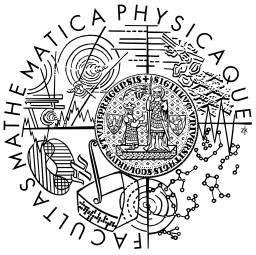


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

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 2022

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. Anne Yaghobovej, Natálii Potočekovej a Davidovi Nápravníkovi za pomoc pri štylizácii.

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. Pre lepšie pochopenie problému popisuje teóriu, myšlienky a históriu viacerých riešiacich algoritmov. Zameriava sa na dôkladný popis implementácie samotného aplikovaného postupu skladania. Približuje výzor prostredia aplikácie a interaktivitu rôznych elementov naprieč jednotlivými úsekmi a zahŕňa popis všetkých aplikačných komponent. 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. Spracováva spätnú väzbu od testovacích subjektov a navrhuje prípadné vylepšenia do budúcnosti.

Klíčová slova: rubikova kocka riešič tutoriál

Title: Rubic'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. For better understanding of the problem we describe the theory, ideas and the history of multiple solving algorithms. The implementation of all used components, chosen solving process for beginners, appearance of the application environment and the interactivity of different elements is 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, handles feedback from test subjects and suggests possible improvements for future development.

Keywords: rubic'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. Zanalyzujeme históriu rôznych riešičov. Vysvetlíme ich myšlienku, algoritmus a porovnáme ich efektivitu. Odôvodníme naše rozhodnutia z hľadiska implementácie jednotlivých komponent. Do najmenších detailov opíšeme prostriedky, ktorými napĺňame naše vopred stanovené ciele. Nakoľko sa jedná o aplikáciu zameranú pre začiatočníkov, vysvetlíme princípy dosiahnutia efektívnosti osvojenia 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.

Keďže sa jedná o aplikáciu vytvorenú pre ľudí, dali sme si záležať na patričnom otestovaní širokou vzorkou respondentov. Opíšeme ich spätnú väzbu a skúsenosti s naším projektom. Vysvetlíme, prečo sme sa rozhodli ich podnety zakomponovať a spôsob ich implementácie.

Nakoniec zhodnotíme celkovú prácu so softvérom. Jeho 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. 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. Tie si viac rozoberieme v kapitole 2.1.

1.1 Model Rubikovej kocky

Klasická varianta tohto puzzle pozostáva z 26 malých kociek poskladaných do kocky o rozmerov 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.

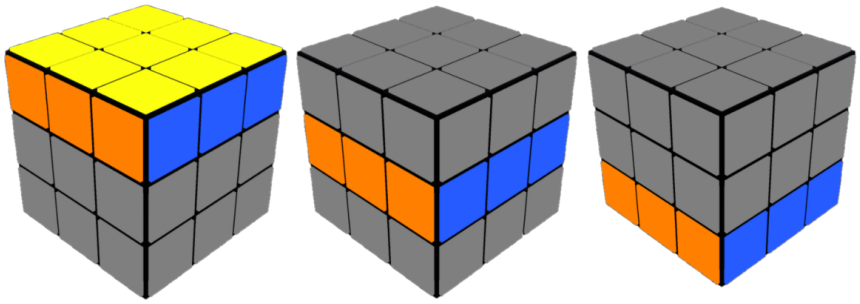
1.2.1 Typy dielikov

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

Názov	Počet farebných stien	Počet celkových dielikov
Stred	1	6
Hrana	2	12
Roh	3	8

Tabuľka 1.1 | Tabuľka s jednotlivými typmi dielikov hlavolamu

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



Obr. 1.2 | Vizualizácia stien

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.

V 2D modeloch použitých v tejto práci, zobrazujeme vždy tri steny hlavolamu. Zväčša sa budeme držať pravidla, že oranžová strana je vždy predná, modrá pravá a žltá horná. Notácie ostatných sú veľmi jednoducho domysliteľné. V prípade otočenia celého hlavolamu, uvedieme presné pozície strán, aby sme predišli zmäteniu čitateľa.

1.2.2 Rotácie strán

Pri manipulácii s Rubikovou kockou sú povolené iba rotácie jednotlivých strán. 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 nazvať. V celom programe používame tzv. **Singmasterovskú notáciu** (viď Joyner, 2002), ktorú sme upravili tak, že sme nahradili symbol ', ktorý reprezentuje otočenie steny v protismere hodinových ručičiek, za písmeno i. Rotácie, ktoré budeme používať sme spísali *do tabuľky 1.2.*

Názov rotácie	Smer rotácie aktuálnej strany
F	Otočenie prednej strany v smere hodinových ručičiek
Fi	Otočenie prednej strany v protismere hodinových ručičiek
F2	Dvojité otočenie prednej strany
B	Otočenie zadnej strany v smere hodinových ručičiek
Bi	Otočenie zadnej strany v protismere hodinových ručičiek
B2	Dvojité otočenie zadnej strany
R	Otočenie pravej strany v smere hodinových ručičiek
Ri	Otočenie pravej strany v protismere hodinových ručičiek
R2	Dvojité otočenie pravej strany
L	Otočenie ľavej strany v smere hodinových ručičiek
Li	Otočenie ľavej strany v protismere hodinových ručičiek
L2	Dvojité otočenie ľavej strany
U	Otočenie hornej strany v smere hodinových ručičiek
Ui	Otočenie hornej strany v protismere hodinových ručičiek
U2	Dvojité otočenie hornej strany
D	Otočenie spodnej strany v smere hodinových ručičiek
Di	Otočenie spodnej strany v protismere hodinových ručičiek
D2	Dvojité otočenie spodnej strany

Tabuľka 1.2 | Zoznam rotácií

Existuje ešte mnoho ďalších rotácií, ako napríklad otočenie strednej vrstvy, no tie sme sa rozhodli pre jednoduchosť vynechať. Navyše je možné ich nahradiť kombináciou niekoľkých otočení spísaných *v tabuľke vyššie 1.2*. Napríklad rotáciu stredného riadka v smere hodinových ručičiek, môžeme substituovať súbežnou rotáciou Ui a D.

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 Existujúce riešiče

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. Preto existuje veľa spísaných algoritmov, ktoré slúžia, ako pomoc začiatočníckym riešiteľom. V tejto kapitole si ukážeme pokročilé spôsoby riešenia, ale aj tie menej zložité.

2.1.1 Twistlewaitov algoritmnus

Tento algoritmus vytvoril matematik Morgen B.Thistlethwaite. Spôsob, akým rieši daný problém patrí k pokročilejším, navyše je praktický iba pre počítače.

Jeho cieľom nie je pri každom kroku umiestiť na správne miesto niekoľko dielikov, ale redukovať možné ťahy, ktoré nás dostanú k riešeniu. Postupne zmenšuje počet možností uloženia danej kocky, až ostane iba jediná pozícia.

Výpočet spočíva v štyroch krokoch. V prvom sú povolené všetky možné ťahy, ktorých je celkovo 18 *(viď Tabuľka 1.2)*. Cieľom je zafixovať orientáciu jednotlivých hrán.

V druhom sa zakazujú ťahy U, Ui, D, Di dovolené sú iba U2 a D2 (brané v oboch smeroch). To zodpovedá skutočnosti, že táto fáza fixuje orientáciu rohov a umiestňuje hrany strednej vrstvy.

Pre tretí krok sú to ekvivalentne ťahy pre prednú a zadnú stenu. Po tejto fáze by mali byť správne umiestnené hrany ľavej a pravej steny vo všetkých troch stredových vrstvách (prstencoch), rohy sú v tetrádach a platí pre ne párna permutácia.

Následne je možné skladať kocku iba za pomoci dvojitých ťahov, ktorých je celkom 6. Na konci tejto fázy je úspešné riešenie Rubikovej kocky. V najhoršom prípade tak nastane po 52 ťahoch.

Heise (2007) popísal spôsob, ktorý vychádzal z Thistlethwaiteho algoritmu. Jendotlivé etapy rozdelil, aby boli jednoduchšie osvojiteľné a zapamätateľné pre ľudí.

Nevýhodou naďalej ostáva jeho zložitosť. Postup nie je vhodný pre začiatočníkov, ktorí iba začínajú spoznávať kompozíciu a funkčnosť hlavolamu.

Navyše samotný Twistlewaitov riešič patrí k jedným z tých najstarších. Otvoril bránu viacerým novým algoritmom, a preto v dnešnej dobe existuje viacero o veľa efektívnejších postupov.

2.1.2 Kociemba

Jeden z najznámejších algoritmov na skladanie Rubikovej kocky je Kociemba. Svoje meno získal po svojom tvorcovi menom Herbert Kociemba. Jeho hlavnou myšlienkou je dvojfázový prístup, ktorý funguje v dvoch krokoch. Najprv vyrieši kocku do stavu s nejakou určitou vlastnosťou a následne jednoducho zloží zbytok kocky.

