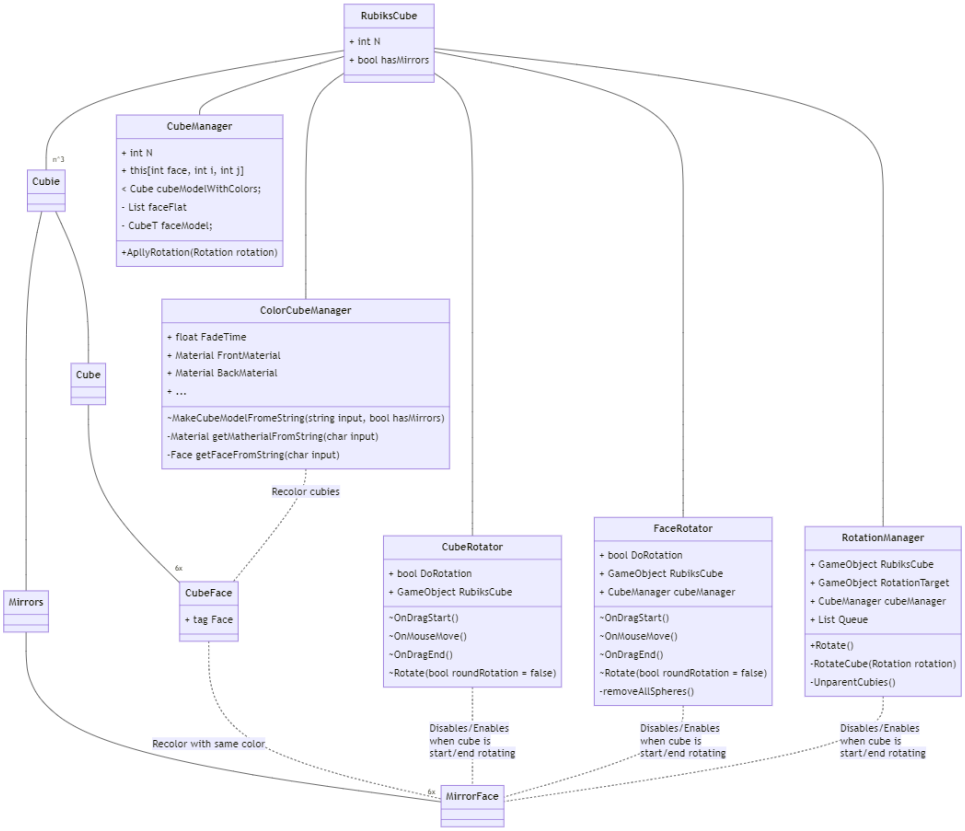
3.2 Komponenty užívateľského prostredia

3.2.1 3D model

Model Rubikovej kocky je v aplikácii vycentrovaný na súradnice (0,0,0). Z dôvodu jednoduchšieho výpočtu rotácií a hľadania malých kostičiek.

Vizuálne sme vypli tiene na modeli, aby sme vyzdvihli kontúry a kocka tým vyzerá viac čitateľnejšie.

Kocka je vytvorená z menších dielikov a poprípade zrkadiel. *Na obrázku 3.7* môžeme vidieť *class diagram* pre daný prefab so skryptami, pripojenými k nemu a ich funkcionalitami.



Obr. 3.7 | Class diagram pre model Rubikovej kocky

3.3 Užívateľom definované riešiče

Nakoľko sme chceli dopriať užívateľovi voľnosť navrhnutia vlastného riešiaceho algoritmu. Naša aplikácia dokáže pracovať so správne nadefinovanými súbormi. Ich štruktúru sme navrhli tak, aby bola čo najjednoduchšia na pochopenie a používanie.

Súbor sa skladá z troch častí – prvotné definície, Tutoriál a akcie. Posledná zložka definuje podciele, ktoré sú potrebné splniť k dosiahnutiu finálneho stavu. Presnú štruktúru a návod, ako dané súbory navrhnúť sme spísali *do kapitoly Užívateľská dokumentácia 5.5.*

Každý súbor definujúci level obsahuje povinný prvok uvádzjúci ťahy, ktoré vygenerujú rozmiešaný model. Uvedomujeme si, že táto komponenta sa môže zdať ako zbytočná, nakoľko by sme mohli použiť rotácie uvedené v časti *akcie*. Chceli sme však dopriať užívateľovi možnosť nakonfigurovať si postupy podľa seba.

Textový formát *.yaml* sme vybrali z dôvodu jeho jednoduchosti a prehľadnosti. Jeho výhodou je užívateľská prívetivosť, nakoľko neobsahuje žiadne znaky, ktoré by mohli používateľa „vystrašiť“. Príkladom je formát *.json*. Ten obsahuje veľké množstvo zátvoriek, ktoré môžu byť pre priemerného človeka zmätočné.

