

Střední průmyslová škola Třebíč

Studijní obor: Informační technologie

Třída: ITA4

Školní rok: 2023/2024 Lukáš Kurtin

Maturitní práce

Rubikova kostka

Profilová část maturitní zkoušky

Zadání práce

ABSTRAKT

Tvorba maturitní práce je jedním z velmi klíčových momentů při studiu. Kvalita zpracování její formální části je pak jedním z nejdůležitějších kritérií při jejím hodnocení. Cíl této práce je popsat jednotlivé kroky během tohoto procesu, doporučit postupy a vytvořit šablonu, která usnadní celý proces.

KLÍČOVÁ SLOVA

maturitní práce, šablona

ABSTRACT

The creation of a graduation thesis is one of the most crucial moments during studies. The quality of the processing of its formal part is then one of the most important criteria in its evaluation. The aim of this work is to describe the individual steps during this process, recommend procedures and create a template that will facilitate the entire process.

KEYWORDS

graduation thesis, template

PODĚKOVÁNÍ

Děkuji Natálii Pistovčákové, která nám pomáhá nepropadnout za plagiát.

V Třebíči dne 25. března 2024 podpis autora

PROHLÁŠENÍ

Prohlašuji, že jsem tuto práci vypracoval/a samostatně a uvedl/a v ní všechny prameny, literaturu a ostatní zdroje, které jsem použil/a.

V Třebíči dne 25. března 2024

podpis autora

Obsah

[Úvod 6](#_Toc162000289)

[1 Teoretická část 7](#_Toc162000290)

[1.1 Rubikova kostka 7](#_Toc162000291)

[1.2 Algoritmizace 7](#_Toc162000292)

[1.2.1 Značení tahů 7](#_Toc162000293)

[1.3 Programová struktura kostky 8](#_Toc162000294)

[1.3.1 Způsob 1 8](#_Toc162000295)

[1.3.2 Způsob 2 8](#_Toc162000296)

[1.4 Grafické vykreslení 8](#_Toc162000297)

[1.4.1 Ortodoxní projekce 9](#_Toc162000298)

[1.5 Použité technologie 9](#_Toc162000299)

[1.5.1 GitHub 9](#_Toc162000300)

[1.5.2 Windows Forms .NET Framework 9](#_Toc162000301)

[2 Praktická část 10](#_Toc162000302)

[Závěr 11](#_Toc162000303)

[Seznam použitých zdrojů 12](#_Toc162000304)

[Seznam použitých symbolů a zkratek 13](#_Toc162000305)

[Seznam obrázků 14](#_Toc162000306)

[Seznam tabulek 15](#_Toc162000307)

[Seznam příloh 16](#_Toc162000308)

Úvod

Jako moji maturitní ročníkovou práci jsem chtěl vypracovat projekt nad moje zkušenosti, u kterého bych strávil více času než na jakémkoliv jiném projektu, který jsem v průběhu školních let dělal. Ideální je proto projekt, který je náročný nejen z hlediska programátorského, ale i grafického.

Vybral jsem si proto simulaci Rubikovy kostky, které jsou mi již léta blízké a vždy mě fascinovala jejich funkčnost. Pro uskutečnění bude potřeba projít několik způsobů, jak naprogramovat základní strukturu a funkce kostky.

Graficky bude kostka zobrazena v krychlové síti a ve 3D prostoru s možností otáčení pohledu. K tomu bude zapotřebí nastudovat projekční matice, které umožňují překlad mezi 3D prostorem a 2D obrazovkou.

Jako vývojové prostředí použiji Microsoft Visual studio a jeho balíček Windows Forms .NET framework, ve kterém se programuje v jazyku C# a který ulehčuje vykreslování na obrazovku a zadávání uživatelského vstupu. Pro řízení projektu a správu verzí využiji platformu GitHub, více popsanou v teoretické části.