Rovnako ako Thistlethwaitov algoritmus je vhodný iba pre počítače. Jeho silnou výhodou je, že patrí k tým najrýchlejším a najefektívnejším, preto sme sa rozhodli ho zakomponovať do tejto práce, i keď nie ako hlavný riešič.

2.1.3 Známy začiatočnícky spôsob riešenia

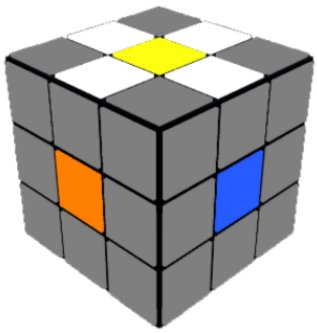
Aby sme vysvetlili začiatočníkom, ako poskladať tento hlavolam, musíme postupovať po jednotlivých krokoch. Budeme skladať kocku po vrstvách a ukazovať jednotlivé sekvencie ťahov, ktoré nás vždy posunú o vrstvu ďalej. Pod slovom vrstvy rozumieme tri riadky modelu. Základnou myšlienkou je ukázať užívateľovi sekvenčný algoritmus, ktorý ho každým krokom priblíži bližšie k cieľu, no nenaruší doterajší pokrok. Opisovaný algoritmus z dielne Rubik's (2020) je jedným z najznámejších.

Pri opise algoritmu, budeme používať spomínanú notáciu *v kapitole 1.2*. Spoločne prejdeme všetkými siedmimi krokmi. Výber steny, ktorou budeme začínať, nemá pre začiatočníka nejaký vplyv na efektivitu riešenia. Preto všeobecne väčšina algoritmov ľahkej úrovne vybrala predvolene bielu farbu. Opísaný postup nižšie nebude výnimkou.

1. Sedmokráska na hornej stene

Úlohou tohto kroku je zoznámenie užívateľa s pohybmi a funkčnosťou Rubikovej kocky. Hlavolam držíme žltou stenou nahor. Naším cieľom bude umiestniť biele hranové plochy k žltému stredu. V prvom kroku dostaneme všetky biele hranové kocky do hornej vrstvy. Nezáleží nám na ich rotácii a poradí. Následne otočíme všetky také kocky, ktorých biela stena nesusedí so žltým stredom. Diel, ktorý budeme chcieť zrotovať presunieme na pravú stenu, pomocou otočenia hornej steny (ekvivalentne sa jedná o niekoľkonásobnú rotáciu U). Aby sme kocku otočili, vykonáme danú sekvenciu ťahov [Ri, U, Fi].

Nasledujúci *obrázok 2.1* ukazuje stav kocky po prvej fáze.

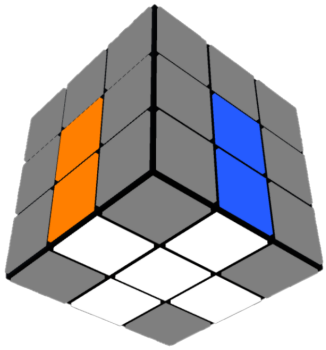


Obr. 2.1 | Stav kocky po prvom kroku začiatočníckeho algoritmu od Rubiks

2. Biely kríž so správne poskladanými hranovými kockami

Kocku držíme žltou stranou smerom nahor. Budeme párovať hranové kocky

vytvárajúce sedmokrásku so stredmi jednotlivých bočných stien. Používame pritom iba rotácie hornej steny. Keď sa dostaneme do stavu, kde stred bočnej steny je zhodný s hranou nad ním, použijeme dvojitú rotáciu danej strany. Tým dostaneme kocku na spodnú stenu. Po splnení tohto kroku so všetkými kockami, dostávame stav zobrazený *na obrázku 2.2.*



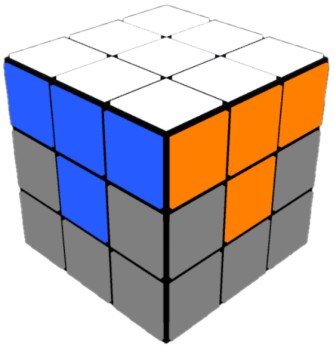
Obr. 2.2 | Stav kocky po druhom kroku začiatočníckeho algoritmu od Rubiks

3. Biele rohy

V tomto kroku si ukážeme, ako vrátime na svoje miesto rohy bielej steny. Hlavolam budeme držať tak, aby biela stena bola hore. Pre jednoduchšiu orientáciu predpokladajme, že sa umiestňovaná rohová kocka nachádza presne pod miestom, kam patrí na hornú vrstvu. Rubikovu kocku otočíme, aby tento roh pre nás patril prednej, pravej a spodnej stene. Následne nám poslúži tabuľka s postupnosťami ťahov.

Pozícia bielej plochy dieliku	Sekvencia pohybov
Pravá stena vľavo dole	Otočíme celou kockou o 90° doľava, následne vykonáme: D L Di Li
Predná stena vpravo dole	Di Ri D R
Spodná stena	F Di Fi D2 (následne vykonáme sekvenciu pre polohu na prednej stene vpravo dole)

Tabuľka 2.1 | Sekvencie ťahov pre poskladanie bielych rohov



Obr. 2.3 | Stav kocky po treťom kroku začiatočníckeho algoritmu

4. Druhá (stredná) vrstva

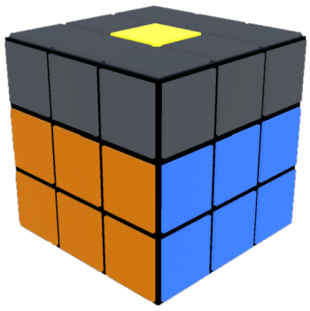
Táto pasáž je pre viacerých riešiteľov zlomová, nakoľko len málo z nich sa

dokáže odtiaľto pohnúť bližšie k vytúženému cieľu bez pomoci. Aby nás horná, už poskladaná vrstva nemýlila, je odporúčané otočiť kocku žltou stranou hore. Vezmime do úvahy túto zmenu pre nasledujúce sekvencie. Na to, aby sme poskladali druhý riadok, budeme používať tzv. ľavý a pravý algoritmus. Ich úlohou je umiestnenie hranového dielu medzi hornou (žltou) a prednou stranou, na svoje patričné miesto do druhého riadka, bez toho, aby porušili poskladanú bielu stenu. Prvým krokom je nájsť hranový diel na vrchnej stene, ktorý patrí druhej vrstve. Otočíme hornú stenu, aby farebná plocha umiestňovaného dielu, ktorá je situovaná na bočnej strane, bola zhodná s farbou stredu tej steny. Ak sa druhá farba tejto kocky zhoduje s ľavou (resp. pravou) stenou modelu použijeme ľavý (resp. pravý) algoritmus.

Názov	Sekvencia pohybov
Ľavý algoritmus	Ui Li U L U F Ui Fi
Pravý agoritmus	U R Ui Ri Ui Fi U F

Tabuľka 2.2 | Ľavý a pravý algoritmus, pre zloženie strednej vrstvy

V prípade, že sa už hranový diel nachádza v druhom riadku, ale jeho orientácia alebo pozícia nie je korektná, použijeme algorimus dvakrát. Prvé použitie ho nahradí iným dielom a druhé dosadí na správne miesto.

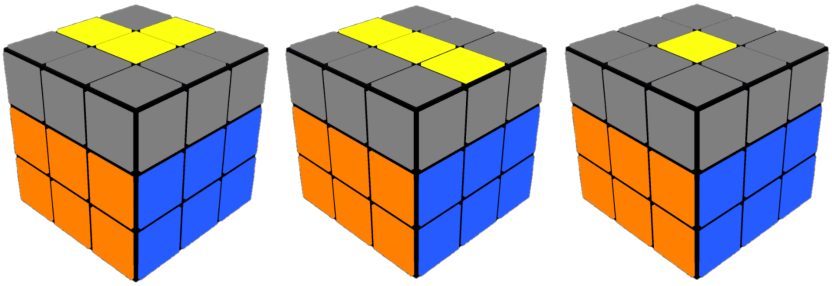


Obr. 2.4 | Stav kocky po štvrtom kroku začiatočníckeho algoritmu

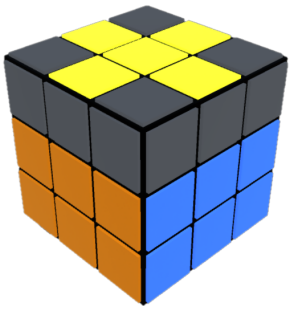
5. Žltý kríž

Teraz, keď máme poskladané prvé dve vrstvy, ostáva nám už iba posledná. Naším cieľom bude vytvoriť tzv. „žltý kríž“ na hornej strane kocky. Zanedbáme nesúladnosť farieb na bočných stranách. Agoritmus, ktorý budeme používať v tomto kroku je nasledovný: F U R Ui Ri Fi

