

MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY



Mozaiky pomocí Voronoiova diagramu

DIPLOMOVÁ PRÁCE

Lenka Zatloukalová

Brno, podzim 2004

Prohlášení

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracovala samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používala nebo z nich čerpala, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: prof. Ing. Ivo Serba, Csc.

Poděkování

Na tomto místě bych ráda poděkovala panu prof. Ing. Ivo Serbovi, Csc., vedoucímu mé diplomové práce, za uvedení do problematiky, jeho ochotu a cenné rady využité při tvorbě této práce.

Shrnutí

Tato práce zahrnuje popis algoritmů a postupů potřebných k tvorbě editoru Voronoiových mozaiek. Editor vytváří mozaiky na základě dvou způsobů. Prvním způsobem je konvertování uživatelem zadaného obrazu na mozaiku a druhým způsobem je tvorba mozaiky s využitím grup symetrií.

V případě tvorby mozaiky ze vstupního obrázku lze používat mnoha různých nastavení ovlivňujících výsledný vzhled mozaiky, tzn. nastavení barevná, různé způsoby generování bodů či nastavení, atd. Uživatel má možnost nastavit různé parametry výstupu a tím dosáhnout požadovaného efektu mozaikování.

V případě tvorby mozaiek s využitím grup symetrií lze vytvářet mozaiky tapetového, pásového a rozetového vzoru. Uživatel tímto způsobem rozmístí body na plátno editoru a editor vypočítá výsledný Voronoiov diagram. Editor opět umožňuje aktivní spolupráci a komunikaci s uživatelem a měnit nastavení různých parametrů.

Klíčová slova

Symmetry groups, Voronoi diagrams, Wallpapers, Frieze, Rosette, Mozaics

Obsah

1	Úvod	2
2	Voronoiovy diagramy	3
2.1	Historie Voronoiových diagramů	3
2.2	Definice a základní vlastnosti Voronoiových diagramů	4
3	Algoritmy pro vytváření Voronoiových diagramů	7
3.1	Dělicí algoritmus	7
3.2	„Plane sweep“ algoritmus	8
3.3	Inkrementální algoritmus	8
4	Návrh editoru	9
4.1	Symetrie a Voronoiovy diagramy	9
4.1.1	Ornamentální mozaiky tvořené Voronoiovým diagramem	11
4.2	Efekt mozaikování pro vstupní obrázek	13
4.3	Vybarvení sítě Voronoiova diagramu	15
5	Implementace editoru Voronoiových mozaiek	17
5.1	Popis pracovního prostředí	17
5.2	Popis nejdůležitějších objektů a komunikace mezi nimi	18
5.2.1	Hlavní formulář	19
5.2.2	Výpočet souřadnic generátorů z grup symetrií	23
5.2.3	Výpočet Voronoiova diagramu	24
6	Popis a ovládání editoru Voronoiových mozaiek	27
6.1	Popis uživatelského prostředí	27
6.2	Menu	27
6.3	Nástrojová lišta	29
6.4	Záložky	30
6.4.1	Záložka Parametry (Parameters)	30
6.4.2	Záložka Vzory (Patterns)	31
6.4.3	Záložka Barvy atd. (Colors etc.)	32
7	Práce s editorem	33
7.1	Tvorba mozaiky vytvořené ze vstupního obrázku	33
7.2	Tvorba mozaiky z vlastních mřížek	36
7.3	Změna obarvení mozaiky	37
8	Závěr	40
A	Příloha	i

Kapitola 1

Úvod

Tato práce se zabývá problémem tvorby mozaiek s využitím Voronoiových diagramů. Chtěla bych zde ukázat a popsat jednu z mnoha oblastí využití Voronoiových diagramů a to oblast výtvarné informatiky. V diplomové práci jsem se snažila využít poznatků z mnoha oblastí a vhodným způsobem je uspořádat a vytvořit tak funkční program, který lze využívat k efektním výstupům.

V následujícím textu bude nejprve definována historie a původ Voronoiova diagramu, ale také jeho základní definice a pojmy potřebné k dalším kapitolám této práce (kapitola dva). Ve třetí kapitole jsou popsány základní algoritmy dělení roviny pomocí Voronoiova diagramu. V následující čtvrté kapitole je popsán návrh editoru, respektive způsob využití jednotlivých algoritmů k tvorbě mozaiek. V kapitole páté je popsána implementace tohoto editoru včetně komunikace jednotlivých komponent programového díla. V dalších dvou kapitolách je popsáno uživatelské prostředí a práce s editorem, jenž je ilustrována na krátkých obrazových ukázkách. V příloze této diplomové práce je obsaženo několik ukázek výstupů editoru Voronoiových mozaiek. Po čtenáři se předpokládá základní znalost geometrických algoritmů, teorie grup a symetrií.

Kapitola 2

Voronoiovy diagramy

2.1 Historie Voronoiových diagramů

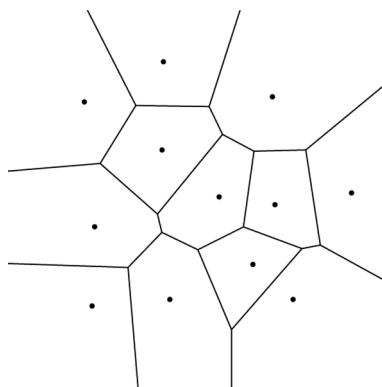
V mnoha oblastech přírodních i společenských věd existuje mnoho oborů, které se zabývají problémem členění prostoru na oblasti. S tímto problémem se můžeme například setkat v oblasti astronomie při studiu struktury vesmíru, v biologii při zjišťování organizace výživy živočišných a rostlinných tkání či v ekologii při rozložení živočišných revírů, sídlišť a ekologických nik.

Zkoumání v těchto oblastech může vést k problému spočívajícím v tom, že existuje konečná množina center, jimž je přiřazena jistá část prostoru. Výsledkem je rozdělení prostoru na systém oblastí – buněk či cel, vesměs téměř nebo úplně vyplňujících prostor a majících společné nejvíce své hranice. V tomto případě dělení nazýváme *teselací*, dělení v rovině *mozaikou*.

Je velmi pravděpodobné, že samotná koncepce Voronoiových diagramů se v historii vyskytovala mnohem dříve, než byla formálně prezentována v 19. století. Podobnou strukturu, jako jsou Voronoiovy diagramy, lze spatřit již v 17. století v díle René Descarta. V dílech *Le Monde de Mr Descartes, ou Le Trait de la Lumière* a *Principia Philosophiae*, publikovaných v roce 1644, se Descartes věnuje uspořádání a rozdělení vesmíru. K tomuto účelu používá těchto diagramů, aby ukázal uspořádání hmoty ve sluneční soustavě. Zde můžeme poprvé spatřit samotnou myšlenku dělení prostoru.

Avšak jako první podali definici a vlastnosti této koncepce Peter Gustav Lejeune Dirichlet (1805–1859) a Georgy Fedoseevich Voronoy (Georges Voronoï) (1868–1908), kteří ve svých studiích pozitivní kvadratické formy používali speciální tvar Voronoiových diagramů. Dirichlet definoval dvou- a tří-dimenzionální Voronoiovy diagramy, zatímco Voronoy obecné m -dimenzionální případy. Není určité překvapující, že první případ použití těchto diagramů se objevil v oblasti krystalografie. Výzkum v této oblasti spadá hlavně do konce 19. a do počátku 20. století a nejvíce se mu věnovali vědci v Německu a Rusku. V této době se Voronoiovy diagramy objevovaly pod mnoha různými jmény jako například Elementární oblasti či Oblast vlivu. V období, kdy se používalo Voronoiových diagramů v krystalografii, se začínají uplatňovat také v oblasti meteorologie při počítání průměru spádu dešťových srážek v jednotlivých oblastech, ale také v geografii, sociologii, v chemii při modelování složitých sloučenin či v kódování.

Přestože výzkum a také aplikace Voronoiových diagramů spadá zejména do oblasti přírodních věd, tak první použití této koncepce lze objevit v mapě zachycující rozšíření cholery ve farnosti St. Jamese ve Westminsteru během roku 1845. Tato mapa ukazuje oblasti, ve kterých se vyskytovalo úmrtí cholerou. Vzdálenosti mezi jednotlivými oblastmi nejsou měřitelné v Euklidovské metrice, nýbrž v termínech vzdáleností, které umožňovala



Obrázek 2.1: Voronoiův diagram

sít' jednotlivých ulic ve Westminsteru, což vytvářelo zároveň sít' Voronoiových diagramů, tedy pojmu, jenž téměř před 150 lety nikdo neznal. Mapa se stala „nejslavnější mapou nemoci“ celého 19. století. Byla velmi citována a reprodukována v mnoha odborných textech kartografů a epidemiologů.

2.2 Definice a základní vlastnosti Voronoiových diagramů

Matematický pojem Voronoiova diagramu lze ilustrovat na následujícím příkladu: Představte si Petriho misku, ve které se pěstují různé druhy barevných bakterií, z nichž každá má na počátku své pevné místo, kam byla umístěna. Každá bakterie tedy zaujímá své místo v misce. V další fázi začne každá bakterie narůstat a zvětšovat tak oblast jí přilehlou v konstantní rychlosti. Až dojde k tomu, že se setkají dvě různé barvy v místě, které je ve stejné vzdálenosti od obou bakterií různé barvy, potom je jejich růst zastaven. Tento proces bude probíhat tak dlouho, až bude celá miska rozdělena na regiony různých barev.

Místo bakterií, které se umísťují na určité místo v misce, můžeme vytvořit Voronoiův diagram s konečnou množinou bodů v rovině. Necht' tedy $P = \{p_1, p_2, \dots, p_n\}$ je množina bodů, jimž se říká *generátory*. Definujeme *Voronoiův diagram* jako soubor $V = \{V_1, V_2, \dots, V_n\}$ podmnožin roviny, nazývaných *Voronoiovy regiony*. Každý V_i je regionem roviny, jenž obsahuje všechny body ležící blíže bodu p_i než k ostatním členům P .

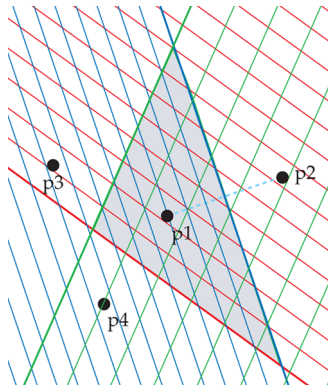
Formálně můžeme Voronoiův diagram v rovině definovat takto [1]:

Definice 1. Necht' $P = \{p_1, p_2, \dots, p_n\} \subset \mathbb{R}^2$, kde $2 < n < \infty$ a $p_i \neq p_j$ pro $i \neq j, i, j \in I_n$. Nazveme region daný

$$V(p_i) = \{x \mid \|x - p_i\| \leq \|x - p_j\| \quad \text{pro } j \neq i, j \in I_n\}$$

rovinným Voronoiovým regionem asociovaným s p_i (nebo také Voronoiův polygon bodu p_i) a množinu danou

$$\mathcal{V} = \{V(p_1), V(p_2), \dots, V(p_n)\}$$



Obrázek 2.2: Voronoiův diagram tvořený průnikem polorovin

nazveme *rovinným Voroniovým diagramem* generovaným množinou P (nebo Voroniovým diagramem množiny P). Body p_i z výrazu $V(p_i)$ nazveme *generátory* i -tého Voroniova polygonu a množinu $P = \{p_1, p_2, \dots, p_n\}$ nazveme *množinou generátorů* Voroniova diagramu \mathcal{V} .

Hranice Voroniova diagramu mohou tvořit úsečky, polopřímky či přímky, nazveme je *Voroniovými hranami* (označení e_i). Sjedením Voroniových hran dostaneme síť, kterou budeme nazývat *Voroniovou sítí*. Vrcholu Voroniovy hrany říkáme *Voronoiův vrchol*.

Pro jednoduchost zavedeme označení VD pro pojem Voroniova diagramu a toto značení budeme dále v textu používat.

Obrázek 2.1 ukazuje příklad rovinného VD. Body jsou generátory, úsečky a polopřímky tvoří Voroniovy hrany. Voronoiův region asociovaný s generátory je vnitřek polygonu obklopující generátor.

V definici 1 je VD tvořen polygony. Jelikož polygony jsou tvořeny pomocí polorovin, můžeme si zavést také alternativní definici VD, jež využívá vlastností polorovin. Abychom mohli ukázat tuto alternativní definici, uvažujeme půlící přímku, jenž je kolmá k úsečce $\overline{p_i, p_j}$ spojující generátory p_i a p_j . Tuto přímku nazveme *osou* mezi body p_i a p_j . Osa dělí rovinu na dvě poloroviny, dostaneme

$$H(p_i, p_j) = \{x \mid \|x - p_i\| \leq \|x - p_j\|\} \quad j \neq i$$

Region $H(p_i, p_j)$ pojmenujeme *dominantní region p_i nad p_j* . V obrázku 2.2 vidíme dominantní regiony p_1 nad p_2 , p_3 a p_4 znázorněný horizontálním, diagonálním a vertikálním šrafováním. V tomto případě vytváří průnik polorovin $H(p_1, p_2) \cap H(p_1, p_3) \cap H(p_1, p_4)$ Voronoiův polygon asociovaný s p_1 . Z tohoto příkladu vidíme, že následující definice je alternativou definice 1 [1].

Definice 2. Necht' $P = \{p_1, p_2, \dots, p_n\} \subset \mathbb{R}^2$, kde $2 < n < \infty$ a $p_i \neq p_j$ pro $i \neq j, i, j \in I_n$. Nazveme region daný

$$V(p_i) = \bigcap_{j \in I_n - \{i\}} H(p_i, p_j)$$

Voronoiův polygon asociovaný s p_i a množinu $\mathcal{V} = \{V(p_1), V(p_2), \dots, V(p_n)\}$ *rovinný Voronoiův diagram* generovaný P .

Kapitola 3

Algoritmy pro vytváření Voronoiových diagramů

V této kapitole si ukážeme některé metody tvorby Voronoiových diagramů. Dále popsané metody jsou zde pouze nastíněné, aby si čtenář mohl udělat představu, jakým způsobem se Voronoiovy diagramy konstruují. Přesnější definice lze nalézt například v [1].

Tyto algoritmy jsou založené na dělení roviny (teselaci) dle určitých přístupů a pravidel. Při dalších úvahách budeme předpokládat splnění následujících předpokladů.

- Numerické výpočty jsou přesné (neuvažujeme omezení daná konečnou délkou reprezentace reálných čísel v počítači).
- Žádné čtyři generátory neleží na společné kružnici.

Nejjednodušší (přirozená) metoda konstrukce vychází přímo z její definice, (viz. definice 2).

Algoritmus 1. NAIVNÍ ALGORITMUS

Vstup: n generátorů p_1, p_2, \dots, p_n

Výstup: VD množiny P

1. Mějme n generátorů $p_1, p_2, \dots, p_n \in P$.
2. Pro každé $i \in \{1, 2, \dots, n\}$ vytvoříme $n - 1$ polorovin $H(p_i, p_j)$, $1 \leq j \leq n, j \neq i$
3. Průnikem polorovin $H(p_i, p_j)$, $1 \leq j \leq n, j \neq i$ je buňka teselace $V(p_i)$.

Tato metoda je intuitivní, ale nevhodná pro algoritmizaci. Počet polorovin a tedy i čas výpočtu roste s n^2 a navíc výpočet průniku velkého počtu polorovin není triviální záležitost. Proto se tato metoda v praxi nepoužívá a dává se přednost metodám rychlejším a algoritmicky jednodušším.

3.1 Dělicí algoritmus

Tento algoritmus (v angličtině zvaný „divide-and-conquer“) vychází z myšlenky, která je základem mnoha efektivních algoritmů. Princip spočívá v rekurzivním dělení problému na menší snadněji řešitelné části. V tomto případě se využívá toho, že sestavit VD pro trojici generátorů je triviální problém. Rozdělení tvoří osy stran trojúhelníka, jehož vrcholy leží v generátorech. Konečné řešení je sloučením jednotlivých rozdělení podmnožin. Postup je následující [3]

Algoritmus 2. DĚLÍCÍ ALGORITHMUS**Vstup:** n generátorů p_1, p_2, \dots, p_n **Výstup:** VD množiny P

1. Generátory p_1, p_2, \dots, p_n uspořádejme podle rostoucí x-ové souřadnice.
2. Je-li $n \leq 3$ sestrojme teselaci ze dvou nebo tří generátorů a pokračujme bodem 4. Je-li $n > 3$ pokračujme bodem 3.
3. Necht' t je celá část $\frac{n}{2}$. Rozdělme P na $P_L = (p_1, p_2, \dots, p_t)$ a $P_R = (p_{t+1}, \dots, p_n)$. Obě teselace řešme odděleně od bodu 2.
4. Spojme teselace generované P_L a P_R . Při spojování dílčích teselací využíváme skutečnosti, že nově vzniklé vrcholy (na hranici mezi teselacemi) leží na hranách jdoucích z nějakého uzlu do nekonečna.

3.2 „Plane sweep“ algoritmus

Metoda pohyblivé přímky (line sweep method) je obecná metoda používaná při řešení rovinných problémů. Dvou-dimenzionální problém se převede na problém jedno-dimenzionální. Metoda spočívá v použití proěsávací přímky, která je buď vertikální či horizontální. S touto přímkou procházíme rovinu buď zleva doprava nebo shora dolů. Postupně zasahuje objekty v rovině a vždy, když dojde k takové události, je vyřešen problém protínající tuto přímku.

Jednoduché aplikaci metody při konstrukci teselací brání skutečnost, že pohyblivá přímka musí při konstrukci vrcholů a hran vzít v úvahu i generátory, které leží před ní. Tento problém se řeší transformací souřadnic generátorů, která však vede k tomu, že hrany teselací jsou částmi hyperbol. Protože je tato metoda implementačně náročná, používá se zřídka.