Konečný stav projektu si představuji jako intuitivní aplikaci, která bude obsahovat samotné funkce Rubikovy kostky, ale i teoretický návod na její složení pomocí nejpoužívanější metody CFOP (pojmenované podle čtyř kroků, které aplikuje). Také bych chtěl implementovat optimalizovaný algoritmus pro automatické složení kostky, popřípadě ho využít jako doplněk naučné stránky.

# Teoretická část

Rubikova kostka je považována za náročný hlavolam. Jakmile se však člověk snaží přijít na to, jak vlastně funguje její mechanismus, zjistí, že to není tak složité.

Kostek je několik variant: 2x2, 3x3, 4x4, 5x5 a mnohem více. Čísla určují, kolik jedna řada z rohu do rohu obsahuje kostiček. Tato práce se zabývá tou původní a nejznámější verzí: 3x3.

## Rubikova kostka

Velký počet lidí si o Rubikově kostce myslí, že tahy zaměňují barvy, což není plně pravda. Skutečně se totiž neotáčí s barvami, ale s jednotlivými kostičkami. Kostka jich má celkově 22, 8 rohů se třemi barvami a 12 hran se dvěma. Jednobarevné středy se do nich nezapočítávají, neboť jejich poloha se nikdy nemění. Jsou totiž středem osy každého otočení.

Každá kostička má distinktivní kombinaci barev, které za normálních okolností nemohou mít duplikát. Může se měnit pouze jejich poloha a tím i její otočení. Silou otáčet jednotlivé kostičky bez použití algoritmu učiní kostku neskladatelnou, dokud se neotočí zpátky.

## Algoritmizace

Jedním z prvků finální verze projektu by měl být algoritmus pro metodické složení kostky. Pokud se kostkou bude náhodně otáčet, složení by zabralo průměrně 43,252×1015 tahů. Existuje proto několik algoritmických sad, ze kterých se vytváří skládací metody. Nejznámějšími a nejpoužívanějšími z nich jsou CFOP a ROUX. Kvůli svojí znalosti o této metodice bude použita metoda CFOP.

Jako zdroj algoritmů se využije veřejná databáze algdb.net. [2]

### Metodika CFOP

Název označuje zkratky postupů, podle jakých se Rubikova kostka skládá. Jsou to: Cross (kříž), tedy jako první krok se poskládá kříž na jedné straně (barvě) kostky. Dále se poskládají první 2 vrstvy okolo vytvořeného kříže. Odtud název F2L (first two layers, první dvě vrstvy). Tyto kroky se většinou provádí podle intuitivního sledování a manipulování kostky, jejich algoritmy se téměř nepoužívají, pouze na nejvyšší kompetitivní úrovni. Zbývající dva kroky jsou naopak pouze algoritmické. Pro sadu OLL (Orientating the Last Layer, orientace poslední vrstvy) je celkem 57 algoritmů. Jejich cílem je pootáčet kostičky tak, aby celá vrchní strana byla kompletní, avšak kostka neposkládaná. O to se stará poslední krok, PLL (Permutating the Last Layer, permutace poslední vrstvy), který přemístí všechny zbylé kostičky. Výsledkem je poskládaná kostka. [4]

### Značení tahů

Algoritmy se zapisují podle celosvětové notace, kde každý tah má své písmeno z anglického názvu. Základní jsou R, L, U, D, F, B (pravá, levá, horní, spodní, přední, zadní). Otáčí se po směru hodinových ručiček. Každý tah se také řídí pravidlem, že pokud se za písmenem nachází apostrof nazývaný „prime“, tah se provede v opačném směru a pokud číslice 2, provede se posun 2krát.Rozšiřující jsou pak široké tahy w, rotace pohledu na kostku x, y, z   
a průřezové M, E, S. [1]

## Programová struktura kostky

Velmi důležitý faktor je, jak vytvořit algoritmus, který bude kostkou otáčet. Na základě toho se vytvoří celý systém a struktura kostky. Po troše hledání jsem došel ke dvěma možným způsobům. Kvůli převažujícím výhodám bude využit způsob 2.