Budeme sa riadiť vzorom, ktoré vytvárajú hranové kocky na žltej strane, vzhľadom na prednú. Otáčame celým hlavolamom, pokiaľ náš hlavolam nebude zhodný s niektorou možnosťou zobrazenou *na obrázku 2.5.*



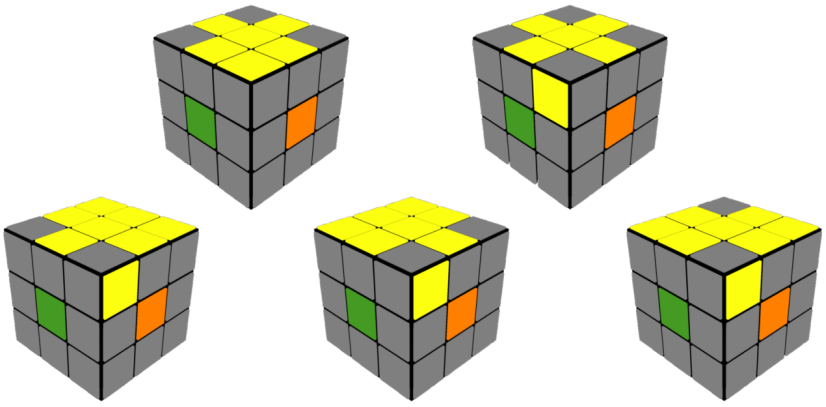
Obr. 2.5 | Možnosti stavov pred zložením žltého kríža



Obr. 2.6 | Stav kocky po piatom kroku začiatočníckeho algoritmu

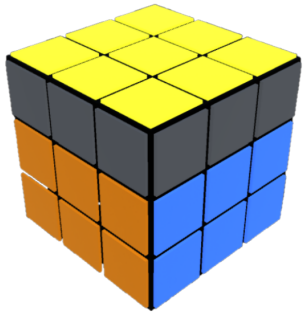
6. Dokončenie žltej steny

Po vytvorení kríža na žltej strane, musíme zabezpečiť presnú rotáciu hranových kociek. To znamená, že nás bude zaujímať iba vytvorenie žltej steny bez ohľadu na bočné plochy hornej vrstvy. Rubikovu kocku potrebujeme otočiť tak, aby spĺňala niektorý z prípadov, ktorý je vyobrazený *na obrázku 2.7*. Predpokladajme pre tento prípad, že oranžová strana je predná.



Obr. 2.7 | Možnosti stavov pred kompletným zložením žltej steny

Vykonáme nasledujúcu sekvenciu ťahov – R U Ri U R U2 Ri. Ak žltá stena nie je kompletná, nájdeme znova zhodnú kocku *z obrázka 2.7*, otočíme celou kockou a zopakujeme už spomínanú sekvenciu ťahov. Je možné, že tento postup budeme opakovať viackrát.



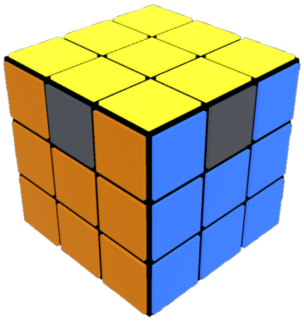
Obr. 2.8 | Stav kocky po šiestom kroku začiatočníckeho algoritmu

7. Umiestnenie žltých rohov na svoje pozície

V tomto kroku potrebujeme dostať všetky rohové kocky hornej vrstvy na svoje miesta. Naprv musíme nájsť kocky, ktoré sa už na svojom mieste nachádzajú. Pokiaľ sa nachádzajú vedľa seba, otočme celou kockou tak, aby sa obe nachádzali na zadnej stene. Vykonáme sekvenciu – [Ri, F, Ri, B2, R, Fi, Ri, B2, R2, Ui].

Ak sú správne dosadené rohy umiestnené diagonálne (nie vedľa seba), použijeme sekvenciu pokiaľ sa nedostaneme do stavu, kedy budeme mať dva susedné diely. Pokračujeme podľa už spomínaného postupu.

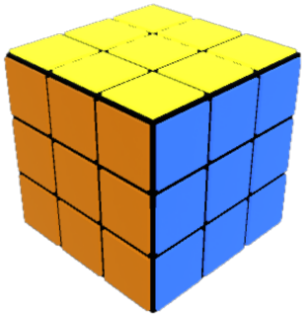
Ak sa naša kocka nezhoduje *s obrázkom 2.9*, opakujeme algoritmus tohto kroku znova.



Obr. 2.9 | Stav kocky po siedmom kroku začiatočníckeho algoritmu

8. Pozícia žltých hrán

Teraz nám už iba stačí presunúť žlté hrany na svoje miesto. Kocku budeme držať tak, aby zadná stena bola kompletná. Následne sa rozhodneme či chceme ostatné tri dieliky otočiť medzi sebou vo smere hodinových ručičiek (resp. proti smere). Podľa smeru použijeme rotácie [F2, U, L, Ri, F2, Li, R, U, F2] (resp. [F2, Ui, L, Ri, F2, Li, R, Ui, F2]).



Obr. 2.10 | Stav kocky po finálnom kroku začiatočníckeho algoritmu

2.1.4 Implementovaný algoritmus

Postup opísaný v časti 2.1.3 obsahuje častokrát kroky, kedy je užívateľ nútený zrotovať celú kocku, sme sa priklonili k úprave sekvencií ťahov jednotlivých fáz. Motiváciou pre túto zmenu bola snaha zabrániť užívateľovi získať zlé návyky, ktoré by mu v budúcnosti mohli spomaliť riešenie hlavolamu.

Preto je počas celého priechodu aplikáciou model Rubikovej kocky fixne orientovaný (až na jedinú výnimku v leveli Freestyle). Všetky používané sekvencie ťahov sú k tomu prispôsobené.

3. Dizajn

V tejto kapitole si priblížime celkový dizajn aplikácie. Nahliadneme ako sme postupovali pri rišení jednotlivých aspektov a zdôvodníme naše rozhodnutia.

Hlavným cieľom aplikácie bolo vytvorenie intuitívneho prostredia v programe Unity, ktoré zabezpečí rýchlejšie osvojenie algoritmu skladania Rubikovej kocky a dopraje príjemný užívateľský zážitok.

3.1 Užívateľské prostredie

Užívateľské prostredie sme sa snažili ladiť do tmavých farieb. Jedným z hlavných dôvodov prečo sme volili túto cestu je, že sme chceli uľahčiť používateľom pohľad a manipuláciu s aplikáciou. Ďalšou výhodou je vyniknutie farebných strán modelu kocky. Chceli sme zamedziť splynutiu hlavne svetloladených stien s pozadím. V celom prostredí aplikácie prevyšuje tmavá šedá až čierna farba s bledožltými a bielymi komponentami.

Počas celého používania aplikácie užívateľa sprevádza neutrálna hudba.

3.1.1 Prostredie

Samotná aplikácia je rozdelená na tri scény, úvodné menu, zoznam levelov a samotné levely. Scénou rozumieme časť programu s jednotným obsahom. Každá obsahuje unikátne samostatné elementy, ktoré ju robia jedinečnou. V nasledujúcich častiach si jednotlivé scény priblížime bližšie.

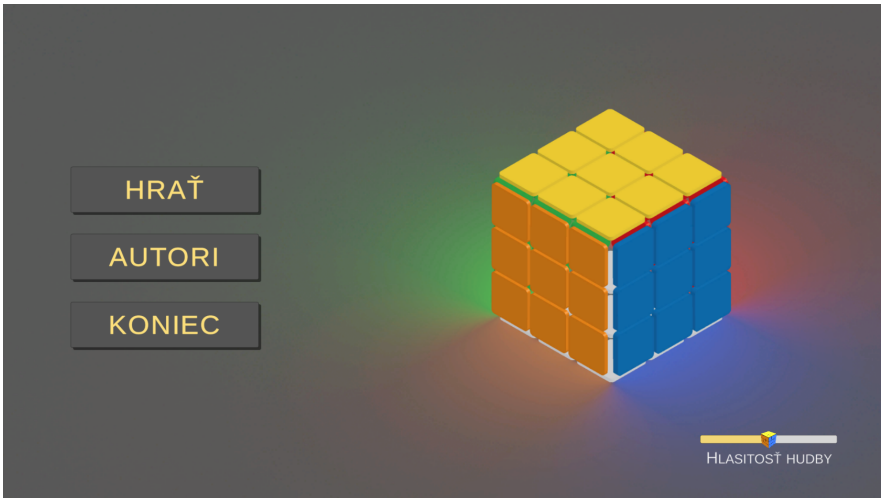
Úvodné menu

Po zapnutí aplikácie, užívateľa privíta hlavné menu, ktorého dôležitou súčasťou sú tri interaktívne tlačidlá a jeden posuvný prvok pre zmenu hlasitosti hudby. Funkcie jednotlivých tlačidiel sú opísané *v tabuľke 3.1*

Názov tlačidla	Funkcia tlačidla
Hrať	Prepne sa do scény so samotými levelami
Autori	Zobrazí autorov
Koniec	Zatvorí celú aplikáciu

Tabuľka 3.1 | Funkcie tlačidiel v menu

Celá kompozícia má ako pozadie tematický obrázok s Rubikovou kockou.



