UNIVERZITA PARDUBICE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2019 **Martin Hencl**

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

zdeněk sýkora: Makrostruktury

Martin Hencl

Bakalářská práce
2019

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Martin Hencl**
Osobní číslo:	**I16085**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**zdeněk sýkora: Makrostruktury**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem práce je popsat principy, které využíval český malíř Zdeněk Sýkora ve svých makrostrukturách. Hlavním úkolem je vytvořit aplikaci, která dokáže tento typ obrazů na bázi Sýkorových pravidel vytvořit. Strukturální obrazy jsou vytvářeny opakovaným použitím několika základních elementů, které mají specifické vnitřní členění a mají čtvercový nebo obdélníkový tvar. Vzájemnou polohu elementů určuje zvolené pravidlo: navazují barvami a tvary, navazují barvami a nenavazují tvary, nenavazují barvami a navazují tvary, nenavazují barvami ani tvary. Při vlastní tvorbě Zdeněk Sýkora používal tzv. partituru s polohou výchozích elementů a znamének plus a mínus. Znaménka určují, kde se přičítá nebo odečítá koeficient, který má funkci urychlení nebo zpomalení přechodů od světlých elementů k tmavým nebo naopak. Makrostruktury vznikají rotací výřezu ze strukturálních obrazů. Je tedy nutné zvolit vhodnou reprezentaci struktur v počítači a naprogramovat volbu z množiny přípustných struktur. Aplikace musí umožňovat získání výřezu, který je poté rotován o zvolený úhel. Součástí práce bude uživatelská příručka.

Rozsah pracovní zprávy: **35**
Rozsah grafických prací: **10**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

KAPPEL, Pavel. Zdeněk Sýkora 90. Praha: Verzone, 2010. 106 s. ISBN 978-80-904546-2-0.

MAREK, Jaroslav a Marie NEDVĚDOVÁ. Historie digitálního umění: náhoda, počítač a linie Zdeňka Sýkory. In: Bečvář, Jindřich a Martina Bečvářová (eds.). 37. mezinárodní konference Poděbrady 2016. Praha: MATFYZPRESS, nakladatelství Matematicko-fyzikální fakulty Univerzity Karlovy, 2016, s. 137-146.

SÝKORA, Zdeněk a Jaroslav BLAŽEK. Computer aided Multi element Geometrical Abstract Paintings. Leonardo. 1970, roč. 3, s. 409.

Vedoucí bakalářské práce: **Mgr. Jaroslav Marek, Ph.D.**
Katedra matematiky a fyziky

Datum zadání bakalářské práce: **31. října 2018**
Termín odevzdání bakalářské práce: **13. prosince 2019**

Univerzita Pardubice Česká republika L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Lukáš Čegan, Ph.D.
pověřený vedením katedry

V Pardubicích dne 14. prosince 2018

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 9. 12. 2019

Martin Hencl

PODĚKOVÁNÍ

Rád bych poděkoval Mgr. Jaroslavu Markovi, Ph.D. za vedení při zpracování tohoto tématu, zvláště za připomínky ke zlepšení teoretické části a cenné rady v praktické části.

ANOTACE

Tato bakalářská práce se věnuje softwarovému uchopení části tvorby českého malíře Zdeňka Sýkory, konkrétně jeho Makrostrukturám. Tento malíř Makrostruktury vyvářel v poměrně krátkém období let 1972–1973, když na ně navázala tvorba Linií. Makrostruktury vznikly šikmým výřezem ze strukturálních obrazů, které měly specifické vnitřní čtvercové členění. Konstrukce Struktur vychází z definovaných pravidel pro vkládání čtvercových elementů (obsahujících většinou černé nebo bílé půlkruhy) na plátno. V první části této práce je popsána Sýkorova tvorba od počáteční krajinomalby, přes Struktury, Makrostruktury až po Linie. Ve druhé kapitole je analyzován algoritmus pro tvorbu Struktur, který má striktně matematický charakter. Diskutován je zejména vliv výchozích elementů a pravidel na výslednou strukturu. Prezentována je počítačová rekonstrukce Struktur a Makrostruktur, když výřez ze Struktur je založen na transformaci obrazu pomocí rotace. Samostatnou kapitolu tvoří popis vývoje aplikace na počítačovou reprezentaci struktur a makrostruktur, která je vytvořena v programovacím jazyku Java. Grafické výstupy jsou realizovány pomocí knihovny JavaFX. V poslední části jsou srovnány některé Sýkorovy Makrostruktury s výstupy generovanými vytvořeným programem.

KLÍČOVÁ SLOVA

Zdeněk Sýkora, digitální umění, náhodnost v umění, makrostruktury, výřez.

Zdeněk Sýkora: Macrostructures.

ANNOTATION

This bachelor thesis is devoted to software grasping a part of the work of the Czech painter Zdeněk Sýkora, specifically his Macrostructures. This painter created macrostructures in the relatively short period of 1972–1973, when they were followed by the creation of the Lines. Macrostructures were created by oblique cut-out of structural images that had a specific inner square segmentation. The structure of Structures is based on defined rules for inserting square elements (containing mostly black or white semicircles) on the canvas. In the first part of this work is described Sýkora's creation from the initial landscape painting, through Structures, Macrostructures to Lines. The second chapter analyzes the algorithm for creating Structures, which has a strictly mathematical character. The influence of starting elements and rules on the resulting structure is discussed. The computer reconstruction of Structures and Macrostructures is presented, when the cutout from Structures is based on image transformation by rotation. A separate chapter consists of a description of the application development for computer

representation of structures and macrostructures, which is created in the Java programming language. Graphical outputs are realized by JavaFX library. The last part compares some of Sýkora's Macrostructures with the outputs generated by the created program.

KEYWORDS

Zdeněk Sýkora, digital art, randomness in art, macrostructures, cutout.

OBSAH

SEZNAM OBRÁZKŮ

Seznam zkratek

JVM	Java Virtual Machine
JDK	Java Development Kit
OpenJFX	Open JavaFX
SDK	Software Development Kit
XML	Extensible Markup Language
JRE	Java Runtime Environment
PDF	Portable Document Format

ÚVOD

Digitální umění ve světě začalo vznikat v 50. letech minulého století. Zaklady vytvořil Ben F. Laposky ve svém článku Oscillons: Electronic Abtractions z roku 1953. V této práci popsal, jak mohou vznikat umělecká díla za pomoci osciloskopu [1]. Dalším světovým „otcem" počítačové grafiky je Ivan Edward Sutherland. Ten ve své doktorské práci v roce 1963 vytvořil a popsal program Sketchpad, za který získal Turingovu cenu v roce 1988. Program ukázal, jak bude vypadat moderní kreslení za pomocí počítače [2]. K dalším pionýrům umělecké tvorby založené na využití počítače patří A. Michael Noll, který se svými kolegy naprogramoval v 60. letech v jazyce FORTRAN aplikaci, která uměla převést na umělecká díla matematické funkce, nebo pseudonáhodné shluky bodů [21]. Další umělec Desmond Paul Henry měl mít dokonce článek o svých Ideografech v časopise Life, avšak kvůli vraždě amerického prezidenta Kennedyho byl článek zrušen [22]. Inspiraci pro využití matematiky a geometrie poskytl zejména François Morellet, který zpracovával svá díla jako křížení čar a geometrické obrazce ve výsledné čtvercové a obdélníkové obrazy [23]. V Československu se prvním umělcem, který využíval k výtvarné tvorbě počítač, stal docent Zdeněk Sýkora. Akademický malíř se věnoval nejdříve krajinomalbě, ale jeho tvorba nakonec směřovala ke geometrickým obrazům založených na matematickém popisu. Počítač posloužil pro generování náhodných čísel, ty se staly vstupními parametry malířových algoritmů. Malíř začal považovat počítač, náhodu a matematiku za nástroj své tvorby. Jeho tvorba byla ovlivněna úzkou spoluprací s malířem Vladislavem Mirvaldem, když inspirací pro Sýkorovy Struktury patrně poskytly Mirvaldovy kaňkáže a zmrzláže. Celý postup tvorby Struktur měl za cíl provést kombinatorické vyčerpání všech možných vzájemných pozic a vlastností elementů dle zadaných pravidel. Při tvorbě Struktur Sýkora spolupracoval s matematikem Jaroslavem Blažkem a používal první počítač na matematicko-fyzikální fakultě Univerzity Karlovy. Algoritmus tvůrci uveřejnili v roce 1970 v časopise Leonardo. V poměrně krátkém období let 1972–1973 malíř vytvářel tzv. Makrostruktury, které získával šikmým výřezem ze strukturálních obrazů. V Makrostrukturách malíře zaujaly křivky tvořené navazujícími elementy. Makrostruktury tak poskytly přechodový můstek k novému typ obrazů — k Liniím. Od roku 1973 se malíř věnoval dalšímu vývoji systémů založených na náhodnosti a přešel ze struktur k liniovým obrazům.

Cílem práce je popsat principy, které využíval český malíř Zdeněk Sýkora ve svých Makrostrukturách. Hlavním úkolem je připravit algoritmus pro softwarové uchopení Struktur a Makrostruktur, založený na bázi Sýkorových pravidel.

Pro vytvoření aplikace bude použit programovací jazyk Java. Vývoj bude probíhat v prostředí IntelliJ IDEA. Grafická část aplikace vznikne s využitím knihovny JavaFX. Grafická podoba bude umožňovat vyplnění vlastní partitury pomocí jednotlivých elementů, či načtení a použití již předem vytvořené vstupní sestavy elementů. Dále aplikace dovolí zvolit pravidla generování Struktury a grafickou podobnu jednotlivých elementů ve výsledné Struktuře. Vykreslená Struktura může být uložena jako výsledný obraz nebo může posloužit jako základ Makrostruktury. V tomto druhém případě lze Strukturu natočit, přiblížit a oříznout a po těchto transformacích ji lze uložit jako Makrostrukturu.

1 ZDENĚK SÝKORA

V první kapitole bude čtenář konkrétněji seznámen s malířem Zdeňkem Sýkorou, pokusíme se o analýzu vývoje a proměn jeho tvorby. Zaměříme se na konstrukci Struktur a Makrostruktur.

Obrázek 1: Zdeněk Sýkora. Zdroj: [3]

1.1 Život a dílo

Zdeněk Sýkora se narodil 3. února 1920 v Lounech. Než začal malovat a kreslit, věnoval se fotografování a fotokolážím. Po otevření vysokých škol po druhé světové válce v roce 1945 začal studovat na Vysoké škole architektury a pozemního stavitelství ČVUT, odkud byl obor výtvarná výchova a deskriptivní geometrie brzy přesunut na Pedagogickou fakultu Univerzity Karlovy. Na této fakultě působil od roku 1947 jako asistent a vyučující. V roce 1966 se stal docentem malby a působil zde až do roku 1980.

Nejdříve maloval realistické malby (krajinomalba), po dokončení studia změnil své zaměření na nejčastěji surrealistické a kubistické malby. Pro Sýkoru byla významná jeho první samostatná výstava v roce 1952. Poté ho oslovuje fauvismus a přechází ke geometrické abstrakci. V roce 1960 tak vstupuje do jeho tvorby geometrie, když vkládá elementy do čtvercového rastru a vytváří z nich první strukturální obrazy. Ovlivněn může být spoluprací s malířem Vladislavem Mirvaldem a jeho kaňkážemi či zmrzlážemi. Při návštěvě Leningradské (Petrohradské) Ermitáže dochází k poznatku, že přechodem na barevné skvrny se zruší hloubka a vytvoří se plocha. Výsledkem tohoto uvědomění je kolekce Zahrady, u kterých začal používat mimo štětce také špachtli. V jeho obrazech se začala objevovat více pravidelná geometrie.

Spolu s dalšími umělci (například s Vladimírem Burdou) založil tvůrčí skupinu Křižovatka, ve které se hlásili k vynálezům poslední doby a jejich využití v tvorbě [4].

Pro tuto práci podstatné jsou programované struktury realizované od roku 1964 ve spolupráci s Jaroslavem Blažkem na počítači LPG-30. Struktury měly vyčerpat všechny kombinatorické možnosti díky sadě programů. Cílem bylo využít všechny vzájemné kombinace prvků bez toho, aby se jednalo o op-art, tedy optical art, geometrické umění optické iluze snažící se o iluzi pohybu a nestability [5].

V roce 1965 měl Zdeněk Sýkora habilitační přednášku na téma Některé otázky vztahu barvy a prostoru v současném výtvarném umění a byla mu udělena akademická hodnost docent na katedře výtvarné výchovy Pedagogické fakulty Univerzity Karlovy v Praze. Zde vyučoval až do roku 1980. Ve stejném roce měl také první výstavu Konstruktivní tendence spolu s dalšími světovými umělci [6].

Jeho tvorba nachází uplatnění v architektuře. Mezi roky 1967 až 1969 pracoval na větracích komínech Letenského tunelu poblíž Národního technického muzea, které jsou od roku 2003 kulturní památkou. Zároveň s tím vytvořil u dnešní kavárny v Jindřišské ulici keramickou stěnu, dříve se jednalo o Polské informační středisko. Dalším zástupcem této tvorby byla opona ve Fučíkově divadle v Lounech, kterou Sýkora vytvořil spolu s Vladislavem Mirvaldem v roce 1963. Tato se naneštěstí ztratila při rekonstrukci divadla v roce 2001. Ve světě zanechal svoji stopu například chodníkem v nizozemském městě Gorinchem, nebo ve Švýcarsku v Basilejské vstupní hale firmy Selmoni AG.

Milníkem jeho uměleckého života je okamžik, když Národní galerie poprvé zakoupila Sýkorův obraz: Strukturu. Jeho obrazy se dostávají postupně do dalších světových galerií a soukromých sbírek, například Nationalgalerie Berlin, nebo McCrory Collection New York.

Svoji poslední výstavu v Československu až do roku 1987 uskutečnil v roce 1970 ve Špálově galerii v Praze. V období mezi lety 1972 a 1973 končí postupně Sýkorovo období makrostruktur a nahradí je liniové obrazy. Při konstrukci linií pracuje s náhodně generovanými čísly a striktně matematickým algoritmem transformovanými do hladce navazujících oblouků kružnic. Linie představuji lidský osud, mají svůj počátek umístěný v rastru obrazu, svoji délku života, tloušťku, barvu. Patologickým případem linie je úsečka nebo bod, pokud jejich délku generátor náhodných čísel stanovil na jedna nebo nula. První výstava Linií je uskutečněna v roce 1979 v nizozemském Gorinchem.

Pro Zdeňka Sýkoru byl velmi důležitý rok 1983, kdy se oženil se svojí bývalou žákyní Lenkou, která byla o 37 let mladší.

Jeho přítel Vít Čapek moderoval od roku 1982 do 1986 rozhovory o jeho práci, které byly vydané v díle Retrospektiva 1947-95 Galerií hlavního města Prahy v roce 1995. V roce 1988 v Domě kultury v Brně uspořádal velkou retrospektivní výstavu 1947-1988. Česká televize o Zdeňku Sýkorovi natočila dokument z řady Evropané [18].