3.3 Inkrementální algoritmus

Tento algoritmus patří mezi nejpoužívanější pro svou jednoduchost a časovou nenáročnost výpočtu. Metoda začíná s jednoduchou Voronoiovou teselací vytvořenou několika generátory. Pro vytvoření Voronoiova diagramu v rovině použijeme na začátku 3 generátory. Tato teselace se pak postupně modifikuje přidáváním dalších generátorů. Hlavní část algoritmu tedy spočívá v transformaci teselace z $i - 1$ na i generátorů pro každé $i \in \{4, 5, \dots, n\}$.

Kapitola 4

Návrh editoru

V této kapitole bych chtěla vysvětlit obecné základy celého Voronoiova editoru. To znamená, na jakém principu je editor založen, z čeho vychází jednotlivé funkce v něm implementované a shrnout celkový návrh programu.

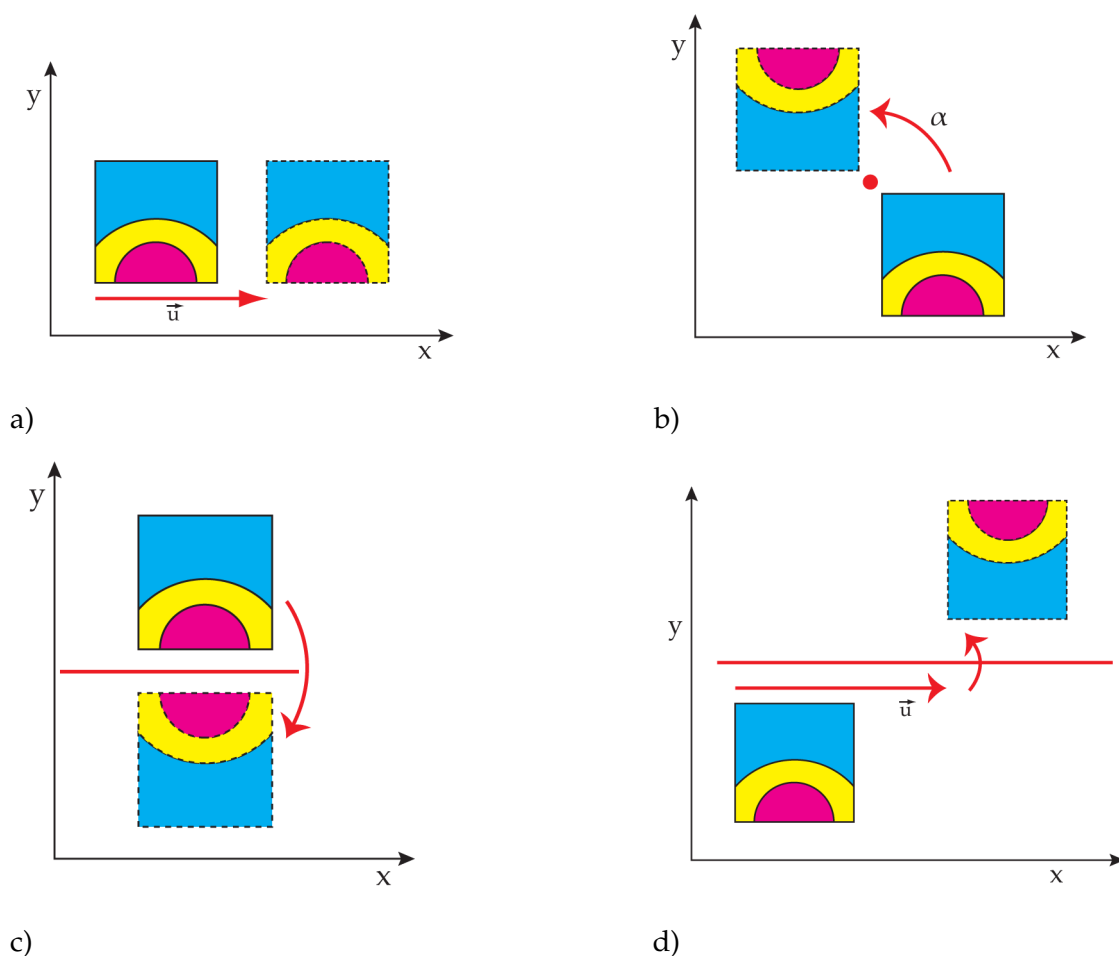
Samotný Voronoioův editor je v současné podobě zaměřen na dvě oblasti tvorby mozaiek s využitím Voronoiových diagramů. Jednou z oblastí, na kterou se editor Voronoiových mozaiek zaměřuje, je oblast ornamentálních mozaiek, vzniklých z aplikací symetrií. V této oblasti je tvořivost ponechána zcela na uživateli, který sám rozhoduje jak bude celková mozaika vypadat. Druhou oblastí, kterou editor pokrývá, jsou mozaiky vzniklé ze vstupního obrázku. Na vstupu je tedy uživatelem zadaný obrázek, na výstupu je mozaika tvořená tímto obrázkem. Jedná se tedy o efekt mozaikování. V další části této kapitoly se tedy zmíním o principech, které mi umožnily celý editor vytvořit.

4.1 Symetrie a Voronoiovy diagramy

V této části kapitoly si vysvětlíme, jakým způsobem je možné vytvářet ornamentální mozaiky z Voronoiových diagramů. K této tvorbě lze použít grup symetrií a izometrií.

Izometrie jsou transformace, které zachovávají vzdálenost. Existují čtyři druhy rovinových izometrií [10]. Tvoří je translace, rotace, reflexe a klouzavá reflexe.

- **Translace** je nejčastější způsob izometrie. Jedná se o posun objektu po přímce na nové místo. Jak je naznačeno v obrázku 4.1 a), objekt se po přímce posunuje o stále stejně velký vektor \vec{u} . Základní vlastností translace je, že nemá pevný bod, tzn. že žádný bod nezůstane na svém místě. Orientace objektu je v případě translace zachována.
- **Rotace** je typem izometrie, která také zachovává orientaci objektu. Jedná se o rotaci objektu kolem určitého středu o nějaký úhel α . Rotace má tedy jeden pevný bod, kterým je střed, kolem něhož objekt rotujeme.
- **Reflexe** je také nazývaná zrcadlením okolo osy. Jedná se o překlopení objektu okolo osy, buď ve vertikálním nebo v horizontálním směru. Pevných bodů je tedy nekonečně mnoho a všechny leží na ose reflexe. Tato izometrie převrací orientaci objektu.
- **Klouzavá reflexe** objekt posune v určitém směru a potom jej překlopí. Objekt se tedy posunuje po přímce po stále stejně velkých úsecích (\vec{u}) a zároveň střídavě mění orientaci. Pevný bod zde stejně jako u translace není žádný.



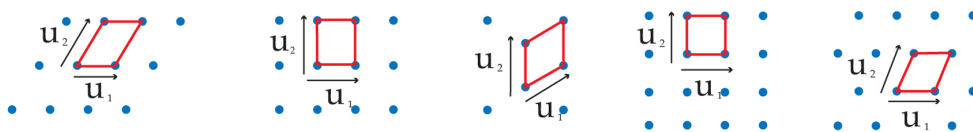
Obrázek 4.1: a) translace b) rotace c) reflexe d) klouzavá reflexe

Obrázek 4.1 ukazuje čtyři základní izometrie.

Symetrie je také možné kombinovat a vytvářet tak kompozice dvou izometrií. Tato kompozice je opět izometrií. Množiny symetrií s určitou strukturou vytvářejí grupy symetrií. Tyto grupy vytvářejí několik základních vzorů. Těmito vzory jsou tapety, vlasy a rozety. Stručně si popíšeme tyto jednotlivé vzory [6].

- **Tapetový vzor** (Wallpaper) se objevuje v oblasti mozaiek nejčastěji. Pro tento vzor (nekonečné pole) platí následující tvrzení: Existují takové lineárně nezávislé vektory \vec{u}, \vec{v} , že množina $\{T(k \cdot \vec{u} + l \cdot \vec{v})\}, \forall k, l \in \mathbb{Z}$, kde T značí translaci, je množinou všech posunutí vzoru.

Opakování vzoru se tedy děje posunem translační jednotky v určitém směru. Tyto translace tak vytvářejí mřížky 2D mozaiky. Mřížky bychom mohli klasifikovat do 5 základních druhů. Translace může tedy probíhat v mřížce rovnoběžníkové, pravoúhlé, kosočtverečné, čtvercové a hexagonální, jak ukazuje obrázek 4.2 Existuje celkem 17 základních grup symetrií tapetového typu. Tyto typy grup symetrií se od sebe odlišují typem základní mřížky, řádem rotací a reflexí okolo osy.



Obrázek 4.2: Základní mřížky tapetového vzoru

- **Vlysový vzor** (Frieze) je nekonečným pruhem. Platí pro něj: Existuje takový vektor \vec{u} , že množina $\{T(k \cdot \vec{u})\}$, $\forall k \in \mathbb{Z}$, kde T označuje translaci, je množinou všech posunutí vzoru.

Pro vlysové vzory existuje celkem 7 grup symetrií, v nichž dochází k posunu translačního razítka v pásu. Používá různé řady rotací a reflexí okolo osy.

- **Rozetový vzor** (Rosette) je také nazýván solitér. Pro tento vzor platí: Existuje takový úhel α , že množina $\{R(k \cdot \alpha)\}$, $\forall k \in \mathbb{Z}$, kde R označuje rotaci, je množinou všech otočení vzoru. Rozetový vzor zahrnuje celkem 11 různých druhů symetrií. Zde nedochází k translacím, ale používá se rotací a reflexí.

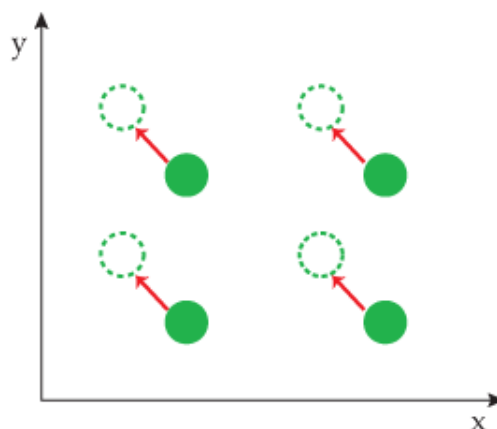
4.1.1 Ornamentální mozaiky tvořené Voronoiovým diagramem

Vlastností grup symetrie a tapetových, vlysových a rozetových vzorů, zmíněných v předchozí části kapitoly, lze využít také v oblasti tvorby Voronoiových diagramů. Pokud tedy známe algoritmy pro tvorbu Voronoiových diagramů, vlastnosti grup symetrií a jednotlivých vzorů, můžeme tyto poznatky vhodně zkombinovat a vytvořit tak zajímavé mozaiky tvořené Voronoiovým diagramem. V této části kapitoly si tedy vysvětlíme, na jakém principu je založena kombinace znalostí Voronoiových diagramů a grup symetrií, jaká pravidla je při jejich tvorbě potřeba dodržovat [9].

Uvažujeme, že máme množinu generátorů P , ze které chceme vytvořit zajímavý a symetrický Voronoiov diagram.

Výsledný Voronoiov diagram množiny generátorů P nezávisí na pozici a orientaci generátorů v rovině, ale na jejich vzájemném uspořádání [4]. V případě, že máme definovanou shodnou transformaci (rigid motion), viz. obrázek 4.3, což je transformace, která se skládá z rotací a posunů a zachovává vzájemné rozmístění generátorů, není rozdíl v tom, zda nejprve aplikujeme transformaci na generátory množiny P a potom spočítáme Voronoiov diagram této množiny nebo zda naopak aplikujeme transformaci na Voronoiov diagram, vytvořený z původního uspořádání generátorů množiny P . Výsledek je naprosto identický. Necht' tedy T značí shodnou transformaci, potom můžeme stručně vyjádřit tento vztah $T(V(P)) = V(T(P))$.

V případě, že shodná transformace T v rovině zobrazuje generátory na samy sebe, mluvíme o symetrii. Jestliže Φ je symetrie množiny generátorů P , $\Phi(P)$ je opět P . Z toho vyplývá, že $V(\Phi(P)) = V(P)$. Protože je symetrie Φ speciálním případem shodné transformace, platí $V(\Phi(P)) = \Phi(V(P))$. Jelikož platí vztah $V(\Phi(P)) = V(P)$, lze říct, že $\Phi(V(P)) = V(P)$. Tzn. symetrie množiny generátorů P je symetrií jejího Voronoiova diagramu. Ve zkratce shrneme tedy následující vztahy.



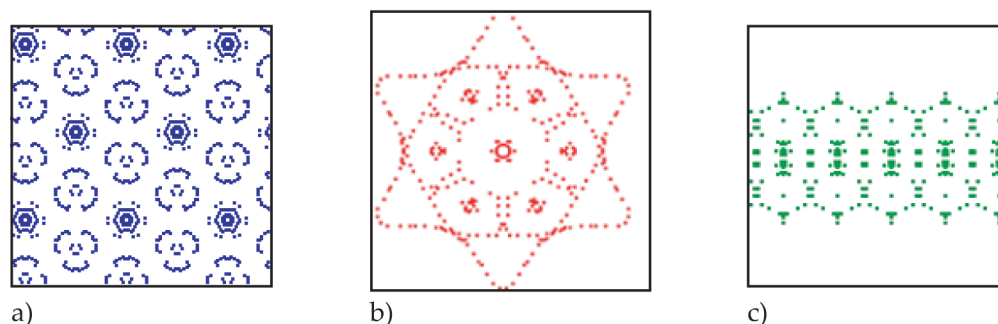
Obrázek 4.3: Shodná transformace

$$\begin{aligned}
 T(V(P)) &= V(T(P)) & T \text{ je shodná transformace} \\
 \Phi(P) &= P & \Phi \text{ je symetrie} \\
 V(\Phi(P)) &= V(P) & \text{Plyne z předchozích dvou} \\
 V(\Phi(P)) &= \Phi(V(P)) & \Phi \text{ je shodná transformace} \\
 \Phi(V(P)) &= V(P)
 \end{aligned}$$

Z těchto jednoduchých vztahů vyplývá, že pokud máme symetricky uspořádané generátory množiny P , bude také jejich Voronoiovův diagram symetrický. Na obrázku 4.4 je příklad generátorů množiny P uspořádaných do tapety, rozety a pásu.

Rozmístění bodů množiny P v rovině ovlivňují také významně dva parametry, které jsou na sobě vzájemně závislé.

- Prvním z nich je parametr *Density*, tedy hustota bodů v daném vzoru. Tento parametr přímo ovlivňuje velikost vektoru \vec{u} při translaci a klouzavé reflexi. Při jeho velké hodnotě bude mít vektor posunu \vec{u} malou hodnotu a dlaždice tvořená soustavou několika Voronoiových generátorů se bude opakovat častěji než při volbě menší hodnoty parametru.
- Druhý parametr *Jitter*, do češtiny též překládán jako „roztřesení“, vnáší do celkového vzhledu určitou míru náhody. Diagram tedy není tak pravidelný a symetrický. Pokud je tedy určena konečná pozice bodů v rovině, tak ještě před jejich vykreslením na obrazovku se k této pozici přičte nějaké náhodné číslo ve směru x -ové a y -ové osy. Výsledné umístění bodu je tedy poněkud „roztřesené“ a do konečného Voronoiova diagramu je tak vnesena nepravidelnost. Náhodné číslo, které bude přičteno k pozici každého bodu z množiny P , závisí samozřejmě na hodnotě parametru *Jitter*, ale také nepřímo na parametru *Density*. Pokud bude hustota bodů v rovině malá, bude celkové roztřesení větší než v případě, že při stejné hodnotě



Obrázek 4.4: a) tapetový vzor b) rozetový vzor c) vlysový vzor

parametru *Jitter* zvolíme hustotu bodů mnohem větší. Příklad trojúhelníkové sítě z různými hodnotami těchto parametru lze najít na obrázku 4.5.

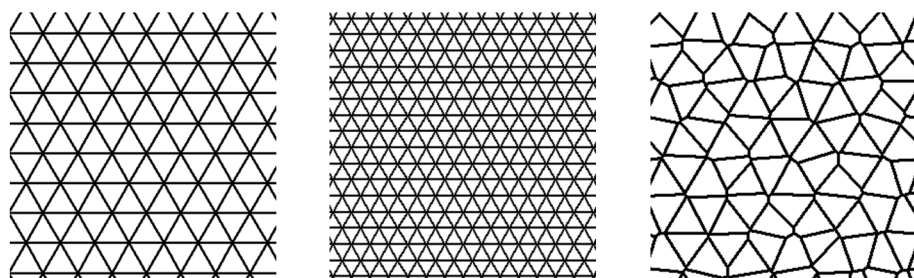
Těchto symetrií lze tedy využít v oblasti ornamentálního designu. Vytvořením mřížky z Voronoiových diagramů a následným obarvením Voronoiových polygonů vznikne zajímavá mozaika. Voronoiovy polygony je možné obarvit podle barvy generátoru či zcela náhodně. Další možností, jak je možné tyto polygony obarvit, je použít barvy z podkladového obrázku. Tento efekt, jenž vytvoří mozaiku z obrázku je popsán v další části 4.2 této kapitoly.

4.2 Efekt mozaikování pro vstupní obrázek

Jak už jsme si v předchozí části kapitoly naznačili, čtenář se zde může dozvědět, na jakém principu je založena tvorba mozaiek ze vstupního obrázku. Základní idea tohoto typu mozaiek, jak ji používá editor Voronoiových mozaiek, je založena na vytvoření sítě Voronoiova diagramu, který se jakoby přiloží na vstupní obrázek. Následně dojde k vybarvení všech Voronoiových polygonů barvou pixelu, který je v obrázku umístěn na stejné pozici jako generátor Voronoiova polygonu.

Vzhled takto vzniklé mozaiky je závislý také na tvaru mřížky, která se na obrázek přikládá. Editor Voronoiových mozaiek pracuje s třemi typy mřížek.