Obr. 3.1 | Dizajn menu aplikácie

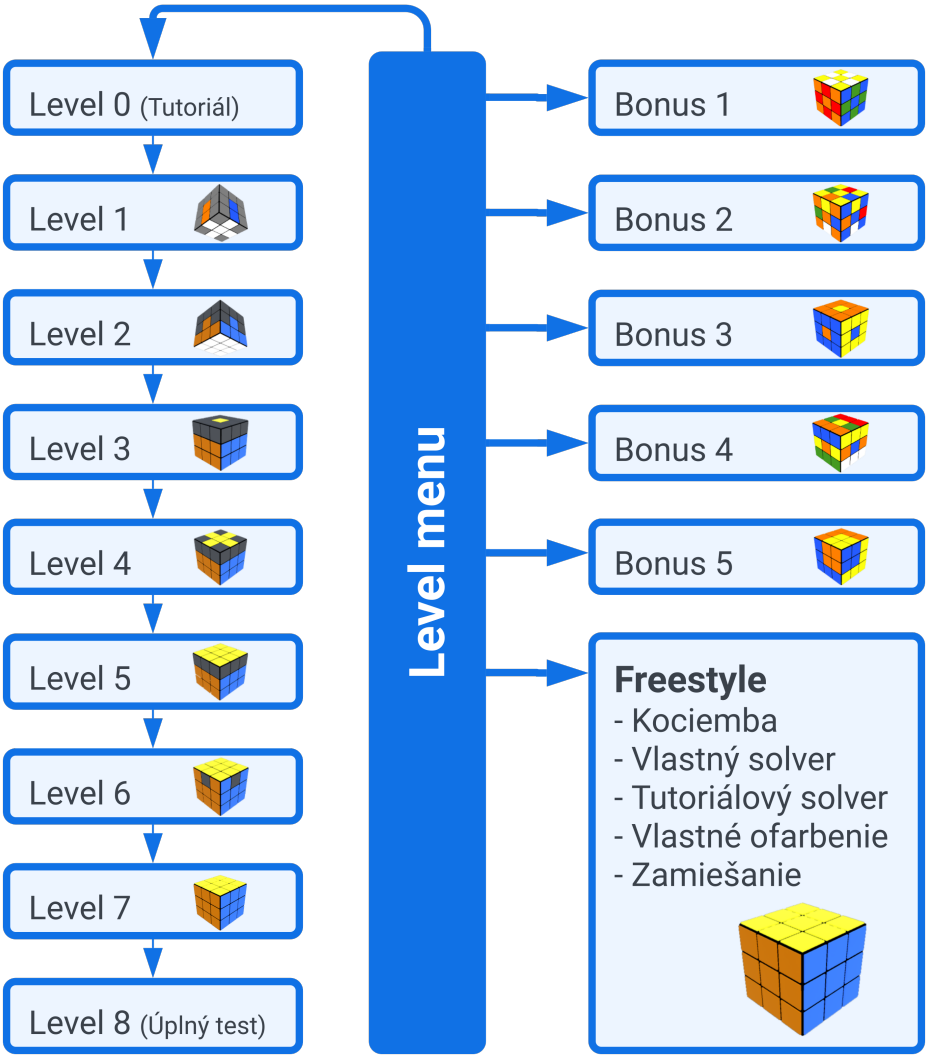
Menu s levelami

Z hľadiska lepšieho osvojenia si algoritmu, je aplikácia rozdelená do niekoľkých levelov. Tie sú všetky usporiadané v jednom kompaktnom menu. Vizuál je ukázaný *na obrázku 3.2*

Obr. 3.2 | Ukážka plne odomknutého menu s levelami

Dizajn výberu levelov pripomína polaroidové fotografie. Vizuálne spracovanie pôsobí intuitívnejšie, nakoľko užívateľ vidí, čo sa v danej fáze naučí. V jednotlivých „fotografiách“ sú zobrazené ikony alebo obrázky Rubikovej kocky, ktoré symbolizujú cieľ každej pasáže.

Celé menu je vizuálne rozdelené do dvoch úsekov. Pre jednoduchší prehľad sme vytvorili *nasledujúci obrázok 3.3.*



Obr. 3.3 | Vizualizačná mapa levelov v menu

Prvá časť je zameraná na výklad a výučbu skladania Rubikovej kocky. Hráč postupne prechádza jednotlivými etapami, ktoré na seba nadväzujú. Presnejšie sa jedná o leveli s názvom „Level 0-8“. Ich opis nájdete *v tabuľke 3.2*

Názov levela	Popis
Level 0	Zoznámenie sa s Rubikovou kockou a prostredím
Level 1	Poskladanie bieleho kríža na spodnej stene
Level 2	Uloženie bielych rohov na svoje miesto
Level 3	Poskladanie stredového riadka
Level 4	Vytvorenie žltého kríža na hornej stene
Level 5	Uloženie žltých rohov tak, aby vytvorili kompletnú hornú stenu (bez ohľadu na bočné plochy)
Level 6	Výmena horných rohových kociek, aby sme ich dostali na svoje miesto
Level 7	Výmena horných hranových kociek, aby sme ich dostali na svoje miesto
Level 8	Poskladanie celej kocky od začiatku

Tabuľka 3.2 | Zoznam levelov, ktoré vysvetľujú algoritmus skladania

Druhá skupina levelov sa zameriava na výučbu zaujímavých spôsobov, ako rozložiť Rubikovu kocku. Jedná sa o takzvané „Bonusové levely“. V aplikácii ich nájdeme celkovo päť. Ich opis a vizuálne zobrazenie si môžeme povšimnúť *na obrázku 3.2*.

Posledný level, ktorý sa nachádza v tejto časti, nesie názov „Freestyle“. V tejto časti sme chceli hráčovi dopriať voľnosť, a preto sme mu poskytli viaceré možnosti pre manipuláciu s modelom Rubikovej kocky.

Viac informácií ohľadom implementácie levelov nájdete v kapitole 3.1.1

Level

Hlavnou komponentou každého levela je 3D model Rubikovej kocky, ktorý je umiestnený v strede celej scény. Užívateľovi je poskytnuté manipulovanie a rýchlou odozvou jeho akcií.

Interaktívne komponenty sú v každej fáze o niečo iné, ale dostatočne konzistenté, aby sme zabezpečili intuitívnosť. Pri ich rozmiestňovaní sme dbali hlavne na užívateľský zážitok a súľad medzi nimi samotnými. Jednootlivé prvky sa navzájom neprekrývajú, užívateľ dokáže sledovať viaceré súbežne a dizajnovo spolu ľadia.

Užívateľovi je hneď na začiatku vysvetlené formou krátkeho tutoriálu, ako jednotlivé komponenty fungujú. V prípade, že sa jedná o pokročilejšie prvky v neskorších leveloch, je každý opatrený textovým vysvetlením jeho funkcie.

3.1.2 Kamera

Prvotné spracovanie práce kamery bolo statické. To znamená, že kamera bola pevne ukotvená na svojej pozícii, kde snímala hornú a dve bočné strany kocky. To spôsobilo, že ak užívateľ chcel nahliadnuť na steny, ktoré v danom momente nevidel, buď bol nútený použiť 2D zobrazenie modelu alebo otočiť celou kockou. Vtedajšia rotácia kocky znamenala otočiť ju v ľubovoľnom z troch smerov o 90 stupňov, čo sme nepokladali za úplne intuitívne. Preto sme sa rozhodli pre lepšiu

variantu a iný pohľad na tento problém. Namiesto statickej kamery, sme ju urobili dynamickú. Teda inými slovami, aby sa nehýbala kocka, ale samotná kamera. Vďaka tomu mal užívateľ pocit väčšieho rozhľadu a pohybu okolo modelu. Ponúkli sme používateľom väčšiu voľnosť, no na druhú stranu sme zamedzili možnosť otočiť kocku iným smerom ako je predvolene, teda žltou stenou hore, preto sa kamera vždz vráti na svoje pôvodné miesto. Dôvodom bol riešiaci algoritmus a myšlienka zabrániť užívateľovi využívať možnosť ľubovoľnej rotácie okolo modelu. Ako sme spomínali v kapitole 2.1.4, chceli sme minimalizovať naučenie sa tohto návyku.