V devadesátých letech vystavoval Sýkora po celé Evropě i doma, v tvorbě se věnoval rozvíjení linií a vrátil se i k tvorbě z 50. let. Po roce 2000 obdržel od francouzského ministra Řád umění a literatury. V roce 2005 také zhotovil největší dílo tvořené liniemi v interiéru budovy Řízení letového provozu v Jenči u Prahy s názvem Létání [8].

K jeho životnímu jubileu proběhl cyklus výstav Zdeněk Sýkora 90 v Praze, v Olomouci a v Karlových Varech. O rok později v 91 letech zemřel doma v Lounech.

Jeho nejdražším obrazem, který se kdy prodal na aukci je dvojobraz makrostruktur Dva trojúhelníky (1973), který se v roce 2008 prodal za 4,484.000 Kč v aukční síni 1. Art Consulting. Nejdražší liniový obraz jsou Linie č. 206 (2002) za 3,660.000 Kč v aukční síni Prague Auctions v roce 2010 a nejdražším obrazem, který není ze sérií Linie ani Struktur je Zahrada (1960), který se vydražil v 1. Art Consulting za 1,003.000 Kč [16].

1.2 Ukázky tvorby

Předchozí kapitola byla věnována životopisu a krátkému popisu malířovy tvorby. V této podkapitole se pokusíme představit čtenáři reprezentanty jeho tvorby vizuálně.

Sýkorova první malířská tvorba byla realistická krajinomalba, nejčastěji zátiší u řeky Ohře a okolí. Ke konci padesátých let prochází jeho tvorba proměnou, od realistické k impresionistické.

Obrázek 2: Ohře v Loužku, 1952. Zdroj: [7]

V šedesátých letech se v Sýkorových obrazech začíná objevovat čím dál víc geometrických obrazců, nejdříve jen jako výsledek barevných přechodů u impresionistické tvorby, později se jedná o záměr. Jako obraz, kde Sýkora již otevřeně umisťuje geometrii je poslední ze série „Zahrad“, „*Zahrada (se čtvercem)“*. Přechod na geometrii provedl Sýkora také proto, aby vyloučil subjektivní zkreslení.

Obrázek 3: Zahrada (se čtvercem), 1960. Zdroj: [7]

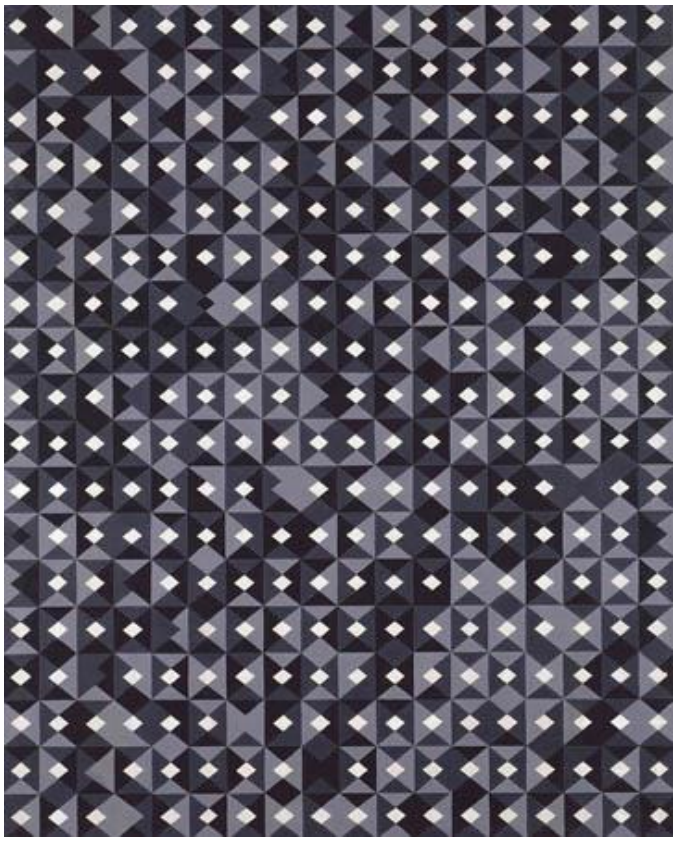
1.3 Struktury

Úplným přechodem ke geometrii je obraz *„Šipka“*. Ten vznikl v reakci na Sýkorovu návštěvu Ermitáže (vizte Životopis výše). Čtvercový obraz vznikl tak, že si umělec narýsoval po obvodu řídící značky (kótami) a ty pospojoval přímkami. Vzniklé plochy byly vybarveny podle cítění a vnímání barev. Původní návrh je na papíru velikosti A4, ale díky souřadnicovému systému nebylo problémem dílo jakkoli zvětšit a výsledný čtvercový obraz tak má rozměr 120 cm.

Obrázek 4: Červená šipka, 1962. Zdroj: [7]

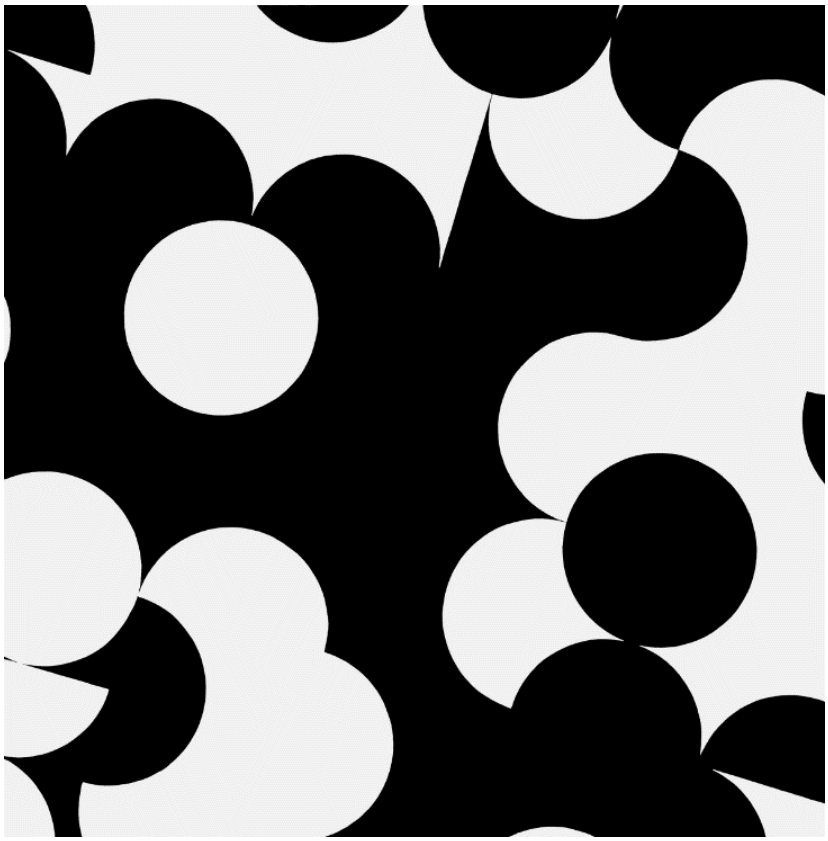
Tento obraz má sice jen pět barevných ploch, avšak řídící linie jsou určeny více než dvaceti body. Další dílo, Šedá struktura, má 324 elementů. V dnešní době by nás napadlo použít počítač především pro usnadnění práce (jednoduché repetitivní činnosti), docent Sýkora však měl počítač jen jako pomocníka. O způsobu práce s náhodnými čísly vizte kapitolu Generování náhodných čísel.

Jak již bylo zmíněno, obraz Šedá struktura je tvořen 18 × 18 = 324 obdélníky. V každém takovém obdélníku jsou dva rovnoramenné protkané trojúhelníky (průnikem je kosočtverec). Vznikne tak pět ploch a každá z nich může nabývat jednou ze 4 barev – černá, bílá nebo 2 odstíny šedé (světle šedá a tmavě šedá). Pro jedno políčko tedy vychází 1024 variant obarvení. Pravidla obarvení jsou taková, že sousední dva prvky (v rámci jednoho elementu) nesmí mít stejnou barvu. Stejný odstín vedle sebe se může vyskytnout jedině pokud bude ve dvou prvcích dvou elementů.



Obrázek 5: Šedá struktura, 1962-63. Zdroj: [9]

Jedním z posledních děl, která věnoval Sýkora strukturám, byl obraz „*Makrostruktura“*, jedná se o zvětšené elementy původních velkých celků. Těmito obrazy opustil svět přímých linek a přešel na křivky. Makrostruktury vznikly tak, že se Sýkora vrátil k původním obrazům, strukturám, vzal bílý rámeček a hledal v nich nové obrazy. U zvětšeniny zmizí striktní charakteristika a vznikne viditelná linie [10].

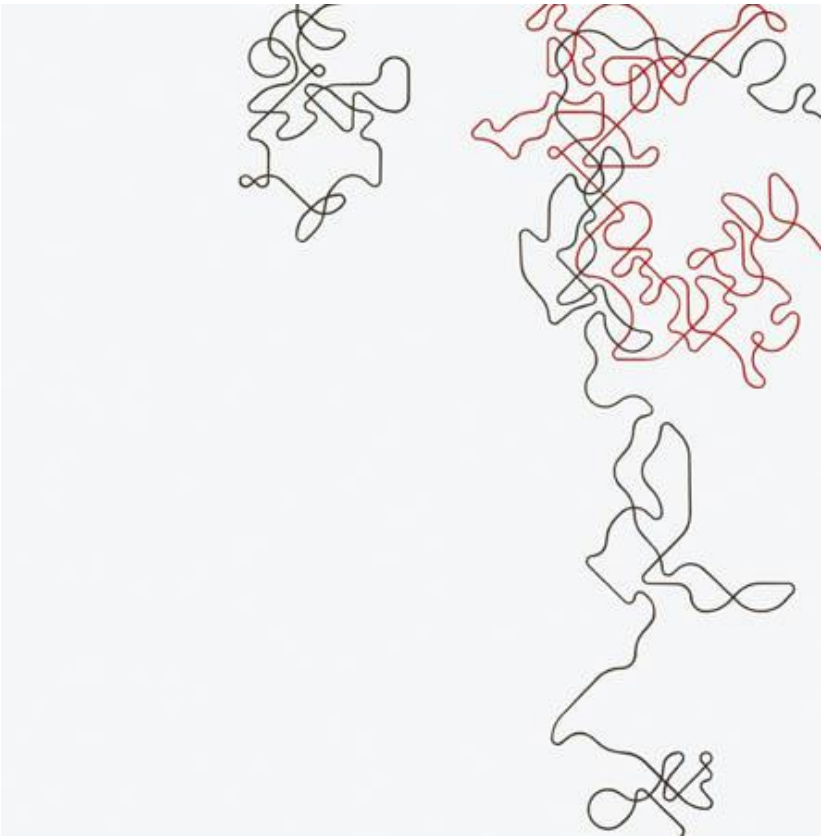


Obrázek 6: Makrostruktura 1972. Zdroj: [10]

1.4 Linie a křivky

Ke křivkám se Sýkora dostal díky makrostrukturám, konkrétně ho zaujal přechod mezi bílou a černou výplní, ostrý přechod křivek. Na rozdíl od přísného řádu ve strukturách jsou linie více uvolněné, nejdříve sice ještě trochu svázané rastrem, ten ovšem postupně úplně opustil. Na plátně jsou linie nespoutané, mohou utíkat ven z plátna a opět se vracet. I ty ovšem potřebují nějaká pravidla pro vytváření a ty se nazývají partitury.

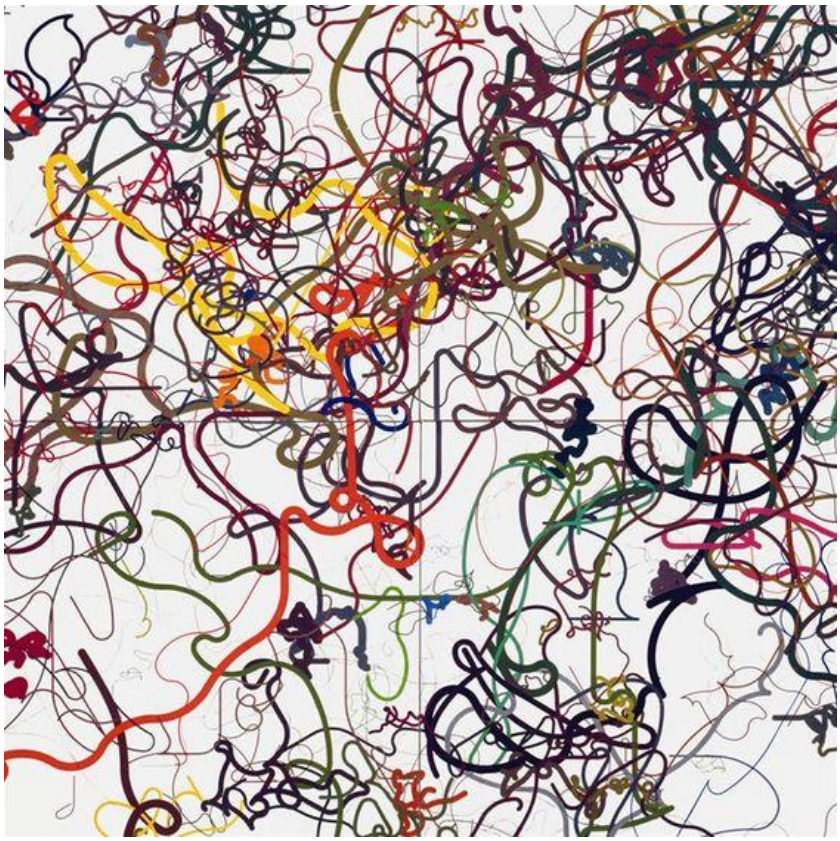
První obraz s liniemi se jmenuje První linie z roku 1973. Je tvořený třemi stejně tlustými křivkami s různou barvou, jejichž směry se nemění příliš intenzivně.



Obrázek 7: První linie, 1973. Zdroj: [9]

Postupně v dalším desetiletí začal Sýkora experimentovat s počtem linií, různou tloušťkou čar, pozicí bodu zrodu i jejich počtem. Například již v roce 1975 namaloval obraz „*Linie č. 3 – 81 linií“*, v roce 1976 „*Linie č. 4 – Jedna linie“* i „*Linie č. 5 – 200 modrých linií“*. Autor si všímá, že volba množiny možných směrů značně ovlivňuje výsledný liniový obraz. U některých obrazů autor vykresluje linie se shodným počátkem.

V osmdesátých letech bylo jeho stěžejním dílem „*Linie č. 24“* nazvaná „*Poslední soud“*. Na něm bylo nakreslených 227 linií a Sýkora toto 3 × 3 metry^2 veliké dílo maloval více než rok.



Obrázek 8: Linie č. 24 – Poslední soud, 1983-1984. Zdroj: [7]

V devadesátých letech pokračuje s experimenty u liniových obrazů společně s manželkou Lenkou, pracuje na „životnosti“ čar (krátké linie s minimem změn ve viditelném spektru), také na obrazech s horizontálními liniemi, nebo liniemi, které vytváří celé velké plochy.