- **Náhodně** vytvořená síť generátorů tvoří jeden typ mřížek. V tomto případě se náhodně vygeneruje určitý počet generátorů v rovině. Tento počet je určen parametrem *Density*. Parametr *Jitter* v tomto případě nemá smysl používat, protože je zde již při vygenerování použito určité náhody.
- **Symetrická** mřížka, jenž vznikla z použitím grup symetrií a jejíž popis čtenář najde v části 4.1, tvoří druhý typ mřížek. Nejčastěji používaný vzor pro mozaiky, které vznikly z obrázku, je tapetový vzor.
- **Trojúhelníková** síť Voronoiova diagramu je třetím typem mřížek, které jsem ve své práci vytvořila. Generátory byly uspořádány tak, aby výsledný Voronoiov diagram



Obrázek 4.5: Příklad sítě Voronoioiva algoritmu s různou hodnotou parametru *Density* a *Jitter*

tvořil trojúhelníkovou síť. Vycházela jsem z bodu umístěného v levém horním rohu, tj. z počátečního bodu celé sítě. Obrázek 4.6 ukazuje postup generování bodů do sítě. Posun ve směru x -ové osy, závisí na parametru *Density*. Čím vyšší tato hustota je, tím menší je posun generátoru ve směru x . Postupně se tak střídá posun ve směru x -ové a y -ové osy. Posun ve směru y -ové osy se střídavě děje nahoru a dolů. Jakmile se dojde na konec řádku a vygeneruje se jeho poslední bod, vrátíme se opět na pozici prvního bodu řádku a vygenerujeme první bod řádku druhého. Proklad mezi těmito dvěma řádky tvoří hodnota z , což je přepona trojúhelníku tvořeného posunem x a posunem y . Ke generování prvního bodu řádku dochází střídavě s hodnotou z a $2 * z$. Postup je znázorněn na obrázku 4.6.

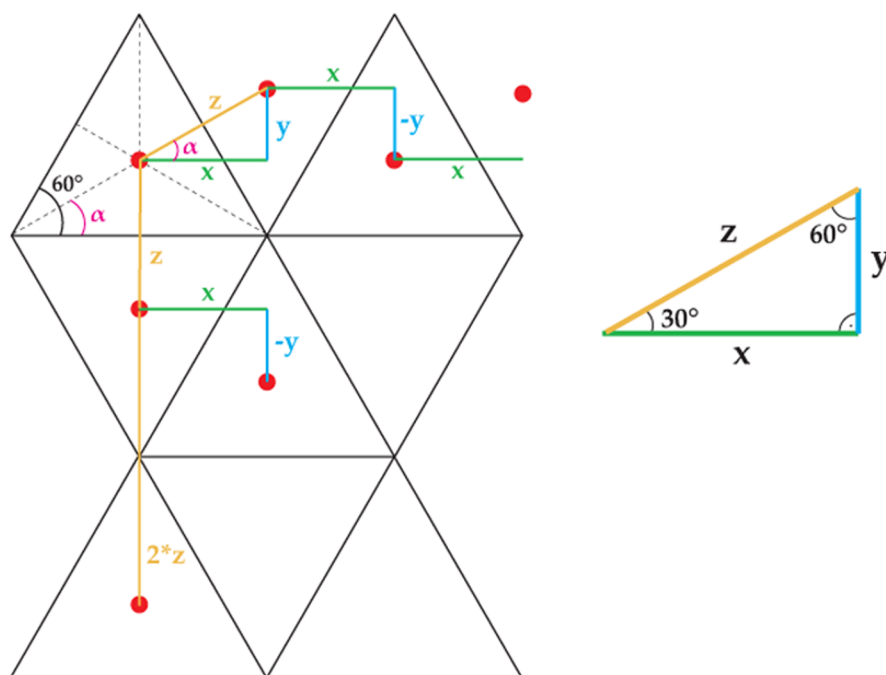
Hodnoty y a z je potřeba dopočítat z hodnoty x a také z hodnot úhlů trojúhelníků. Jelikož celá síť je tvořena rovnostrannými trojúhelníky, jejichž vnitřní úhly mají každý 60° . Z tohoto můžeme vyjádřit vztahy pro hodnoty y a z .

$$\frac{x}{\sin 60^\circ} = \frac{y}{\sin 30^\circ} = \frac{z}{\sin 90^\circ}$$

$$\frac{x \sin 30^\circ}{\sin 60^\circ} = y \quad \Rightarrow \quad y = \frac{x}{\sqrt{3}}$$

$$\frac{x \sin 90^\circ}{\sin 60^\circ} = z \quad \Rightarrow \quad z = \frac{2x}{\sqrt{3}}$$

U trojúhelníkové sítě můžeme opět použít vlastností parametrů *Jitter* a *Density*, příklad trojúhelníkové sítě s různými hodnotami těchto parametrů může čtenář vidět na obrázku 4.5.



Obrázek 4.6: Trojúhelníková mřížka

4.3 Vybarvení sítě Voronoiova diagramu

V předchozích částech této kapitoly jsme si popsali jakým způsobem může editor Voronoiových mozaiek uspořádat generátory v rovině a tím dosáhnout různých efektů při tvorbě mozaiek vzniklých z těchto diagramů. S tvorbou sítě VD je spojeno také vybarvení Voronoiových polygonů. Vybarvit tyto polygony lze v editoru třemi způsoby. Tyto způsoby, nebo-li módy, jsou závislé na typu mozaiky, kterou si uživatel přeje vytvořit. Pro každý generátor máme k dispozici jeho souřadnice v rovině, jeho aktuální barvu a také index do pole náhodných barev.

- Mód vybarvení jednotlivých polygonů podle **vstupního obrázku** je založen na přiložení mřížky na vstupní obrázek a následném vybarvení jednotlivých polygonů. U každého generátoru sítě VD je nutné si zapamatovat pozici generátoru v rovině, tj. jeho souřadnice. Při přiložení mřížky na vstupní obrázek se vybere barva pixelu ležícího na x-ové a y-ové pozici každého generátoru.
- Mód obarvení polygonu podle **barvy generátoru** využívá aktuální barvy generátoru. Používá se při vybarvování sítě, kterou vytvořil uživatel s použitím jednotlivých grup symetrií. Uživatel si zvolí barvu generátorů a při kliknutí do plátna editoru se vytvoří síť bodů. Barva, kterou si uživatel pro jednotlivé generátory vybral je barvou aktuální. Touto barvou se obarví Voronoiovy polygon a je možné takovým způsobem vytvářet vícebarevné mozaiky.

- Mód vybarvení polygonu **náhodnou barvou** používá ukazatel do pole náhodných barev. Počet náhodných barev v poli určuje uživatel. Každému polygonu je tedy náhodně přiřazena barva z tohoto pole. Editor také umožňuje náhodným způsobem změnit barvu již obarvených polygonů v mozaice. Implementace tohoto obarvovacího algoritmu je popsána v kapitole 5.

Kapitola 5

Implementace editoru Voronoiových mozaiek

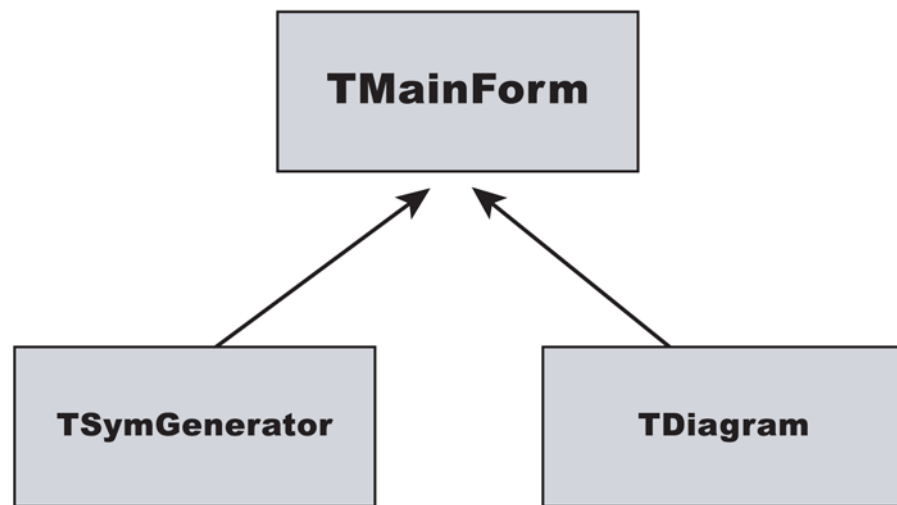
Editor Voronoiových mozaiek byl vytvořený kompletně v prostředí Borland Delphi 6. Pro výpočet Voronoiova diagramu jsem použila program *Qhull* [7]. Tento program byl vytvořen na Minnesotské univerzitě v Geometry Center. Je volně k dispozici pokud jsou dodrženy licenční podmínky. Soubor s licencí je obsažen na přiloženém CD. Spočítá Voronoiov diagram na základě vstupních údajů, které tvoří soubor s počtem Voronoiových generátorů a jejich souřadnicemi. Jeho výstupem je opět soubor, tentokrát se souřadnicemi Voronoiových polygonů. Program nevykresluje Voronoiovy polygony, pouze spočítá vrcholy jejich souřadnic, což je jedna z mnoha funkcí, ke které ho lze využít. Mimo jiné je možné tento program využít k výpočtu konvexních obalů, Delaunayovy triangulace či k výpočtu Voronoiova diagramu tvořeného průnikem polorovin.

Tato kapitola se zaměřuje na popis implementace jednotlivých prvků editoru, při čemž je kladen důraz na popis těch nejdůležitějších částí programu.

5.1 Popis pracovního prostředí

Prostředí Borland Delphi umožňuje pracovat s již předem připravenými grafickými komponentami a různými nástroji, což usnadňuje a urychluje tvorbu GUI. Takovými prvky jsou například tlačítka, zaškrtávací boxy, políčka textového vstupu, seznamy, záložky a další. Základní prvky Delphi tvoří následující čtyři části:

- **Object Inspector** umožňuje nastavit konkrétní vlastnosti jednotlivých komponent. Lze zde nastavit například titulek komponenty, vertikální a horizontální rozměry, barva povrchu nebo pozadí, typ písma použitého uvnitř komponenty, řetězec, který bude použit pro plovoucí nápovědu, viditelnost komponenty a další.
- **Object Treeview** znázorňuje stromovou hierarchii použitých komponent, což usnadňuje práci a přístup k jednotlivým komponentám.
- **Okno programu** ukazuje vzhled výsledného editoru a tím umožňuje vytvořit si představu o výsledném vzhledu editoru. V tomto okně se vytváří konečná podoba programu. Na formulář programu se přidávají jednotlivé ovládací prvky.
- **Okno se zdrojovým kódem** obsahuje zdrojový kód programu. V tomto okně se nastavuje funkčnost jednotlivým přidaným komponentám. Tyto komponenty je nutno mezi sebou propojovat a vytvořit k nim zpětné vazby, tzn. akce, které se provedou při vzniku určité události, např. stisku tlačítka.



Obrázek 5.1: Komunikace mezi nejdůležitějšími objekty editoru

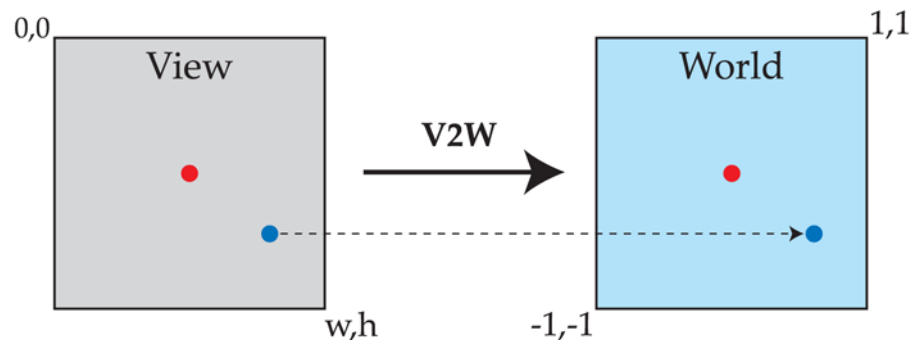
5.2 Popis nejdůležitějších objektů a komunikace mezi nimi

Mezi nejdůležitější objekty, které používá editor Voronoiových mozaiek patří **TMainForm**, **TSymGenerator** a **TDiagram**.

TMainForm je hlavní formulář, který reaguje na požadavky uživatele. Z části je vytvořen prostředím Delphi. A pokud do aplikace přidáme nějaký prvek, např. tlačítko, tak se automaticky přidá do atributů **TMainForm**. V tomto přidání nové komponenty do svých atributů se liší od objektů **TSymGenerator** a **TDiagram**, se kterými spolupracuje. Jejich spolupráce je založena na tom, že hlavní formulář volá tyto dva objekty a požaduje zpřístupnění informací. Příkladem takto popsané spolupráce je výpočet diagramu hlavního formuláře. K tomu, aby mohl být diagram spočítán je potřebné znát pozice jednotlivých generátorů vytvořených jednou z grup symetrií v ploše. Pozice generátorů získá hlavní formulář od objektu **TSymGenerator**. Tento objekt dostane na vstup souřadnice jednoho bodu a na výstup vrátí pole souřadnic bodů vygenerovaných určitou grupo symetrií. Tento objekt v sobě obsahuje definice všech grup symetrií a podle typu grupy symetrií a souřadnice prvního bodu vygeneruje body další. Poté dojde k zjištění jejich souřadnic, uspořádání do pole a předání této informace hlavnímu formuláři.

Objekt **TDiagram** je objekt, který slouží jako správce diagramů. Pokud je dán požadavek od hlavního formuláře spočítat Voronoiov diagram, tak tento objekt vytvoří soubor s aktuálními informacemi o diagramu (tj. aktuálními pozicemi všech generátorů) a předá ho na vstup programu **Qhull**. Program spočítá, jak bude diagram vypadat a vrátí objektu **TDiagram** opět soubor, ve kterém se nachází informace o polygonech výsledného Voronoiova diagramu. Objekt **TDiagram** může v tomto okamžiku opět předat požadované informace hlavnímu formuláři. Tento princip komunikace mezi třemi výše zmíněnými objekty ukazuje obrázek 5.1.

Ještě před tím, než bude v další části kapitoly věnován prostor k popisu jednotlivých objektů, bych chtěla zmínit, jakým způsobem dochází k přepočtu souřadnic, které používá



Obrázek 5.2: Převod souřadnic z View do World

editor na souřadnice, které používá program Qhull pro výpočet samotného Voronoiova diagramu. Tento převod ze souřadnic typu *View* do souřadnic světových, tzn. *World*, používají objekty *TMainForm*, *TDiagram* i *TSymGenerator*.

Souřadnice typu *View* jsou souřadnice, které používá editor Voronoiových mozaiek k vykreslení. Jsou to tedy pixelové souřadnice, které se udávají v rozmezí 0 až výška a 0 až šířka okna. Program Qhull však potřebuje k výpočtu Voronoiova diagramu pozici bodu v reálných souřadnicích, též označených světové souřadnice, tj. souřadnice typu *World*. Tyto světové souřadnice jsou definovány pro hodnoty od -1 – -1 se středem v 0 a to pro šířku i výšku. Procedura *V2W* řeší tento převod. Je zde použito explicitního výpočtu, ale tento problém je možné řešit i obecnějším přístupem, který využívá lineární transformace. Tento převod je pro větší názornost ukázán na obrázku 5.2.

V další části této kapitoly je věnován prostor popisu jednotlivých objektů.

5.2.1 Hlavní formulář

Objekt *TMainForm* se stará o vykreslení diagramu, komunikaci s uživatelem, tzn. zpětnou vazbu mezi použitými komponentami a o vizuální stránku editoru. Skládá se z mnoha atributů, z nichž některé jsou vygenerované prostředím Borland Delphi, proto si popíšeme pouze některé nejdůležitější atributy tohoto hlavního formuláře.

Privátními atributy hlavního formuláře jsou:

- **ShowGenerators** je atribut typu *Boolean*, který si uchovává informaci o tom, zda se mají vykreslovat generátory na plátno či zda mají zůstat neviditelné. Tato hodnota je na počátku nastavena na *true*.
- **ShowEdges** je atribut typu *Boolean*, který jako v předchozím případě si uchovává informaci o tom, zda mají být vykreslené hrany polygonů. Počáteční hodnota je opět nastavena na *true*, takže hrany Voronoiových polygonů jsou zpočátku vykresleny.
- **ShowPolys** podobně jako v předchozích dvou případech se jedná o atribut typu *Boolean*, který nese informaci o vykreslování polygonů.

- **ColorMode** je atribut výčtového typu (`RandomC`, `PictureC`, `GeneratorC`) a určuje, který vykreslovací mód je použit pro vykreslení mozaiky. To znamená, že se určí, jakým způsobem se vybarví polygony Voronoiova diagramu. Je možné polygony vybarvit zcela náhodným způsobem v případě typu `RandomC`, barvou generátoru pokud je zvolen typ `GeneratorC` anebo barvou podkladového obrázku jestliže byl zvolen typ `PictureC`.
- **RandomColors** je atribut typu `array of TColor`. Jedná se o pole náhodných barev, ve kterém se uchovávají jednou vygenerované náhodné barvy, které se použijí v případě každé akce při vykreslení, aby nedocházelo k neustálému generování barev nových.
- **Background** je typu `TBitmap`, což je atribut, který uchovává obrázek, jenž se vkládá na pozadí.
- **Diagram**, který je typu `TDiagram` a slouží k výpočtu diagramu. Spolupracuje s programem `Qhull`. Tento objekt bude popsán dále v této kapitole podrobněji.
- **SymGenerator** je objekt typu `TSymGenerator`. Slouží k práci s 36 grupami symetrií. Bude popsán dále v této kapitole podrobněji.