3.1.3 Svetlo

Jedno z prvých riešení bolo vytvorenie iba jediného statického zdroja svetla. Ten osvetľoval celú kompozíciu rovnomerne. Nakoľko sme v neskorších štádiách rozmýšľali nad vylepšením práce kamery, nemohli sme už ostať iba pri statickom osvetlení. Potreba osvietiť všetko každý element, na ktorý sa užívateľ práve pozerá, sa stala ešte dôležitejšou pri zmene kamery na dynamickú. Z tohto dôvodu, nám napadlo riešenie pripevnenia svetla na pozíciu kamery. Teda pri každom pohybe kamery, by sa spolu s ňou hýbalo aj svetlo.

3.2 Interaktívne prvky

Pre responzívnosť a uživateľsky príjemnú manipuláciu s interaktívnymi elementami sme použili rôzne efekty stmavnutia či zosvetlenia pri pohybe myšou v ich oblasti alebo pri kliknutí na ne. Väčšinu vizuálnych aspektov tlačidiel zastrešuje TextMeshPro renderer, ktorý nahrádza Unity UI Text.

Všetky komponenty sú ukotvené na stabilných miestach, ktoré sú vypočítané z hľadiska rohu aplikácie, pri ktorom sú najbližšie. Ich pevné pozície museli byť nastavené, nakoľko by užívateľ mohol meniť rozlohu okna aplikácie, a to malo za následok posun, ba až zmiznutie interaktívnych elementov z dosahu používateľa.

3.2.1 Model Kocky

Model kocky vznikal s myšlienkou jednoduchého vytvárania a pekného dizajnu. Celý objekt bol vymodelovaný z jednej kocky, ktorá je nakopírovaná 27-krát. Pri tomto procese, bolo potrebné dávať pozor, aby stred celého telesa bol vycentrovaný na súradniciach (0,0,0). Dôvodom je jednoduchšie vypočítavanie neskôr pridaných rotácií a práca s kamerou.

Jednotlivé dieliky sú obalené z každej strany šiestimi malými farebnými plochami. Iba niektoré sú však pre užívateľa viditeľné. Farebné plochy reprezentujúce strany modelu sú vždy z dvoch strán zmenšené o desať percent a z tretej o 90 percent. Výber, ktorá zo strán je zmenšená o viac ako tie ostatné dve, je odôvodnený jeho pozíciou. Celý dizajn má za následok vytvorenie reálnejšieho 3D modelu.

Materiál používaný na steny kociek je matný a farebne vybraný, aby bol čo najpodobnší originálu.

Model je predvolene otočený žltou stranou nahor.

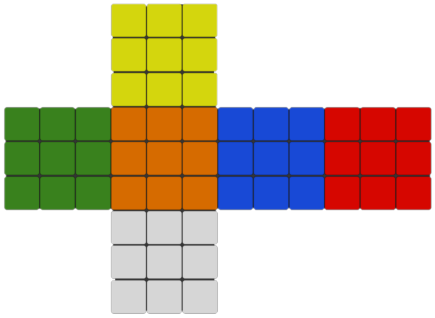
3.2.2 Rotácie

Pri zapracovávaní rotácií stien sme zohľaňovali dva myšlienkové postupy. Najprv rotácie jednotlivých stien fungovali tak, že stačilo kliknúť na stred strany, ktorú chcel otočiť a následne potiahnuť rovným pohybom smerom mienenej rotácie. Táto implementácia sa nám zdala málo intuitívna, nakoľko viacero testovaných užívateľov si neosvojilo dané ťahy a tým sme tento prvok pôsobil neintuitívne. Preto sme celý proces rotácií zmenili od základu. Pridali sme tlačidlá, ktoré dané rotácie obsluhujú. Každé z nich má priradený farbený popis, ktorý napovedá užívateľovi. Sú špeciálne usporiadané podľa typu a vzťahu k jednnotlivým stenám.

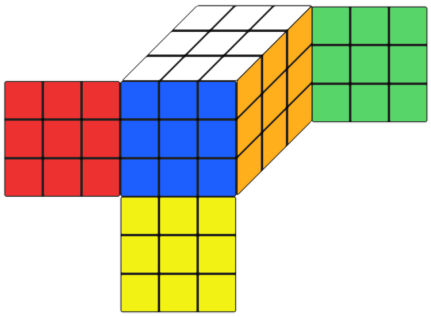
Nad nimi sa nachádzaju intuitívne ikony, ktoré zjednodušuujú vyhľadávanie v zoskupení tlačidiel. Navyše prejdením kurzorom po jednotlivých tlačítkach zvýrazní rotujúcu stranu.

3.2.3 2D mapa

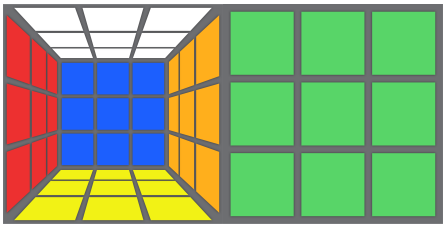
Pri zobrazovaní 3D modelu sme si uvedomili nedostatočnú orientácia užívateľa v priestore. Vedeli sme, že musíme pridať komponentu, ktorá by nám pomohla zvýšiť pohodlné používanie našej aplikácie. Rozmýšľali sme nad tromi zobrazeniami ukázaných nižšie. Nakoniec sme sa zhodli pre ten z nášho pohľadu najituitívnejší, a to *obrázok 3.4*.



Obr. 3.4 | Ukážka aktuálneho 2D zobrazenia modelu



Obr. 3.5 | Ukážka alternatívneho 2D zobrazenia modelu



Obr. 3.6 | Ukážka ďalšieho 2D zobrazenia modelu

Aktualizácia 2D modelu po následnom otočení niektorou zo stien kocky, bola prvotne vytvorená pomocou Rays. To znamenalo, že na každú malú farbenú plochu kocky smeroval jeden z nich. Toto riešenie sme neskôr zhodnotili ako veľmi neefektívne z hľadiska viacerých aspektov. V prvom rade pri rýchlejšej manipulácii s kockou, počas jej animácie, sa model častokrát zdeformoval.

Z toho dôvodu sme eliminopvali všetky vytvorené rays a snažili sa vymyslieť efektívnejší princíp. Momentálne, keď kocka otočí jednou zo svojich stien, vieme povedať, ktorými to bolo na základe interakcie užívateľa. Následne sledujeme iba otáčané elementy a po ich otočení zmeníme farbu na korešpondujúcom mieste 2D modela.

Aplikácia poskytuje užívateľovi prefarbiť farebné plochy modelu. Po zaznamenaní kliknutia sa jeho vybraná farba ihneď aplikuje aj na 2D model kocky.

3.2.4 Možnosti pomiešania modelu

V aplikácii sme naimplementovali viacero možností rozloženia hlavolamu, aby si sám užívateľ mohol vybrať, ktorý z nich mu vyhovuje najviac.

Tlačidlo rozmiešania

Pre uľahčenie a zrýchlenie učebného procesu užívateľov, sme pridali samostatné tlačidlo, ktoré im pripravý rozložený hlavolam. Stačí naň kliknúť a program sám vygeneruje sekvenciu ťahov, ktoré sa automaticky začnú vykonávať. Notácia ťahov je opísaná *v tabuľke 1.2*. Používateľ je schopný kedykoľvek zastaviť animáciu a presúvať sa v rade príkazov.

Pohyby a rotácie

Ako sme už spomínali v kapilote vyššie, pohyb okolo celého modelu kocky zabezpečuje kamera, ktorá rotuje po orbite okolo. Aby sme zjednodušili ovládanie aplikácie pre užívateľov, sme pridali funkcionalitu otáčania jednotlivých strán kocky. Steny sa môžu otáčať iba v rotáciách, ktoré neobsahujú ich stredné časti. Využíva sa oficiálne pomenovanie *z tabuľky 1.2*.

Pri otáčaní jednotlivých stien sa využíva princíp opísaný v kapitole (ref analýza rotácie). Užívateľ jednotlivými tlačidlami dokáže otáčať zvolenou stranou.

Farbenie presných častí kocky

Na rozdiel od rozloženia kocky pomocou rotácií a ťahov, sme pre uľahčenie prístupu zvolili aj iný spôsob namapovania ich aktuálneho problému, ktorý chcú

poskladať. Pre lepší prehľad, ako dostal užívateľ určité časti na svoje miesta. Z tohto dôvodu naša aplikácia podporuje zmenu režimu z rotácií na farbenie každej malej časti zvlášť podľa vybranej farby. Farba premaľovávanie je predvolene biela. Pre zvolenie a zafarbenie danej časti stačí užívateľovi vybrať farbu a kliknúť ľavým tlačidlom myši na danú časť. Tento proces funguje nasledovne. Každá kocka, ktorá reprezentuje farebnú časť strany, má nastavený tzv. box collider, ktorý zaznamenáva interakcie s daným objektom. Kocka, ktorá má k sebe tieto komponenty pridružené, má box collider vypnutý. Hlavným dôvodom je, že ak by užívateľ klikol veľmi blízko hranice, ktorá oddeľuje farebnú časť od čierneho podkladu, mohlo by sa stať, že by narušil jednotnosť a materiál základnej stavebnej jednotky modelu. Preto je vypnutá možnosť s ňou nejak interagovať.