V novém miléniu oba malovali hlavně odlehčená díla, využívali zkušeností z předchozí tvorby, společně rozvíjeli myšlenky z uplynulých desetiletí. Za vrchol tvorby tohoto období je považován obraz *„Linie 220“*.

Obrázek 9: Linie č. 220, 2003. Zdroj: [7]

2 ALGORITMY

V této kapitole se nejprve podrobněji zaměříme na způsoby, kterými Sýkora získával data pro své systémy umožňující generovat obrazy. Základní pilíře Sýkorových obrazů tvořil autorův nápad společně se stanovenými pravidly. Podle autora je nejdůležitější to, jak na diváka působí výsledný obraz, než to, jak se k němu došlo, jaký byl princip jeho vzniku [10]. Navržené generativní systémy pro tvorbu výtvarných děl vyžadovaly získání náhodných vstupních čísel. Proto bude v této kapitole popisům systémů pro generování Struktur, Makrostruktur a Linií předcházet podkapitola věnovaná náhodně generovaným číslům.

2.1 Generování náhodných čísel

Zdeněk Sýkora potřeboval pro své systémy generující obrazy náhodná čísla. V úplně prvních experimentech používal hody hracími kostkami, nebo tahal žetony s čísly z pytlíčku. Zkoušel také náhodná telefonní čísla ze seznamu. Později začal potřebovat delší série náhodných čísel. Například zmíněný *Poslední soud* obsahuje 227 linií. Pro zadání jedné linie je nutné vygenerovat 1 + 2 + 1 + 1 + 2L parametrů (délka života L, souřadnice počátku linie, barva, tloušťka, délka tečny, směr). Střední hodnota délky života v tomto obraze určitě přesahovala 20, celkem tedy potřeboval nejméně 227 x (1 + 2 + 1 + 1 + 2 x 20) = 9085 náhodných čísel. Získávání náhodných čísel předchozími metodami přestávalo být možné. Podle Sýkory generování nebylo úplně náhodné, když vliv měla únava, či lenost [10]. Díky spolupráci s matematikem Jaroslavem Blažkem začal využívat počítač LPG-30. Blažek mu dodával kartičky s požadovanými intervaly (nejdříve 1–2, 1–3, 1–7, 1–12, 1–24, později 1–30, 0–360, 0–150, 0–200). Jindřich Ploch a Pavel Raiman mu v 80. letech dodali počítačový program pro generování pseudonáhodných čísel.

[11]

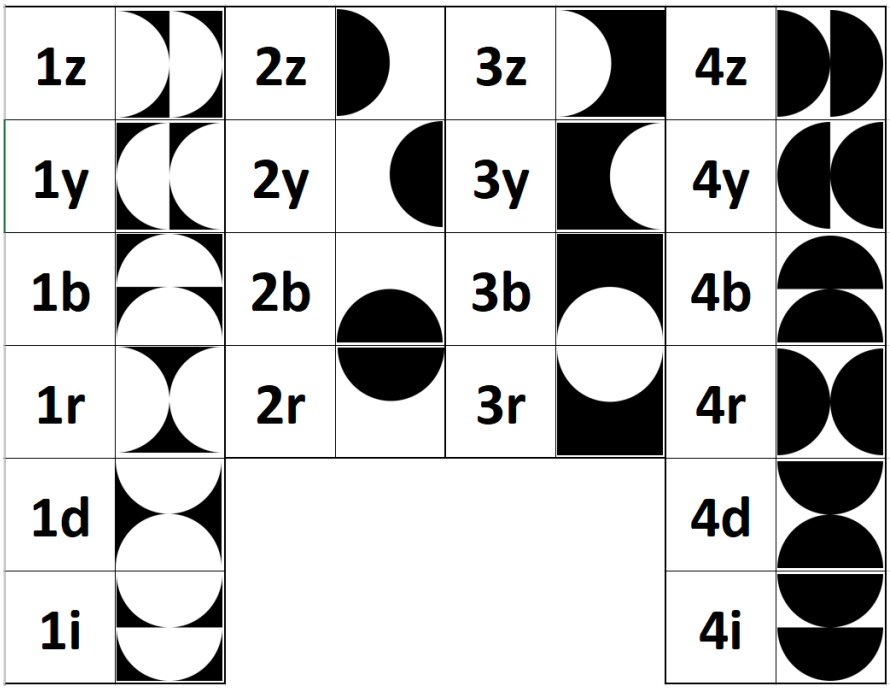
2.2 Algoritmus pro struktury

Algoritmus pro Černobílou strukturu popsal Zdeněk Sýkora společně s Jaroslavem Blažkem v článku [11].

V tomto obraze jsou elementy čtvercové, když obsahují půlkruhy ve třech variantách (jeden půlkruh, dva půlkruhy za sebou, dva půlkruhy proti sobě). Tyto elementy potom dál otáčel o 90° a získal tím kombinatoricky výčet deseti možných tvarů, v jakých se mohou elementy na ploše vyskytovat. Po přidání černé a bílé barvy výplně (negativy) se počet elementů zdvojnásobil na dvacet. Těchto dvacet elementů rozdělil do čtyř skupin podle světlosti (první je nejsvětlejší

a čtvrtá nejtmavší) na obrázku 10 uspořádaných do sloupců. V každé skupině světlosti přiřadil jednotlivým elementům písmenné označení *z, i, y, b, r, d.*

[11]



Obrázek 10: Elementy. Zdroj: Vlastní

Pro strukturu nejsou důležité pouze tvary, ale také pravidla, podle kterých bude povoleno těmto elementům se skládat do výsledného obrazu. Pravidla jsou čtyři a jako pomocné znaky znaménka plus a minus.

„Říkáme, že barvy pokračují, pokud je barva podél strany prvku stejná jako barva podél sousedního okraje sousedního prvku. Říkáme, že tvary pokračují na straně prvku, pokud se každý půlkruh otevřený na hraně elementu spojí s půlkruhem hraničního prvku a vytvoří úplný kruh nebo pokud se spojí dva vzory, z nichž žádný není půlkruh otevřený do strany.“

Pravidla jsou:

- V = 0 – pokračuje barva; pokračuje tvar,
- V = 1 – pokračuje barva; nepokračuje tvar,
- V = 2 – nepokračuje barva; pokračuje tvar,
- V = 3 – nepokračuje barva, nepokračuje tvar.

Barva má přednost před tvarem. Pro vysvětlení výše uvedené citace a pravidel „V“ je podstatné zmínit také to, že se berou v potaz všechny strany, lépe řečeno hrany, kolem vypočítávaného elementu.

Znaménka plus a minus slouží k popsání míst, kde se buď přičítá, nebo odečítá předem zvolený koeficient (C = x) k náhodnému číslu elementu. Tohle je poslední část, kterou určuje autor.

[10]

Shrnutí toho, co si připravil autor:

- Strukturu (s počtem řádků a sloupců),
- zvolené pravidlo (V = 1–4),
- koeficient přechodu C (4 > C > 0),
- umístění počátečních elementů do libovolných buněk,
- umístění znamének + a - do libovolných buněk.

Další kroky jsou určeny počítačem. Vysvětlení bude uvedeno na příkladu tak, jak ho popsal Sýkora ve svém textu *Můj systém* [10].

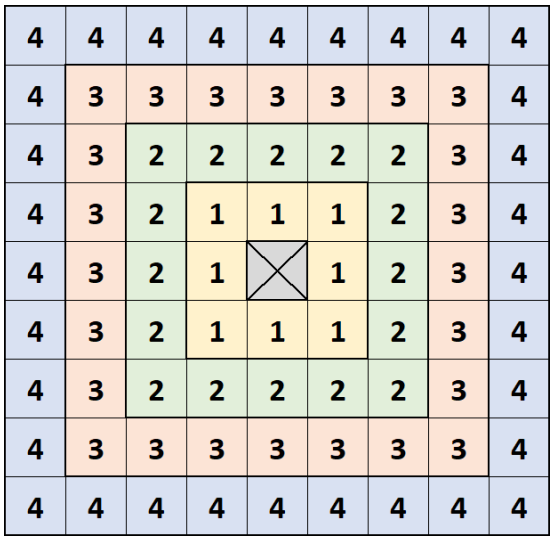
Počítač postupuje podle algoritmu. Začíná v elementu [1;1] v levém horním rohu a postupuje zleva doprava v lichých řádcích a zprava doleva v sudých. Pro každý element (předem nevyplněný) zkontroluje všechny okolní elementy. Tato kontrola probíhá tak, že se spočítá průměr těchto okolních prvků (hodnota je dána příslušností ke skupině 1–4, vizte obrázek 10: Elementy) podle předvyplněných elementů a těch, které byly spočteny v předchozích iteracích. V dalším kroku se k vypočítanému číslu přičte nebo odečte koeficient, (pouze pokud je v elementu + nebo -). Pokud výsledný koeficient okolí není jednoznačný (například 2,5) začne počítač prohledávat širší okolí elementu, nejdále však o čtyři elementy dál. Názornější popis v podobě vývojového diagramu je v následující kapitole algoritmus jak pro struktury, tak i pro makrostruktury.

1z	2z	3z	4b	4d
2r	2b	3y	4d	3z
2b	2r	+		-

Obrázek 11: Výpočet elementu. Zdroj: Vlastní

Na obrázku 11 si uvedeme příklad celého procesu výpočtu elementu. Zvolili jsme pravidlo „pokračuje barva, nepokračuje tvar“ V = 1. Koeficient pro úpravu je 0,75. Podtržené elementy jsou předem zvolené, také znaménka jsou předem dána. Nyní chceme vědě, co dosadí algoritmus na pozici [3;3], předchozí prvky jsou vypočítány v dřívějších iteracích.

První část je výpočet koeficientu určujícího to, ze které skupiny se potom element bude vybírat. Sečtou se prvky 2b, 3y, 4d a 2r kolem. Tedy 2 + 3 + 4 + 2 = 11. Počet prvků, které se zapojily je 4, tedy 11/4 = 2,75. Nyní se přičte předem dané číslo 0,75, protože v hledaném políčku je +, tedy 2,75 + 0,75 = 3,50. Desetinná část je 0,5 a algoritmus musí rozšířit okruh prohledávání o 1. Elementy budou tedy: 1z, 2z, 3z, 4b, 4d, 2r, 2b, 3y, 4d, 3z, 2b a 2r, tedy $(1 + 2 + 3 + 4 + 4 + 2 + 2 + 3 + 4 + 3 + 2 + 2) / 12 = 2{,}66;\ 2{,}66 + 0{,}75 = 3{,}41.$ Tuto hodnotu už můžeme zaokrouhlit přímo na 3 a vybírat element z 3. skupiny. Podle pravidla V1 „navazuje barva, ale ne tvar“. Důležité je, že se koukáme jak na prvek 2r, tak i 3y. Pokud bychom chtěli dodržet plně pravidlo V1, museli bychom mít prvek, který má na své levé hraně bílý půlkruh (pro 2r) a černý půlkruh seshora (pro 3y). Ten ovšem neexistuje, proto se spokojíme s dodržením pravidla, že bude pokračovat jen barva, tedy zleva bílá a seshora černá. Tomu odpovídá prvek 3z.



Obrázek 12: Oblast dodatečného prohledávání. Zdroj: Vlastní

2.3 Algoritmus pro makrostruktury

Docent Sýkora se k makrostrukturám dostal tak, že se vrátil k původním strukturám s bílým rámečkem a začal v nich hledat nové obrazy.

Algoritmus struktur je tedy rozšířen tak, že výsledný strukturální obraz je tvůrcem otočen v libovolném úhlu a může být i zvětšen. Samotné otočení v programu může proběhnout buď za použití rodičovské procedury, nebo pomocí maticové operace pro rotaci bodu:

$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$, kde x' a y' jsou výsledné souřadnice otočeného bodu, α úhel otočení a x a y původní souřadnice bodu.

Změna měřítka může probíhat stejně jako při otáčení za pomocí poděděné procedury, nebo maticově:

$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} k_x & 0 \\ 0 & k_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$, kde k je koeficient měřítka (k_x a k_y se mohou lišit, potom dojde ke zmenšení bez dodržení měřítka).

2.4 Linie inspirované Makrostrukturami

O Strukturách Zdeněk Sýkora uváděl, že mu připomínají odlesky na vodní hladině. V Makrostrukturách začal Sýkora spatřovat linie tvořené hranicemi ploch vznikajících spojením ploch půlkruhů. Volnost hraničních křivek ho přitahovala. Chtěl najít logiku v jejich realizacích, stejně jako ji měla realizace struktur. To se mu podařilo. Čtvercový pravoúhlý rastr nejprve doplnil o úhlopříčky a zvolil ho jako nové východisko. Později volil i více jiných směrů.

„Linie jsme definovali „školním způsobem" jako stopu po bodu. Ten se mohl v základním rastru pohybovat tak, že vertikály, horizontály a diagonály sloužily jako tečny budoucích linií. Tím bylo dáno, že bod se může pohybovat 8 různými směry, označenými číslicemi 0–7. Sled čísel 0–7 jsme získávali náhodným způsobem, a sice pomocí hodu kostkou a binárního principu: lichá = 1, sudá = 0.

000 = 0; 001 = 1; 010 = 2; 011 = 3; 100 = 4; 101 = 5; 110 = 6; 111 = 7;

Kromě pokračování anebo vracení se po téže linii, byly ještě 4 možné změny směru. Linie vznikaly tak, že každé změně (každému úhlu) odpovídal určitý oblouk různé délky a poloměru. Vše bylo náhodné, počet tečen, výchozí bod a jeho souřadnice x/y. Měl jsem zase jednou v historii možnost a právo vystavit čisté bílé plátno, když by souřadnice a průběh křivky vyšly mimo daný formát.

Charakter těchto linií nemohl zakrýt rastrový základ. Dále mi vadila nestejná délka tečen, jelikož diagonály byly delší. Po delším trápení jsem našel jednoduché řešení. Jako na ciferníku hodin zvolil jsem 12 směrů a délka tečny byla pro všechny směry stejná. V koncovém bodě jedné tečny pokračoval další krok v novém směru. Rastrový charakter zcela zmizel a křivky získaly přirozenost a spontánnost, odpovídající jejich náhodnému charakteru."

[12]

Pro tvoření linií bylo také důležité, jak se linie vzájemně kříží, záleželo na tom, která křivka povede přes kterou a v jakém okamžiku, nebo jakou mají linie „délku života" a unikátní barvy.

Před každým liniovým obrazem si Sýkora vytvořil nejdříve tzv. partituru („skóre“) obrazu. Ta sloužila k zápisu všech informací o obraze, souřadnice počátečního bodu, šířku, barvu, délku a směry linie.

Generování linií je založeno na přesném matematickém algoritmu. Linie na plátno vykresloval pomocí kružítka a později vybarvoval. Při jejich tvorbě dále spolupracoval s doktorem Blažkem a využíval počítačově generovaná náhodná čísla.

Konstrukci oblouků Zdeněk Sýkora nikde nepublikoval. V jeho knize Grafiky se objevili ukázek tzv. partitur, tedy tabulek náhodných čísel generujících linie. Na základě nich se podařilo v článcích [12], [17], [19] jeho algoritmus zrekonstruovat.