4. Implementácia

V tejto kapitole sa zameriame na celkové riešenie a implementáciu tejto práce. Vyberieme najzaujímavejšie časti spísaného kódu a pozrieme sa na ich funkčnosť.

4.1 Model

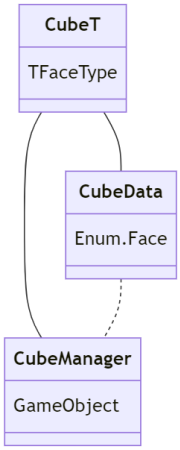
4.1.1 Datová štruktúra

Model je uložený v dvoch formátoch. V prvom si ukladáme konfiguráciu kocky z hľadiska farieb. To využívame napríklad pri kontrole poskladaného modelu.

Druhým spôsobom je zoznam GameObject, v ktorom udržujeme stabilnú pozíciu farebných plôšok 3D modelu, aby sme ich vedeli rýchlo vyhľadať a prefarbiť, keď bude potrebné.

Na oboch týchto komponentách je potrebné aplikovať *rovnaké rotácie*, aby sme udržovali presné prepojenie s 3D modelom.

K tomu slúži trieda *CubeT*, ktorá vykonáva rovnakú prácu s dvomi rôznymi typmi dát. Prvým sú CubeData, ktoré pracujú s enumom Face (farba), a cubeModel, ktorý je už spomínaný 2D model GameObjectov. Trieda *CubeManager* poskytuje prístup k obom zoznamom.



Obr. 4.1 | Class diagram triedy CubeT

4.1.2 Generovanie

Prefab výslednej kocky je generovaný za pomoci skriptu *CubeGeneratorScript.cs* v priečinku *Editor*. Ten vezme jednu malú Rubikovu kocku o veľkosti 1x1x1 (v prípade kocky so zrkadlami, vlastní ich aj táto predloha), a tú rozkopíruje, na správne súradnice. Strany nevidené užívateľom odstráni.

Prepojenie so zrkadlami

V prípade rotácií kockou je nutné následne skontrolovať, že priľahlé zrkadlá odrážajú správnu konfiguráciu zadných stien. Z hľadiska jednoduchosti

implementácie sú prepojené s modelom kocky. Znamená to, že pokiaľ užívateľ manuálne otočí modelom, otáčajú sa aj zrkadlá. Tie sme však pri ich presune vypli. Po dokončení rotácie sa znova obnovia.

4.1.3 Prepínanie scén

Presuny medzi scénami zabezpečuje skript *SceneLoader.cs*. Obsahuje metódu, ktorá podľa zadaného mena načíta scénu.

Nakoľko musíme zabezpečiť, že niektoré z našich dát budú k dispozícii scéne a uložené aj po prechode. Používame preto návrhový vzor *Singleton*[1], ktorým si uchovávame už spracované súbory, veľkosť vybranej kocky či index aktuálneho levelu.

4.2 Spracovanie Yaml súborov

Spracovanie *yaml* súborou zabezpečuje *asset*[2] importovaný z Unity Asset Store. Jednou z jeho výhod je odchytávanie syntaktických chýb, ktoré následne spracovávame a predkladáme užívateľovi.

Zanalyzovaný text vkladáme do verejnej datovej štruktúry, kde hlavnou triedou je samotný *Level*.

Ukážka datovej štruktúry levelov

```
public class LevelDefinitionDto
{
    [YamlMember(Alias = "Level")]
    public Level Level { get; set; }
}

public partial class Level
{
    [YamlMember(Alias = "Name")]
    public string Name { get; set; }

    [YamlMember(Alias = "Description")]
    public string Description { get; set; }

    [YamlMember(Alias = "Cube_Model")]
    public CubeModel CubeModel { get; set; }

    [YamlMember(Alias = "Cube_Shuffle_Moves")]
    public string[] CubeShuffleMoves { get; set; }

    [YamlMember(Alias = "Tutorial")]
    public Tutorial[] Tutorial { get; set; }
```

[1] https://gamedev.stackexchange.com/questions/116009/in-unity-how-do-i-correctly-implement-the-singleton-pattern/151547#151547

[2] https://assetstore.unity.com/packages/tools/integration/yamldotnet-for-unity-36292

```
        [YamlMember(Alias = "Actions")]
        public Action[] Actions { get; set; }
    }
```

4.2.1 Rotácie

Datová štruktúra

Rotáciu reprezentujeme trojicou osa, uhol a zoznam otáčaných vrstiev. Implementáciou je štruktúra s názvom *Rotation*, uložená v súbore *Types.cs*.

```
    public struct Rotation
    {
        public Axis axis;
        public int angle;
        public List<int> rotatedLayers;
    }
```