Z hlavnej kamery sa nám vysiela laser. Každý Box Collider pre jednotlivé farebné plôšky je zväčšený o desať percent z každej strany, čo zabezpečuje vyplnenie celej steny kocky, a teda sa nestane, že by laser prešiel na druhú stranu.

Užívateľovi sme zamedzili možnosť prefarbiť viacero stien jednou farbou. Ide o veľmi jednoduché riešenie. Každá stredová plôška farebnej steny má nastavenú značku (angl. tag) s názvom Centre (prekl. stred). Preto, keď kamera vyšle laser na pozíciu, kde ukazuje kurzor myši a zasiahne objekt s touto značkou, bude ignorovať nasledujúcu zmenu materiálu, ktorá by za normálnych okolností nastala. Teda používateľovi nie je dovolené zmeniť farbu stredovej časti strany modelu.

3.2.5 Posun v sekvencii ťahov

Užívateľ je schopný kedykoľvek si zastaviť a znova pustiť animáciu modelu. Pri pozastavení je možné posúvať sa po jednotlivých krokoch dopredu a späť. Intuitívnosť dodáva aj voľba dizajnu tlačidiel. Všetky zvolené popisy boli vyberané a prispôsobované známym symbolom.

Pri iterovaní cez zoznam pohybov je užívateľovi ukázaný úsek aktuálnych ťahov, ktoré mu slúžia pre lepšiu orientáciu a predikciu nasledovných krokov. Používateľovi je ukázaný jeden spätný, aktuálny a päť nasledujúcich krokov algoritmu.

Obr. 3.7 | Dizajn tlačidiel, ktoré zabezpečujú posun v sekvencii ťahov

3.2.6 Reset modelu

Pod zobrazením 2D modelu kocky sa nachádza tlačidlo s názvom Reset. Jeho úlohou je vrátiť stav prostredia do pôvodného stavu.

3.3 Výučba jednotlivých levelov

Nakoľko hlavnou myšlienkou práce je vytvoriť stabilnú aplikáciu, ktorá má za cieľ efektíívne naučiť používateľa skladať hlavolam Rubikovej kocky, zakomponovali sme do aplikácie niekoľko komponent, ktorých vlastnosti napomáhajú užívateľovi osvojiť si daný problém.

Celý postup začiatočníckeho algoritmu skladania Rubikovej kocky sme rozdelili do jednotlivých etáp. Hráč poostupnee prechádza všetkými a po dokončení levelu sa mu odomkne nasledujúci. Tento postup sme zvolili z toho, dôvodu, aby používateľ nepreskočil nejaký fázu a aby sme mali pekný dohľad nad jeho krokmi.

Každý level obsahuje štyri fázy. Najprv sa hráčovi interaktívne ukáže postup, ktorý sa v tejto časti naučí. Počas celej ukážky sme zo scény eliminovali akékokľvek prvky, ktoré by mohli užívateľa rozptylovať. Navyše sme prerobili 3D model, aby poukazoval vždy iba na tie dieliky, ktoré sú v daný moment najpodstatnejšie.

Keď užívateľ dokončí návodnú ukážku, prichádza rad na neho, aby si postup vyskúšal. Používvateľ celkovo zopakuje algoritmus trikrát, no po každom dokončení mu zvyšujeme náročnosť. Najprv je model Rubikovej kocky zjednodušený a hráč vidí iba podstatné dieliky. Počas skladania hráčovi poskytujeme neustálu spätnú väzbu. Radíme presnými ťahmi, ale aj menším heslovitým popisom, čo daná sekvencia rotácii vykoná.

V druhej iterácii hráč pracuje už s reálnym modelom Rubikovej kocky, ktorý obsahuje všetky farby. Náročnosť tretej fázy spočíva v tom, že jedinou nápovedou je heslovitý text. Hráč už nedisponuje presnými ťahmi.

Užívateľ nie je nútený počas plnenia levelu dodržovať presný postup, ktorý ho učíme. Ak by sa pomýlil, alebo by sa dostal k riešeniu inou cestou, aplikácia ho upozorní, ale nebude mu brániť v pokračovaní. V prípade, že by sa v danom postupe stratil, je mu poskytnutá pomoc vo forme resetovacieho tlačítka, ktoré ho vráti do stavu, ktorý bol najbližší k finálnemu cieľu.

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 Riešič

Pri implementácii riešičov, sme zakomponovali dva algoritmy. Jedným je začiatočnícky princíp, ktorý sa snažíme vysvetliť užívateľovi. Druhý je pokročilejší. Jedná sa o algoritmus Kociemba, ktorý je spomínaný v kapitole 2.1.2, keďže nepatrí k tým najľahším a navyše je určený skôr pre počítače ako ľudské bytosti, sme ho pridali z dôvodu uľahčenia a rýchlejšieho vrátenia modelu do pôvodného stavu či rozšírenia povedomia používateľov o viacerých druhoch riešičov.

Priechod sekvenciou pohybov, ktoré riešiče vygenerujú zabezpečuje zakomponovaný návrhový vzor Iterator.

4.1.1 Začiatočnícky riešič

Myšlienka

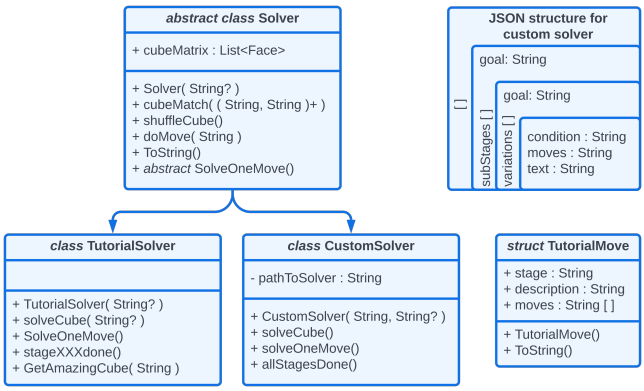
Hlavnou myšlienkou je poskladať Rubikovu kocku po krokoch. To znamená začať pomaly presúvať jednotlivé kocky na svoje miesto, bez toho, aby sa porušil doterajší pokrok. Teda sekvencia ťahov musí aktuálny stav hlavolamu priblížiť bližšie k riešeniu. Vstup od užívateľa musíme predtým skontrolovať, či náhodou nie je chybný. Ak by používateľ používal iba tlačidlo „Shuffle“ a rotácie jednotlivých strán, tak by táto kontrola bola zbytočná. Nakoľko sme pridali funkciu zafarbovanie jednotlivých plôšok, je istá pravdepodobnosť chyby zadávania vstupu od užívateľa. Preto musíme zamedziť takémuto stavu.

Vstup

Riešič príma vstup vo forme reťazca znakov. Script CubeState.cs prečíta 2D mapu a prenesie jednotlivé steny do premennej typu String. Príklad tohto postupu je opísaný aj v kapitole 5.7.1.

Reprezentácia

Riešič je obsiahnutý v samostatnej knižnici RC Solver. Prikladáme vizualizáciu objektového návrhu na obrázku nižšie.



Obr. 4.1 | Objektový návrh knižnice RC Solver

Výhody

Riešič obsahuje viacero metód, ktoré zabezpečia zloženie hlavolamu, iba do určitého štádia. Túto funkciu využívvame pri každom leveli, nakoľko potrebujeme zložiť model vždy do cieľového stavu predchádzajúceho levela.

Ďalším pozitívnym aspektom je.

4.1.2 Kociemba

Algoritmus Kociemba je spomínaný v kapitole 2.1.2 Na implementáciu tohto riešiča sme použili už napísanú knižnicu zo stránky gitHub `https://github.com/Megalomatt/Kociemba/tree/Unity`. Celá knižnica je už vopred pripravená pre prostredie Unity, čo nám veľmi uľahčilo prácu. Rovnako ju používame na zisťovanie či daná permutácia Rubikovej kocky je zložiteľná. Ak by užívateľ chcel použiť tento algoritmus, stačí ak zatlačí tlačidlo s názvom „Kociemba“, ktoré sa nachádza v leveli s názvom „Freestyle“.

4.1.3 Vyzualizácia návrhu vlastného riešiča

Ako už sme spomínali, každý užívateľ je schopný navrhnúť si svoj vlastný riešič. Pri spraccovávvaní vstupu sme používali dva balíčky Json.NET Schema a Json.NET.