3 PRAKTICKÁ ČÁST

Praktická část se zabývá tvorbou programu pro reprodukci Sýkorových struktur. Ačkoliv je cílem celé práce vytvářet makrostruktury, tak struktury jsou jejich základem, a proto je jim věnována hlavní část programu.

Vývoj probíhá ve vyšším programovacím jazyce Java. Tento jazyk byl vybrán z několika důvodů:

- Probíhala v něm většina výuky programování během studia na fakultě,
- v programu je snadné a zároveň povinné dodržovat typovou kontrolu,
- Java je nezávislá na architektuře zařízení, výsledný bajtkód je interpretován pomocí virtuálního stroje JVM,
- aby nedošlo k nedorozumění o originalitě této práce, nebo záměně s diplomovou prací ing. Jiřího Hory na téma Struktury (jeho program je psaný v jazyce C#).

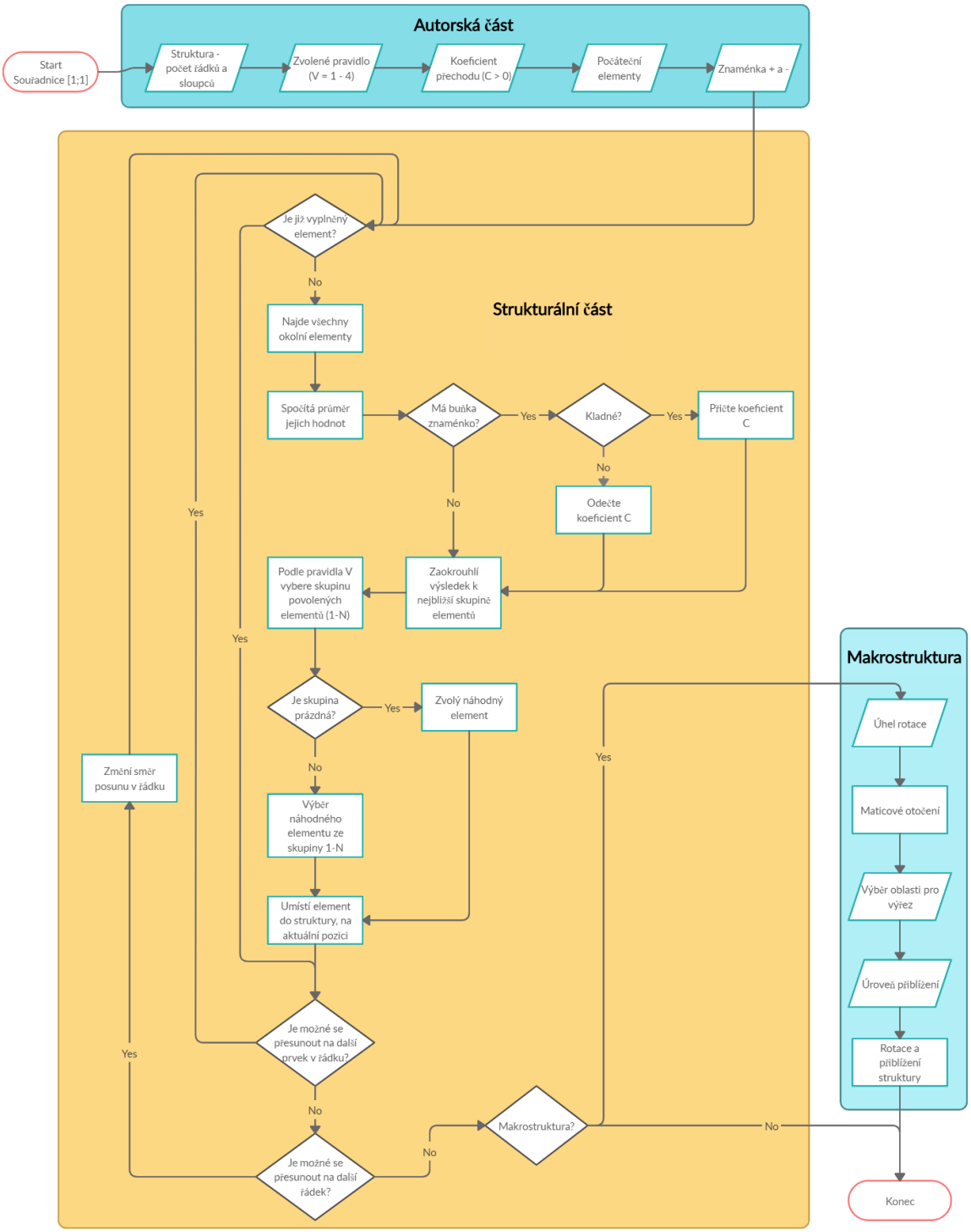
Pro zobrazení je využita grafická nadstavba JavaFX. Ta byla, mino jiné, vybrána opět kvůli tomu, že byla vyučována v předmětu Počítačová grafika. Další předností je jednoduchý návrh a práce s rozhraním výsledné aplikace. Za nevýhodu by se dalo považovat, že JavaFX již není oficiálně vyvíjena společnosti Oracle (není součástí oficiálního JDK), byla přesunuta pod OpenJDK a další údržby se ujala pouze komunita jako OpenJFX [13].

Vlivem toho, že JavaFX již není součástí oficiálního balíčku, potřebuje projekt zvláštní parametry nastavení. Jako SDK projektu je nastavený adresář JDK (v případě této aplikace Java 11). Samotná JavaFX je definována v části externí knihovny, v nastavení projektu v záložce Libraries je přidána absolutní cesta ke složce javaFx/lib/. Dále je nutné přidat tuto knihovnu do Modulů, aby balíček věděl, kde a jak ji hledat a tomuto modulu nastavit příznak Kompilovat. Toto nastavení se v prostředí IntelliJ Idea provede: File > Project Structure > Modules > Dependencies > + a zde vložit JavaFX knihovnu. Závislosti na dalších knihovnách a podmínky spuštění aplikace jsou popsány v části uživatelská příručka.

3.1 Aplikace

Všechny dále popisované zdrojové soubory jsou přiložené k bakalářské práci a dají se využít k sestavení aplikace v jiném vývojovém prostředí, nebo i jen z příkazové řádky přímo pomocí Java compiler.

Aplikace je tvořená stromovou strukturou, kořenem je systémem daný *src,* každá složka je v Javě nazvána jako package, neboli balík. Každá třída (class) musí být v balíku, nelze ji mít přímo v kořenovém. Rozdělení na balíčky je voleno hlavně kvůli přehlednosti a snadnějšímu hledání závislostí.



Obrázek 13: Algoritmus vytvoření struktury a makrostruktury. Zdroj: [19]

Umístění hlavní třídy, která spouští celý program zjistíme ze souboru *MANIFEST.MF* ve složce *MANIFEST*, v našem případě se jedná o třídu *MainApp.java* ve složce *windows*. To je hlavní spouštěcí třída celého programu a slouží k otevření hlavního okna.

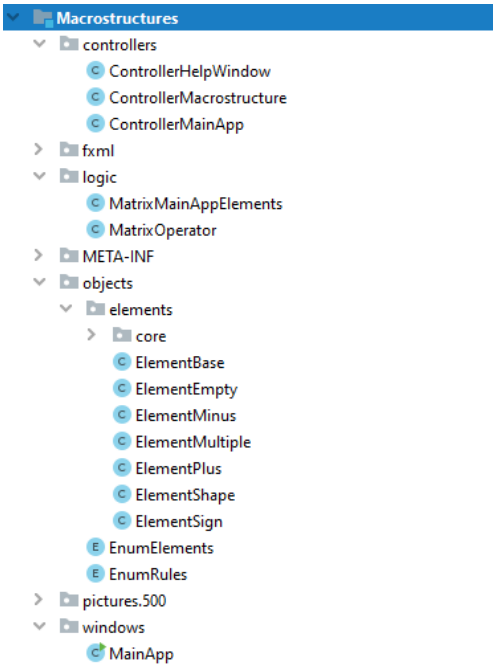
Ve složce *controllers* jsou ovládací třídy pro jednotlivá okna aplikace starající se o manipulaci s grafickými prvky na oknech, získávání a zapisování informací z prvků, které vidí a ovládá uživatel a předávají je dalším třídám.

Ve složce *fxml* jsou soubory využívané JavaFX ke vzhledu oken, jedná se o upravené XML soubory, které byly vytvořeny za pomocí programu Gluon.

Ve složce *objects* jsou třídy elementů a jejich vlastností. Tento balík má více úrovní, kdy v základní úrovni jsou výčtové typy pro názvy elementů a pravidel. Ve složce *elements* jsou elementy, které jsou buď předkem dalších, nebo mají ze základních nějakou nadřazenou vlasnost. Ve složce *core* jsou již jen základní elementy.

Další hlavní složkou je balík *logic*, ve kterém je řešena práce s maticí struktury, ať už realizuje maticové výpočty na nejnižší úrovni, nebo aplikace těchto výpočtů na vzhled.

Balík pictures obsahuje obrázky ve vysokém rozlišení všech elementů, včetně prázdného, nebo znamének. Tento balík zůstal ve zdrojových balíčcích z důvodů hlubokého zanoření do programové struktury, a protože bez nich by program ztratil veškerou funkčnost. Dají se nahradit jinými elementy, jak bude zmíněno dále v uživatelské příručce.



Obrázek 14: Struktura programu. Zdroj: Vlastní

3.2 Balík controllers

Toto je část programu, která se stará o manipulaci s daty na oknech a realizaci fxml souborů do programové podoby. Třídy zde obsažené až na výjimky data nijak nezpracovávají a jen vytváří

a volají ostatní třídy, které data vlastním specifikovaným způsobem zpracují. Třída *ControllerHelp* je primitivní a slouží pouze k otevření uživatelského manuálu, nebo této bakalářské práce z prostředí aplikace.

3.2.1 Třída ControllerMainApp

Toto je v podstatě nejdůležitější třída celého programu, bez ní by se vůbec nic nezobrazilo a ostatní třídy by spolu neuměly dobře komunikovat. Na začátku třídy jsou pro grafické prvky navržené v MainApp.fxml s identifikátorem přidány jejich objektové typy, v podstatě se jedná o propojení vizuální části s programem. Dále je v několika metodách (začínajících většinou *load*Něco) definované základní chování některých těchto prvků, pod pojmem chování můžeme brát jak nastavení grafického výstupu, tak i propojení některých prvků mezi sebou (například textové pole zobrazující hodnotu je spárováno s posuvníkem této hodnoty). Také jsou zde definovány obsahy rozbalovacích nabídek podle výčtových objektů a přepínacím tlačítkům (*radio buttons*) jsou připojeny jejich obrázky, aby se uživatel nemusel orientovat pouze v textové reprezentaci.

V další části třídy jsou definovány vlastnosti při vyvolání události (event) některého prvku. Úplně korektní by byl postup, že každá metoda zachytávající událost pouze vyvolá obslužnou metodu, v případě celého programu je však většinou obslužný kód již v této metodě. Za zmínku stojí metody *b_resize_onMouseClicked*, *b_generate_onMouseClicked* a *mi_load*.

První metoda je vyvolána kliknutím na tlačítko pro změnu velikosti struktury. Přečte hodnoty velikostí plátna a společně s odkazy na objekty popisků a skupiny přepínačů elementů předá tyto informace třídě MatrixMainAppElements (popsané dál), která z nich vytvoří matici elementů a přidá je na hlavní okno.

Druhá metoda (zkráceně) *generate* nejdříve nechá naklonovat do hloubky vyplněnou strukturu (pouhé přiřazení novému objektu předá referenci na starý objekt a dále by se měnila stará data) a tento nový objekt nechá spočítat opět hlavní třídu pro práci s maticemi. Výsledek pošle do nově vytvořeného okna (s danými vlastnostmi jak podle vizuálního souboru, tak kódem) a čeká, dokud není nové okno zavřeno.

Metoda *mi_load* je vyvolána z nabídky okna a dokáže načíst uloženou strukturu z textového souboru. Nechá hlavní maticovou třídu načíst informace a v případě úspěchu vyvolá vykreslení na okno. K ukládání těchto struktur slouží druhá metoda *mi_save*, která dělá proces opačný, obě funkce budou zmíněny v příslušné třídě.

3.2.2 Třída ControllerMacrostructure

Tato třída se stará o obsluhu okna, které je vyvoláno po vypočítání struktury. Slouží k vykreslení výsledné struktury, uložení obrazu a případné další manipulaci se strukturou a jejím přetransformováním na makrostrukturu.

Pro samotné vykreslení vygenerované struktury jako obrazu slouží metoda *drawMatrixAsImageView*, vyvolaná ještě před samotným otevřením tohoto okna. V této metodě se nejdříve obnoví všem elementům jejich grafická podoba, protože z neznámého (autor práce nenašel vysvětlení) důvodu jsou obrázky některých elementů po výpočtu struktury zdeformované, ačkoli se na tuto vlastnost jednotlivých elementů nesahalo. Dále se využívá metody společné všem grafickým prvkům z JavaFX *snapshot*. Tato funkce vezme objekt (node) a doslova ho vyfotí ve stávajícím stavu, nemusí být ani viditelný a umístěný v okně. Této funkce se využívá v programu ještě několikrát. Výsledný obraz je vykreslen na aplikaci a povolí se uživateli prvky pro další úpravy, nebo uložení obrazu ve stávajícím stavu. Metoda snapshot má však jednu nepříjemnou chybu, pro jejíž vysvětlení je nutné vědět, že JavaFX pracuje nejen s procesorem, ale k vykreslování a zpracování obrazu využívá grafickou kartu, resp. grafický operační procesor. Pokud se však obraz do grafické paměti karty (může se jednat i o integrovanou) nevejde, tak celá funkce „spadne“ a vyvolá se výjimka o nedeklarovaném obraze [20]. Tato chyba se dá vyřešit jedině navýšením grafické operační paměti, nebo zmenšením vykreslovaného plátna. Ukázka části kódu této metody:

```
@FXML
void b_generate_onMouseClicked(MouseEvent event) {
try {
        MatrixMainAppElements matrixCalculated =
        (MatrixMainAppElements)matrixMain.clone();
        matrixCalculated.calculateElementsByRuleAndCoeficient(
        cb_rules.getSelectionModel().getSelectedItem(), sl_coeficient.getValue());
(…)
        ControllerMacrostructure controllerMacrustructure =
        fxmlLoaderMacrostructure.getController();
        controllerMacrustructure.setStageMainAppStage(MainApp.getMainStage());
        controllerMacrustructure.setCalculatedMatrix(matrixCalculated);
        stageMacrostructure.showAndWait();
} catch(Exception e) {
    showAllert(WARNING,"Generování","Nevygenerováno!“);
}
```