Spracovanie Singmasterovskej notácie

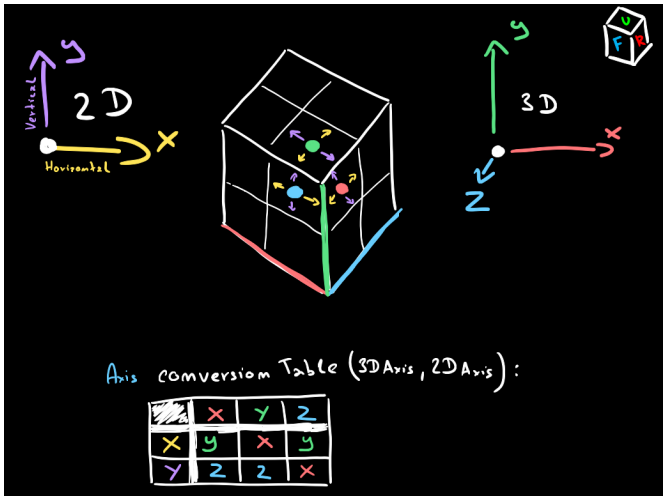
Aby náš program dokázal pracovať s notáciou rotácií použitou v jednotlivých súboroch, spracovávame ju do spomínanej štruktúry Rotation za pomoci *kompilovaného regulárneho výrazu*. Ten zabezpečuje vyšší výkon, nakoľko volanie spracovania textového reťazca je časté, hlavne pri parsovaní užívateľských súborov.

```
    public static Regex BuildRegex()
    {
        var opts = RegexOptions.Compiled
            | RegexOptions.Singleline;
        return new Regex
        (@"^(((\d?)([FBRLUD])(w?)('?)(2?))|([MES])('?)(2?)|([xyz])('?)(2
            ?))$", opts);
    }
```

Rotácie 3D modelu

Rotácie vyvolané užívateľom sa začínajú počítať hneď po kliknutí ľavým alebo stredným tlačidlom. Po zachytení vstupu od myši sa vyšle Ray, ktorý pri trafení plôšky modelu vráti hodnotu true. Následne vypočítame normálu, aby sme zistili na ktorej strane kocky bol klik vykonaný.

Aby sme korektne vykonali rotáciu dielikov, rozdelíme ich podľa súradníc v priestore a normály. Následne ich pridelíme objektu s názvom Rotator, ktorý sa nachádza na v strede. Podľa osi ťahu myši a osi získanej z normály vypočítame konečnú os otočenia.



Obr. 4.2 | Ukážka výpočtu rotácie 3D modelu

V prípade automatickej rotácie, ktorá sa využíva pri ukážke skladania Rubikovej kocky. Sú rotácie implementované jednoduchšie, a to za pomoci importovaného assetu DoTween.

4.3 Testovanie

Pri vývoji aplikácie sme si dali záležať na otestovaní viacerých algoritmických častí. Ich samotný kód je k nahliadnutiu v priečinku `Assets/Scripts/Tests`.

V *CubeTests.cs* testuje aplikáciu rotácie na dátovej štruktúre. Overujeme, že reprezentácia vykonaných rotácií je totožná s otočením reálneho modelu Rubikovej kocky.

Súbor *SolverTests.cs* obsahuje overenie algoritmu zloženia za pomoci Iterative Deepeningu na pomiešanom hlavolame.

OthersTests.cs testuje prevod reťazca rotácie za pomoci regulárneho výrazu do dátovej štruktúry *Rotation* a naopak.

Všetky testy sú prispôsobené a testované na všetkých implementovaných veľkostiach kocky, čím sme si overili ich korektnú prácu s modelmi párnych aj nepárnych rozmerov.

Testy sú spustiteľné za pomoci Unity editoru v okne *Test Runner*. Prikladáme ukážku testu zo súboru *OthersTests.cs*.

```
[Test]
public void ComplexParseTest()
{
    var input = "3Uw'2";
    var output = parser.Parse(input, 3);
    List<int> rotatedLayers = new List<int> { 0, 1, 2 };
    var expected = new Rotation(Axis.y, -180, rotatedLayers);
    Assert.IsTrue(RotationEquality(expected, output));
}
```

Navyše sme našu aplikáciu úspešne otestovali na zariadení s parametrami:

- OS: Windows 10.0.19043
- CPU : AMD Ryzen 7 4800H with Radeon Graphics
- GPU : NVIDIA GeForce RTX 3060
- RAM : 16GB

5. Užívateľská dokumentácia

V tejto kapitole poskytneme užívateľom návod, ako pracovať s naším projektom. Vysvetlíme presnejšie ovládanie a manipuláciu v priestore aplikácie a opíšeme postup vytvárania nového riešiča.

5.1 Minimálne systémové požiadavky

Užívateľ by mal disponovať zariadením s operačným systémom Windows 10. Pre pohodlnejšie ovládanie odporúčame používať myš.

5.2 Spustenie

Priložený balík je formátu .zip. Po extrahovaní celého obsahu získame priečinok s piatimi elementmi. Pre zapnutie aplikácie je potrebné spustiť súbor s názvom RubicsCube.exe.