Program se skládá z mnoha **procedur**, které vytvářejí zpětnou vazbu na nějakou provedenou akci. Popíšeme si zde funkčnost pouze některých nejdůležitějších procedur.

- **FormCreate** je konstruktor a plní úlohu počáteční inicializace. Zpočátku nastaví na hodnotu `true` atributy `ShowGenerators`, `ShowPolys` a `ShowEdges`. Vytvoří instanci objektů `TDiagram` a `TSymGenerator` a přiřadí jim počáteční nastavení. Nastaví také mnoho dalších parametrů, včetně obarvovacích módů.
- **FormDestroy** je destruktorka, který na konci práce s programem smaže všechny parametry nastavené při inicializaci a ukončí práci s diagramem.
- **DiagramGenerateExecute** je procedura, která je svázaná s tlačítkem *Generate Points* umístěným v editoru. Tato procedura zavolá odpovídající procedury, které vytvoří síť generátorů buď náhodně nebo tak, aby jejich diagram vytvořil trojúhelníkovou síť. V případě typu sítě záleží na volbě, kterou uživatel označil.
- **DiagramComputeExecute** zavolá odpovídající metodu objektu `TDiagram` pro výpočet diagramu.
- **ViewGeneratorsExecute**, **ViewEdgesExecute**, **ViewPolygonsExecute** změní hodnotu typu `Boolean`, tj. přepíná vykreslování hran, generátorů a polygonů a zavolá vykreslovací proceduru `DrawDiagram`, která bude popsána dále v textu.
- Program je tvořen také skupinou procedur, které změní nastavení výstupu diagramu a zavolají vykreslovací proceduru. Těchto procedur je větší množství, proto zde uvádím pouze některé. Například:

- **EditDensityChange** je procedura, která změní hustotu sítě, tj. množství vygenerovaných bodů. Ovlivňuje generování nových bodů.
 - **EditJitterChange** změní roztřesení bodů v ploše. Ovlivňuje generování nových bodů.
 - **TrackBarJitterChange** je procedura svázaná s posuvníkem, který určuje míru roztřesení použitého při generování sítě bodů.
 - **TrackBarXOffserChange** je opět procedura spojená s posuvníkem určujícím posun sítě bodů ve směru x-ové osy
 - **DiagramClearExecute** vyčistí plátno editoru.
- Další skupinou procedur jsou procedury, které zpracovávají signály z kliknutí a pohybu myši. Jsou to procedury:
 - **ImageMouseMove** je procedura, která mění souřadnice pozice kurzoru myši na plátně, tj. souřadnice typu *View* na souřadnice typu *World* a ty následně zobrazuje ve stavovém řádku umístěném ve spodní části editoru pod plátnem. Tímto je usnadněna orientace na plátně.
 - **ImageMouseDown** zpracovává kliknutí myši. V případě, že bylo kliknuto levým tlačítkem myši na plátno editoru, převedou se x-ové a y-ové souřadnice z typu *View* na typ *World* a dají se na vstup objektu *TSymGenerator*, který podle uživatelem zvolené grupy symetrií vrátí vygenerované pole souřadnic opět typu *View*. Následně dojde k volání vykreslené funkce a vykreslení vygenerovaných bodů.
Pravé tlačítko myši slouží k přebarvení jednoho polygonu konkrétní barvou, kterou si uživatel před tím, než obarví polygon vybral v dialogovém okně barev. Při pravém kliknutí tlačítka na plochu plátna dojde opět k převodu x-ové a y-ové souřadnice k převodu souřadnic z typu *View* do typu *World*. Zjistí se vzdálenost tohoto bodu se všemi dalšími generátory diagramu a ten, který je našemu bodu nejbliž se přebarví barvou uživatelem vybranou. Dojde k překreslení diagramu.
 - **SpeedButtonClick** je procedura, která je volána při stisku kteréhokoliv tlačítka označujícího jednu z 36 grup symetrií. Tyto tlačítka se od sebe liší tagem, tj. označením. Podle této značky se nastaví chování objektu *TSymGenerator* a předá se mu na vstup informace o používané grupě symetrií.
 - Jednu z dalších skupin procedur tvoří procedury, které pracují se souborem. Patří mezi ně například
 - Procedura **FileSavePictureAccept**, která ukládá vytvořenou mozaiku z plátna ve formátu *bmp* a *jpg*.
 - **FileSaveExecute** uloží všechna nastavení diagramu, tj. hustotu sítě, používaná hodnota roztřesení, šířka a barva hran atd., včetně pozice a aktuální barvy generátorů do souboru. Tím je umožněno při otevření a načtení souboru pracovat se stejnými hodnotami dále.

- **FileOpenAccept** slouží k načtení informací ze souboru a nastavení parametrů na stejné hodnoty. Načte také pozice generátorů a zavolá proceduru k vykreslení diagramu.
- **NewCanvas** je procedura, která vytvoří nové plátno o rozměrech, které uživatel zadal do dialogového okna.
- Důležitou skupinkou procedur jsou procedury, které řeší různá barevná nastavení. Patří mezi ně následující procedury:
 - **TrackBarRandomChange** mění počet náhodných barev použitých při vybarvení polygonů náhodným způsobem. Je zde implementováno pole náhodných barev o velikosti tohoto počtu náhodných barev, který si uživatel volí posuvníkem. Je možné zvolit hodnotu v rozmezí 1 až 10. Každý generátor si uchovává informaci o své pozici, aktuální barvě a také ukazatel do tohoto pole náhodných barev. Při každé změně počtu použitých náhodných barev se zavolá tato procedura, vytvoří se pole náhodných barev a generátorům se přiřadí nový index do tohoto pole. Poté dojde k vykreslení diagramu.
 - **ActionRandomColorNewExecute** je spojena s tlačítkem *New Colors* a naplní položky v poli náhodných barev náhodně zvolenou barvou.
 - **DiagramChangeColorsExecute** umožňuje změnit barvy polygonů. Používá náhodných barev, ale zachovává stejnou barevnost polygonů s identickou barvou. Nejdříve se zjistí počet použitých barev v mozaice. Tzn. projdou se všechny generátory a postupně se ukládají jejich barvy do pole dvojic. To znamená, že každá nově použitá barva, která není v poli ještě obsažena se zařadí na konec tohoto pole a vygeneruje se k ní barva náhodná. Tak vznikne pole dvojic, které pro každou použitou barvu mozaiky uchovává barvu náhodnou. Ve druhém průchodu přiřadím vždy generátorům, které měly stejnou barvu novou barvu z pole dvojic.
- **DrawDiagram** je privátní procedurou, která slouží k vykreslení diagramu a je volána mnoha jinými procedurami. Po svém zavolání dojde k vyčištění plátna. Pokud je přepínač pro vykreslování hran polygonu (tj. *ShowEdges*) nastaven na hodnotu *true*, nastaví se kreslicí pero tak, jenž má barvu takovou, jakou si uživatel zvolil v barevném dialogovém okně a vykreslí hrany polygonů. Jestliže je hodnota atributu *ShowPolys* nastavena na hodnotu *true*, tak se nastaví vykreslovací štětec a jeho barva se nastaví podle zvoleného vykreslovacího módu. Pokud je zvolen vykreslovací mód náhodný, tak generátor u sebe nese index do pole náhodných barev, tj. zvolí se tato náhodná barva. Pokud je zvolen mód obarvení podle obrázku, tak se zjistí barva pixelu v obrázku nacházející se na stejné pozici jako generátor a pokud je zvolen vykreslovací mód podle generátoru, zvolí se aktuální barva generátoru a přiřadí se do štětce. Takto se vybarví polygony. Nakonec se v případě, že je hodnota atributu *ShowGenerators* nastavena na *true* vykreslí generátory.

5.2.2 Výpočet souřadnic generátorů z grup symetrií

Objekt `TSymGenerator` spolupracuje s hlavním formulářem, který mu předává na vstup souřadnice jednoho bodu a číslo konkrétní grupy symetrií a tento objekt vrací pole souřadnic bodů vygenerovaných touto konkrétní grupou symetrií. K definici jednotlivých grup symetrií pro tapetové, pásové a rozetové vzory objektu `TSymGenerator` jsem využila část programu *Java Kali*[8], který je volně k dispozici. Jsou zde definovány transformace pro každou z grup symetrií zvlášť. Zdrojový kód jsem přepsala z prostředí Java do Borland Delphi.

Objekt `TSymGenerator` využívá k výpočtu souřadnic generátorů také pomocné datové typy a konstanty. Jsou zde definovány konstanty, které se využívají při jednotlivých transformacích prováděných v grupách symetrií, např. $\sqrt{2}$ je konstanta, která se používá při rotaci generátorů. Práci s grupami symetrií také usnadňují tři následující pomocné datové typy typu `record`.

- Typ **TDVector** definuje vektor, jenž se používá při transformacích s grupami symetrií.
- Typ **TDMatrix** definuje matici, která se rovněž využívá při práci s grupami symetrií.
- Typ **TSymmetryGroup** je reprezentací jedné konkrétní grupy symetrií, v níž se nachází definice typu reflexe, klouzavé reflexe, řádu rotace, počtu translací a konkrétní translace.

Atributy objektu `TSymGenerator`, které objekt využívá při práci s grupami symetrií jsou:

- **CurrentGroup** určuje číslo aktuální grupy symetrií. Grupy symetrií jsou očíslovány od 0–35, při čemž grup tapetového vzoru je 17 (tzn. hodnoty 0–16), pásový vzor je definován 7 grupami symetrií (tzn. hodnoty 17–23) a rozetový vzor určuje 12 grup (hodnoty 24–35).
- **Density** je hodnota typu `Integer`, která udává hustotu sítě generátorů. Tato hodnota je předána hlavním formulářem `TMainForm` a ovlivňuje velikost vektoru, jenž se používá při transformacích v grupách symetrií.
- **Jitter** je stejně jako předchozí hodnota předáván hlavním formulářem a určuje jaká míra náhody je použita při generování bodů. Po určení konečné pozice bodu se k tomuto bodu přičte určitý šum, jehož velikost je závislá na tomto parametru.
- **Groups** je typu `array of TSymmetryGroup`, což je pole všech grup symetrií definovaných v objektu `TSymGenerator`. Při inicializaci dojde k naplnění tohoto pole jednotlivými grupami symetrií, respektive jejich konkrétními parametry.
- **RotationMatrix** je pomocná matice, která slouží při výpočtech rotací.
- **TmpArray** je pomocné pole, ve kterém se uchovávají souřadnice již vygenerovaných bodů. Toto pole se (po vygenerování všech bodů a uložení jejich souřadnic do tohoto pole) předává jako výstup objektu `TSymGenerator` hlavnímu formuláři, který potom tento výsledek zpracovává dále.

Procedury objektu `TSymGenerator` jsou následující:

- Konstruktor **Create** je zodpovědný za počáteční inicializaci tohoto objektu. Zpočátku si vytvoří matice rotací a předpočítá všechny rotace pro každý typ této transformace.
V další fázi postupně vkládá do pole **Groups** jednotlivé grupy symetrií společně s jejich parametry. O toto postupné vkládání grup do pole se stará funkce **SGLoad**, která podle aktuálního čísla grupy symetrií vloží do pole informace o ní. Takto budou do pole načteny všechny grupy symetrií.
- Procedura **MakePoints** je volána z hlavního formuláře. Dostane na vstup souřadnice jednoho bodu a vrátí pole obsahující souřadnice bodů vygenerovaných dle aktuálního čísla grupy, kterou si uživatel zvolil pro svou práci. Implementace této procedury je založena na celé skupině metod, která se postupně aplikuje na počáteční bod. Skládá se ze čtyř fází, které odpovídají provádění následujících transformací: reflexe, klouzavá reflexe, rotace a translace.
 - V první fázi je na vstupní bod aplikována procedura **DrawPointRfGlRtTr**, tzn. nejprve se aplikuje transformace reflexe, jejíž druh je závislý na čísle grupy symetrií. Při reflexi mohou vzniknout nové body. Tyto body se včetně bodu počátečního zpracovávají ve fázi druhé.
 - V druhé fázi se tedy na všechny body aplikuje procedura **DrawPointGlRtTr**, která provede klouzavou reflexi závislou na konkrétní grupě symetrií. Opět se všechny nově vniklé body včetně bodů původních předají na vstup třetí procedury.
 - V třetí fázi se na všechny body opět aplikuje procedura **DrawPointRtTr**, která provede rotaci všech bodů podle aktuální grupy symetrií. Všechny body se opět předají na vstup poslední z této skupiny procedur.
 - V poslední čtvrté fázi se na body aplikuje procedura **DrawPointTr**, která aplikuje translaci na všechny body. Translace je zde definována zvlášť pro jednotlivé typy vzorů. Při rozetovém vzoru se translace neprovádí, na body se v poslední fázi aplikují pouze atributy **Density** a **Jitter**. Při pásovém vzoru se translace provádí pouze v pásu jedním translačním vektorem. A nakonec při tapetovém vzoru se body generují po celé ploše pomocí dvou translačních vektorů. Výsledné souřadnice bodů se zapíší do pole **TmpArray** a to se předá hlavnímu formuláři.
- Další procedury jsou pouze pomocné procedury. Patří sem již výše zmiňovaná procedura **SGLoad** a další pro operaci s vektory.

5.2.3 Výpočet Voronoiova diagramu

Objekt `TDiagram` spolupracuje s hlavním formulářem a také s programem `Qhull`, který se stará o výpočet Voronoiova diagramu. Uchovává také souřadnice generátorů a vrcholů polygonů VD. Spolupráce s hlavním formulářem spočívá ve výpočtu VD na základě

souřadnic bodů, které objekt `TDiagram` obdrží od hlavního formuláře. Tento objekt také slouží ke generování sítě bodů náhodným způsobem a do trojúhelníkové sítě.

`TDiagram` používá k výpočtu VD několik pomocných datových typů. Tyto datové typy tvoří následující:

- Typ **`TPointW`** je typu `record` a udává reálné souřadnice typu *World*. Slouží ke komunikaci s programem `Qhull`, který pracuje s reálnými souřadnicemi v rozmezí od -1 do 1.
- Typ **`TPointV`** je opět typu `record` a určuje celočíselné souřadnice typu *View*. Kromě těchto souřadnic uchovává také informace o aktuální barvě generátoru a index do pole náhodných barev, jehož velikost je závislá na počtu náhodných barev, které se použijí při náhodném vybarvení polygonů, a které si uživatel zvolí na posuvníku v editoru Voronoiových mozaiek.
- Typ **`TPolysI`** je typu `array of array of Integer`. Je to typ pole polygonů, jenž jsou zadané indexy do pole `VerticesV`, což je pole, které uchovává souřadnice vrcholů Voronoiova polygonu.
- Typ **`TPolysV`** je typu `array of array of TPointV`, tzn. jedná se typ pole polygonů, jenž jsou zadány souřadnicemi typu *View*.

Atributy objektu `TDiagram`, které objekt používá jsou následující:

- **`ImageWidth`, `ImageHeight`** jsou atributy typu `Integer`, které uchovávají informace o rozměrech plátna, jenž jsou potřebné k převodu souřadnic z typu *View* do *World* a naopak.
- **`Density`, `Jitter`** jsou celočíselné atributy, které udávají hodnotu hustoty a roztřesení, použitých při generování bodů do trojúhelníkové sítě či náhodným způsobem.
- **`QVPath`** uchovává cestu k programu `QVoronoi.exe`.

- Dalšími atributy je skupina atributů, které uchovávají informace o VD. Souřadnice generátorů se uchovávají v reprezentaci *View* a *World* současně. K jejich uchování slouží atributy **`GeneratorsW`** a **`GeneratorsV`**.

Programu `Qhull` jsou na vstup předány souřadnice bodů typu *World*. Výstup programu `Qhull` tvoří souřadnice vrcholů jednotlivých polygonů VD. Tyto souřadnice se ukládají v poli **`VerticesW`**, uchovávají se také v reprezentaci *View* a to v poli **`VerticesV`**.

K uchování polygonů slouží atributy **`PolysV`** a **`PolysI`**, které uchovávají souřadnice vrcholů polygonů indexem a také v konkrétních hodnotách. Toto dvojí uchovávání souřadnic slouží k rychlejšímu výpočtu programu.

Procedury objektu `TDiagram` jsou následující:

- Konstruktor **`Create`** plní funkci inicializace celého objektu.
- **`ClearVP`** smaže polygony a vrcholy a tím vyčistí plátno editoru.

- **MakeRandomGenerators** je procedura, která generuje síť bodů náhodným způsobem. Na vstup dostane barvu, kterou budou všechny generátory vytvořeny a tím následně vybarveny všechny polygony a také velikost pole náhodných barev. Vytvoří náhodné body rozložené po celém plátně. Počet takto vytvořených bodů je omezen parametrem *Density*. Zároveň s generováním bodů vytváří pro každý bod index do pole náhodných barev.
- Procedura **MakeTriangleGenerators** generuje síť bodů do trojúhelníkové mřížky postupem popsaným v 4.2. Hustota a pravidelnost této sítě je ovlivněna parametry *Density* a *Jitter*. Na vstup opět jako v předchozím případě dostane barvu generátorů a velikost pole náhodných barev. Při vygenerování bodů do trojúhelníkové sítě zároveň vytvoří pro každý generátor index do pole náhodných barev.
- **MakeBoundingGenerators** vytvoří takzvané hraniční generátory, což jsou 4 generátory vytvořené mimo plátno editoru, které slouží k tomu, aby hrany diagramu neutíkaly do nekonečna.
- **ComputeDiagram** je procedura, která předá souřadnice generátorů v reprezentaci *World* programu Qhull. Přesněji řečeno, zapíše tyto souřadnice do souboru, který předá programu na výpočet Voroniova diagramu. Program Qhull vrátí opět soubor, v němž jsou uloženy souřadnice vrcholů polygonů a samotné polygony, které se zde nachází ve formě indexů na polygony.