Celkovú schému riešiča si môžeme prezrieť nižšie.

```
{
  "type": "array",
  "items": {
  "type": "object",
  "properties": {
    "goal": {
    "type": "string"
    },
    "subStages": {
    "type": "array",
    "items": {
      "type": "object",
      "properties": {
```

```
            "goal": {
              "type": "string"
            },
            "variations": {
              "type": "array",
              "items": {
              "type": "object",
              "properties": {
                "condition": {
                "type": "string"
                },
                "moves": {
                "type": "string"
                },
                "text": {
                "type": "string"
                }
              },
              "required": ["condition", "moves", "text"]
              }
            }
            },
            "required": ["goal", "variations"]
          }
          }
        },
        "required": ["goal", "subStages"]
        }
      }
```

Prikladáme aj jednoduchšiu schému.

```
      [{
        goal: String,
        subStages: [{
          goal: String,
          variations: [{
            condition: String,
            moves: String,
            text: String
          }]
        }]
      }]
```

4.2 Vyobrazenie levela

Scéna, v ktorej sa užívateľ dostáva do styku so samotným levelom, je vždy jednotná pre všetky levely. Využívame skript s názvom LevelInitialization.cs, v ktorom sú navrhnuté jednotlivé pasáže. Máme nadefinovanú hlavnú triedu

Level, od ktorej dedia ostatné levely. Každé dieťa obsahuje všetký reťazce, zobrazenia a postupy, ktoré bude potrebovať.

4.3 Testovanie

V tejto kapitole si priblížime spätnú väzbu, ktorú sme získali od testovacích užívateľov. Opíšeme objekty, ktoré boli implementované vďaka ich podnetu.

4.3.1 Testovanie začiatočníckeho riešiča

Ako sme spomínali vyššie, náš riešič je obsiahnutý v samostatnom projekte a do aplikácie je pridávaný formou knižnice. V tejto zložke rovnako nájdeme testy, ktorými sme si overovali, že náš riešič dokáže vyriešiť ľubovoľnú konfiguráciu Rubikovej kocky.

Testy sme spúšťali na jednom milióne rôznych hlavolamov. Po desiatich minutách sme si overili, že náš riešič mal sto percentnú úspešnosť. V nnajhoršom prípade doobehol do 310 ťahov.

4.3.2 Testovanie užívateľmi

Projekt bol testovaný desiatimi rôznymi používateľmi. Pri výbere respondentov sme dbali na ich rôznorodosť skúseností s Rubikovou kockou. Vďaka tomuto rozhodnutiu sme získali rozmanité dáta, ktoré sme následne implementovali do aplikácie.

Jedným z dôležitých návrhov k zmene bol presun jednotlivých interaktívnych komponent, tak aby poskytovali lepší užívvateľský zážitok.

Po otestovaní projektu, väčšina užívateľov nám posunula pozitívnu spätnú väzbu. Po prečítaní návodu, nemal ani jeden testovací používateľ problém s manipuláciou a pohybom v priestore. Myslíme si, že sme cieľ intuitívnosti splnili.

4.3.3 Odomknutie všetkých levelov

V prípade ďalšieho testovania jednotlivých levelov, poskytujeme spôsob odomknutia celého postupu. Príloha obsahuje súbor save.txt, ktorý postačí nahrať do zložky `%appdata%\..\LocalLow\DefaultCompany\RubicsCube`. Pri ďalšom spustení, budú všetky levely otvorené.

5. Užívateľská dokumentácia

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

5.1 Minimálne softvérové požiadavky

Užívateľ by mal disponovať zariadením s operačným systémom Windows 10.

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 miniatúrou Rubikovej kocky a názvom RubicsCube.exe.

Je pravdepodobné, že ak na vašom zariadení používate nejakú formu antivírusu, program sa vám nespustí automaticky. V tom prípade je potrebné zakliknúť, že dôverujete nášmu softvéru.

5.3 Ovládanie

Celá aplikácia sa ovláda iba pomocou myši.
Ľavé tlačidlo interaguje s jednotlivými prvkami.
Pravé tlačidlo otáča celým 3D modelom Rubikovej kocky.

5.4 Levely

Po zakliknutí tlačidla „Hrať“ v úvodnom menu sa vám objaví zoznam všetkých levelov. Tie sú rozdelené do dvoch kategórií.

Ľavá kategória vám postupne vysvetlí a naučí jednoduchý postup skladania Rubikovej kocky. Naopak druhá strana zoznamu vám poskytuje rôzne bonusové levely so zaujímavými vzormi, ktoré sa na tomto hlavolame dajú vytvoriť.

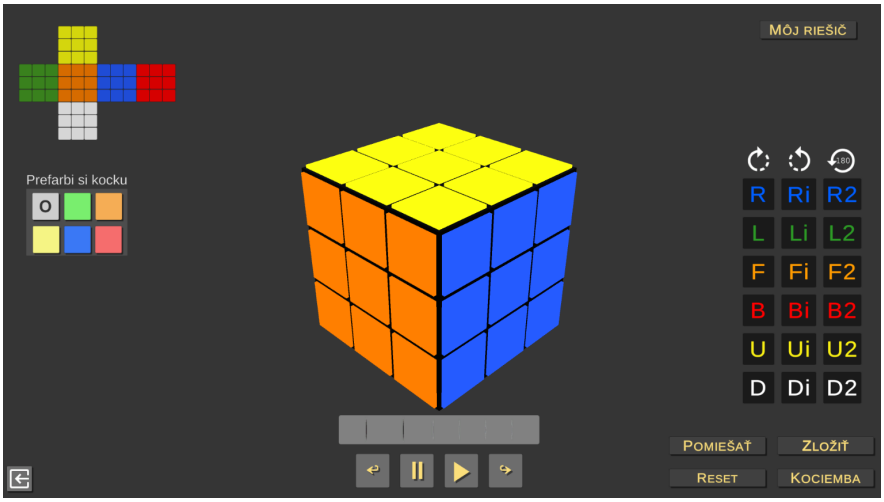
Posledný level s názvom „Freestyle“ vám poskytuje voľný priestor. Môžete tu zadať vlastnú konfiguráciu Rubikovej kocky. Následne ju dokážete poskladať vaším, začiatočníckym alebo pokročilým (Kociemba) algoritmom. To, ako si navrhnete vlastný postup, si vysvetlíme neskôr.
Každé tlačidlo poskytujúce výber levelu obsahuje signifikantnú farbu, podľa jeho doterajšej splniteľnosti. Zelené levely sú dokončené, žltý aktuálny ešte nedokončený, šedé uzamknuté.

5.4.1 Odomykanie levelov a ukladanie postupu

Na začiatku má každý užívateľ odomknuté bonusové levely a level s názvom „Level 0“. Po úspešnom dokončení každého levelu, ktorý sa zaoberá vyučovaním začiatočníckeho postupu skladania hlavolamu, sa odomkne nasledujúci a automaticky uloží postup.

5.5 Hlavné prvky

Na obrázku 5.1 môžeme vidieť všetky interaktívne komponenty nachádzajúce sa v aplikácii, ktoré sú opísané v odsekoch nižšie.



Obr. 5.1 | Stav kocky po prvom kroku začiatočníckeho algoritmu od Rubiks

3D model

V strede každého levela sa nachádza 3D model Rubikovej kocky. Otáčanie celým objektom vykonáte za pomoci konzistentného stlačenia pravého tlačidla myši a následným pohybom. Vo vyučovacích modeloch sa model vráti vždy na pôvodné miesto tak, aby bolo vidieť hornú (žltú), prednú (oranžovú) a pravú (modrú) stenu.

Rotácie jednotlivých strán zabezpečuje osemnásť jednotných tlačidiel napravo od modelu. Viac informácií nájdete v kaitole 5.5.

2D mapa

2D mapa je vždy aktuálnym zobrazením 3D modelu, ktorý napomáha k predstave o polohe jednotlivých dielikov. Nachádza sa vždy v ľavom hornom rohu.

Tlačidlá rotácií

Pozícia tlačidiel rotácií je vždy napravo od 3D modelu kocky. Farba textu tlačidla reprezentuje stranu, na ktorej vykoná rotáciu. Rovnako po prejdení kurzorom po niektorom z tlačidiel sa zvýraznia kocky veľkého modelu, ktoré budú zrotované.

Kontrolné tlačidlá a časová os

Navigačné tlačidlá sú zobrazené pod hlavným modelom Rubikovej kocky. Ich funkcie sú zaznamenané *v tabuľke 5.1*.

Tlačidlo so symbolom	Funkcia
Pauza	Zastaví animáciu kocky
Spustenia	Spustí animáciu kocky
Šípky doľava	Vráti sa o jeden ťah dozadu
Šípky doprava	Prejde o jeden ťah dopredu

Tabuľka 5.1 | Kontrolné tlačidlá a ich funkcie

Nad nimi sa nachádza časová os, ktorá poukazuje na posledný minulý, aktuálny a budúce ťahy.