5.3 Ovládanie modelu

Aplikácia je ovládaná výlučne iba myšou. V nasledujúcej tabuľke sú spísané tlačidlá s ich funkciami.

Tlačidlo	Funkcia
Ľavé	Otáčanie vybranou vrstvou kocky
Stredné	Otáčanie celým hlavolamom zľava doprava a naopak
Pravé	Výber viacerých rotovaných vrstiev

Tabuľka 5.1 | Tlačidlá myši pre ovládanie modelu kocky

Pre vykonanie rotácie hlavolamu je potrebné kliknúť na požadovaný dielik a potiahnuť myšou vo vami požadovanom smere. V prípade, ak by ste chceli otočiť viacerými vrstvami naraz, kliknite pravým tlačidlom na jednotlivé plôšky a potiahnite ľavým tlačidlom podobne ako s jednou vrstvou. Tento výber vám vizuálne vyznačí rotované dieliky.

V prípade zobrazenia nápovedy ovládania počas spusteného levelu zvoľte tlačidlo *Control System*.

5.4 Orientácia v aplikácii

Aplikácia vás privíta v menu s dvomi tlačidlami a vizuálnym modelom Rubikovej kocky. Stlačením *Play* sa dostanete do menu, kde sa vám zobrazí výber veľkosti hlavolamu a riešiča.

Tlačidlom *Quit* zatvoríte aplikáciu.

5.4.1 Výber riešiča a veľkosti modelu

Po stlačení tlačidla *Play* v úvodnom menu sa vás aplikácia opýta na výber Rubikovej kocky a k nej príslušného riešiča. Pre zvolenie modelu kliknite na biele tlačidlá s nápisom danej veľkosti (napr. 3x3).

Následne v dolnej časti obrazovky vyberte riešič, predvolene je nastavený postup *Default*. Potvrďte svoj výber tlačidlom *Play*.

5.4.2 Výber levelu

Po zvolení modelu a systému jeho riešenia sa vám zobrazí menu s levelmi. V strede obrazovky sa nachádza Rubikova kocka, ktorá odzrkadľuje finálny vizuál po úspešnom dokončení levela. Prechod cez jednotlivé fázy zabezpečujú šípky po jej stranách. Svoj výber potvrdíte tlačidlom *Start*.

5.4.3 Level

V strede obrazovky sa nachádza vami vybraný model Rubikovej kocky. V jeho okolí sú rozmiestné „zrkadlá“, ktoré ukazujú konfigurácie stien, ktoré v danom momente nie sú užívateľom videné.

V ľavej časti obrazovky sa nachádza panel, zobrazujúci text návodu, ktorý vás usmerní pre úspešné dokončenie levela. Opis jednotlivých tlačidiel sa nachádza *v tabuľke 5.2*.

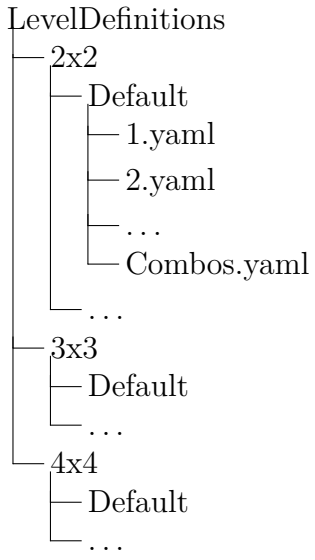
Tlačidlo	Funckcia
Control system	Zobrazí panel s nápovedou ovládania
Rotation system	Zobrazí panel s nápovedou o notácii rotácií
Next	Posun v tutoriáli
Play	Spustí ukážkovú sekvenciu rotácií
Skip tutorial	Preskočí tutoriál
Reset	Vráti kocku do poslednej splňujúcej konfigurácie
Help	Poskytne nápovedu

Tabuľka 5.2 | Tlačidlá v leveli

5.5 Vlastný riešič

V tejto podkapitole si vysvetlíme, ako si správne nadefinujeme vlastný riešič. Aby aplikácia bola schopná nájsť vami napísané súbory, musia byť uložené pod priečinkom na adrese:
`%appdata%\..\LocalLow\DefaultCompany\RubicsCube\LevelDefinitions`.
Jeho štruktúru si môžeme povšimnúť nižšie.

Štruktúra priečinka



Pod priečinkom *LevelDefinitions* sa nachádzajú tri zložky pre dané veľkosti hlavolamu. Aplikácia v momentálnom štádiu podporuje iba tri.

V prípade vypracovania vlastného riešiča je odporúčané vytvorenie vlastnej zložky na úrovni *Default*, v ktorej sa nachádza nami napísaný algoritmus vo formáte *.yaml*.

5.5.1 Spísanie levelu

V priečinku s riešičom sa nachádzajú už samotné fázy výpočtu, ktoré reprezentujú jednotlivé levely aplikácie.