Po vykreslení obrazu struktury na okně jsou odblokovány ovládací prvky pro další úpravy. Posuvník rotace slouží k natočení struktury o požadovaný úhel a tlačítko *Ukotvit* vyvolává událost *b_dock_onAction*, která podle úhlu rotace aktuálního obrazu struktury udělá nový obraz (opět pomocí funkce *snapshot*), který obsahuje již rotovaný obraz. Tento krok byl zvolen kvůli zjednodušení odečítání souřadnic pro oříznutí a také proto, že komponenta *Slider Pane*, na

kterém je obraz umístěn, vracela při stejném sestavení pokaždé různé souřadnice obsaženého obrazu. Když je rotovaný obraz vykreslený, odemkne se možnost vybrat oblast k vyříznutí. Ukázka části kódu této metody:

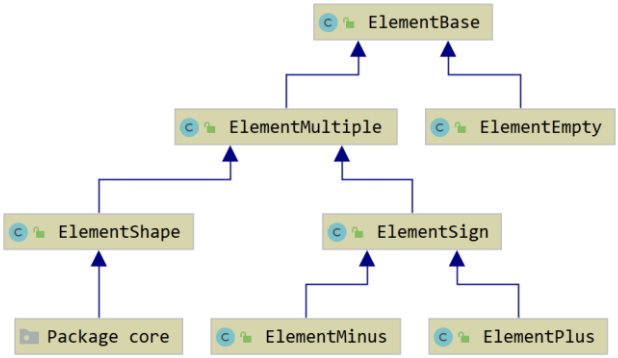
```
@FXML
void b_dock_onAction(ActionEvent event) {
      (…)
      SnapshotParameters params = new SnapshotParameters();
      params.setFill(Color.TRANSPARENT);
      gridPaneRotated.setRotate(sl_rotation.getValue());
      int rotatedWidth = getRotatedWidth();
      int rotatedHeight = getRotatedHeight();
      imageRotated = new WritableImage(rotatedHeight, rotatedWidth);
      gridPaneRotated.snapshot(params, imageRotated);
      imageViewRotated = new ImageView(imageRotated);
      (…)
      }
```

Uvnitř této třídy je ještě jedna privátní třída *RubberBandSelection* sloužící k výběru oblasti a určení ohraničení výběru. Tato třída existuje od doby ukotvení obrazu a její funkce je taková, že snímá stisknutí tlačítka myši na obraze, ukládá souřadnice a vykresluje výběrový obdélník. Ten se využívá v metodě *b_crop_onAction* vyvolané tlačítkem Oříznout. Podle vybrané oblasti se vytvoří nový menší obrázek, který je zároveň uložen na disk, kam uživatel chce. Výsledný obraz makrostruktury může mít průhlednou oblast, která vznikla při výběru oblasti i mimo původní strukturu a není považována za závadu.

3.3 Balík objects

V tomto balíčku jsou obsaženy všechny elementy vyskytující se na struktuře a jejich vlastnosti. Jedna z nich je třída s výčtem pravidel *EnumRules,* ve kterých je přiřazená interpretaci k číslu pravidla. Druhý výčtový typ patří *EnumElements*, který popisuje všechny elementy v matici struktury, včetně těch nevizuálních jako je prázdný element.

Dalším podřízeným balíčkem tohoto je *elements*. V něm se nacházejí třídy, které jsou nějakým způsobem nadřízené těm, které jsou obsažené v nejnižším balíčku *core*. Nadřízené jsou buď hierarchicky, kvůli členění dokumentu jako takového, nebo kvůli tomu, že vlastnosti a metody z nich využívá více dalších poděděných objektů.



Obrázek 15: Hierarchie elementů. Zdroj: Vlastní

3.3.1 Třída ElementBase

Každý element má nejvyššío předka ve třídě *ElementsBase*, kde je obsloužena zároveň většina společné činnosti. Ostatní třídy jsou spíš pro udržení hierarchie a jednodušší práci s výslednými objekty. Podstatné jsou dvě vícerozměrná pole:

```
private EnumElements[][][][] array4dOfNeighboursByColorOnly;
private EnumElements[][][][] array4dOfNeighboursByColorAndShape;
```

První proměnná je definice povolených elementů pouze podle barvy, v druhé podle barvy i podle tvaru. Zde jsou uchována pravidla pro každý vizuální element o svých možných sousedech, o potenciálních sousedních elementech. Myšlenka je taková, že první rozměr pole je index pravidla (0–3), druhý rozměr je index strany (0 = sever, 1 = východ, 2 = jih, 3 = západ), třetí rozměr je skupina elementů (1–4; v programu 0–3) a ve čtvrtém rozměru je již samotná skupina povolených prvků.

Na příklad pro pravidlo V=1, od severního sousedního elementu a s vypočteným koeficientem 3 bude dotaz takový:

```
EnumElements[] arrayEnumColor =
elementUp.getAllPossibleEnumElementsForRuleSideAndCoeficientByColorOnly(1, 2, 3);
```

Tento přístup má jak výhody, tak nevýhody. Výhodou je rychlost, skupina povolených typů elementů je získána bez výpočtu s konstantní složitostí. Za nevýhodu se dá považovat větší velikost každého objektu elementu, avšak pokud je element vytvářen pro jiné účely než pro výpočet okolních, tak toto vícerozměrné pole neobsahuje. Zásadnější nevýhodou je napevno dané pravidlo pro každý element, nelze tedy jednoduše definovat nové elementy. Od začátku byl důraz kladen na rychlost výpočetního algoritmu a složitost případných změn pravidel pro elementy se objevila až s hotovým řešením. Ukázka samotné metody:

```
public EnumElements[]
getAllPossibleEnumElementsForRuleSideAndCoeficientByColorOnly(
int ruleNumber, int side, int resultCoeficient) {
    if ((ruleNumber >= 0) && (ruleNumber <= 3)) {
        if ((side >= 0) && (side <= 3)) {
            if ((resultCoeficient >= 1) && (resultCoeficient <= 4)) {
                return
this.array4dOfNeighboursByColorOnly[ruleNumber][side][resultCoeficient - 1];
            }
        }
    }
    return null;
}
```

3.4 Balík logic

V tomto balíku jsou dvě třídy pro práci s maticí struktury. Jedná se o nejdůležitější třídy, co se týče logiky a výpočtů celého programu. Na začátku byla myšlenka taková, že třída *MatrixMainAppElements* bude zpracovávat data mezi grafickým výstupem a maticí a bude jen obsluhovat události vznikající na elementech struktury a třída *MatrixOperator* bude pouze provádět výpočty s maticí.

3.4.1 Třída MatrixMainAppElements

U další třídy bylo oddělení funkčnosti dodrženo (což je důležitější), avšak do této se vetřely i některé metody, které by měly být pravděpodobně v jiných, nebo tuto třídu ještě rozdělit na dílčí (například průchod maticí řádek po řádku by měl být také v další třídě).

V této třídě je hlavní metoda pro vykreslování na poskytnutý element v argumentu. Přečte aktuální stav všech elementů v matici a umístí je do mřížky na prvek. Zároveň také určí, jak bude daný prvek vypadat, což je jedna z částí, která by mohla být přesunuta, pro dosažení univerzálnosti pro všechny grafické prvky. Na jiné grafické prvky ovšem pro umístění stejně nelze použít funkci *node.setContent(node)* a proto metoda zůstala stejná.

Tato třída také umí převést matici na proud dat a uložit ho do souboru. Stejně tak se stará o načtení takto uložených dat z textového souboru, každý element je přeložen na dvouznakový identifikátor, na začátek souboru se připíší rozměry matice a potom již elementy, pozice v souboru odpovídá pozici v matici. Načtení probíhá podobně jako ruční vytváření nové partitury, nejdříve se ze zvoleného souboru načtou rozměry výsledné partitury a potom řádek po řádku načtou řetězce s dvouznakovými názvy elementů a ty se jeden po druhém vytváří a ukládají do matice.

3.4.2 Třída MatrixOperator

V této třídě probíhají všechny výpočty struktury, přesněji práce s maticí elementů. Postup zpracování při zavolání výpočtu elementu je následující: Metoda *calculateAndRoundOneCoeficient* spočítá koeficient pro hledaný element (metoda výpočtu vizte Algoritmus pro struktury). Tento koeficient nám řekne, z jaké skupiny elementů budeme výsledný požadovat. Metoda *findAllPossibleElementTypesByCoeffiecient* funguje tak, že se podívá na všechny čtyři směry kolem hledaného a odtud získá umístěné elementy. V každém z těchto elementů jsou již připraveny dva seznamy povolených (vizte kapitolu *Třída ElementBase*). Nejdříve se vyzkouší přísnější pravidlo jak pro barvu, tak zároveň tvar. Udělá se průnik všech čtyř seznamů okolních elementů. Pokud zbyde jeden a více elementů, víme, že ty se mohou umístit do matice. Pokud ovšem žádný nezůstane, musíme tu stejnou operaci provést se čtyřmi seznamy povolených elementů s pravidlem pouze pro návaznost (nebo nenávaznost) podle barvy. Pokud ani z tohoto průniku nevzejde žádný kandidát na umístění, vezme se náhodný element z předchozí skupiny (navazuje alespoň na jedné straně podle pravidla barvy). Ukázka jednoho průniku povolených prvků v kódu:

```
List<EnumElements> arrayListOfComonElementTypesByColorOnly = new
ArrayList<>(arrayListElementUpColor);
arrayListOfComonElementTypesByColorOnly.addAll(arrayListElementRightColor);
arrayListOfComonElementTypesByColorOnly.addAll(arrayListElementDownColor);
arrayListOfComonElementTypesByColorOnly.addAll(arrayListElementLeftColor);
arrayListOfComonElementTypesByColorOnly.retainAll(arrayListElementUpColor);
arrayListOfComonElementTypesByColorOnly.retainAll(arrayListElementRightColor);
arrayListOfComonElementTypesByColorOnly.retainAll(arrayListElementDownColor);
arrayListOfComonElementTypesByColorOnly.retainAll(arrayListElementLeftColor);
if (!arrayListOfComonElementTypesByColorOnly.isEmpty()) {
    return arrayListOfComonElementTypesByColorOnly.
stream().distinct().collect(Collectors.toList());
}
```

4 ROZBOR OBRAZŮ GENEROVANÝCH NAPROGRAM. APLK.

V této části budeme prezentovat grafické výstupy z naprogramované aplikace a porovnáme je se Sýkorovými obrazy. Pro hodnocení je nutno vzít v potaz to, že rozhodovací algoritmus umisťování elementů do struktury pracuje s náhodností a generováním vždy získáme odlišný výsledek. Elementy budou vždy ze stejné skupiny, avšak z ní mohou nabývat různých úrovní (z, y, apod.), vzhledem k tomu, že nejpřísnější pravidlo, uplatnění pravidla jak dle barvy, tak dle tvaru, je relativně striktní a tak dochází častěji k uvolnění daného pravidla na výběr dle barvy, nebo dokonce pouze na celou skupinu.

Výpočet struktury je navíc také postupný, začíná vždy v levém horním rohu a končím v levém dolním rohu struktury (v případě lichého počtu řádků), nebo v pravém dolním rohu struktury (v opačném případě). Element ze spodní řady může mít vliv na horní řadu pouze v případě předvyplnění, element vypočítaný v průběhu algoritmu již do horních řad nezasáhne, avšak odlišný element z horních řad může měnit strukturu elementů i o několik iterací vzdálených.

Reverzní analýza makrostruktur je téměř nemožná, jak vzhledem k výše zmíněným proměnným, díky úhlu natočení (úhel může být 45º, 135 º, nebo 225º) a také díky oblasti výřezu. Pro rozbor makrostruktury je nutná znalost jak partitury, tak výsledného obrazu.

V následujících kapitolách bude snaha o rozbor makrostruktury s německým názvem Rosa-grün-struktur 2 (Červeno-zelená struktura 2) z roku 1970, která se prodala v roce 2010 v aukční síni za 40 000 Kč [14], v dnešní době je její cena 75 000 Kč [15].

4.1 Porovnání makrostruktur

Shrneme si informace, které jsme schopni vyčíst ze Sýkorovy Červeno-zelené makrostruktury (zobrazené na obrázku 16). Toto nám umožňují zkušenosti z experimentů při generování vlastních Struktur. Plátno původní struktury bylo minimálně 13 elementů vysoké a 13 široké. Pokud nebylo ve struktuře definováno velké množství elementů, potom se dá odhadnout pravidlo na „Nenavazuje barvou ani tvarem“ s vyšším počtem znamének plus a minus. Koeficient byl pravděpodobně vysoký, protože se prvky ze skupiny jedna potkávají se skupinou čtyři. Proto koeficient odhadneme číslem dva.

Obrázek 16: Červeno-zelená makrostruktura 1970. Zdroj: [15]

Z těchto vypozorovaných parametrů mohla být vytvořena partitura podle výše určených pravidel. Pro účely výzkumu byly dále vykresleny 4 struktury dle každého pravidla. V každé této struktuře proběhlo hledání makrostruktury, která by odpovídala nejvíce originální makrostruktuře. Pro zpětnou analýzu může existovat obrovské množství zadání, proto je zde jako výsledek uvedena celá skupina obrazů, podle každého pravidla. Je tak vidět rozdíly při stejné partituře, stejném úhlem natočení a přibližně stejné oblastí výřezu. Výsledek se dá rozporovat, ale stejná makrostruktura by šla vykreslit pouze v případě téměř úplného zaplnění partitury. Ale v takovém případě není potřeba program ke generování a lépe by posloužil grafický editor.

Obrázek 17: Replika 1, V = 0. Zdroj: Vlastní

Obrázek 18: Replika 2, V = 1. Zdroj: Vlastní

Obrázek 19: Replika 3, V = 2. Zdroj: Vlastní

Obrázek 20: Replika 4, V = 3. Zdroj: Vlastní

5 UŽIVATELSKÁ PŘÍRUČKA K APLIKACI

Program sestavený v této závěrečné práci slouží k reprodukci struktur a makrostruktur Zdeňka Sýkory, když využívá algoritmus popsaný v časopise Leonardo. Příručka popsaná v této kapitole usnadňuje orientaci v aplikaci a zrychlí užívání.

