



**FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ** **ústav automatizace
a měřicí techniky**

**MAPV – Projekt 7
Brno 2016**

Rekonstrukce obrazu s využitím evolučního algoritmu

Vedoucí projektu:

Ing. Petr Nováček

Vypracovali:

Bc. Aliaksei Halachkin

Bc. Denis Fedor

Zadání

Cílem projektu je navrhnout evoluční algoritmus schopný rekonstruovat obraz překládáním mnohoúhelníků. Navrhněte vlastní evoluční operátory (křížení, mutace) a vlastní fitness funkce. Navržené evoluční operátory a fitness funkce následně vyhodnoťte a rozeberte v dokumentaci. Vstupem algoritmu bude vždy snímek (bitmapa), jenž má být rekonstruován, výstupem pak rekonstruovaný snímek doplněný o grafy průběhu fitness funkce. Rekonstruovaný snímek bude ve formátu bitmapy a zároveň bude výstupem soubor (například svg) obsahující souřadnice a barvu jednotlivých obrazců z nichž je výsledný obraz složen. V dokumentaci budou rozebrány jednotlivé části algoritmu, především pak budou porovnány a vyhodnoceny vlastní evoluční operátory. Vlastní fitness funkce budou případně porovnány s IQA (Image Quality Assessment) algoritmy (například s algoritmem SSIM - Structural Similarity Index). Projekt bude zpracován v jazyce C++ s využitím například knihovny OpenCV a pravidelně konzultován (zhruba po 14ti dnech).

Vstupy:

- Snímek (předloha), jenž má být rekonstruován.

Výstupy:

- Rekonstruovaný snímek.
- Evoluční algoritmus pro rekonstrukci snímků, navržené operátory a fitness funkce.
- Dokumentace algoritmů s důrazem na vyhodnocení a porovnání navržených operátorů a fitness funkcí.

Testovací obrázky:



Vypracování

Prvý návrh programu

Prvé kroky vypracovania zadaného projektu spočívali v inštalácii programu Microsoft Visual Studio 2015, následne sme vytvorili build OpenCV a to taktiež nainštalovali. OpenCV sme prepojili s našim projektom ga_art vo Visual Studiu a samotný projekt umiestnili na git (Bitbucket). Prvotným návrhom bol program, ktorý využíval jednoduché genetické algoritmy spočívajúce v prekladaní mnohouholníkov.

Každý mnohouholník bol označený ako gén. Každá množina mnohouholníkov, teda obrázok, bol označený ako jedinec. Množina obrázkov bola nazvaná populáciou. Ako fitness funkcia, určujúca zhodu medzi zadaným a generovaným obrázkom - jedincom, bola použitá stredná kvadratická odchýlka. Výber najlepších jedincov z danej generácie zaisťovala selekcia typu Roulette wheel alebo Rank selection. Program využíval elitárstvo (zachovanie najlepších jedincov z predchádzajúcej generácie) a jednoduché kríženie – nové jedince pre ďalšiu generáciu vznikali premiešaním génov (mnohouholníkov) z rodičovských jedincov, ich rozdelením na polovicu a prepojením. Jednoduchá mutácia spočívala v náhodnom výbere niektorého mnohouholníku a náhodnej zmene jeho parametrov. Tento algoritmus sa nám však neosvedčil, jeho výsledky neboli uspokojujúce, preto sme sa po konzultácii s vedúcim projektu rozhodli prejsť z tohto návrhu na program fungujúci na báze Hill Climbingu.