Kapitola 6

Popis a ovládání editoru Voronoiových mozaiek

Tato kapitola se věnuje popisu prostředí, práce a ovládání editoru Voronoiových mozaiek. Čtenář se zde dozví jak je vytvořeno uživatelské prostředí a jakým způsobem je možné s programem pracovat. Editor se skládá ze spustitelného souboru *Voronoι.exe* a také je potřebné ve stejném adresáři mít program *QVoronoι.exe*, což je část programu *Qhull* a zpracovává vstup a výstup, je ovládán přepínačem, jenž ovlivňuje běh a výstup programu. Další část editoru tvoří soubory *Voronoι.hlp* a *Voronoι.cnt*, které vytvářejí nápovědu k práci s editorem.

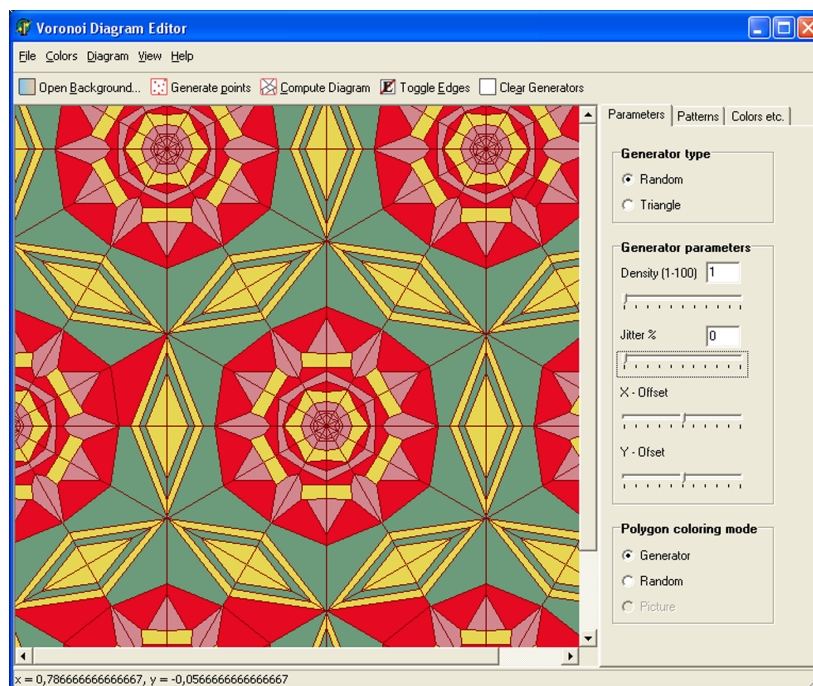
6.1 Popis uživatelského prostředí

Po otevření souboru *Voronoι.exe* se spustí editor Voronoiových mozaiek, jehož pracovní prostředí je znázorněno na obrázku 6.1.

Toto pracovní prostředí můžeme rozdělit do třech základních částí. V horní části editoru se nachází hlavní menu, které umožňuje uživateli pracovat se základními funkcemi editoru. Pod ním lze nalézt nástrojovou lištu, která obsahuje nejčastěji používané funkce v editoru, takže uživatel má možnost pracovat s touto lištou, aniž by musel listovat v hlavním menu. Střed editoru vyplňuje pracovní plátno, jenž má nastavitelnou šířku a délku. Pokud uživatel vytváří mozaiku z vlastního obrázku, plátno se přizpůsobí velikosti tohoto obrázku. Jestliže je naopak vytvářena mozaika, která používá grup symetrií, je možné si nastavit velikost plátna, na kterém se bude pracovat. Při pohybu myši po plátně, se v dolní části editoru zvané stavový pruh objevují x-ové a y-ové souřadnice kurzoru myši. Důležitou roli při kreativní tvorbě v editoru hrají také tři záložky, které se nachází v pravé části editoru. Je možné se mezi nimi libovolně přepínat a volit si tu ze záložek, která je zrovna potřebná k práci v editoru. Jednotlivé položky v editoru jsou přístupné přes klávesové zkratky `alt+podtržené písmeno`. Následující část této kapitoly je věnována stručnému popisu jednotlivých částí editoru.

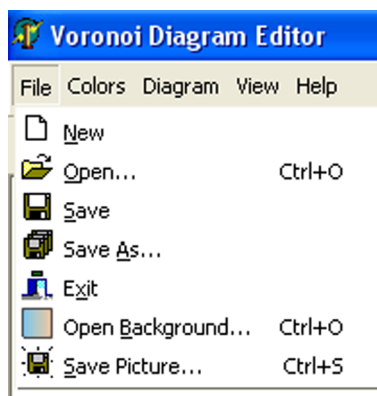
6.2 Menu

- Položka **Soubor (File)** se používá pro základní práce s editorem a k ukončení programu.
 - **Nový (New)** – Otevření a nastavení velikosti nového plátna pro kreslení mozaiek.
 - **Otevřít (Open)** – Načtení souboru, který byl v minulosti uložen.



Obrázek 6.1: Editor Voronoiových mozaiek

- **Uložit (Save)** – Zapiše pozici a aktuální barvu jednotlivých generátorů do souboru a tím umožní příště otevřít soubor s mozaikou a pokračovat v její tvorbě.
- **Uložit jako (Save as)** – Uloží soubor pod určitým jménem.
- **Konec (Exit)** – Ukončí práci s editorem.
- **Otevřít pozadí (Open Background)** – Otevře vstupní obrázek a přizpůsobí plátno jeho velikosti.
- **Uložit obrázek (Save Picture)** – Uloží uživatelem vytvořenou mozaiku jako obrázek ve formátu *bmp, jpg,....*
- **Příkaz Barvy (Colors)** slouží k barevným nastavením a změnám v mozaice.
 - **Vyber barvu generátorů (Select Generator Colors)** – Vyvolá dialogové okno pro zadání barvy generátorů.
 - **Vyber barvu hran (Select Edge Colors)** – Vyvolá dialogové okno pro zadání barvy hran Voronoiova polygonu.
 - **Změň barvy (Change Colors)** – Změní obarvení mozaiky. Polygony s identickou barvou budou přebarveny náhodně vybranou barvou. Vzhled mozaiky zůstane zachován.

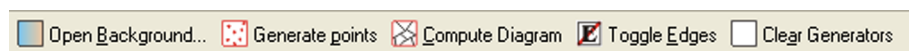


Obrázek 6.2: Menu editoru

- Položka **Diagram** se věnuje vytvoření a úpravám Voronoiova diagramu na plátně editoru.
 - **Vygeneruj body (Generate Points)** – Vytvoří síť Voronoiova diagramu buď náhodně nebo do trojúhelníkové sítě, podle toho, jaké parametry zvolil uživatel na záložce **Parametry (Parameters)**.
 - **Spočítej diagram (Compute Diagram)** – Vypočítá Voronoiov diagram z vytvořené sítě generátorů.
 - **Vyčisti generátory (Clear Generators)** – Vyčistí plátno.
- Příkaz **Zobrazit (View)** umožňuje měnit nastavení zobrazení jednotlivých částí Voronoiova diagramu.
 - **Vypni / Zapni zobrazení generátorů (Toggle Generators)** – umožňuje zapnout a vypnout viditelnost generátorů v mozaice. .
 - **Vypni / Zapni zobrazení hran (Toggle Edges)** – Zapíná a vypíná viditelnost hran polygonů ve VD.
 - **Vypni / Zapni zobrazení polygonů (Toggle Polygons)** – Přepíná viditelnost barevné výplně polygonů, takže umožňuje v případě, že je barevná výplň vypnuta, vidět, jak mřížka přiléhá na vstupní obrázek.
- Položka **Nápověda (Help)** podá uživateli nápovědu, jak pracovat s editorem. Obsahuje také konkrétní příklady a ukázky obrázků pro jednotlivé případy tvorby mozaiky. V nápovědě je možné se orientovat pomocí obsahu a vyhledávání v něm.

6.3 Nástrojová lišta

Nástrojová lišta (toolbar) obsahuje základní příkazy, které se při tvorbě mozaiky používají. Usnadňuje práci uživateli tím, že položky této nástrojové lišty jsou umístěny vedle sebe na ploše a uživatel nemusí listovat v menu. Nástrojová lišta obsahuje následující položky, jejichž funkce je vysvětlena v části popisu menu.



Obrázek 6.3: Nástrojová lišta

- Otevřít pozadí (Open Background)
- Vygeneruj body (Generate Points)
- Spočítej diagram (Compute Diagram)
- Vypni/Zapni zobrazení hran (Toggle Edges)
- Vyčisti generátory (Clear Generators)

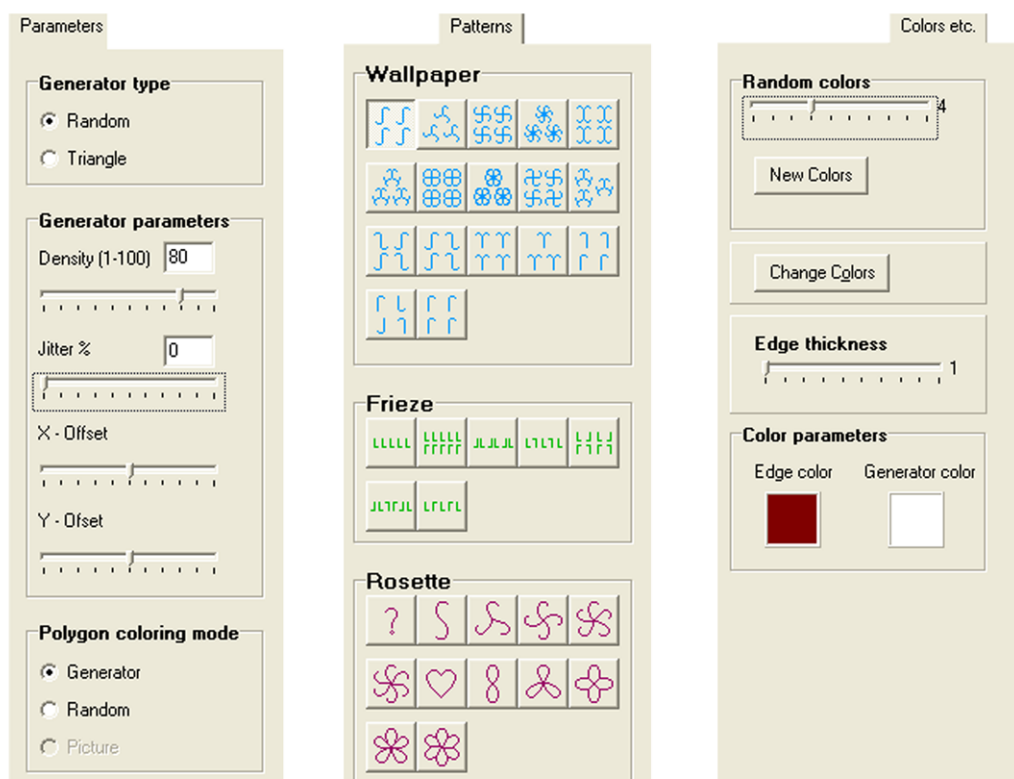
6.4 Záložky

Záložky tvoří důležitou část editoru. Umožňují komunikovat s uživatelem a vytvářet tak postupně mozaiky. Lze je rozčlenit na tři části. První záložka se věnuje parametrům generátorů při vytváření sítě VD, v druhé záložce najde uživatel tlačítka pro tvorbu mozaiek s využitím grup symetrií a ve třetí záložce je možné nastavit parametry, jenž ovlivňují vzhled mozaiky. Obrázek 6.4 ukazuje všechny tři záložky editoru Voroniových mozaiek.

6.4.1 Záložka Parametry (Parameters)

Záložka **Parametry** se věnuje nastavení hodnot potřebných pro výpočet sítě VD.

- Položka **Typ generátoru (Generator type)** určuje, jakým způsobem se vygeneruje síť VD.
 - **Náhodný (Random)** – Generátory se budou generovat náhodným způsobem.
 - **Trojúhelník (Triangle)** – Generátory budou vytvářet trojúhelníkovou síť.
- Položka **Parametry generátorů (Generator Parameters)** určuje konkrétní vlastnosti generátorů v rovině.
 - **Hustota (Density)** – Určuje jakou hustotu bude mít síť VD. Je možné volit hodnotu v rozmezí 1–100.
 - **Roztřesení (Jitter)** – Určuje jaké bude celkové roztřesení sítě bodů, jaká bude použita náhoda při jejich generování. Použití této položky má smysl pouze v případě trojúhelníkové sítě a při použití grup symetrií. Je možné volit hodnotu v rozmezí 0–100.
 - **Posun po x-ové ose (X-Offset)** – Posuvník, který určuje o kolik se posune mřížka ve směru x-ové osy.



Obrázek 6.4: Záložky: Parametry, Vzory, Barvy atd.

- **Posun po y-ové ose (Y-Offset)** – Posuvník, který určuje o kolik se posune mřížka ve směru y-ové osy.
- Položka **Obarvovací mód polygonů (Polygon coloring mode)** určuje jakým způsobem se vybarví polygony v mozaice.
 - **Generátor (Generator)** – Barva pro výplň polygonů je určena barvou generátoru, kterou zvolil uživatel.
 - **Náhodný (Random)** – Barva polygonu je zvolena náhodně.
 - **Obrázek (Picture)** – Vybarvení polygonů podle vstupního obrázku. Barva polygonu je závislá na barvě pixelu, který se nachází na stejné pozici v obrázku jako generátor Voronoiova polygonu.

6.4.2 Záložka Vzory (Patterns)

Záložka **Vzory** slouží uživateli k vytváření mozaiek s využitím grup symetrií. Uživatel si zvolí jednu z grup symetrií a kliknutím na plátno vytváří mozaiky. Tato záložka obsahuje celkem 36 grup symetrií, rozdělených do třech vzorů.

- **Tapetový vzor (Wallpapers)** – Obsahuje celkem 17 grup symetrií.
- **Vlysový vzor (Frieze)** – Obsahuje celkem 7 grup symetrií.
- **Rozetový vzor (Rosette)** – Obsahuje celkem 12 grup symetrií.

6.4.3 Záložka Barvy atd. (Colors etc.)

Tato záložka je věnována nastavení parametrů důležitých pro vzhled mozaiky. To znamená barevná nastavení generátorů, hran, ale také tloušťku hran.

- Položka **Náhodné barvy (Random Colors)** obsahuje posuvník s hodnotami 1–10, který je implicitně nastaven na hodnotu 4. Zde si uživatel nastaví kolik náhodných barev se má při vybarvení polygonů nastavit. Tlačítko **Nové barvy (New Colors)** přiřadí mozaice tolik náhodných barev, kolik jich uživatel na posuvníku nastavil.
- Tlačítko **Změň barvy (Change Colors)** změni barvy v mozaice, takovým způsobem, že polygony vybarvené identickou barvou budou přebarvené jinou náhodně vybranou barvou, tak že počet barev v mozaice se nezmění.
- Posuvník **Tloušťka hrany (Edge Thickness)**, který má hodnoty 1–10 s implicitně nastavenou hodnotou 1, určuje tloušťku hran Voronoiova polygonu.
- Položka **Vlastnosti barev (Color Parameters)** obsahuje dvě barevné okna s označením **Barva hran (Edge Color)** a **Barva generátorů (Generator Color)**. Při kliknutí na každé z těchto oken se objeví dialog pro zadávání obarvení hran a generátorů. Zde si uživatel volí, jakou barvou budou vybarveny polygony a jejich hrany.

Kapitola 7

Práce s editorem

V této kapitole se budu snažit naznačit způsob práce s programem. Budou zde ukázány základní postupy, jak získat požadovaný výstup ve formě obrázku či souboru. Kapitola je doprovázena krátkými ukázkami těchto postupů. Obsáhlejší výstupy jsou prezentovány v příloze této diplomové práce. A konkrétní nastavení všech jednotlivých parametrů je k dispozici v nápovědě, jež je součástí programu. Zde najde uživatel jejich popis a také detailní ukázky. Tato kapitola si klade za cíl vysvětlit princip práce s editorem, za předpokladu, že čtenář je obeznámen s funkcí programu popsanou v předchozí kapitole 6.

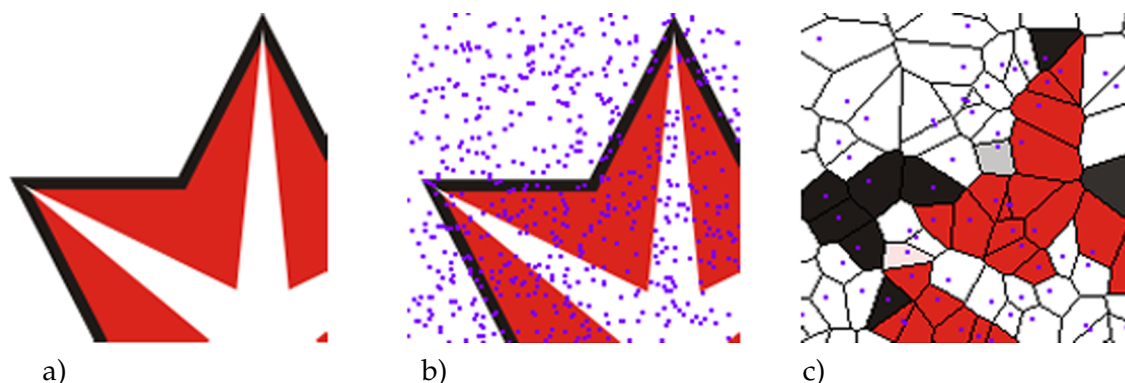
7.1 Tvorba mozaiky vytvořené ze vstupního obrázku

Celý postup bude prezentován na obrázku 7.1 a). Obsahuje pouze dvě barvy a hrany, takže plně postačí pro náš účel předvedení jednotlivých funkcí programu. Pro tvorbu mozaiky z uživatelem zadaného obrázku je potřeba mít obrázek v jednom z těchto formátů: *jpg*, *bmp*, *ico*, *wmf*, *emf*. Obrázek se načte na pozadí příkazem *File > Open Background*. Je možné také použít klávesovou zkratku *Ctrl+B*. Tímto se otevře obrázek, se kterým budeme dále pracovat. V záložce *Parameters* se automaticky nastaví způsob obarvování na volbu *Picture*. Velikost plátna se nastaví na velikost obrázku. Další krok je závislý na typu mřížky, kterou si uživatel přeje vytvořit. Mřížka může být vygenerována automaticky nebo vytvořena ručně uživatelem.