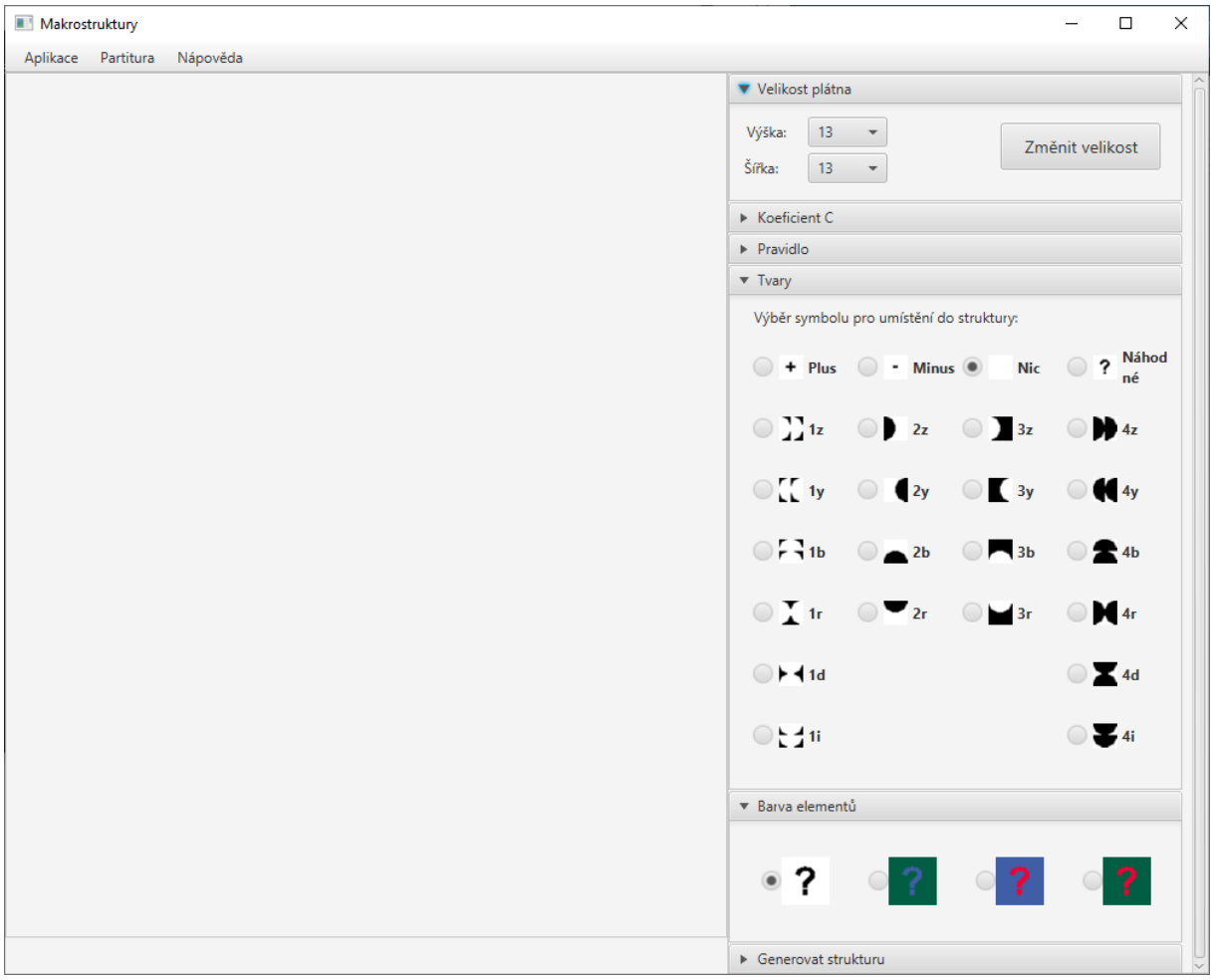
Spuštění programu má dvě varianty. Doporučuje se první varianta, pouze dvojklikem na soubor *spustit_nemamNic.bat*. Tento soubor nevyžaduje žádné další komponenty v uživatelském počítači a spustí hlavní program. Ten obsahuje hlavní program *Macrostructures.jar*, složku *jre* a dávkový soubor *spustit_nemamNic.bat*. Celý program se v tomto případě spouští dávkovým souborem bez dalších potřebných znalostí o běhu. Složka *jre* obsahuje celé prostředí nutné pro běh programu, jedná se o Java 11 a JavaFX zkompilované do jednoho balíku. Dávkový soubor již jen spustí program bez parametrů. Výhoda této varianty je bezobslužnost a dlouhodobá funkčnost celé aplikace.

Druhá varianta spuštění je pro pokročilejší uživatele s prostředím Java 11 a JavaFX již v počítači. Při tomto spuštění je nutný pouze program *Macrostructures.jar* a dávkový soubor *spustit_mamJava11aJavaFX.bat,* ve kterém je potřeba změnit aktuální cestu k JavaFX v proměnné %PATH_TO_FX%. Spuštění programu touto variantou má výhodu např. při přenosu do jiného počítače v menší velikosti přesouvaného balíku (celá složka *jre* při této variantě není třeba).

5.1 Uživatelské rozhraní hlavního okna

Hlavní okno, které se otevře po spuštění, je rozděleno na tři části. První část je horizontální nabídka, ve které jsou možnosti pro ukončení aplikace, uložení a nahrání partitury a nápověda. Možnost uložení je aktivní, pouze pokud existuje partitura. Naopak nahrát se dá pouze partitura do prázdného okna. V nápovědě je zkrácený popis a odkazy na bakalářskou práci a manuál.

Druhá část je vertikální nabídka v pravé části okna. Zde jsou rozbalovací panely pro ovládání celého průběhu generování. První nabídka slouží k výběru velikosti struktury a vytvoření mřížky (do třetí části hlavního okna). Další záložka slouží ke zvolení koeficientu, který se přičítá nebo odečítá. Následující záložka je volba pravidla jedna až čtyři. V záložce Tvary jsou všechny elementy, které jdou vložit do mřížky struktury, včetně prázdného elementu, nebo náhodného (který se vybere ze všech ostatních). Předposlední záložka je pro volbu barevného složení výsledné struktury. V poslední nabídce je tlačítko pro generování celé struktury.



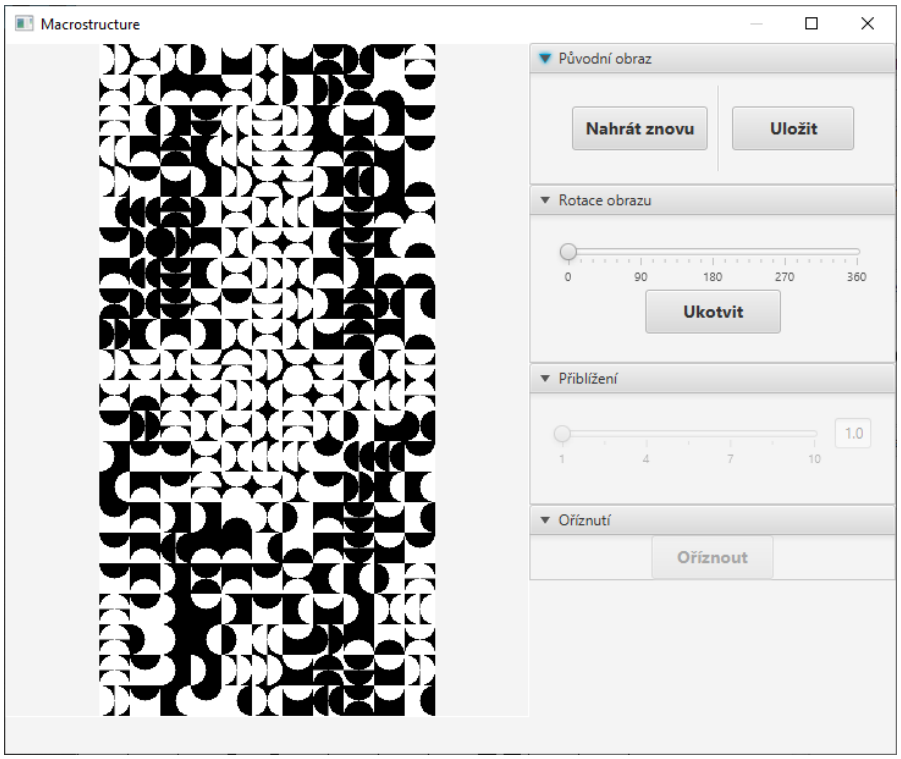
Obrázek 21: Hlavní okno aplikace. Zdroj: Vlastní

Postup tvoření v programu je tento:

1. Po spuštění aplikace vybrat buď načít partituru ze souboru přes horní nabídku, nebo zvolit velikost partitury a změnit velikost,
2. dále upravit koeficient,
3. vybrat vhodné pravidlo z nabídky,
4. levým tlačítkem myši přepínat elementy z nabídky tvarů a stejně je umisťovat do mřížky,
5. zvolit barvu pro výslednou makrostrukturu,
6. nechat program vypočítat strukturu tlačítkem Generuj.

5.2 Okno pro strukturu a makrostrukturu

Po vygenerování se otevře nové okno s vypočítanou strukturou. Toto okno má dvě části: strukturu a v pravé části ovládání. V menu je možné obnovení výchozí struktury a také její přímé uložení jako nový obrázek. Další jsou možnosti natočení celé struktury, uzamknutí s touto rotací, přiblížení celé struktury a po výběru oblasti oříznutí na makrostrukturu.



Obrázek 22: Okno makrostruktury. Zdroj: Vlastní

Postup práce s tímto oknem je tento:

1. Po otevření okna je možné strukturu uložit tlačítkem,
2. v případě tvorby makrostruktury se dá posuvníkem nastavit rotace,
3. tlačítkem uzamknout strukturu na požadovaném úhlu,
4. dalším krokem může být přiblížení celé struktury,
5. nyní je již povolený výběr myší pro oblast oříznutí,
6. finálním krokem je ořez tlačítkem a zároveň uložení nové makrostruktury.

Struktura se dá uložit do samostatného souboru obrázku buď po spuštění okna, nebo při opětovném nahrání celé struktury tlačítkem. Výběr tlačítkem myši je povolený po uzamčení rotace struktury. Po zavření tohoto okna je možné vrátit se do hlavního okna s partiturou a provést zde změny i nové generování struktury.

ZÁVĚR

Hlavním cíle celé bakalářské práce bylo analyzovat způsob vytváření struktur a makrostruktur docenta Zdeňka Sýkory. Nejdůležitějším zdrojem byl článek [11]. Dalším krokem na základě této analýzy vytvořit program, který dokáže vytvářet struktury a makrostruktury.

Rozbor algoritmu byl založen na rozboru originálního článku a jeho překladu. Dále potom na překladu německy psaného popisu vzniku struktur v knize Zdeněk Sýkora 90, kde je zároveň český překlad. Sýkora si nejdříve vymyslel pravidla, vytvořil obrazy a až poté teprve algoritmus sepsal a pevně definoval. Způsob vytváření byl celkem snadno pochopitelný a až na jednu drobnou nejasnost dobře popsaný. Diplomová práce Jiřího Hory posloužila pro případné ujasnění nejasností, nebo k analýze chyb. Způsob vygenerování replik zvolené Makrostruktury byl založen na rozboru této předlohy.

Aplikace dokáže vygenerovat struktury podle Sýkorových postupů a principů. Uživatel může definovat velikost struktury, jednotlivé elementy do struktury, koeficient intenzity přechodu, pravidlo návaznosti a barvu generovaných elementů. Dále může uživatel výslednou strukturu uložit, nebo ji upravit podle svých představ. Může zvolit úhel natočení, násobek přiblížení a oblast, která bude oříznuta na makrostrukturu. Aplikace by mohla být vylepšena například spoluprací více vláken, nebo vektorovým vykreslováním elementů.

Výsledné obrazy jsou vytvářené sestavováním čtvercových elementů o hraně 300 pixelů, nevznikají vektorově. I přiblížené Makrostruktury však mají křivky dostatečně hladké a až při vytištění na příklad na velkoplošné tiskárně by mohl být rozlišen rastr elementů, na běžném displeji jsou obrazy úplně ostré.

POUŽITÁ LITERATURA

[1] LAPOSKY, Ben F. Oscillons: Electronic Abstractions. *Leonardo*. 1969, **2**(4), 345-356. DOI: 10.2307/1572117. ISSN 0024094X. Dostupné také z: https://www.jstor.org/stable/1572117?origin=crossref.

[2] HOSCH, William L. Ivan Edward Sutherland. *Encyclopædia Britannica* [online]. Chicago: Encyclopædia Britannica, 2019 [cit. 2019-12-06]. Dostupné z: https://www.britannica.com/biography/Ivan-Edward-Sutherland.

[3] SÝKOROVÁ, Lenka. Zdeněk Sýkora – životopis. *Zdeněk Sýkora*. [online]. Louny, 2019. [cit. 2019-12-06]. Dostupné z: http://www.zdeneksykora.cz/?s=zivotopis.

[4] BURDA, Vladimír, JAREŠ, Michal, ed. *Lyrické minimum*. Praha: Torst, 2004. ISBN 80-721-5235-1.

[5] xxx9. Op-art. *Referáty-seminárky.cz* [online]. Praha: Referáty-seminárky.cz, 2006 [cit. 2019-12-06]. ISSN 1802-422X. Dostupné z: http://referaty-seminarky.cz/op-art/.

[6] VALENTA, Jakub. Konstruktivní tendence. *Jakubvalenta.cz* [online]. Praha: jakubvalenta.cz, 2019 [cit. 2019-12-06]. Dostupné z: https://www.jakubvalenta.cz/statnice/3_08_konstruktivni_tendence.html.

[7] SÝKOROVÁ, Lenka. Dílo Zdeňka Sýkory. *Zdeněk Sýkora*. [online]. Louny, 2019. [cit. 2019-12-06]. Dostupné z: http://www.zdeneksykora.cz/?s=galerie.

[8] SÝKOROVÁ, Lenka. Realizace v architektuře. *Zdeněk Sýkora*. [online]. Louny, 2019. [cit. 2019-12-06]. Dostupné z: http://www.zdeneksykora.cz/?s=realizace_v_architekture.

[9] DUSAN. Zdeněk Sýkora. *Monoskop* [online]. Bratislava: Monoskop.org, 2016 [cit. 2019-12-06]. Dostupné z: https://monoskop.org/Zden%C4%9Bk_S%C3%BDkora.

[10] SÝKORA, Zdeněk a Pavel KAPPEL. Zdeněk Sýkora 90. Praha: Verzone, 2010. ISBN 978-80-904546-1-3.

[11] SYKORA, Zdenek, BLAZEK, Jaroslav. Computer-Aided Multi-Element Geometrical Abstract Paintings. Leonardo [online]. 1970, 3(4) [cit. 2019-12-06]. DOI: 10.2307/1572257. ISSN 0024094X. Dostupné z: https://www.jstor.org/stable/1572257?origin=crossref.

[12] SVOBODA, M. - MAREK, J. Construction and statistical analysis of Zdeněk Sýkora's lines. In APLIMAT 2014: 13th Conference on Applied Mathematics: proceedings. Bratislava: Slovenská technická univezita v Bratislave, 2014. s. 387-392. ISBN 978-80-227-4140-8.

[13] Java FX. *Oracle* [online]. Redwood City: Oracle Corporation, 2019 [cit. 2019-12-06]. Dostupné z: https://www.oracle.com/technetwork/java/javase/overview/javafx-overview-2158620.html.

[14] Sýkora Zdeněk - Červeno - zelená struktura. *1. ART CONSULTING* [online]. Brno: 1. ART CONSULTING BRNO CZ, 2010 [cit. 2019-12-06]. Dostupné z: http://www.acb.cz/Sykora-Zdenek-Cerveno-zelena-struktura-35645.

[15] Zdeněk Sýkora. *PRAGUE AUCTIONS* [online]. Praha: PRAGUE AUCTIONS, 2017 [cit. 2019-12-06]. Dostupné z: https://www.pragueauctions.com/nakup-prodej/primy-prodej/zdenek-sykora/?id=56-16965.

[16] SKŘIVÁNEK, Jan. Top 10 Sýkora. *ART+* [online]. Praha: AmbitMedia, 2011 [cit. 2019-12-08]. Dostupné z: https://www.artplus.cz/cs/aukcni-zpravodajstvi/1/top-10-sykora.