Jedným z prvých problémov, ktorý sme museli prekonať už v prvotnom návrhu programu bolo kreslenie mnohouholníkov do generovaného obrázku. Samotný problém spočíval v tom, že OpenCV pri vykresľovaní mnohouholníkov natívne nepočíta s využitím priehľadnosti (Alpha, 4. kanál farby), to sa nám však podarilo obísť využitím funkcie addWeighted, ktorá spája 2 obrázky s určitou váhou. Každý mnohouholník sa preto vždy vykreslí do nového obrázku ako nepriehľadný a tento obrázok sa spojí s generovaným obrázkom s váhou odpovedajúcou hodnote priehľadnosti daného mnohouholníku. Keďže takáto zložitá operácia pri vykresľovaní stoviek mnohouholníkov zaberala značný výpočtový čas, skúmali sme možnosti jeho zmenšenia. To nás viedlo k použitiu rýchlejšej funkcie fillConvexPoly pre vykresľovanie mnohouholníkov k čomu však bolo nutné overiť, že všetky vykresľované mnohouholníky budú konvexné (to je pri troch vrchoch zaručené vždy, avšak pri viacerých vrchoch môže byť niektorý z mnohouholníkov nekonvexný a vykreslí sa nevyplnený, neovplyvňuje to však funkčnosť algoritmu a ošetrenie tejto výnimky je jednoduché).

Popis programu

Evolučný algoritmus

Pre samotnú rekonštrukciu obrazu sme použili horolezecký algoritmus (Hill Climbing) fungujúci na podobnom princípe ako evolučný algoritmus, ktorý sme navrhli ako úplne prvý. Rozdiel spočíva v tom, že Hill Climbing počíta iba s jedným jedincom, teda generuje iba jeden obrázok a nemôže tak využívať metódy kríženia alebo elitárstvo. My sme však tento algoritmus rozšírili o niekoľko úprav zlepšujúcich jeho výsledky. Hill Climbing má gradientny charakter a má problémy s lokálnymi extrémami, preto jednou z týchto úprav je použitie metódy Simulated annealing. Tá pomáha zabráňovať zaseknutie algoritmu v lokálnych maximách (fitness) tým, že

umožňuje náhodné pridanie mnohouholníku aj v prípade, že sa tým nezlepší celková fitness generovaného obrázku (zhodu so zadaným obrázkom), avšak s rastúcim časom sa pravdepodobnosť výskytu takýchto náhodných pridaní znižuje úmerne celkovému počtu pridaných mnohouholníkov. Ďalšie vylepšenie bolo inšpirované myšlienkou o zbytočnosti mnohouholníkov, ktoré sú úplne prekryté inými mnohouholníkmi a majú tak zanedbateľný vplyv na výsledok. Táto modifikácia preto umožňuje odobratie mnohouholníku, ak sa takouto akciou nezmení celková fitness generovaného obrázku o vopred danú percentuálnu časť jej aktuálnej hodnoty. Spojením týchto úprav s pôvodným algoritmom Hill Climbingu dostávame použitý algoritmus, ktorý môžeme myšlienkovo vyjadriť nasledujúcim spôsobom:

```

VYBRÁNO = 0
GENERACE = 0
DOKUD (GENERACE < POČET GENERACÍ)
    VYGENERUJ POČÁTEČNÍ N-TICI MNOUÚHELNIKŮ
    SPOČÍTEJ FITNESS N-TICE – F1
    ZMUTUJ N-TICI
    SPOČÍTEJ FITNESS ZMUTOVANÉ N-TICE – F2
    JESTLI ( F2 < F1)
        NEBO (rand(0.0, 1.0) < K1 / VYBRANO)
        NEBO (BYL ZMAZÁN MNOHOÚHELNIK a |F1 –F2| < K2 * F2/ 100.0 ) )
            VYBER ZMUTOVANOU N-TICI
            VYBRÁNO++
    GENERACE++

```

Reprezentácia mnohouholníkov

Dôležitou vlastnosťou nášho programu je tiež univerzálnosť obsiahnutých funkcií vzhľadom k možnosti použitia dvoch typov mnohouholníkov ku generovaniu výsledného obrázku – mnohouholníky môžu byť definované karteziánsky alebo polárne (pomocou config súboru popísaného v ďalších kapitolách).