- **Automatické vytvoření sítě bodů**

Při této volbě se uživatel rozhodne, zda si přeje vygenerovat body náhodně v ploše či do trojúhelníkové sítě.

- V záložce *Parameters* si uživatel vybere typ generátorů a označí příslušnou položku, tzn. označí buď volbu *Random* (v případě náhodně rozmístěných bodů) nebo označí volbu *Triangle*, pokud mají být generátory umístěny do trojúhelníkové mřížky.
- V záložce *Generator Parameters* je potřeba nastavit hustotu sítě generátorů. Tento parametr se nastaví v posuvníku s označením *Density*, kde hodnota 1 značí nejmenší hustotu generátorů a hodnota 100 hustotu největší. V posuvníku *Jitter* se zvolí, zda chceme, aby při tvorbě Voronoiových generátorů byla použita nepravidelnost. Tato volba má smysl pouze tehdy, pokud generujeme síť trojúhelníků a umožňuje zavést do jejich tvorby mírnou až velkou nepravidelnost a tím ovlivnit celkový vzhled mozaiky. Pokud si uživatel přeje posunout síť



Obrázek 7.1: a) Vstupní obrázek b) Vygenerované body c) Sít' VD

generátorů po x-ové či y-ové ose, použije posuvníky s označením *X-Offset* a *Y-Offset*.

- V dalším kroku se vypočítá sít' generátorů dle uživatelského nastavení. Příkazem *Diagram > Generate Points*, se vytvoří sít' Voronoiových generátorů. Tuto situaci ilustruje obrázek 7.1 b). Generátory zde byly vytvořeny náhodně, hustota sítě je nastavena na hodnotu 40.
- Příkazem *Diagram > Compute Diagram* vznikne mozaika tvořená Voronoiovým diagramem. Příklad náhodně vygenerované sítě na obrázku 7.1 c), příklad trojúhelníkové sítě je na obrázku 7.2 c).

• Uživatelem vytvořená sít' generátorů

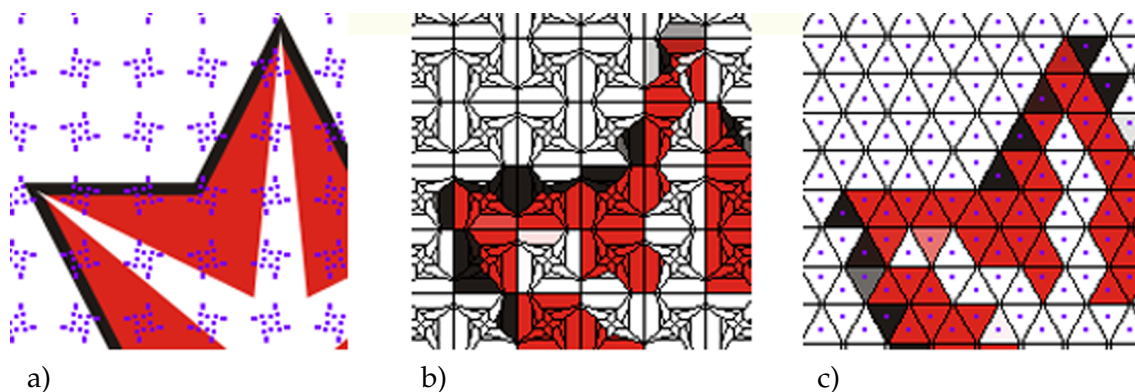
Pokud si uživatel přeje vytvořit sít' generátorů sám, umožňuje mu to záložka *Patterns*. Zde jsou k dispozici tři paletky s přednastavenými skupami symetrií, které určují, jakým způsobem se budou body generovat. Tyto jednotlivé vzory byly popsány v kapitole 6.2.

V paletce *Wallpapers* je možné vybírat ze 17 grup symetrií. Tyto grupy symetrií pokrývají plochu plátna rovnoměrně po celé jeho šířce a výšce, způsobem, který odpovídá určité, uživatelem zvolené symetrii.

V paletce *Frieze* je možné vybírat ze 7 grup symetrií. Při této volbě se budou generátory vytvářet pouze v pásu umístěného uprostřed obrazovky. Tato volba nemá u mozaiek tvořených ze vstupních obrázků příliš velké uplatnění.

V paletce *Rosette* je možné vybírat ze 12 grup symetrií, které vytvářejí sít' generátorů do kulatého tvaru, nepokrývají (stejně jako v případě volby *Frieze*) celé plátno rovnoměrně, proto jejich použití má spíše význam pro mozaiky vytvářené ručně, než ze vstupního obrázku.

- V záložce *Generator Parameters* je potřeba v posuvníku *Density* nastavit hustotu sítě generátorů a v posuvníku *Jitter* se zvolí, zda chceme, aby při tvorbě Voronoiových generátorů byla použita nepravidelnost. V případě, že by měla být



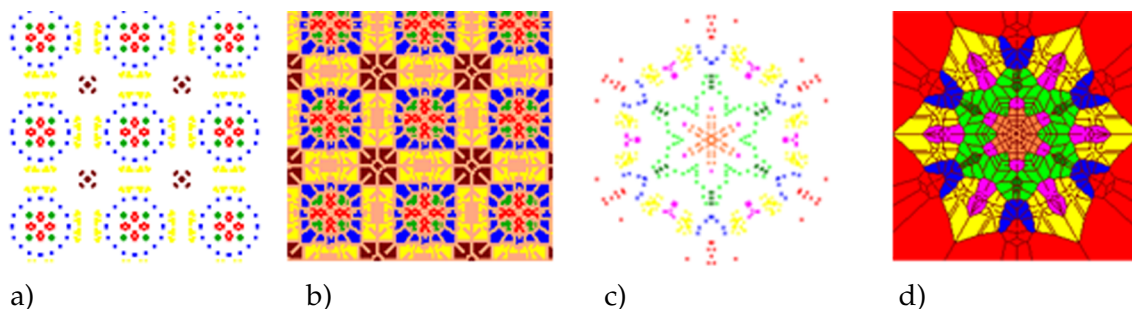
Obrázek 7.2: a) Vstupní obrázek se sítí bodů b) Mozaika c) Trojúhelníková síť VD

síť bodů posunuta v horizontálním či vertikálním směru, zvolí se tato možnost v posuvnících *X-Offset* a *Y-Offset*.

- V záložce *Colors etc.* je k dispozici položka *Color Parameters*, ve které se nachází okno *Edge Color*. Zde si uživatel zvolí barvu hran sítě VD. A v posuvníku *Edge Thickness* si nastaví tloušťku hran Voronoiových polygonů.
- Po těchto základních nastaveních si uživatel v záložce *Patterns* zvolí jednu z grup symetrií. Kliknutím na plátno, na kterém je na podkladě umístěn vstupní obrázek, se vygenerují body. Na obrázku 7.2 a) je příklad vstupního obrázku, na který byla umístěná síť tvořená jednou z grup symetrií tapetového vzoru.
- Příkazem *Diagram > Compute Diagram* vznikne mozaika tvořená Voronoiovým diagramem. Příklad mozaiky tvořené vstupním obrázkem a uživatelem vytvořené mřížky demonstruje obrázek 7.2 b).

Editor Voronoiových mozaiek umožňuje změnit barevné nastavení a tloušťku hran, i když mozaika byla již vytvořena. Pro tuto změnu se opět v záložce *Colors etc. > Color Parameters > Edge Color* vybere nová barva a hrany se automaticky přebarví. V posuvníku *Edge Thickness* je možné dodatečně změnit sílu hran. Změna se opět projeví automaticky. Příklad mozaiek vzniklých ze vstupního obrázku z různě nastavenými parametry je k dispozici v příloze této diplomové práce.

V případě, že uživatel není s výsledkem své práce spokojený, může příkazem *Diagram > Clear Generators* síť generátorů smazat a začít s prací znovu. V opačném případě je možné uložit rozmístění bodů v ploše do souboru příkazem *File > Save*. Uloží se pozice všech generátorů a při otevření souboru příkazem *File > Open* se síť bodů opět umístí do plochy do stejné pozice. Tato volba je vhodná v případě, že si uživatel přeje pracovat a vylepšovat síť po delší době. Jestliže považuje uživatel práci na mozaice za ukončenou, uloží obrázek příkazem *File > Save Picture*. Obrázek lze uložit ve formátech *jpg*, *bmp*.



Obrázek 7.3: a) Sít' VD tvořená vícebarevnými body v tapetovém vzoru b) Mozaika vzniklá z této sítě c) Body v rozetovém vzoru d) Mozaika typu rozeta

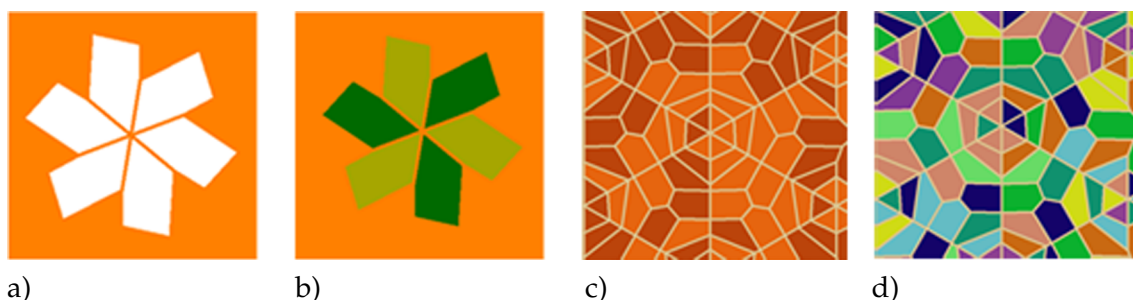
7.2 Tvorba mozaiky z vlastních mřížek

Voronoi Editor umožňuje uživateli vytvořit mozaiku, která vznikne vygenerováním sítí Voronoiových generátorů uživatelem. Princip je založený na generování různobarevných bodů na ploše dle určitých pravidel a následnému spočítání Voronoiova diagramu. Uživatel může využívat mnoha parametrů v nastavení Voronoi editoru a tím aktivně měnit vzhled samotné mozaiky.

Postup při tvorbě mozaiky tohoto typu se skládá z následujících kroků:

• Tvorba mřížek

- Při práci s vlastními mřížkami si uživatel na začátku nejdříve nastaví velikost plátna, na kterém bude pracovat. Toto nastavení je možné provést příkazem *File > New*, při jehož vyvolání se objeví dialogové okno, do kterého uživatel zadá rozměry plátna, na kterém bude pracovat.
- V záložce *Patterns* si uživatel vybere jednu z grup symetrií, kterou bude používat.
- V záložce *Generator Parameters* je potřeba nastavit hustotu a roztřesení sítě generátorů. Tyto parametry se opět nastaví v posuvnících *Density* a *Jitter*. Pokud si uživatel přeje posunout síť generátorů po x-ové a y-ové ose, použije posuvníky s označením *X-Offset* a *Y-Offset*. Posunutí ve směru těchto os se využije zejména při tvorbě pásového či rozetového vzoru. Například při generování rozety se může stát, že uživatel neklikne přesně do středu a kousek mozaiky je tak schován pod hranicí viditelnosti a tímto posunem je možné mozaiku zviditelnit celou.
- V záložce *Generator Parameters* se automaticky nastaví obarvovací mód na položku *Generator*, což znamená, že Voronoiovy polygony se budou vybarvovat barvou generátoru.



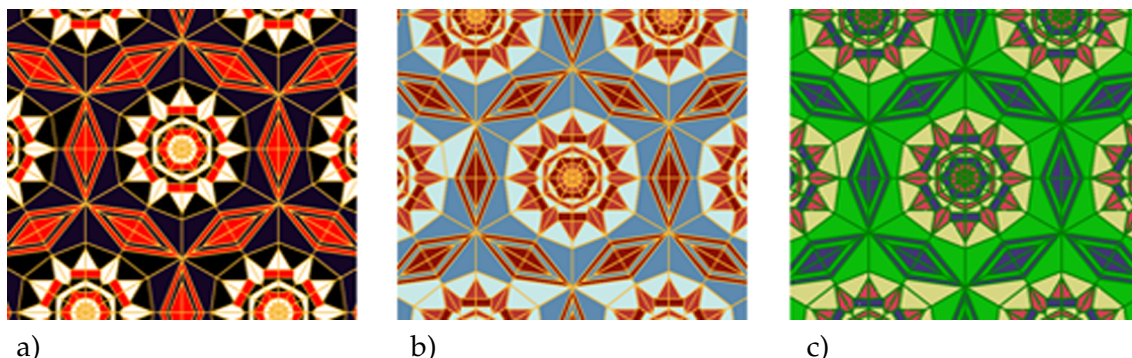
Obrázek 7.4: a) Vytvořená mozaika b) Mozaika, která má různě obarvené polygony c) Mozaika vybarvená náhodným způsobem dvěma barvami d) Mozaika vybarvená náhodným způsobem deseti barvami

- V záložce *Colors etc.* je k dispozici položka *Color Parameters*, ve které si uživatel zvolí barvu generátorů, podle nichž se potom vybarví Voronoiovy polygony. V této položce si také vybere barvu hran polygonů. Po kliknutí na okna *Edge Color* a *Generator Color* se objeví barevná dialogová paleta, ze které je možné zvolit požadovanou barvu.
- Po těchto základních nastaveních uživatel kliknutím na plátno vytvoří určitý vzor. V tomto okamžiku je možné opět zvolit novou barvu generátorů v položce *Colors etc. > Generator Color*. V okamžiku dalšího kliknutí na plátno se vytvoří nová skupinka barevných bodů. Takto je možné postupovat dle uvážení ještě několikrát. Příklad takto vytvořené sítě VD ukazuje obrázek 7.3. V případě a) lze vidět body uspořádané do tapetového vzoru a z nich vzniklou mozaiku, v případě c) je ukázka sítě bodů uspořádaných do rozetového vzoru.
- Příkazem *Diagram > Compute Diagram* vznikne výsledná mozaika.

Pokud si uživatel přeje vypnout zobrazení generátorů, hran či polygonů Voronoiova diagramu, zvolí příkaz *View > Toggle Generators (> Toggle Edges)(> Toggle Polygons)* stejnými příkazy je možné zobrazit generátory, hrany a polygony zpět. Při otevření souboru příkazem *File > Open* se síť bodů opět umístí do plochy do stejné pozice. Výsledný obrázek je možné uložit ve formátech *jpg*, *bmp* a to příkazem *File > Save Picture*.

7.3 Změna obarvení mozaiky

V případě, že si uživatel vytvořil zajímavou mozaiku, má k dispozici mřížku generátorů, ale není spokojený s barvami, které pro svoji mozaiku použil, je i přesto možné toto obarvení Voronoiových polygonů změnit. V podstatě lze využít dvou přebarvovacích principů ke změně barev v mozaice. Barvy lze nahradit zcela náhodným způsobem nebo zachovat polygony, které jsou identicky obarvené a jen změnit jejich barvu.



Obrázek 7.5: Mozaika tapetového vzoru, která byla dvakrát přebarvena způsobem zachovávajícím počet barev.

- **Náhodné obarvení polygonů**

- Při náhodném obarvení polygonů se v záložce *Parameters* automaticky nastaví obarvovací mód na položku *Random*.
- Uživatel bude pracovat se záložkou *Colors etc.*, ve které je k dispozici položka *Random Colors*. V posuvníku v ní umístěném si zvolí kolik náhodných barev by mělo být při přebarvení mozaiky použito a zmáčkne tlačítko *New Colors*.
- Po každém zmáčknutí tlačítka se vygeneruje určitý počet náhodných barev nastavených v posuvníku. Příklad mozaiky, která byla obarvena náhodným způsobem různými počty barev lze vidět na obrázku 7.4 c) a d). V případě c) byly použity dvě náhodné barvy a v případě d) bylo použito deset barev.

- **Změna obarvení mozaiky se zachováním počtu barev**

V záložce *Parameters* se v tomto případě obarvovací mód nastaví na hodnotu *Generator*. Polygony, které mají identickou barvu změní barvu generátoru (ta bude opět u těchto polygonů stejná) se přebarví novou náhodně vybranou barvou. Tato změna se provede u všech polygonů v mozaice. V záložce *Colors etc.* je umístěno tlačítko *Change Colors*. Po jeho stisknutí se provede požadovaná změna přebarvení se zachováním počtu barev v mozaice. Postup je opět možné opakovat až dosáhneme líbivého obarvení mozaiky. Tento případ přebarvení mozaiky demonstruje obrázek 7.5.

- **Změna barvy jen určitého počtu polygonů**

Pokud byla vytvořena mozaika a uživatel je spokojený s tvarem mřížky i s celkovým obarvením a požaduje změnu barvy pouze jednoho nebo určitého počtu polygonů v mozaice, přichází i tato možnost v úvahu.

Uživatel bude pracovat se záložkou *Colors, etc.* V položce *Color Parameters* se otevře paletka pro obarvování generátorů (kliknutím na volbu *Generator Color*). Uživatel si vybere vhodnou barvu a pravým tlačítkem myši klikne do oblasti polygonu, který chce přebarvit. Změní se tak barva polygonu a je možné zvolit další barvu a tím obarvit další polygon či polygony. Příklad uvádí obrázek 7.4. V případě a) je mozaika původní, v případě b) byly polygony dodatečně obarveny výše popsáním způsobem.