[17] MAREK, Jaroslav, NEDVĚDOVÁ, Marie. Historie digitálního umění: Náhoda, počítač a Linie Zdeňka Sýkory: 37. mezinárodní konference Historie matematiky. Praha: Matfyzpress, 2016. s. 137-146 s. ISBN 978-80-7378-317-4.

[18] BRABEC, Jaroslav. Zdeněk Sýkora. Česká televize [online]. Praha: Česká televize, 2001 [cit. 2019-12-08]. Dostupné z: https://www.ceskatelevize.cz/porady/1026666614-evropane/20136219568-zdenek-sykora/.

[19] HORA, Jiří. *Zdeněk Sýkora: Struktury.* Pardubice, 2017. Diplomová práce. Univerzita Pardubice. Vedoucí práce Mgr. Jaroslav Marek.

[20] RUSHFORT, Kevin. JDK-8088198. *JDK Bug System* [online]. Redwood City: Atlassian, 2012 [cit. 2019-12-08]. Dostupné z: https://bugs.openjdk.java.net/browse/JDK-8088198.

[21] A. Michael Noll. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2019 [cit. 2019-12-08]. Dostupné z: https://en.wikipedia.org/wiki/A._Michael_Noll.

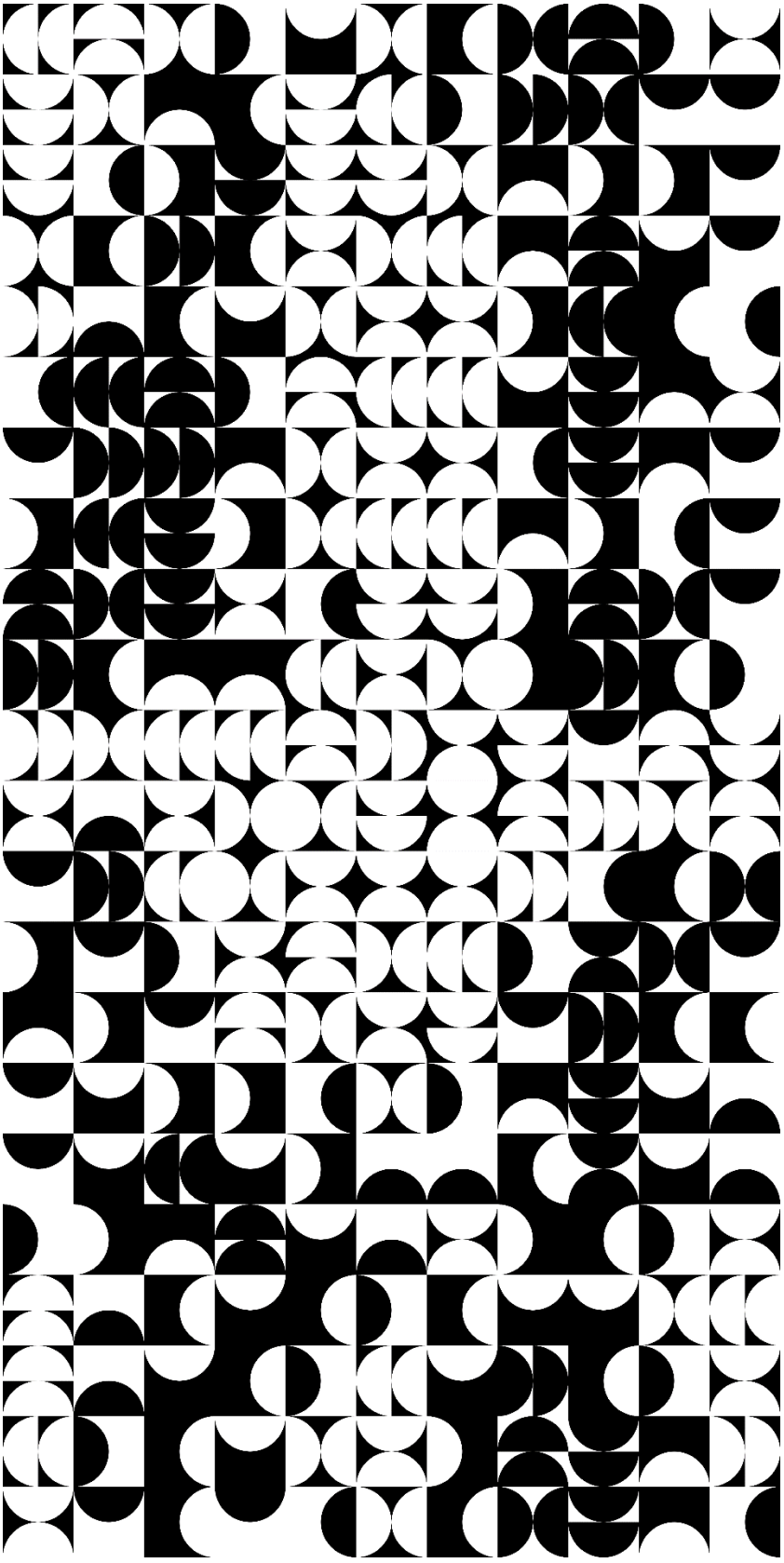
[22] Desmond Paul Henry. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2019 [cit. 2019-12-08]. Dostupné z: https://en.wikipedia.org/wiki/Desmond_Paul_Henry.

[23] François Morellet. *Artnet* [online]. Německo: artnet, 2016 [cit. 2019-12-08]. Dostupné z: http://www.artnet.com/artists/fran%C3%A7ois-morellet/.

PŘÍLOHY

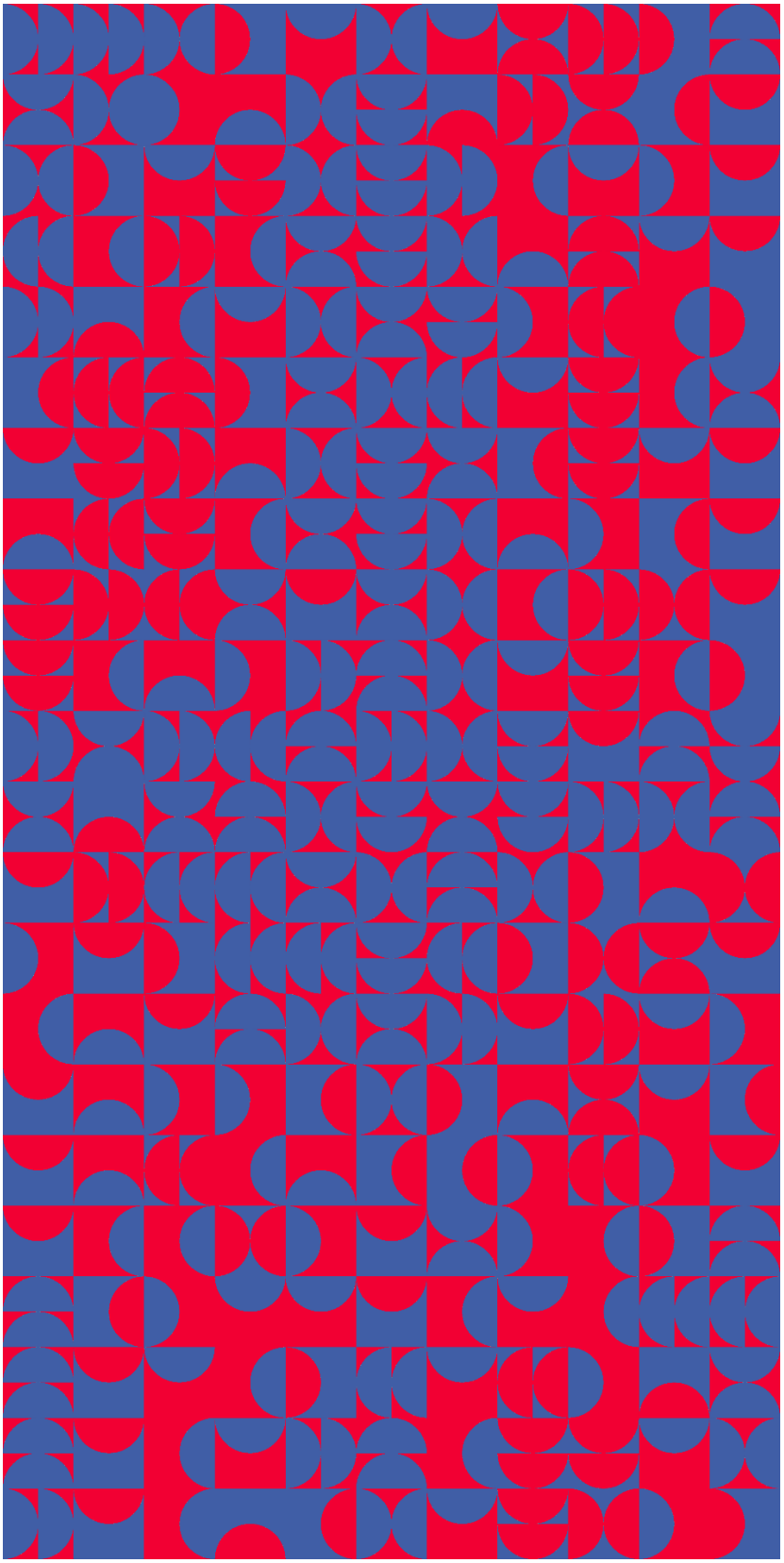
PŘÍLOHA A – ČERNOBÍLÁ STRUKTURA

Nejznámější Sýkorova struktura, pravidlo V = 2, C = 0,75.



PŘÍLOHA B – ČERVENOMODRÁ STRUKTURA

Stejná partitura jako struktura výše, pravidlo V = 2, C = 0,75.

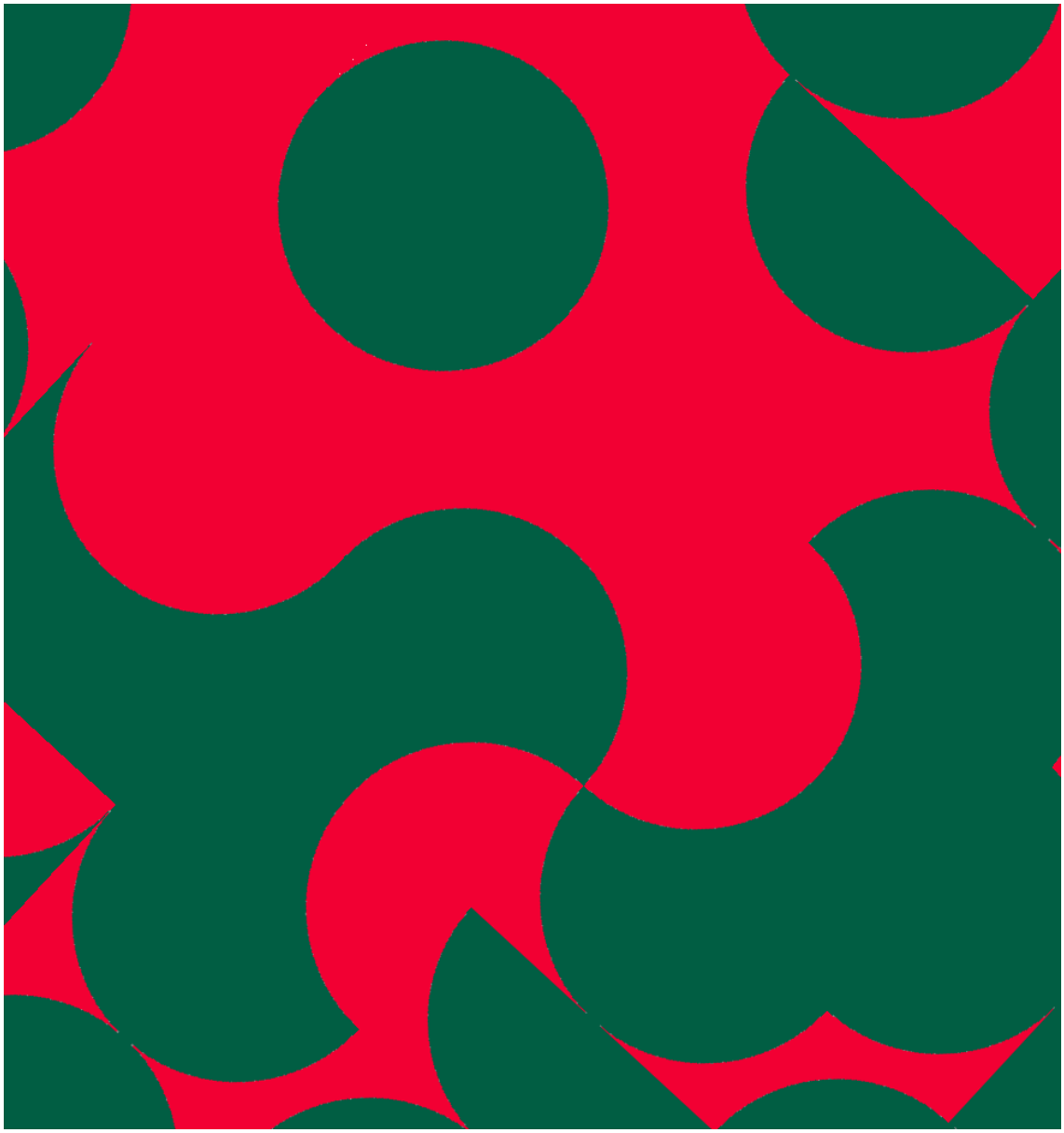


PŘÍLOHA C – ČERVENOZELENÁ MAKROSTRUKTURA

Stejná partitura byla využita pro vytvoření makrostruktury, pravidlo V = 0, C = 0,5, úhel 45°.