Použili sme teda dve možnosti reprezentácie mnohouholníkov v priestore. Prvá spočíva v náhodnom umiestnení N bodov (vrcholov), ktoré definujú samotný mnohouholník:

$$Poly_{cartesian} = \{(x_i, y_i)\}$$

$$x_i = rand(0, w), y_i = rand(0, h), i = 1, 2, \dots, N$$

kde w a h sú rozmery obrázku.

Druhá reprezentácia, ktorú sme použili spočíva v použití polárnych súradníc. Náhodne zvolíme N uhlov a N polomerov okolo počiatku súradníc, potom počiatok súradníc náhodne posunieme:

$$Poly_{polar} = \{(r_i, \varphi_i, x_0, y_0)\}$$

$$r_i = rand(0, Offset)$$

$$\varphi_i = rand(0, 2\pi)$$

$$x_0 = rand(Offset, w - Offset)$$

$$y_0 = rand(Offset, h - Offset)$$

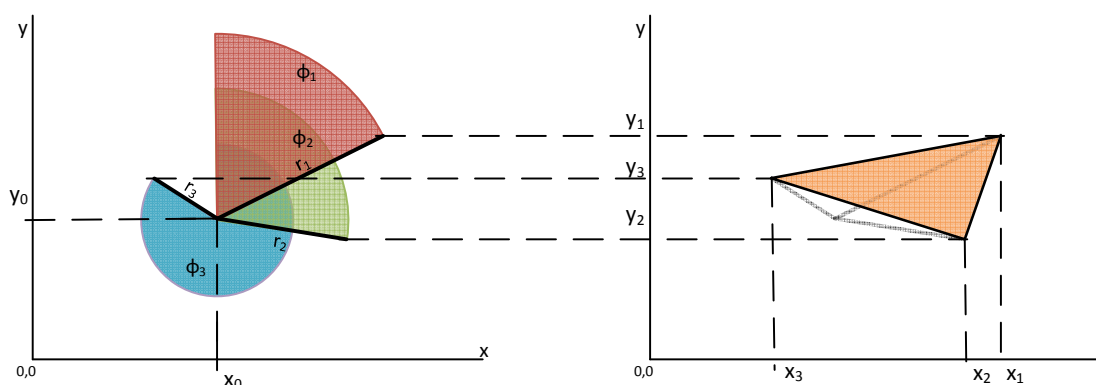
kde *Offset* určuje, ako ďaleko posúvame počiatok súradníc a zároveň rozmer mnohoúhelníkov. Aby parameter *Offset* bol intuitívny, prepočítavame ho z iného parametru *Scale*:

$$Offset = Scale \cdot \frac{\min(h, w)}{2}, Scale \in [0.0, 1.0]$$

Čím väčšia hodnota *Scale*, tým väčšie dostaneme mnohoúhelníky. Prepočítať jeden tvar na druhý môžeme nasledujúcim spôsobom:

$$x_i = r_i \cos(\varphi_i) + x_0$$

$$y_i = r_i \sin(\varphi_i) + y_0$$



Obr. 1 – Súvislosť medzi dvoma reprezentáciami mnohoúhelníkov

Ak pridáme do mnohoúhelníkov farbu a mieru priehľadnosti, potom usporiadaná n-tica mnohoúhelníkov definuje celý obrázok.

Ďalšie vlastnosti programu - fitness funkcie, mutácie, RGB/Grayscale

Pri vyhodnotení zhodnosti zadaného a generovaného obrázku používame tri metódy fitness funkcie – strednú kvadratickú odchýlku (MSE), špičkový pomer signálu k šumu (PSNR) alebo index štruktúrálnej podobnosti (SSIM), ktorý rešpektuje ľudské vnímanie scény. K mutácii využívame 6 rôznych spôsobov úpravy náhodného mnohoúhelníku – odstránenie mnohoúhelníku, posunutie mnohoúhelníku o náhodnú vzdialenosť, posunutie jedného bodu mnohoúhelníku o náhodnú vzdialenosť, náhodná zmeny farby mnohoúhelníku, náhodná zmena priehľadnosti mnohoúhelníku alebo prehodenie poradia vykresľovania dvoch mnohoúhelníkov.