Kapitola 8

Závěr

Hlavním úkolem mé diplomové práce bylo vytvořit programové dílo, které by dokázalo tvořit mozaiky založené na Voronoiových diagramech. Měly by to být mozaiky, které vzniknou z uživatelem zadaného obrázku, ale také mozaiky, které uživatel vytvoří sám. Výsledkem by měl být výstupní obrázek ve vhodném formátu, který je možné dále použít či dále upravovat v jiných grafických editorech.

Snažila jsem se využít také algoritmů a programů, které byly již vytvořeny a naprogramovány, a tím urychlit běh programu a také vše skloubit a vytvořit tak funkční program pro tvorbu mozaiek. Použití programu Qhull mi usnadnilo práci při výpočtech Voronoiových diagramů.

Při prvním návrhu samotného editoru jsem zamýšlela vytvořit pouze algoritmus pracující se vstupním obrázkem. Ale při postupném seznamování se s touto tematikou jsem našla mnoho jiných zajímavých problémů, které by stály za to, abych se nimi ve své diplomové práci dále zabývala. Velmi mne nadchla myšlenka vytvořit si vlastní mřížky, které se dále přiloží na vstupní obrázek a výsledek má potom určitou pravidelnost. Začala jsem se tedy zabývat nápadem umístit Voronoiovy generátory v rovině takovým způsobem, aby výsledek tvořil symetrickou mřížku. K tomuto jsem využila vlastností grup symetrií. Jenže s tímto nápadem byl spojen ihned nápad další, a to rozšířit samotný editor o tvorbu mozaiek tapetového, pásového a rozetového typu, které budou tvořeny Voronoiovým diagramem a nebudou závislé na vstupním obrázku, ale předpokládají spolupráci s uživatelem. Takto vznikl návrh editoru popsaného v této práci.

Určitý čas jsem se také zabývala volbou vhodného programovacího prostředí. Původně jsem program začala psát v programovacím jazyce C++ a prostředí KDE. Uvědomila jsem si však, že takto napsaný editor by nemusel být snadno přístupný ve výuce, proto jsem raději zvolila prostředí Borland - Delphi 6. Editor vytvořený v tomto prostředí je snadno spustitelný na všech počítačích v učebně pro výuku výtvarné informatiky.

S programem se podle mého názoru pracuje dobře, protože jeho rychlost je velká. Práci s editorem usnadňuje program Qhull, který byl pro výpočet Voronoiových diagramů použit. Takže během několika sekund lze vytvořit mozaiky založené na Voronoiových diagramech. Také uživatelské prostředí jsem se snažila navrhnout tak, aby bylo na první pohled intuitivní. Editor se skládá z několika komponent, jež jsou věnované vždy určité oblasti práce s editorem.

Největším problémem, se kterým jsem se při tvorbě editoru setkala, byla definice transformací jednotlivých grup symetrií. Tyto definice jsem převzala z volně dostupného programu Java Kali a přepsala z programovacího jazyka Java do prostředí Borland Delphi.

V textu této diplomové práce jsem se snažila popsat vývoj takového editoru. Od teorie,

přes popis návrhu až po implementaci a prezentaci výsledků mého programu. Zaměřila jsem se zejména na popis a funkčnost samotného programu.

Snahou mé práce bylo tedy vytvořit editor, který by se dal využít v oblasti výtvarné informatiky. Existuje mnoho dalších nápadů, které by bylo možné začlenit do funkcí editoru Voronoiových mozaiek a tím jeho funkčnost rozšířit. To by však musela součást další diplomové práce, protože již tak jsou funkce mého programu velmi obsáhlé. Velmi mě zaujala myšlenka tvořit fraktály s využitím Voronoiových diagramů, která je popsána v [5]. Možná by tato idea mohla být námětem další diplomové práce.

Literatura

- [1] Okabe, A., Boots, B., Sugihara, K.: *Spatial Tesselations: Concepts and Applications of Voronoi diagrams.*, Chichester: John Wiley and Sons, c2000.
- [2] Slovák, J.: *Geometrické algoritmy*, Masarykova Univerzita v Brně, c2003.
- [3] Shamos, M.I., Hoey, D.: *Closest-point problem.*, Proc. of the 16th Annual IEEE Symposium of Foundations of Computer Science, c1975.
- [4] Kaplan, C.S.: *Voronoi diagrams and ornamental design.*, University of Washington: Department of Computer Science and Engineering.
- [5] Clifford, A.P.: *Chaos and fractals: A computer graphical journey: ten years compilation of advanced research* / edited by Clifford A. Pickover / Amsterdam: Elsevier, c1998
- [6] Serba, I., Staudek, T.: *Výtvarná informatika.*, Fakulta informatiky MU v Brně, c2002
- [7] <http://www.qhull.org/>
- [8] <http://www.geom.uiuc.edu/java/Kali/welcome.html>
- [9] <http://mathworld.wolfram.com>
- [10] <http://www.scienceu.com>