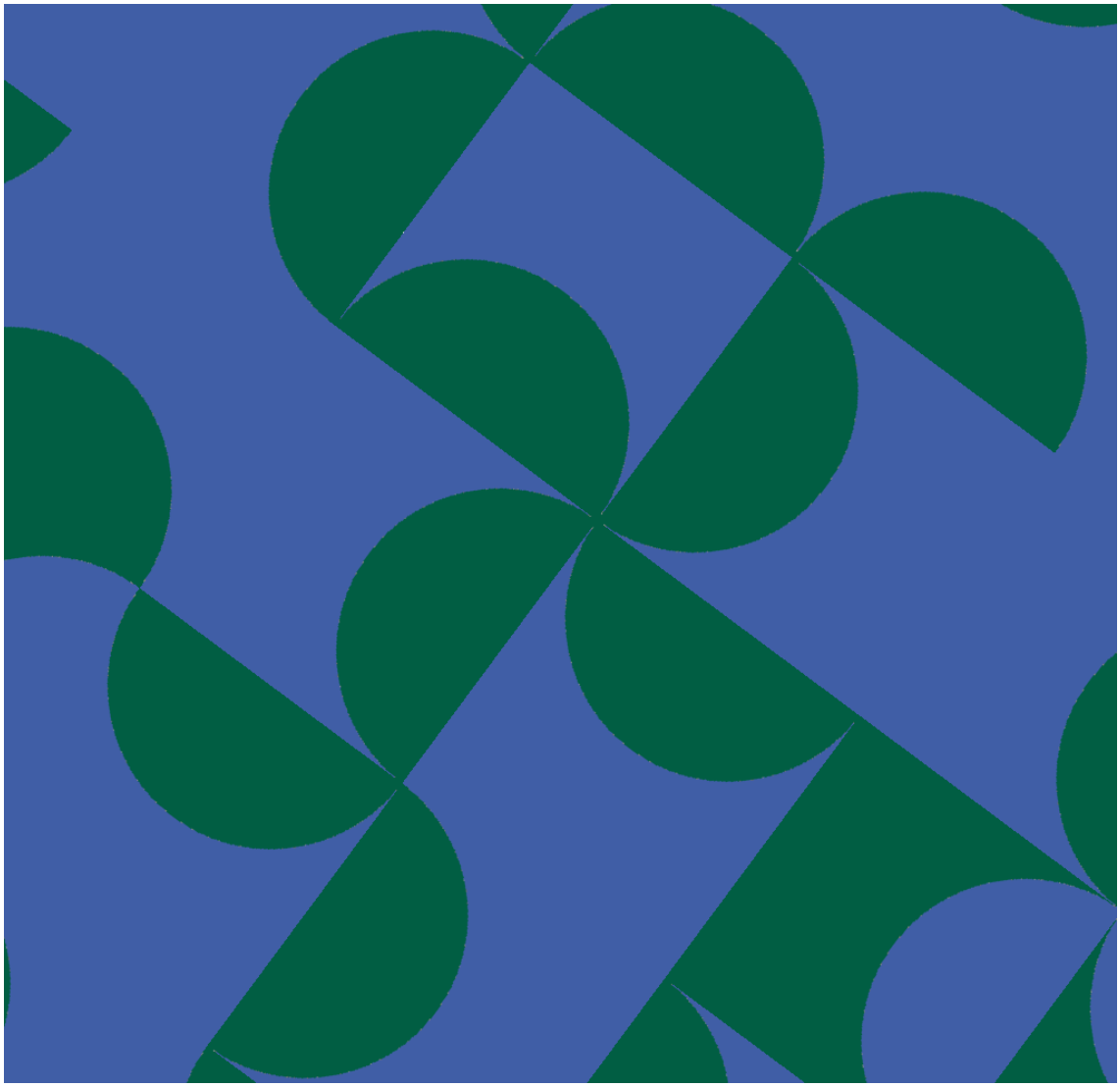
PŘÍLOHA D – ČERVENOZELENÁ MAKROSTRUKTURA

Stejná partitura byla využita pro vytvoření makrostruktury, pravidlo V = 0, C = 0,5, úhel 45°.



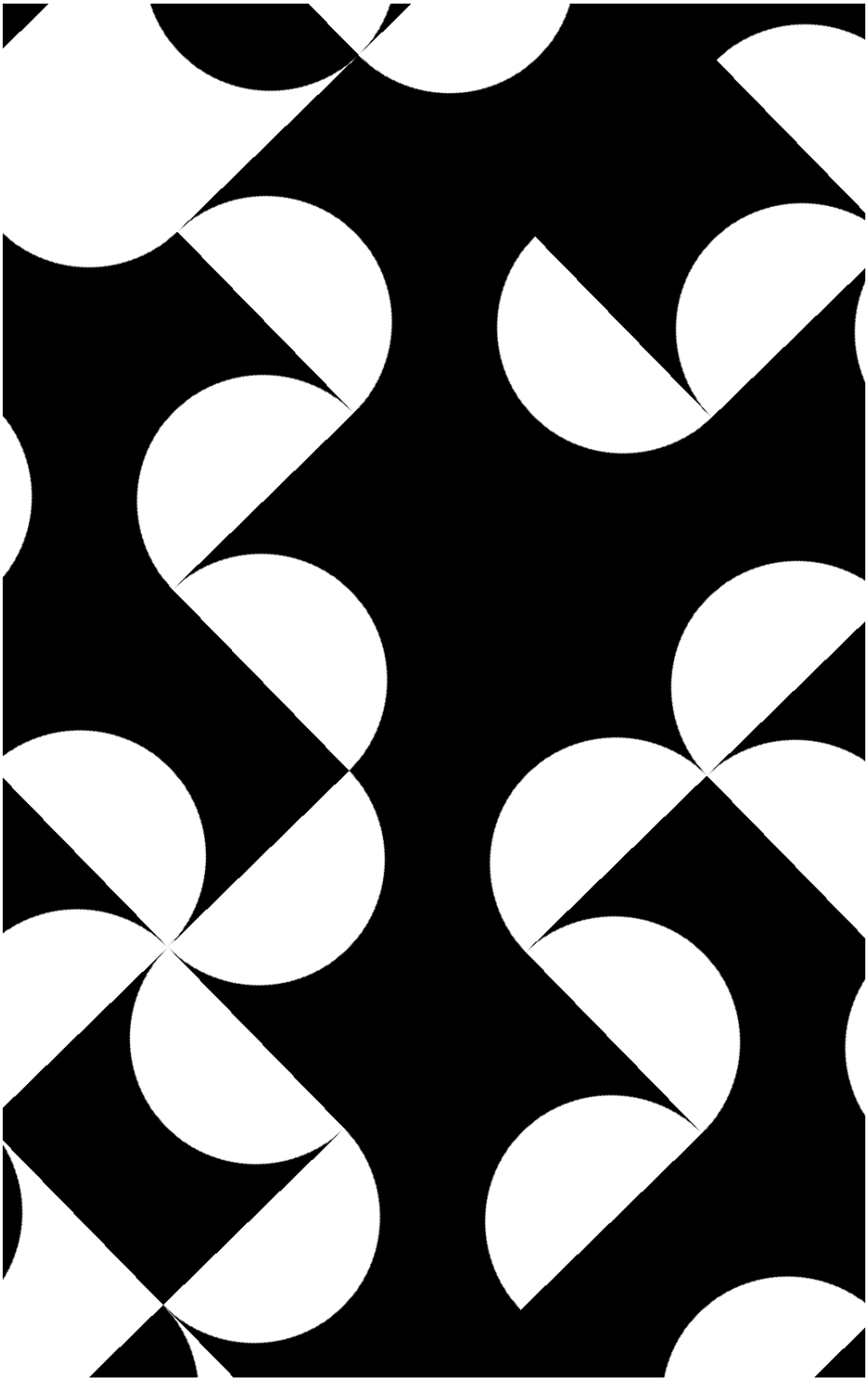
PŘÍLOHA E – MODROZELENÁ MAKROSTRUKTURA

Náhodná partitura, pravidlo V = 3, C = 0,0, úhel 125°.



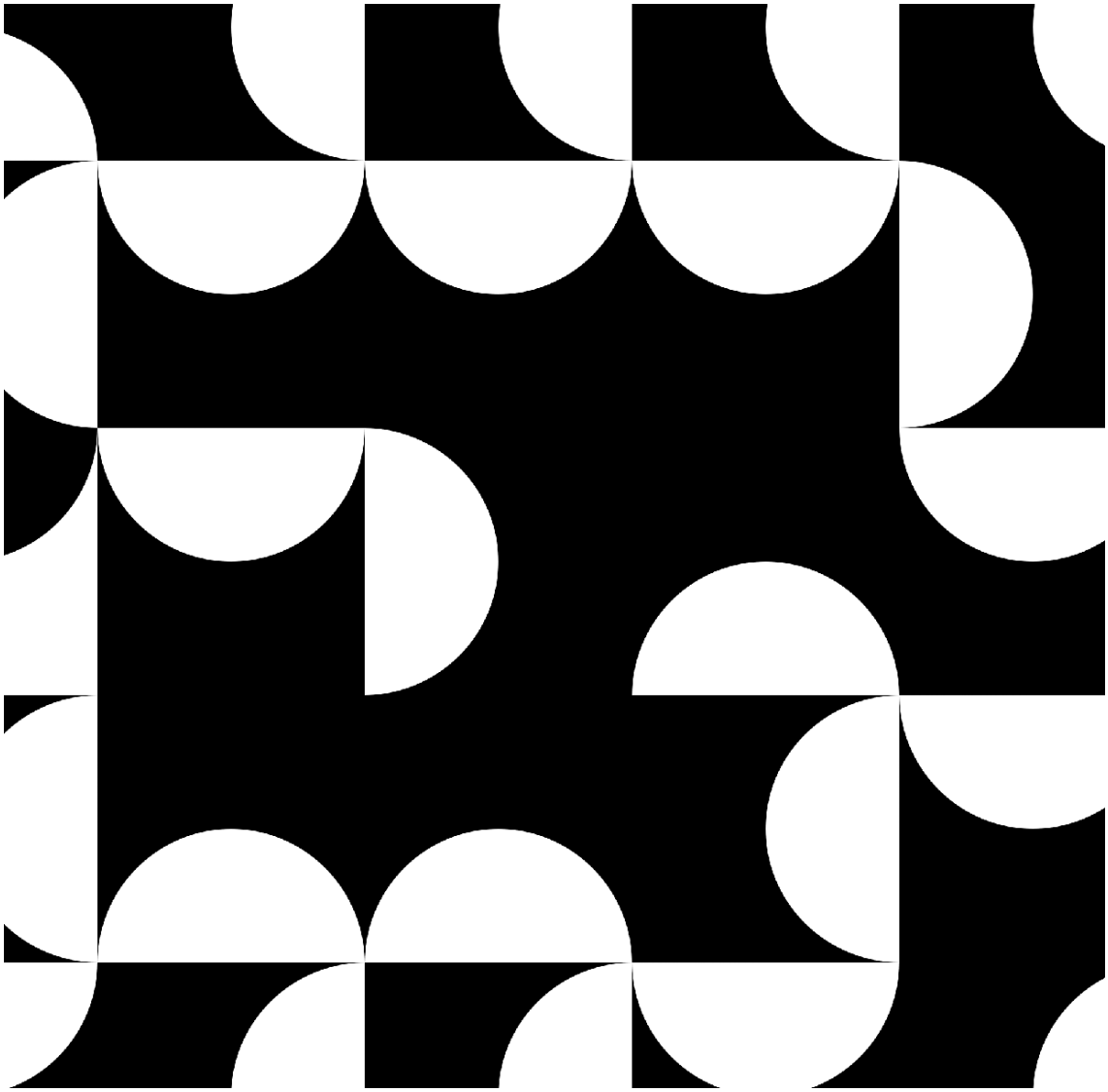
PŘÍLOHA F – ČERNOBÍLÁ MAKROSTRUKTURA

Náhodná partitura, pravidlo V = 2, C = 0,3, úhel 125°.



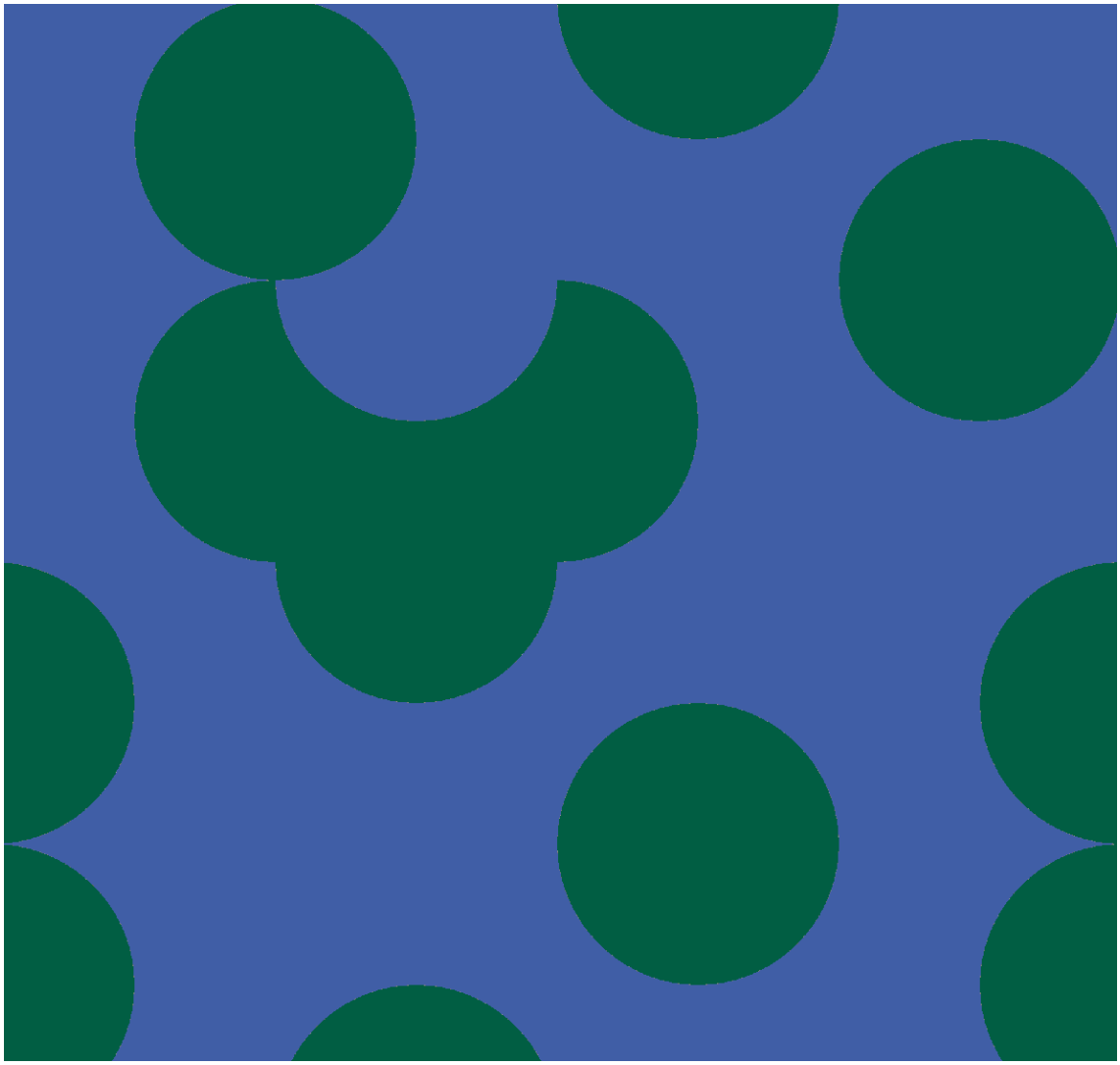
PŘÍLOHA G – ČERNOBÍLÁ MAKROSTRUKTURA

Náhodná partitura, pravidlo V = 1, C = 0,3, úhel 0°.



PŘÍLOHA H – MODROZELENÁ MAKROSTRUKTURA

Náhodná partitura, pravidlo V = 0, C = 0,3, úhel 90°.



PŘÍLOHA I – MODROZELENÁ MAKROSTRUKTURA

Náhodná partitura, pravidlo V = 0, C = 0,3, úhel 75°.

PŘÍLOHA J – MODROZELENÁ MAKROSTRUKTURA

Náhodná partitura, pravidlo V = 0, C = 0,0; úhel 125°.