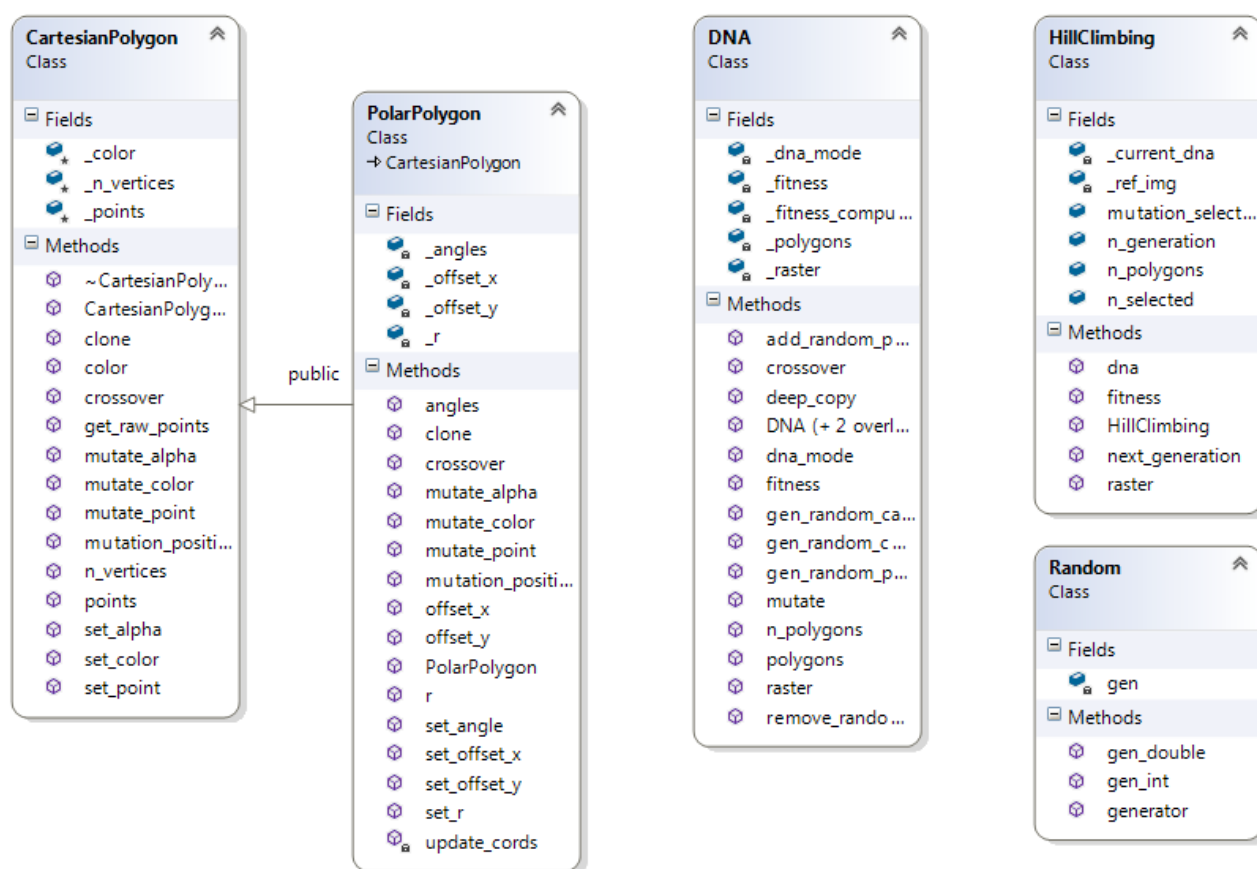
Rozšírenie programu o detekciu čiernobielych (Grayscale) zadaných obrázkov a snaha o zrýchlenie výpočtového času pri ich generovaní oproti farebným (RGB) obrázkom viedla k ďalšiemu zdvojeniu funkcií.