Prefarbenie 3D modelu

Aplikácia podporuje zadanie vlastného vstupu Rubikovej kocky. V leveli Freestyle sa pod 2D mapou nachádza šesť farebných tlačidiel. Po výbere jedného z nich, stačí kliknúť na niektorú z farebných plôch 3D modelu. Pri nesprávne zadanom vstupe vás aplikácia upozorní.

Ďalšie tlačidlá

Zoznam ďalších tlačidiel a ich funkcie nájdete *v tabuľke 5.2*. Väčšina z sa nachádza iba v leveli s názvom „Freestyle“.

Tlačidlo	Funkcia
Reset	Vráti model Rubikovej kocky do pôvodného stavu
Zložiť	Spustí začiatočnícky riešič
Kociemba	Spustí Kociemba riešič
Môj riešič	Spustí riešič navrhnutý užívateľom
Tlačidlo Pomiešať	Pomieša model

Tabuľka 5.2 | Zoznam ďalších interaktívnych prvkov a ich funkcií

5.6 Permutácia hlavolamu

V aplikácii existujú tri rôzne spôsoby, akým užívateľ môže rozložiť model.
1. Tlačidlo: aplikácia pomieša model sama
2. Rotácie strán: užívateľ môže pomiešať model za pomoci rotačných tlačidiel
3. Vyfarbenie jednotlivých plôch: užívateľ môže zvoliť farbu a následne klikaním na jednotlivé plochy 3D modelu, namapovať jeho permutáciu hlavolamu.

5.7 Riešiče

Zavolanie riešiča môže užívateľ vyvolať pomocou troch tlačidiel menom „Zložiť“, „Kociemba“, „Môj riešič“. Tlačidlo Zložiť poskytne užívateľovi začiatočnícky princíp skladania, ktorý je obsiahnutý vo výučovacích leveloch.

Kociemba je zaujímavým doplnkom, ktorý veľmi rýchlo a efektívne dokáže dostať kocku do základného tvaru, no jeho princíp je zložitejší. Ak máte doma vlastný model Rubikovej kocky, ktorý chcete rýchlo zložiť do pôvodného tvaru, požite tento algoritmus, ktorý vám zaručene pomôže.

Aplikácia podporuje navrhnutie vlastného riešiča. Návod k jeho implementácii nájdete v nasledujúcej kapitole.

5.7.1 Navhrnutie vlastného riešiča

Vlastný riešič si nadefinujete pomocou súboru s názvom customSolver.json v priečinku – `%appdata%\..\LocalLow\DefaultCompany\RubicsCube`.

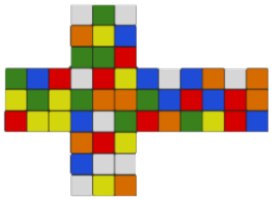
Vybrali sme ukážkovú časť súboru, ktorá reprezentuje zadávanú štruktúru.

```
[{
    "goal": "----U--------L--L-----F--F-----R--R-----B--B--
      D-DDD-D-",
    "subStages": [{
        "goal": "012=D 301=R",
        "variations": [{
            "condition": "010=DR 101=DR",
            "moves": "Ri U Fi Ui",
            "text": "Zeleno biela - Otocit nadol"
        }]
    }]
}]
```

Základom súboru je zoznam objektov, ktoré majú definovaný svoj cieľ (goal) vo formáte 54-znakového reťazca, ktorý definuje požadovanú konfiguráciu kocky. Jedná sa o reprezentáciu modelu pomocou špecifických znakov, ktoré udávajú polohu plochy v zloženom hlavolame. 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 steny v poradí (horná, ľavá, predná, pravá, zadná a spodná), kde každá obsahuje deväť znakov, ktoré odzrkadľujú stav samostatnej steny. Príkladom nám bude *obrázok 5.2*, ktorého reprezentácia reťazcom by bola rovná – DLDFURLLBLRBULUBUUDBULFFRDLRDRLRBBFLFDFRBFUBRFBU RDDDUF.

Znak pomlčky znamená, že dané farebné dieliky nás pre túto fázu nezaujímajú. Z ukážkovej časti zo súboru si môžeme povšimnúť, že daný reťazec reprezentuje *obrázok 2.2*.



Obr. 5.2 | Príklad rozloženej Rubikovej kocky v 2D zobrazení

Každý hlavný cieľ má definovaných niekoľko menších. Znova sa pozrieme na ukážku súboru. Podcieľ definovaný reťazcom – 012=D 301=R znamená, že budeme chcieť umiestniť diel patriaci spodnej a pravej stene na pozíciu definovanú danými trojcifernými číslami. Prvá cifra reprezentuje poradie steny, druhá riadku a tretia políčka v danom riadku. Indexujeme od nuly. Z uvedeného príkladu si rozoberieme trojčíslie 012 – horná stena, stredný riadok, posledný štvorček.

Posledný atribút, ktorý si musíme vysvetliť je trojica condition, moves, text. Condition definuje pozíciu dieliku, ktorý chceme dostať na lokáciu opísanú v podcieli. Spôsob akým ho tam vieme dosadiť je opísaný v „moves“. Táto premenná definuje sekvenciu ťahov. Posledný parameter s názvom „text“ je vyčlenený na pridanie komentára.

V prílohe tejto práce je súbor, ktorý vám poslúži ako nápoveda k definovaniu vášho vlastného riešiča. Vo vypracovanom príklade je spracovaný algoritmus, ktorý používame vo vyučovacích leveloch.

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, či 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 taký príklad nájdeme na webovej adrese: `https://rubikscu.be/`.

Pri našom prieskume sme objavili aplikáciu od spoločnosti Rubik's. 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 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.

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 aplikácia je cielená pre užívateľov, ktorí s Rubikovou kockou majú elementárne skúsenosti, zvolili sme dva ukážkové spôsoby riešenia. Jedným je algoritmus Kociemba, ktorý ukazuje rýchle a efektívne riešenie skladania kocky. Keďže sa táto metóda považuje za pokročilú, nespĺňa požiadavky jednoduchého postupu pre začiatočníkov. Preto sme pridali menej náročný riešič. 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.

Prostredie aplikácie spĺňa požiadavku intuitívnosti. Užívateľovi je poskytnutá pomoc vo forme úvodného levelu. Ten obsahuje popis všetkých komponent a ovládania programu. Po jeho splnení, by používateľ mal byť schopný pracovať vo vytvorenom prostredí.

Jednotlivé interaktívne prvky sú rozmiestnené po celom priestore. V centre je situovaný 3D model hlavolamu, ktorý je prepojený s 2D mapou. Tá poskytuje užívateľovi rozhlad na všetkých šesť stien kocky. V prípade, že by užívateľ chcel nahliadnuť na strany, ktoré momentálne nevidí, môže rotovať kamerou okolo stredu scény, čo mu poskytne pohľad na kompozíciu z iného uhla.

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. Používateľovi je umožnené nakonfigurovať si svoju vlastnú permutáciu hlavolamu, nakoľko môže svojvoľne otáčať všetkými viditeľnými stranami kocky a rovnako je schopný prefarbiť si jednotlivé plochy zvolenou farbou. Ak by sa užívateľ dostal do situácie, kde by chcel model vrátiť do pôvodného stavu, môže tak urobiť aj za pomoci resetovacieho tlačidla.

Prostredie dopĺňa textové pole, ktoré naviguje používateľa cez jednotlivé kroky. Nad ním sa nachádza skupina tlačidiel pre ovládanie animácie rotácií 3D modelu. Užívateľ je schopný kedykoľvek proces zastaviť, spustiť, či postupovať po jednotlivých krokoch.

Mysleli sme aj na pokročilejších užívateľov, a preto sme naimplementvali možnosť napísania si vlastného riešiča. Postup a rady sme spísali do užívateľskej dokumentácie.

Aplikáciu sme otestovali na viacerých užívateľoch. Skúmali sme aspekty ako intuitívnosť či efektivitu. Väčšina používateľov hodnotilo aplikáciu kladne. Zozbierané poznatky sme zakomponovali.

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
- vylepšenie začiatočníckeho algoritmu (napr. eliminovať rotácie, ktoré sa navzájom negujú)
- zlepšenie výkladu prvých levelov
- pridanie prvkov, ktoré by motivovali užívateľa v používaní naďalej (skóre, vylepšenia prostredia...)
- pridanie viacerých hlavolamov (napr. kocky 2x2, 4x4...)
- doimplementácia pokročilejších rotácií strán kocky
- 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. Rovnako bola prínosná práca s testovacími užívateľmi, ktorá vniesla do problematiky iný pohľad a pomohla doviesť aplikáciu do stavu, v akom sa nachádza práve teraz.

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 Rubics cube application
- programátorská dokumentácia v zložke Programers documentation, súbor index.html
- zástupce adresy, kam patria súbory save a customSolver
- súbor save, ktorý odomyká celú aplikáciu
- customSolver.json, ktorý je ukážkou užívateľského reišiča
- link pre gitlab repository `https://gitlab.mff.cuni.cz/bosaniom/rubics_cube`