### Způsob 1

Síla tohoto způsobu je nízká náročnost programovatelnosti.

Kostka by se skládala z 6 polí o dvou dimenzích neboli 3x3 matice. Každý prvek by znamenal jedno barevné pole na kostce.

Spočívá v tom, že při zadání příkazu na otočení strany se každá barva na každé ovlivněné kostičce natvrdo posune na její požadovanou pozici. Nevýhodou je neaplikovatelnost v chytrých algoritmech, neboť by se při nich musely samostatně kontrolovat všechny barvy. To by začinilo extrémní časovou náročnost programu  
a zbytečně dlouhý a nepřehledný kód.

### Způsob 2

Další varianta je těžší na programování, ale je kompatibilnější s dynamickým vývojem.

Namísto šesti matic se využijí pouze 3, jedna na každou vrstvu kostky. Jejich objekty by byly třídy kostičky, které by obsahovaly vlastnosti otočení a pevné očíslování základní pozice. Kostičky by měnily svoji aktuální polohu po směru hodinových ručiček relativně k ostatním. Pro získávání stavů kostky pro užití vhodného skládacího algoritmu by se kontrolovala pouze rotace a očíslování kostičky (vztahy mezi větším počtem kostiček).

## Grafické vykreslení

Pro vykreslení kostky se použijí 2 způsoby: síťové zobrazení kostky pro 2D  
a zobrazení ve 3D s možností rotace.

### 2D

Zobrazení celé kostky ve dvou dimenzích je možné pouze oddělením všech jejích stran tak, aby se mohla rozložit na plochu. Takové zobrazení je sice lehké vytvořit, je však velmi nepřehledné.

### 3D

Vykreslení ve třech dimenzích je optimální pro lidské pochopení. V takovém prostředí žijeme, tudíž je to pro nás přírodní. Dosáhnutí toho je však náročnější z programové stránky.

Je nutné vytvořit digitální prostředí pro objekty, které se budou pohybovat po třech světových osách (x, y, z). Obrazovka počítače je dvoudimenzionální, tudíž je nemožné prostor bez úprav zobrazit. Tento problém řeší ortodoxní projekce.

### Ortodoxní projekce

Prostor je zrealizován pomocí třídimenzionálních vektorů. Ty se ale nedají vykreslit na 2D plochu, neměly by hloubku a nic by z nich nešlo poznat. Proto se jednotlivé vektory vynásobí tzv. projekčními maticemi a výsledek je 2D bod na obrazovce s vypočítanými faktory pohledu, jako je vzdálenost od kamery, zorné pole rotace atd.

## Použité technologie

### GitHub

GitHub je online služba, která poskytuje ukládání projektů a jejich přenos mezi zařízeními. Dělá se tak pomocí repositářů neboli prostorů na internetu, kam se ukládají data.

### GitHub Desktop

Aplikace, která ulehčí práci s platformou GitHub. Detekuje všechny změny v souborech daného staženého repositáře na zařízení a umožňuje jejich nahrání nebo stažení pomocí jednoho tlačítka.

### Windows Forms .NET Framework

Tento framework používá programovací jazyk C#. Ulehčuje grafickou stránku aplikací přes svoje komponenty UI (uživatelské rozhraní) a formulářový základ. Lehce se implementuje s programem pomocí událostí pro jednotlivé akce na komponentách formuláře. Také poskytuje nástroje pro kreslení vlastních tvarů do aplikace.

Lze kreslit na samotný formulář, ale tato metoda je velice neefektivní pro výkonnostní prostředky počítače. Využije se pro to komponentu PictureBox, která je mnohem lépe optimalizovaná pro vykreslování.

### C#

### .NET

# Praktická část

## Třídy 3D prostoru

Pro správnou funkci kostky je potřeba vytvořit 3D prostor nejen pro její funkčnost, ale i vykreslení. Skládá se třemi třídami, každá z nich se posouvá o dimenzi výše.