Vstupy a výstupy programu

Program umožňuje nastavenie množstva premenných (celkový zoznam popísaný nižšie) z externého súboru vo formáte JavaScript Object Notation (.json) nezávislom na počítačovej platforme. Neoddeliteľnou súčasťou nášho programu je tiež zaznamenávanie priebehu

rekonštrukcie zadaného obrazu v textovom formáte (.csv) so zaznamenaním priebehu fitness, počtu celkovo vybraných mnohoúhelníkov, aktuálneho počtu mnohoúhelníkov, počtu pridaných a odobraných mnohoúhelníkov i počtov použítí jednotlivých typov mutácií a v dvoch grafických formátoch – rastrovom (.png) s bezstratovou kompresiou a vektorovom (.svg) s možnosťou prezerania jednotlivých mnohoúhelníkov.

Popis hlavných tried a funkcií



Obr. 2 – Model znázorňujúci najdôležitejšie triedy použité v programe a ich súčasti

Polygon

`class CartesianPolygon` - karteziánsky mnohoúhelník

`protected:`

(Atribúty)

`_n_vertices` - počet vrcholov

`_color` - farba vo formáte BGRA

`_points` - vektor súradníc vrcholov

`public:`

(Metódy)

`CartesianPolygon` - konštruktor

`clone` - vytvorí hlbokú kópiu mnohoúhelníka

`n_vertices` - getter

`color` - getter

`points` - getter

`get_raw_points` - getter

```

    set_point - setter
    set_color - setter
    set_alpha - setter
    crossover - kríženie
    mutate_point - mutácia (posunutie vrcholu)
    mutation_position - mutácia (posunutie mnohouholníku)
    mutate_color - mutácia (zmena farby)
    mutate_alpha - mutácia (zmena priehľadnosti)
class PolarPolygon : public CartesianPolygon - polárny mnohouholník
private:
(Atribúty)
    _r - vrcholy v polárnych súradniciach
    _angles - vrcholy v polárnych súradniciach
    _offset_x - súradnice stredu mnohouholníku
    _offset_y - súradnice stredu mnohouholníku
(Metódy)
    update_cords - prepočet polárnych súradníc vrcholov na karteziánske
public:
    PolarPolygon - konštruktor
    Clone - vytvorí hlbokú kópiu mnohouholníka
    r - getter
    angles - getter
    offset_x - getter
    offset_y - getter
    set_r - setter
    set_angle - setter
    set_offset_x - setter
    set_offset_y - setter
    crossover - kríženie
    mutate_point - mutácia (posunutie vrcholu)
    mutation_position - mutácia (posunutie mnohouholníku)
    mutate_color - mutácia (zmena farby)
    mutate_alpha - mutácia (zmena priehľadnosti)
check_boundaries - overí, či sa vrchol nenachádza mimo obrazu

```

DNA

```

class DNA
private:
(Atribúty)
    _dna_mode - výber medzi polárnymi a karteziánskymi mnohouholníkmi
    _polygons; - vektor mnohouholníkov
    _raster - generovaný obraz
    _fitness - zhoda so zadaným obrazom
    _fitness_computed - informácia o aktuálnosti fitness
public:
(Metódy)
    deep_copy - hlboká kópia
    gen_random_color - generuje náhodnú farbu formátu BGRA
    gen_random_cartesian_polygon - generuje náhodný karteziánsky mnohouholník
    gen_random_polar_polygon - generuje náhodný polárny mnohouholník
    DNA - konštruktor
    add_random_polygon - pridá do DNA náhodný mnohouholník
    remove_random_polygon - odoberie z DNA náhodný mnohouholník
    n_polygons - getter
    dna_mode - getter
    polygons - getter
    fitness - výpočet zhody generovaného obrazu so zadaným
    raster - získa generovaný obrázok špecifikovaných rozmerov z mnohouholníkov v DNA
    crossover - kríženie
    mutate - mutácia