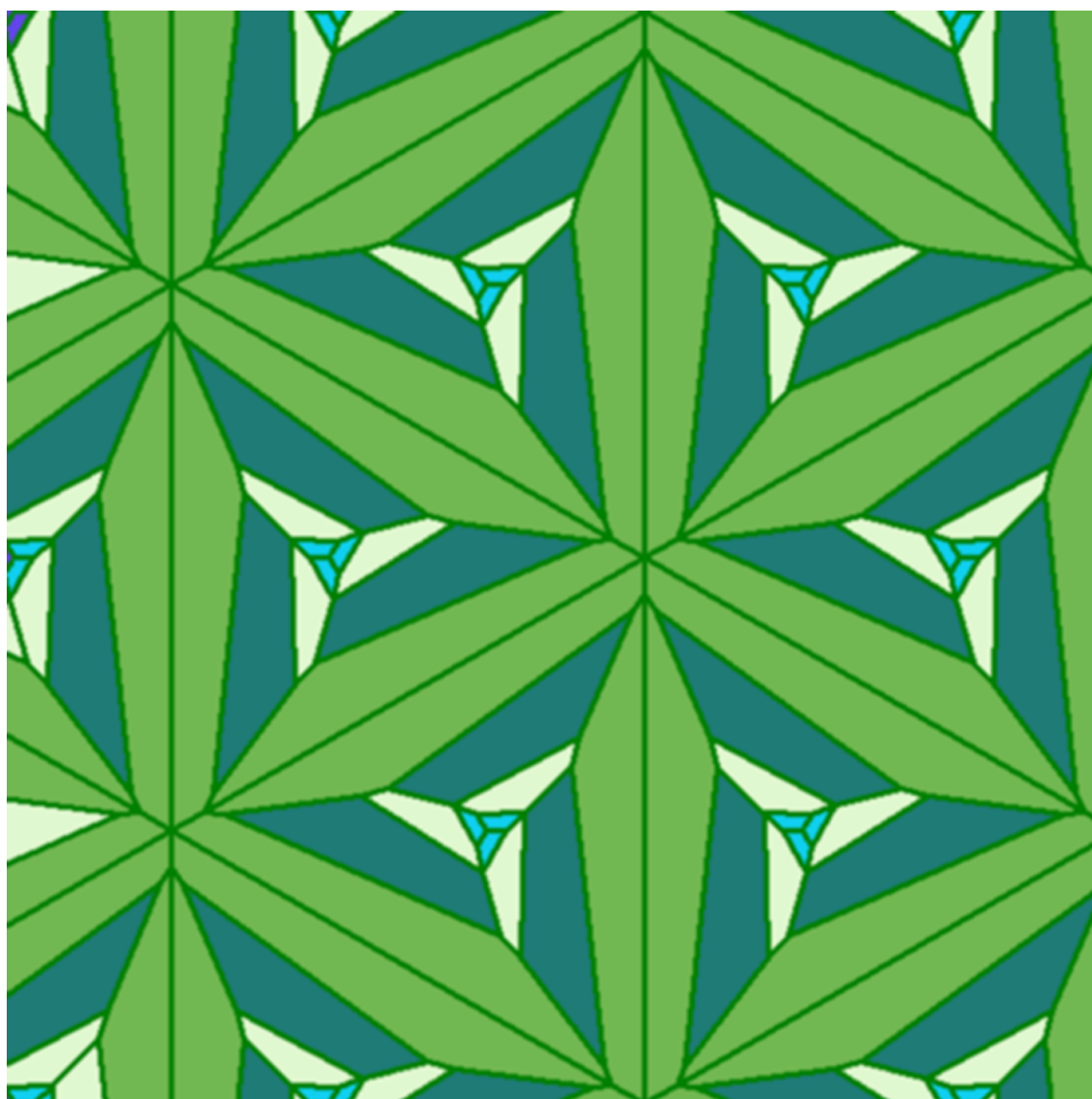
Příloha A

Příloha

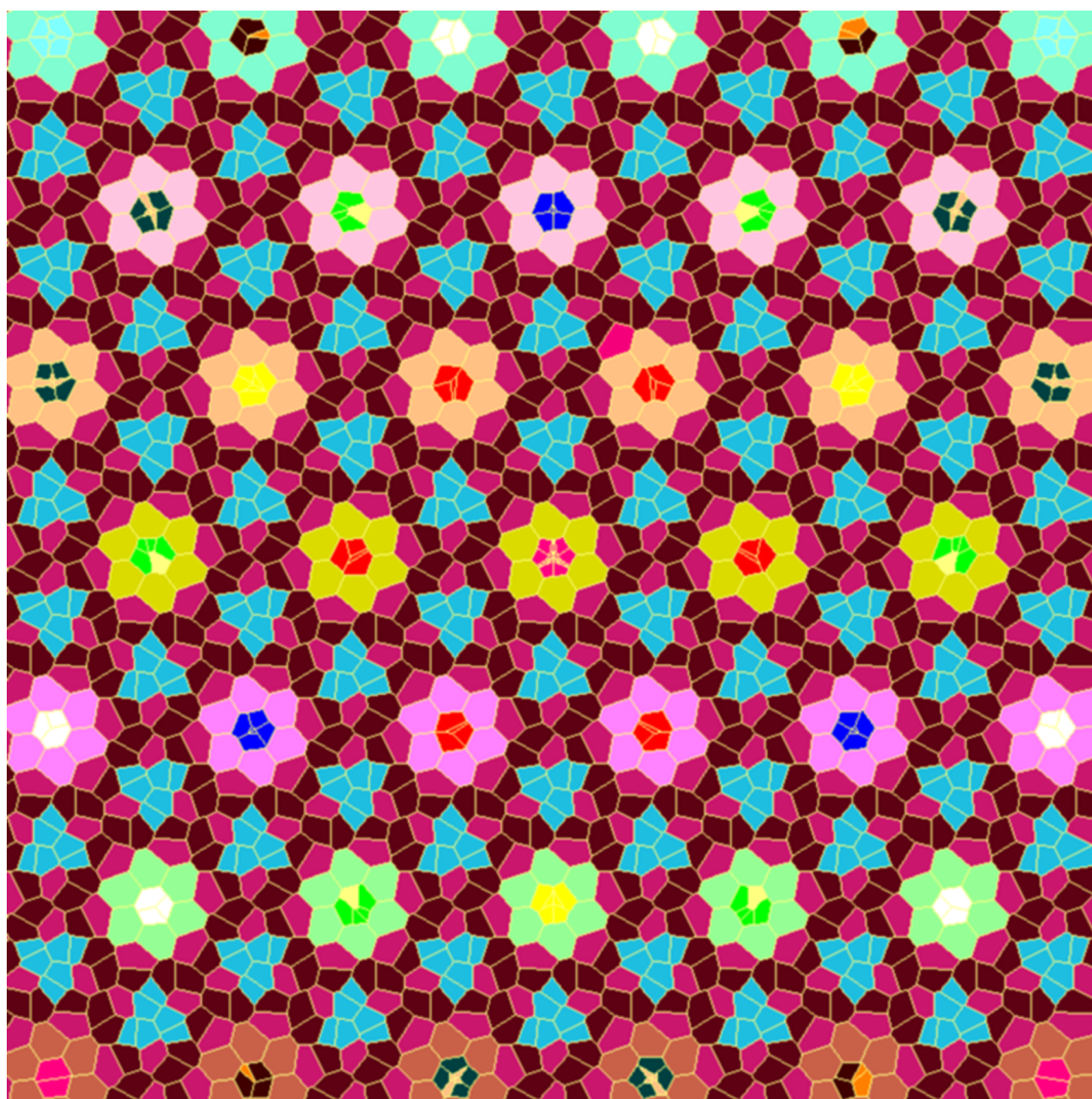
Obrázek A.1: Tapeta č.1



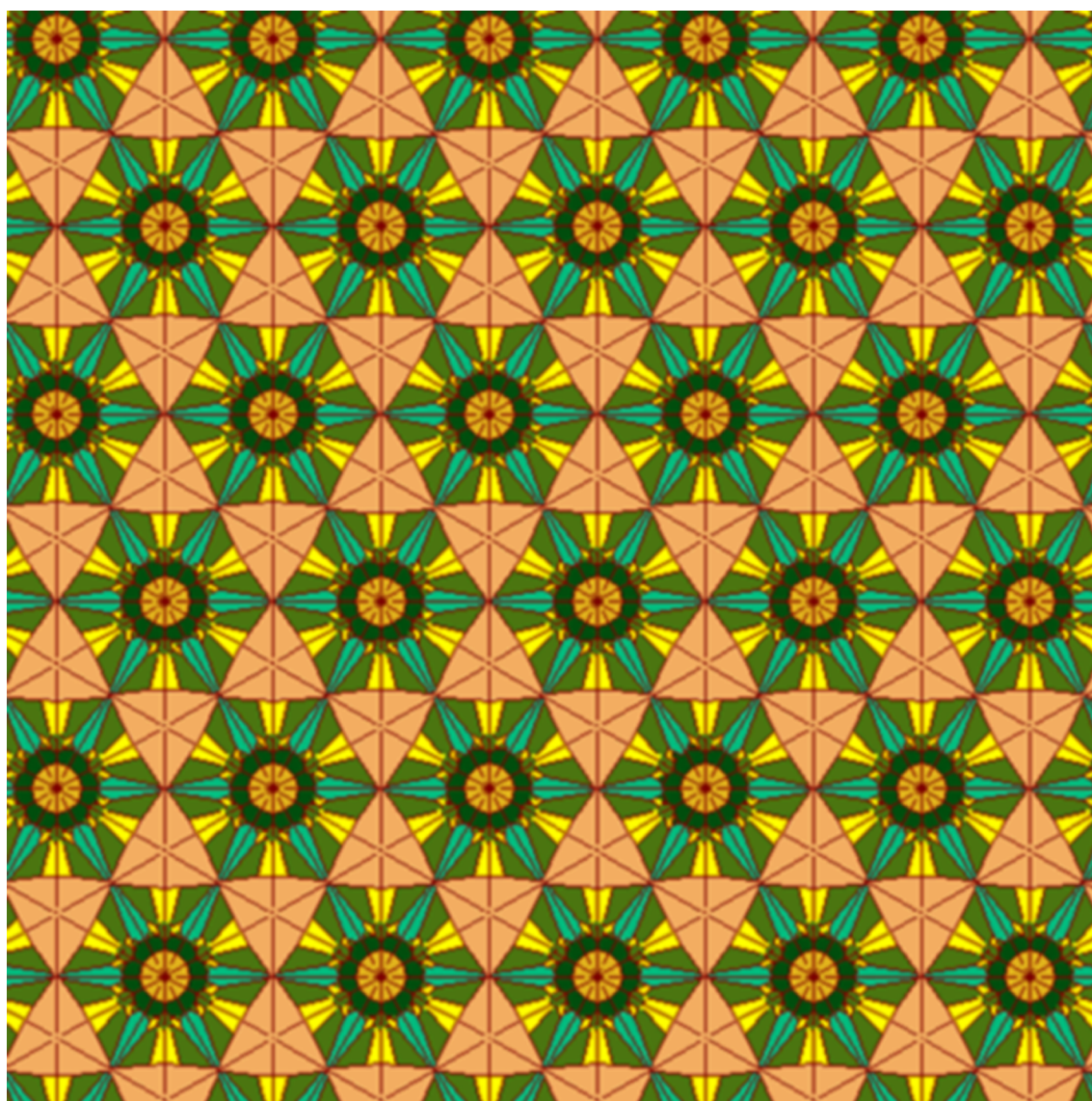
Obrázek A.2: Tapeta č.2



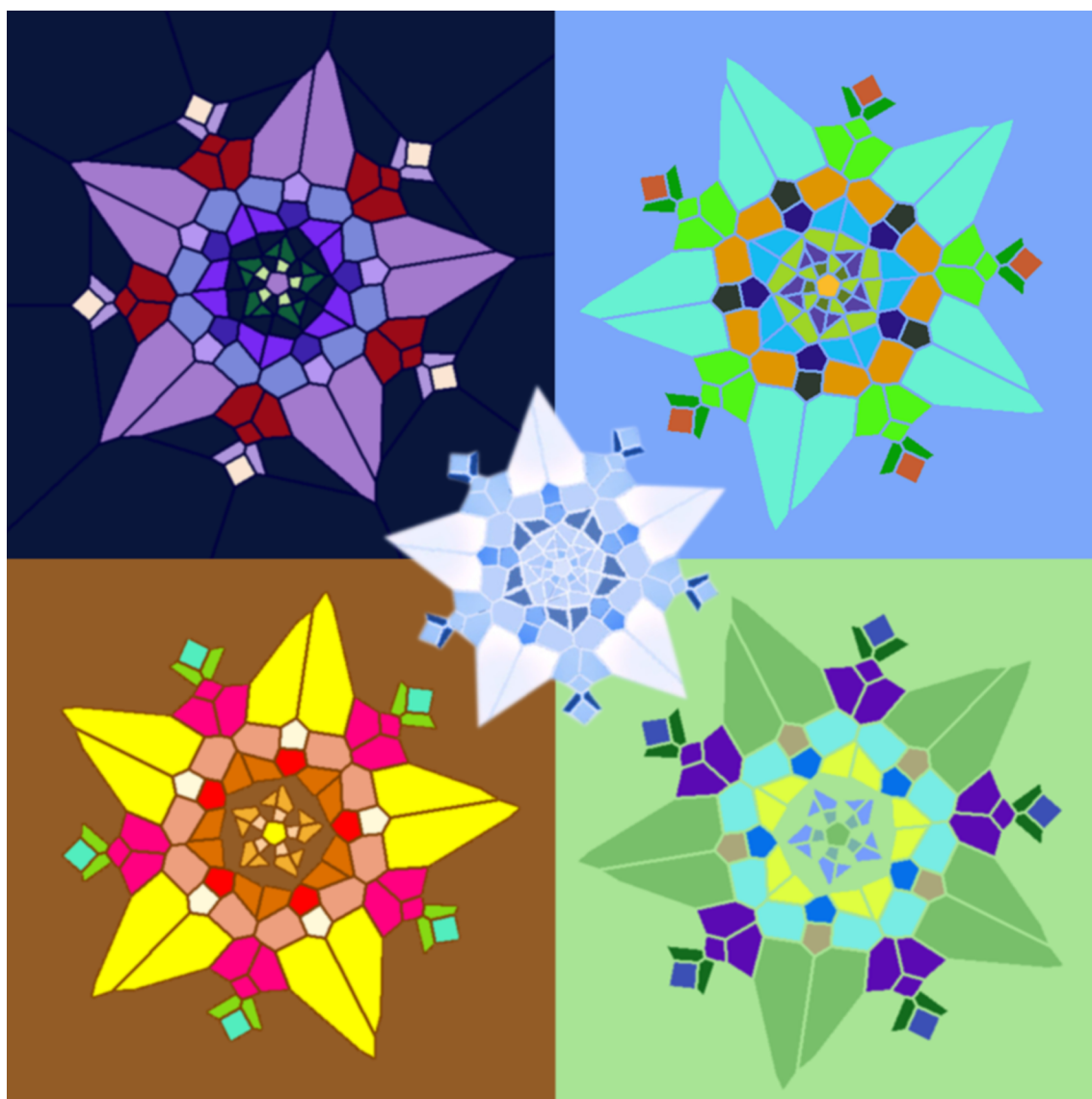
Obrázek A.3: Tapeta č.3



Obrázek A.4: Tapeta č.4



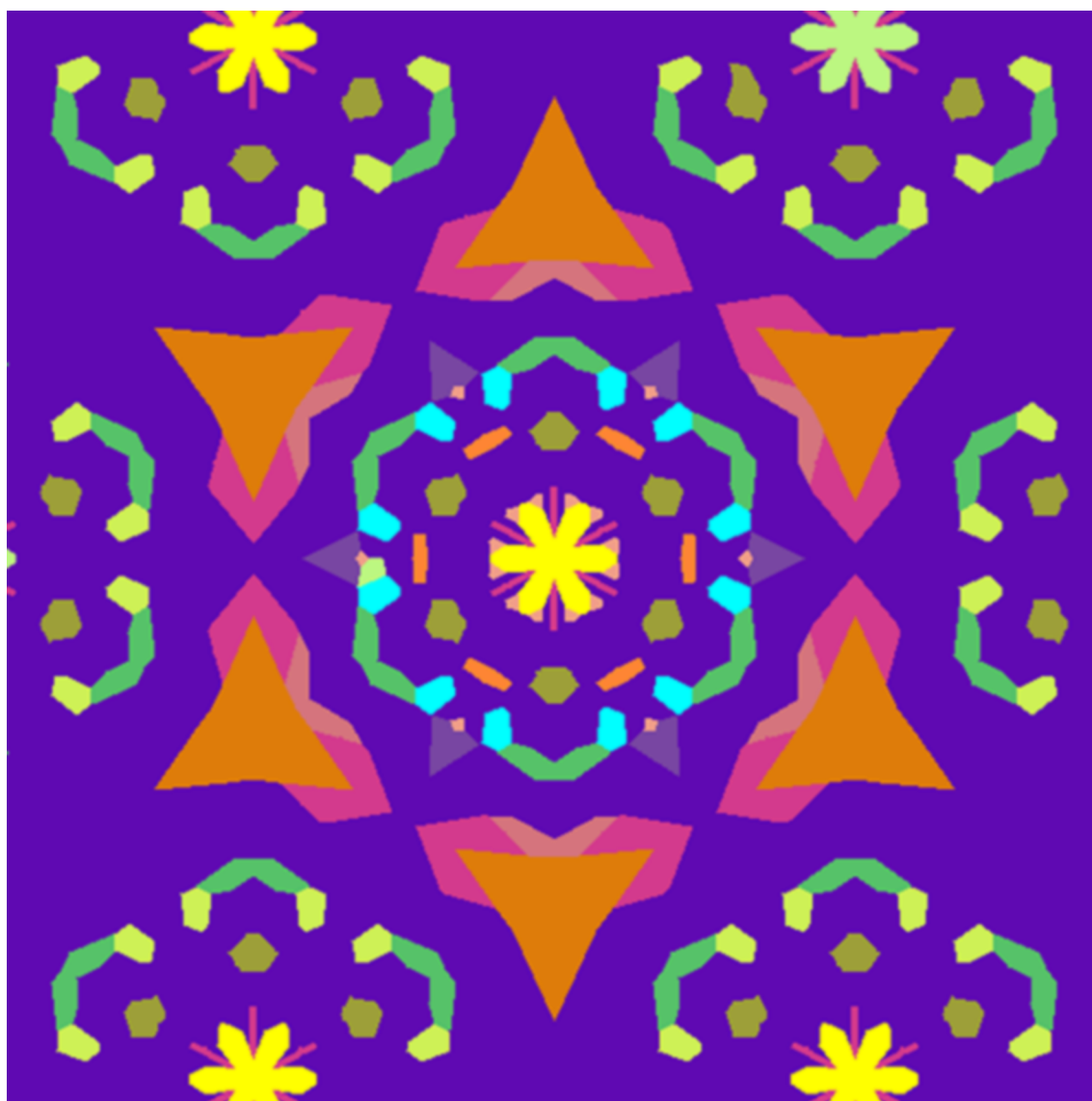
Obrázek A.5: Rozeta vybarvená několika způsoby



Obrázek A.6: Rozeta č.1



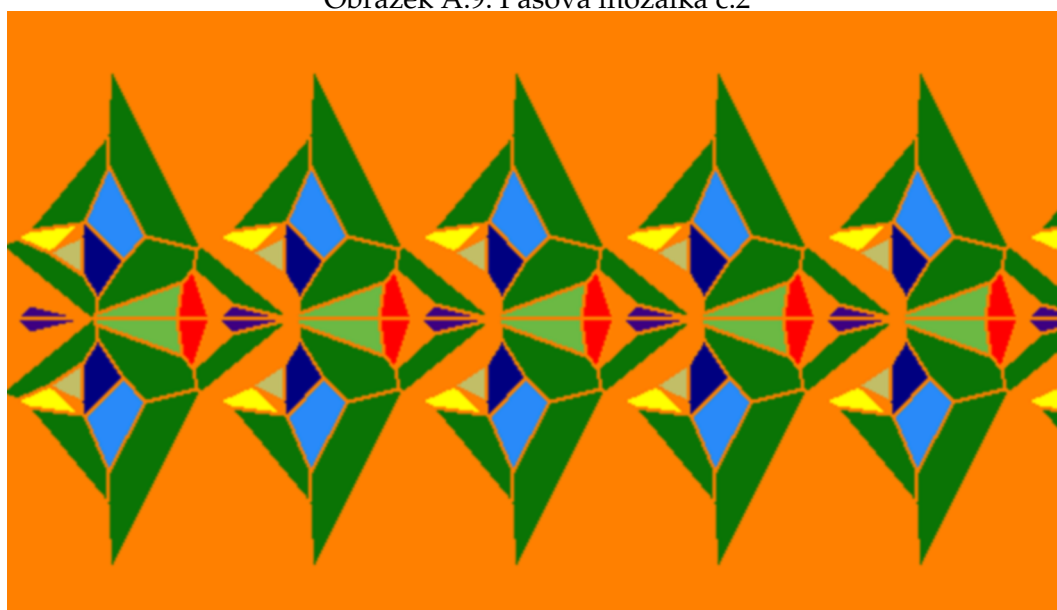
Obrázek A.7: Rozeta č.2



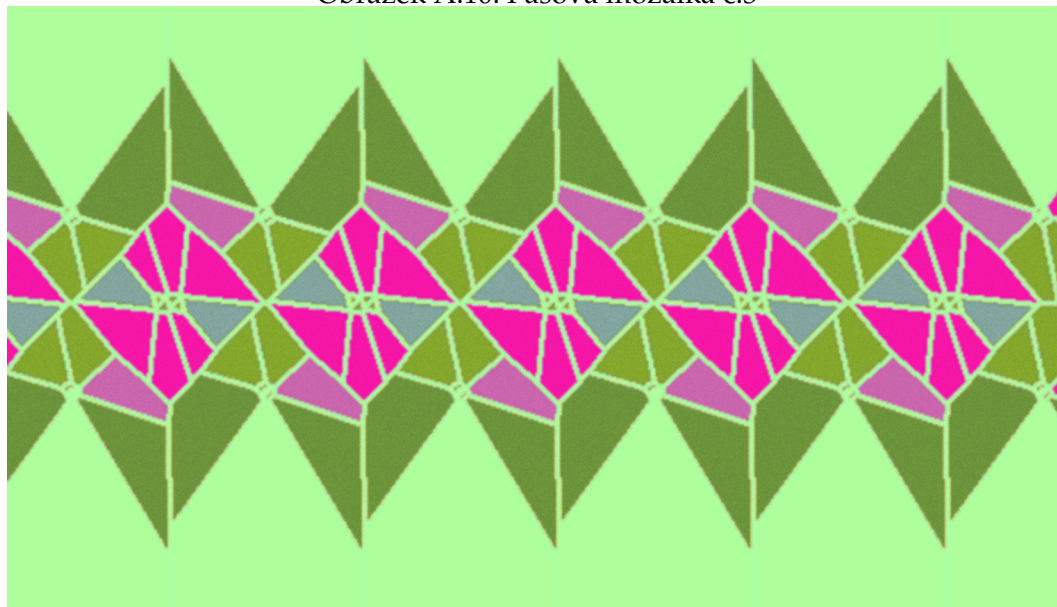
Obrázek A.8: Pásová mozaika č.1



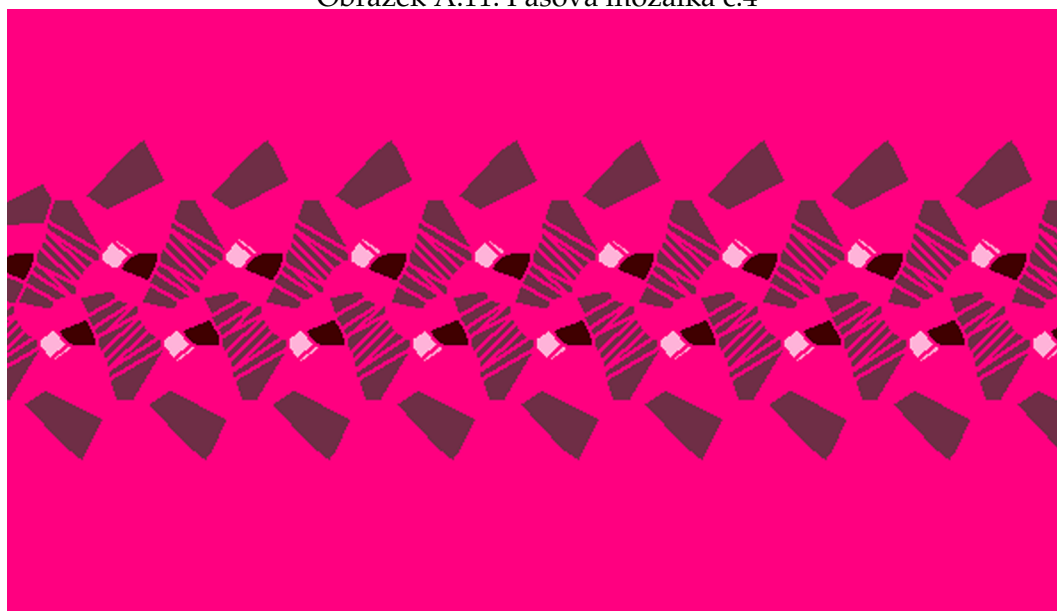
Obrázek A.9: Pásová mozaika č.2



Obrázek A.10: Pásová mozaika č.3



Obrázek A.11: Pásová mozaika č.4



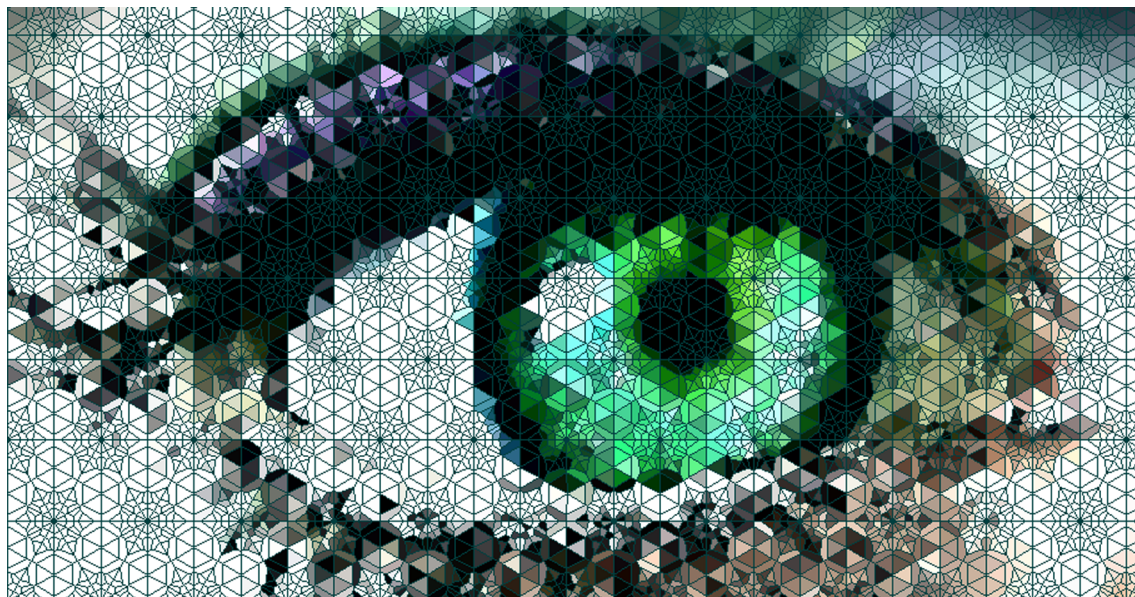
Obrázek A.12: Mozaika ze vstupního obrázku z mřížky pomocí grup symetrií, se zobrazením hran



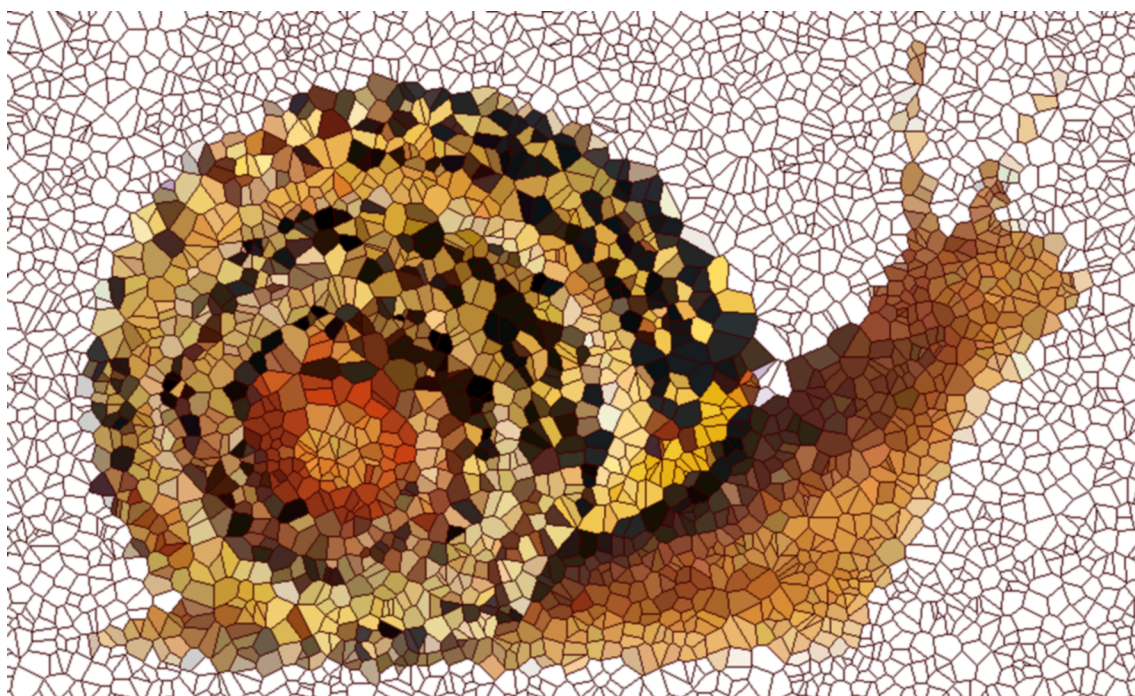
Obrázek A.13: Mozaika ze vstupního obrázku z náhodně vygenerované mřížky s hranami (nahore), z trojúhelníkové mřížky bez hran (dole)



Obrázek A.14: Mozaika ze vstupního obrázku z mřížky pomocí grup symetrií, se zobrazením hran



Obrázek A.15: Mozaika ze vstupního obrázku z náhodně vygenerované mřížky se zobrazením hran



Obrázek A.16: Mozaika ze vstupního obrázku z mřížky pomocí grup symetrií, bez zobrazení hran