Tie musia byť pomenované prirodzeným číslom počínajúc 1 až po vami vybraný počet. Povšimnite si príklad v štruktúre pre hlavolam o veľkosti dva.

Navyše každý riešič musí obsahovať súbor *Combos.yaml*. Ten si však rozoberieme neskôr.

Samotný level reprezentuje nejakú ucelenú časť riešiča, napríklad poskladanie jednej farby, vrstvy a pod.

Ako sme už spomínali, súbory sú písané vo formáte *.yaml*, ktorý je založený na oddeľovaní sémantických častí tabulátorom. Presnú štruktúru levela si môžeme povšimnúť nižšie. Upozorňujeme, že všetky prvky sú povinné. Výnimku tvoria prvky znázornené po pomlčkou. Napríklad Tutoriál slúži ako zoznam niekoľkých krokov (značených pomlčkami), ktoré obsahujú aspoň jednu z definícií – *Description*, *Cube_visualisation* a *Rotation_visualisation*.

Štruktúra súboru s levelom

- Level
  - Name
  - Description
  - Cube_Model
    - Final_Goal_Cube
      - up
      - left
      - front
      - right
      - back
      - down
  - Cube_Shuffle_Moves
  - Tutorial
    - -
      - Description
      - Cube_visualisation
        - up
        - left
        - front
        - right
        - back
        - down
      - Rotation_visualisation
    - ...
  - Actions
    - -
      - Description
      - Subgoal_Cube
        - up
        - left
        - front
        - right
        - back
        - down
      - Allowed_Rotations
      - Allowed_Combos
    - ...

Jednotlivé prvky sme kvôli prehľadnosti spísali spolu s ich funkciami *do tabuľky 5.3.*

Element	Popis
Name	Meno, ktoré sa zobrazí v menu s levelmi
(Level) Description	Popis levelu, ktorý sa zobrazí v menu s levelmi
Final_Goal_Cube	Vizuál Rubikovej kocky po dokončení levelu
Cube_Shuffle_Moves	Rotácie použité pri vygenerovaní kocky
(Tutorial) Description	Popis kroku tutoriálu
(Tutorial) Cube_visualisation	Konfigurácia pre prefarbenie modelu
(Tutorial) Rotation_visualisation	Sekvencia ťahov, ktoré sa vykonajú po vykreslení
(Actions) Description	Opis, ktorý bude zobrazený užívateľovi v čase riešenia daného podcieľa
(Actions) Subgoal_Cube	Podcieľ
(Actions) Allowed_Rotations	Povolené rotácie pre splnenie zadaného podcieľa
(Actions) Allowed_Combos	Povolené kombá pre splnenie zadaného podcieľa

Tabuľka 5.3 | Prvky levelu yaml súboru

Priblížme si ešte presnejšie zadávanie niektorých zo spomínaných prvkov.

Zadávanie rotácií

Všetky zadávané rotácie dodržujú Singmasterovskú notáciu spísanú. Vizuálne spracovanie je k nahliadnutiu *na webovej stránke*[1].

V prípade otočenia viacerých vrstiev požívajte zápis pomocou písmena „w“. Príklad notácie v správnom poradí obsahujúci všetky vlastnosti – *3Fw'2*.

V prípade sekvencie viacerých rotácií je potrebné ich oddeliť od seba medzerou.

Existuje ešte jedna možná skratka, ktorá uľahčuje prácu so spisovaním riešiča. Reťazec „all“ symbolizuje množinu všetkých možných rotácií.

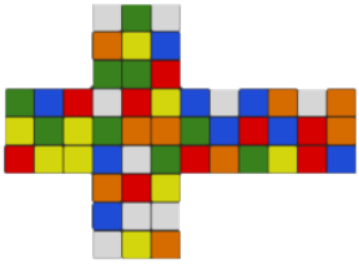
Jej ďalšou výhodou je, že pokiaľ je zadaná pod prvok *Cube_Shuffle_Moves*, generovanie kocky prebieha ako rozloženie modelu a následné jeho poskladanie až do predchádzajúcej fázy.

Spísanie vizuálu 3D modelu

Znaky reprezentujú prvé písmená názvov jednotlivých stien v anglickom jazyku. Príkladom je písmeno U (z ang. up), ktoré značí hornú stenu kocky. Reťazec si vieme lepšie predstaviť ako šesť častí reprezentujúcich stien v poradí (horná, ľavá, predná, pravá, zadná a spodná), kde každá obsahuje deväť znakov, ktoré odzrkadľujú stav samostatnej steny. Znak „X“ znamená, že dané farebné dieliky nás pre túto fázu nezaujímajú. Príkladom nám bude *obrázok 5.1*, ktorého reprezentácia by bola rovná:

[1] `https://jperm.net/3x3/moves`

```
Final_Goal_Cube:
    up:     DLD FUR LLB
    left:   LRB ULU BUU
    front:  DBU LFF RDL
    right:  RDR LRB BFL
    back:   FDF RBF UBR
    down:   FBU RDD DUF
```



Obr. 5.1 | Rozmiešaný 2D model Rubikovej kocky

Príklad súboru s levelom

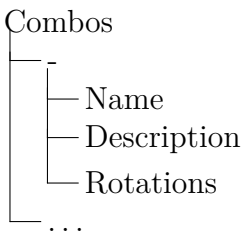
```
Level:
    Name: "DAISY"
    Description: "CREATE A~DAISY"
    Cube_Model:
        # Set final rotation of the cube, highlighted
           cubies on each face
        # and the level goal
        # U~= yellow, L = green, F = orange, R = blue, B
            = red, D = white, X = no color
        Final_Goal_Cube:
            up:     XDX DUD XDX
            left:   XXX XXX XXX
            front:  XXX XXX XXX
            right:  XXX XXX XXX
            back:   XXX XXX XXX
            down:   XXX XXX XXX
    Cube_Shuffle_Moves:
        - all

    Tutorial:
        -
            Description: |
                Look at the middle layer.
                Move EDGE pieces that
                have a~WHITE tile from
                the MIDDLE layer into
                the top layer.
            Cube_visualisation:
                up:     XXX XUX XXX
                left:   XXX XXX XXX
```

```
                front: XXX XXD XXX
                right: XXX FXX XXX
                back:  XXX XXX XXX
                down:  XXX XXX XXX
            Rotation_visualisation: R
        -
            ...

    Actions:
        -
            Description: |
                Move EDGE pieces that
                have a~WHITE tile into
                the top layer.

                Be careful not to bump
                out the white edges
                already in the daisy.
                Rotate the UP (U) face to
                move a~white edge out of
                the way before moving
                another white edge into
                the daisy.

                It is not necessary to place the cubies
                   in order.
            Subgoal_Cube:
                up:    XDX XUX XXX
                left:  XXX XXX XXX
                front: XXX XXX XXX
                right: XXX XXX XXX
                back:  XXX XXX XXX
                down:  XXX XXX XXX
            Allowed_Rotations:
                - all
                - ...
            Allowed_Combos:
                - Rotate_Edge_Piece_in_Daisy_Level
                - ...
        -
            ...
```

5.5.2 Spísanie súboru s kombami

Ako ste si už všimli, štruktúra súborov povoľuje zadanie tzv. komb do súboru *Combos.yaml*. Tieto prvky slúžia ako „premenné“ za rôzne sekvencie rotácií. Navyše sa používajú spolu s rotáciami vo výpočte riešenia hlavolamu.

Štruktúra súboru s kombami



Príklad súboru s kombami

```
Combos:
    -
        Name: Rotate_Edge_Piece_in_Daisy_Level
        Description: "Flip the Edge"
        Rotations: R' U~F'
    -
        ...
```

5.5.3 Užitočné rady a tipy

V tejto podkapitole vám ponúkneme užitočné rady a tipy, ktoré vám pomôžu pri spisovaní vlastného riešiča.

Popisy

Pri zadávaní popisov v prvku *Description*, začnite so symbolom „|“. Ten zabezpečí písanie viacriadkového opisu.

Podciele

Z hľadiska efektívnosti algoritmu, ktorý rieši Rubikovu kocku, sme boli nútení obmedziť počet rotácií pre zloženie zadaného podcieľa na konštantu päť.

Preto je nutné vytvoriť algoritmus, čo najefektívnejšie. K tomu napomáha vytvorenie čo najväčšieho množstva komb, ktoré sú rovné jednej rotácii.

Taktiež je výhodné rozdeľovať samotné „hlavné ciele“ levelu na menšie podciele, ktoré budú dosiahnuteľné rýchlejšie.

Odporúčame si vopred naštudovať nami napísané levely a držať sa podobných pravidiel.

Prepis Default riešiča

Neodporúčame užívateľovi prepisovať riešič v priečinku *Default*, nakoľko pri každom štarte aplikácie sa resetuje prepísaním na pôvodnú verziu.

6. Diskusia

V nasledujúcej kapitole zhodnotíme a porovnáme nami vytvorenú aplikáciu, s už existujúcimi projektami. Predstavíme ich nedostatky, ale aj pozitívne aspekty, ktoré sme sa snažili zakomponovať do nášho riešenia.

6.1 Existujúce projekty

Pri vypracovávaní tejto práce sme sa vžili do role začiatočníka, ktorý má za cieľ poskladať hlavolam Rubikovej kocky. Naše pátranie po zdrojoch nebolo náročné, nakoľko existuje viacero stránok, blogov alebo videí, ktoré túto problematiku plnohodnotne vysvetľujú.