```

HillClimbing

class HillClimbing

private:

(Atribúty)

_current_dna - aktuálna DNA

_ref_img - zadany obraz

public:

n_generation - počet generácií

n_selected - počet vybraných mnohoúhelníkov

mutation_selected - počet mutácií

n_polygons - aktuálny počet mnohoúhelníkov

(Metódy)

HillClimbing - konštruktor

next_generation - nová generácia

fitness - výpočet zhody generovaného obrazu so zadným

raster - získa generovaný z mnohoúhelníkov v DNA

dna - getter

run_hill_climb

run_hill_climb - spustí algoritmus Hill Climbing

pretty_print - formátuje výstup záznamu

split - rozdeľuje reťazec záznamu na jednotlivé premenné

get_time_date_as_str - vracia aktuálny dátum a čas ako reťazec

polygons_to_svg - ukladá mnohoúhelníky aktuálnej DNA do súboru vo formáte .svg

hill_climb_time_meas

hill_climb_time_meas - spustí algoritmus Hill Climbing s meraním času

fitness

fitness - zvolí metódu výpočtu zhody generovaného obrazu so zadným

mean_square_err - výpočet fitness metódou strednej kvadratickej odchýlky

psnr - výpočet fitness metódou pomeru signálu k šumu

ssim - výpočet fitness metódou indexu štruktúrálnej podobnosti

Configs

load_configs - načíta premenné z .json súboru

draw_polygons

draw_polygons - vytvorí generovaný obraz z vektoru mnohoúhelníkov

Popis nastaviteľných konštánt (config súbor)

"DISPLAY":

- "DISPLAY_IMG_H" – výška zobrazovaného obrazu
- "DISPLAY_IMG_W" – šírka zobrazovaného obrazu
- "DISPLAY_EVERY_N_GEN" – zobrazovanie obrazu každú N-tú generáciu

"EVOLUTION":

- "ANNEALING_SIMULATION" – použitie Simulated annealing
- "ANNEALING_SIMULATION_RATE" – pravdepodobnosť Simulated annealing
- "MAX_N_POLYGONS" – maximu aktuálnych mnohoúhelníkov
- "N_GENERATIONS" – počet generácií
- "REMOVING_POLYGON" – použitie funkcie odstraňovania mnohoúhelníkov
- "REMOVING_POLYGON_TOLERANCE" – pravdepodobnosť odstránenia mnohoúhelníkov

"INTERNAL_RESOLUTION":

- "IMG_H" – skutočná výška generovaného obrazu
- "IMG_W" – skutočná šírka generovaného obrazu

"LOGGING_P":

- "LOGGING" – použitie funkcie záznamu
- "LOG_IMG_EVERY_N_GEN" – zaznamenávanie v grafickom formáte každých N gener.
- "LOG_TO_CSV_EVERY_N_GEN" – zaznamenávanie v textovom formáte každých N gener.

"METRIC":

- "FITNESS_MODE" – voľba metódy výpočtu fitness (MSE, PSNR, SSIM)
- "GAUSSIAN_SIGMA_X" – parameter použitého Gaussian rozostrenia pri výpočte SSIM

"MUTATIONS_PROBS":

- "ADD" – pravdepodobnosť mutácie (pridanie mnohoúhelníku)
- "MUTATE_ALPHA" – pravdepodobnosť mutácie (zmena priehľadnosti mnohoúhelníku)
- "MUTATE_COLOR" – pravdepodobnosť mutácie (zmena farby mnohoúhelníku)
- "MUTATE_POINT" – pravdepodobnosť mutácie (posunutie vrcholu mnohoúhelníku)
- "MUTATE_POS" – pravdepodobnosť mutácie (posunutie mnohoúhelníku)
- "REMOVE" – pravdepodobnosť mutácie (odstránenie mnohoúhelníku)
- "SWAP" – pravdepodobnosť mutácie (prehodenie poradia mnohoúhelníkov)