### Vector3

Nejnižším možným prvkem ve všech dimenzionálních prostorách je bod neboli vektor. Tato třída obsahuje tři hodnoty pro posun po osách X,Y a Z. Spojením několika vektorů se dají vytvořit složitější tvary. Většinou se však používají pouze trojúhelníky nebo čtverce, ze kterých se dané tvary skládají.

Dalšími vlastnostmi třídy Vector3 jsou výpomocné proměnné, které určují, jak má vektor měnit svoji pozici při otáčení kostky. Všechny tři vlastnosti jsou vlastní třídy Vector3. Každý vektor kostky má daná pravidla pro každou osu zvlášť.

LengthFrom0 – Vzdálenost vektoru od středu kostky;

Displacement – Udává úhel (v radiánech), který vektor svírá s danou osou;

AnimState – Ukládá změněnou pozici neboli otočení okolo osy

Funkce CalcDisplacement() vypočítává délku a úhel od nuly. Nelze je na pevno nastavit, protože po každém otočení se hodnoty mění a je potřeba je přepočítat.

### Square

Ze čtyř vektorů se skládá jeden čtverec. Jediná další vlastnost je barva čtverce, která se nastavuje manuálně při vytváření kostiček.

Čtverce mají na rozdíl od vektorů fyzickou podobu, takže se jimi vykresluje kostka.

### Cube

Přidáním třetí dimenze čtverci vzniká kostka – požadovaný tvar. Konkrétněji se jedná o kostičku, ze kterých se sjednocuje celá Rubikova kostka. Obsahuje pole šesti čtverců, které se dají nastavit osmi vektory a šesti barvami při vytváření objektu. Vlastnost cubeIndex udává správnou pozici kostičky. Její hodnoty jsou unikátní čísla 0 až 26, pro každou vrstvu 9. Využívá se především pro algoritmické skládání. Nezbytnou vlastností je rotateAxis neboli osa, po které se mají kostičky otáčet. Nastavuje se manuálně a pouze pro středové kostičky, které mají pouze jednu venkovní (nečernou) stranu. Pokud se s kostkou otočí tak, aby se změnila přední strana, tyto hodnoty se zkopírují společně se zbytkem vlastností a zajistí se tak správná rotace pro každou stranu.

## Struktura kostky

### Vytvoření kostky

## Tahy

### Programový pohyb

### Vizuální změna polohy

## Vykreslování

### Projekční matice

### Pořadí vykreslení

## Automatické poskládání

Závěr

Seznam použitých zdrojů

[1] [Rubik's Cube Move Notation (jperm.net)](https://jperm.net/3x3/moves)

[2] [3x3 (algdb.net)](http://algdb.net/puzzle/333)

[3] Code-It-Yourself! 3D Graphics Engine Part #1 - Triangles & Projection. 2019. Dostupné z: https://www.youtube.com/watch?v=ih20l3pJoeU ,[cit. 2024-02-08]

[4] [Rubik's Cube solution with advanced Fridrich (CFOP) method (ruwix.com)](https://ruwix.com/the-rubiks-cube/advanced-cfop-fridrich/)

Seznam použitých symbolů a zkratek

|  |  |  |
| --- | --- | --- |
| Zkratka | Celý název |  |
| CFOP | Kříž F2L OLL PLL |  |
| F2L | První 2 vrstvy |  |
| OLL | Orientace poslední vrstvy |  |
| PLL | Permutace poslední vrstvy |  |
|  |  |  |
|  |  |  |

Seznam obrázků

[Obr. 2.1 Obsah 11](#_Toc147493921)

[Obr. 2.2 Příklad umístění legendy obrázku 13](#_Toc147493922)

Seznam tabulek

[Tab. 2.1 Legenda k tabulce 13](#_Toc147493615)

Seznam příloh

Prázdná šablona maturitní práce