To ale znamená, prečítať si niekoľko strán textu alebo niekoľkonásobné prehratie rôznych súborov. Navyše žiadna z týchto možností neposkytuje pomoc, keď sa hráč stratí v riešení, alebo ak by rád vrátil hlavolam do pôvodného stavu.

Čo sa týka webových aplikácií našli sme viacero podobných projektov. Nanešťastie väčšina z nich fungovala iba ako riešič Rubikovej kocky. Užívateľ mal možnosť zadať svoj vlastný vstup a program mu ukázal najrýchlejšie riešenie. Jedným z takých príkladov nájdeme *na webovej adrese*[1].

Pri našom prieskume sme objavili aplikáciu od spoločnosti *Rubik's* [2]. Tá sa ako jediná z mnohých iných snaží o interaktívne vysvetlenie postupu skladania tohto hlavolamu.

Obsahuje tri režimy hrania: komentovaná realita, virtuálny model a stopky. Práve prvý spomínaný mód je zameraný na pomoc začiatočníkom zorientovať sa v modeli a rotáciách. Pozitívnym aspektom je, že aplikácia podporuje nasnímanie vlastného hlavolamu pomocou fotoaparátu. Čo je rozdiel oproti našej aplikácii, kde je užívateľ nútený si svoj vstup naklikať manuálne. Každopádne sme sa týmto riešením inšpirovali a zakomponovali ho do možných budúcich plánov. Veľkou nevýhodou tejto aplikácie je, že hráčovi postup nevysvetľuje dostatočne. Užívateľ je navigovaný k riešeniu jednotlivými ťahmi. Vidí podciele, ktoré musí splniť, ale nie je mu poskytnutý tutoriál priamo v aplikácii.

Ako sme spomínali, prvou fázou každého vyučovacieho levela v našej aplikácii je vysvetlenie jednotlivých pohybov. Navyše každý pohyb má popis, ktorý presnejšie definuje, prečo danú rotáciu vykonávame. Ďalším rozdielom je, že mobilná aplikácia od spoločnosti Rubiks učí skladanie kompletne celého hlavolamu naraz. My sme však implementovali opakovanie každého podcieľa, aby si užívateľ osvojil dané sekvencie ťahov predtým, ako postúpi do ďalšieho levela.

Celkovo si myslíme, že z aplikácie vieme čerpať nápady do budúcna. Dizajn aplikácie je intuitívny a jednoduchý. Musíme však uznať pár negatívnych aspektov, ktoré vnímame. Aplikácia nie je koncipovaná primárne pre začiatočníkov. Vnímame fakt, že užívateľ by mal najprv preštudovať

[1] `https://rubikscu.be/`

[2] `https://apps.apple.com/gb/app/rubiks-official-cube/id1504482335`

písomný návod a až následne vyskúšať prácu s aplikáciou. Na druhú stranu projekt obsahuje širokú škálu režimov, ktoré uspokoja aj riešiča Rubikových kociek pokročilej kategórie.

Ďalšou skúmanou aplikáciou bola *Magic Cube*[3]. Jej pozitívnym aspektom je široká škála rôznych modelov Rubikovej kocky. Hlavolam má príjemnú a plynulú animáciu rotácií.

Na druhú stranu pri väčších kockách program neposkytuje možnosť otočenia viacerými vrstvami naraz ako naša aplikácia. Pri rotácii celého modelu nezaokrúhľuje pozíciu, čo môže spôsobiť chvíľkovú stratu orientácie v 3D priestore.

Výukový materiál spočíva prehratím videa z aplikácie *Youtube*, počas ktorého je užívateľovi poskytnutá virtuálna Rubikova kocka. Počúvanie a videnie uľahčuje zapamätanie postupu používateľom efektívnejšie ako čítanie textu v našej aplikácii. Nevýhodou tohto prístupu ale môže byť časté pretáčanie a hľadanie presnej stopáže.

Počas celej výukovej časti nie je užívateľovi poskytnutá žiadna nápoveda vo forme ďalšieho kroku.

[3] `https://play.google.com/store/apps/details?id=com.maximko.cuber&hl=cs&gl=US`

Záver

Projekt je vytvorený za účelom zjednodušenia manipulácie a osvojovania si začiatočníckeho riešenia hlavolamu Rubikova kocka. Pri vývoji sme zohľadňovali viacero aspektov ako efektivita algoritmu či intuitívnosť prostredia. V dôsledku toho sa nám podarilo vytvoriť stabilnú aplikáciu, ktorá spĺňa zadané požiadavky.

Zanalyzovali sme rôzne typy riešičov. Nakoľko je aplikácia cielená pre užívateľov, ktorí s Rubikovou kockou majú elementárne skúsenosti, zvolili sme postup, ktorý presne toto spĺňa. Jeho úlohou je ukázať užívateľovi sekvenčný postup skladania. Dôsledkom je algoritmus prechádzajúci cez jednotlivé vrstvy. Výsledkom je rozdelenie zložitého problému na etapy, ktoré si užívateľ lepšie osvojí.

Výpočet riešenia hlavolamu zabezpečuje algoritmus Iterative Deepening. Ten za pomoci štyroch parametrov vyhľadá riešenie a vráti sekvenciu rotácií, ktorá poskladá model do požadovaného stavu.

Prostredie aplikácie spĺňa požiadavku intuitívnosti. Užívateľovi je poskytnutá pomoc vo forme návodných panelov. Tie sú prítomné v každom leveli, vďaka čomu do nich môže užívateľ kedykoľvek nahliadnuť.

Jednotlivé interaktívne prvky sú rozmiestnené po celom priestore. V centre je situovaný 3D model hlavolamu, ktorý je prepojený so zrkadlami. Tie poskytujú užívateľovi rozhľad na tri zadné steny kocky. Manipulácia s 3D modelom pripomína skladanie reálnej Rubikovej kocky. Toho sme dosiahli implementáciou všetkých možných spôsobov rotácií, akých sa v realite dá naskytnúť. Užívateľovi je dovolené vybrať si viacero vrstiev a otočiť s nimi naraz. Rovnako je povolená aj rotácia kompletného modelu.

V okolí sa nachádza viacero tlačidiel, ktoré responzívne reagujú na interakciu od užívateľa. Navyše väčšina z nich obsahuje popis ich funkcie, čo napomáha intuitívnosti programu. Prostredie dopĺňa textové pole, ktoré naviguje používateľa cez jednotlivé kroky.

Mysleli sme aj na pokročilejších užívateľov, a preto sme naimplementovali možnosť napísania si vlastného riešiča. Postup a rady sme spísali do užívateľskej dokumentácie. Aplikácia užívateľov riešič otestuje a dodá mu spätnú väzbu s prípadnými chybami. Nakoľko podporujeme rôzne rozmery hlavolamov, užívateľovi je poskytnutá možnosť spísania riešičov pre každý z nich.

Počas celého vývoja aplikácie sme ju testovali Unity testami.

Plány do budúcnosti

Na projekte by sme radi pracovali naďalej, a preto sme spísali pár návrhov, ktoré by mohli byť implementované v budúcnosti:

- podpora viacerých jazykov
- pridanie prvkov, ktoré by motivovali užívateľa v používaní naďalej (skóre,

odmeny atď.)

- história ťahov
- podpora pre mobilné zariadenia
- sken kocky za pomoci fotoaparátu
- zameranie sa na pokročilejších hráčov (napr. pridanie stopiek)

Celkovo hodnotíme prácu s projektom pozitívne. Čas strávený pri implementácii nám pomohol zdokonaliť sa v prostredí Unity a v jazyku C Sharp.

Zoznam použitej literatúry

Demaine, E. D., Demaine, M. L., Eisenstat, S., Lubiw, A. a Winslow, A. (2011). *Algorithms for Solving Rubik's Cubes*. Springer, Berlin, Heidelberg. ISBN 978-3-642-23719-5.

Heise, R. (2007). Human thistlethwaite algorithm. *com*.

Joyner, D. (2002). *Adventures in group theory: Rubik's Cube, Merlin's machine, and Other Mathematical Toys*. Baltimore : Johns Hopkins University Press, https://archive.org/details/adventuresingrou0000joyn/page/7/mode/2up. ISBN 0-8018-6947-1.

Korf, R. E. (1997). *Finding optimal solutions to rubik's cube using pattern databases*. AAAI Press, http://dl.acm.org/citation.cfm?id=1867406.1867515. ISBN 0-262-51095-2.

Rokicki, T., Kociemba, H., Davidson, M. a Dethridge, J. (2014). *The diameter of the rubik's cube group is twenty*. SIAM Review.

Rubik's (2020). *YOU CAN DO THE Rubik's Cube Solution Guide*. Spin Master, https://assets.ctfassets.net/r3qu44etwf9a/6kAQCoLmbXXu29TTuArrk1/404118e1f9bfb6f9997157a284bbc572/Rubiks_Solution-Guide_3x3.pdf.

Thistlethwaite, M. (1981). *Thistlethwaite's 52 move algorithm*. com, https://www.jaapsch.net/puzzles/thistle.htm.

Tsoy, M. (2018). *Kociemba*. com, https://github.com/muodov/kociemba.

Zoznam obrázkov

Zoznam tabuliek

A. Prílohy

A.1 Zdrojové súbory

Zazipovaný súbor, ktorý obsahuje:

- spustiteľný súbor s názvom *RubicsCube.exe*
- zástupca adresy, kam patria všetky spísané riešiče
- Priečinok *Ukazkove solvery*, ktorý obsahuje nami spísané riešiče
- link pre gitlab repository s programátorsky okomentovaným kódom `https://gitlab.mff.cuni.cz/teaching/nprg045/majerech/monika-bosaniova-rubikovka/rubikovka`