"POLYGON":

- "DNA_MODE" – voľba spôsobu reprezentácie mnohoúhelníkov (Cartesian, Polar)
- "N_VERTICES" – počet vrcholov mnohoúhelníkov
- "SCALE" - zväčšenie

Výsledky testovania

Defaultné nastavenie konštánt:

```
"DISPLAY":  
  "DISPLAY_IMG_H": 500  
  "DISPLAY_IMG_W": 500  
  "DISPLAY_EVERY_N_GEN": 500  
"EVOLUTION":  
  "ANNEALING_SIMULATION": true  
  "ANNEALING_SIMULATION_RATE": 0.001  
  "MAX_N_POLYGONS": 1000  
  "N_GENERATIONS": 0  
  "REMOVING_POLYGON": true  
  "REMOVING_POLYGON_TOLERANCE": 0.1  
"INTERNAL_RESOLUTION":  
  "IMG_H": 200  
  "IMG_W": 200  
"LOGGING_P":  
  "LOGGING": true  
  "LOG_IMG_EVERY_N_GEN": 3000  
  "LOG_TO_CSV_EVERY_N_GEN": 1000  
"METRIC":  
  "FITNESS_MODE": "MSE"  
  "GAUSSIAN_SIGMA_X": 1.5  
"MUTATIONS_PROBS":  
  "ADD": 0.143  
  "MUTATE_ALPHA": 0.143  
  "MUTATE_COLOR": 0.143  
  "MUTATE_POINT": 0.143  
  "MUTATE_POS": 0.143  
  "REMOVE": 0.143  
  "SWAP": 0.143  
"POLYGON":  
  "DNA_MODE": "Cartesian"  
  "N_VERTICES": 3  
  "SCALE": 0.2
```

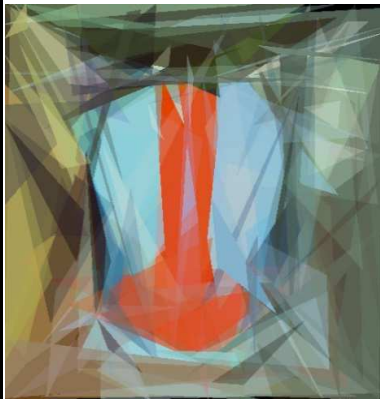
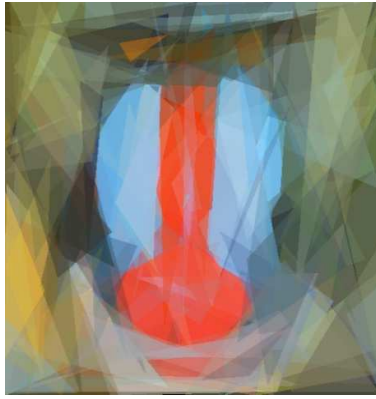
Porovnanie SSIM a MSE

Zmeny v nastavení konštánt u oboch testov:

"N_GENERATIONS": 300000

"ANNEALING_SIMULATION": false

"REMOVING_POLYGON": false

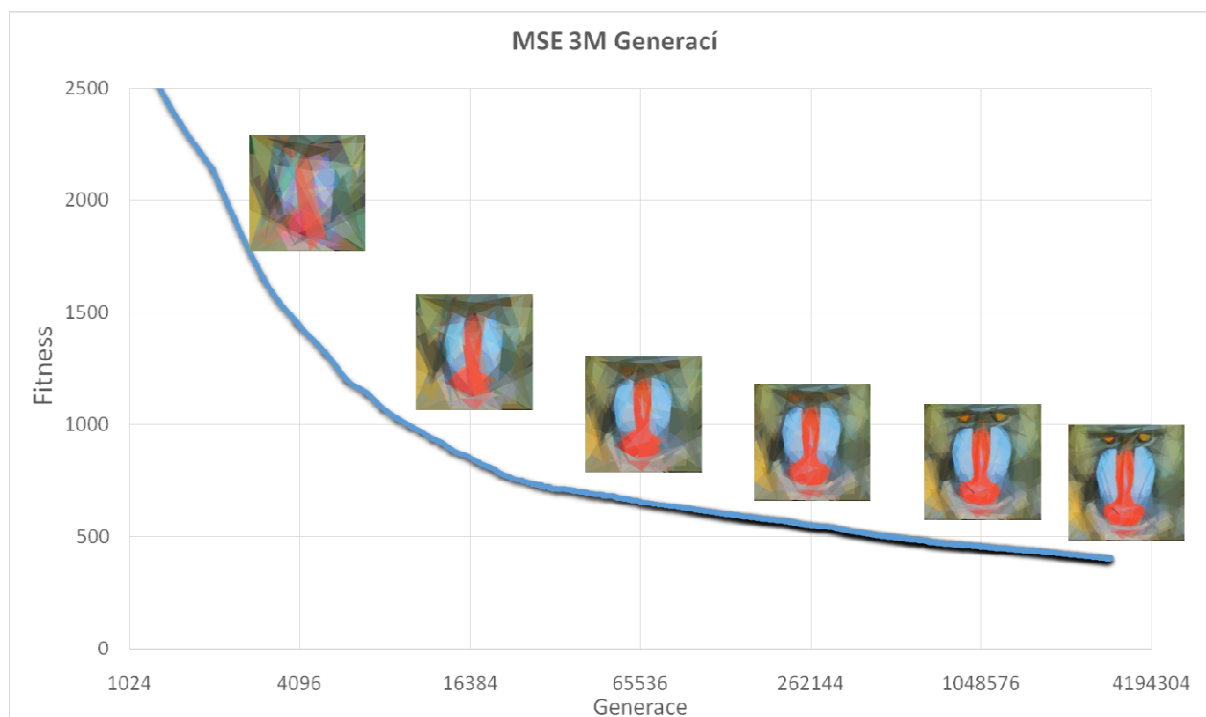
SSIM		MSE	
			
FITNESS	0,58	FITNESS	517
SELECTED	7303	SELECTED	8088
N POIYGONS	115	N POIYGONS	105
ADD	118	ADD	134
REMOVE	4	REMOVE	30
MUTATE_POS	185	MUTATE_POS	405
MUTATE_POINT	522	MUTATE_POINT	935
MUTATE_COLOR	3769	MUTATE_COLOR	3717
MUTATE_ALPHA	2350	MUTATE_ALPHA	2554
SWAP	355	SWAP	313

Vývoj hodnoty fitness funkcie MSE

Zmeny v nastavení konštánt:

"N_GENERATIONS": 3000000

"REMOVING_POLYGON_TOLERANCE": 0.000001



Overenie vplyvu pridaných modifikácií Hill Climbingu

Zmeny v nastavení konštánt pri malom vplyve modifikácií:

"N_GENERATIONS": 1500000

"REMOVING_POLYGON_TOLERANCE": 0.000001

Zmeny v nastavení konštánt pri väčšom vplyve modifikácií:

"N_GENERATIONS": 1500000

"ANNEALING_SIMULATION_RATE": 0.05

"REMOVING_POLYGON_TOLERANCE": 0.005

"ADD": 0.25

"MUTATE_ALPHA": 0.1

"MUTATE_COLOR": 0.1

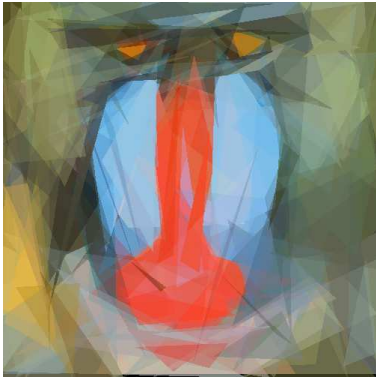

"MUTATE_POINT": 0.1

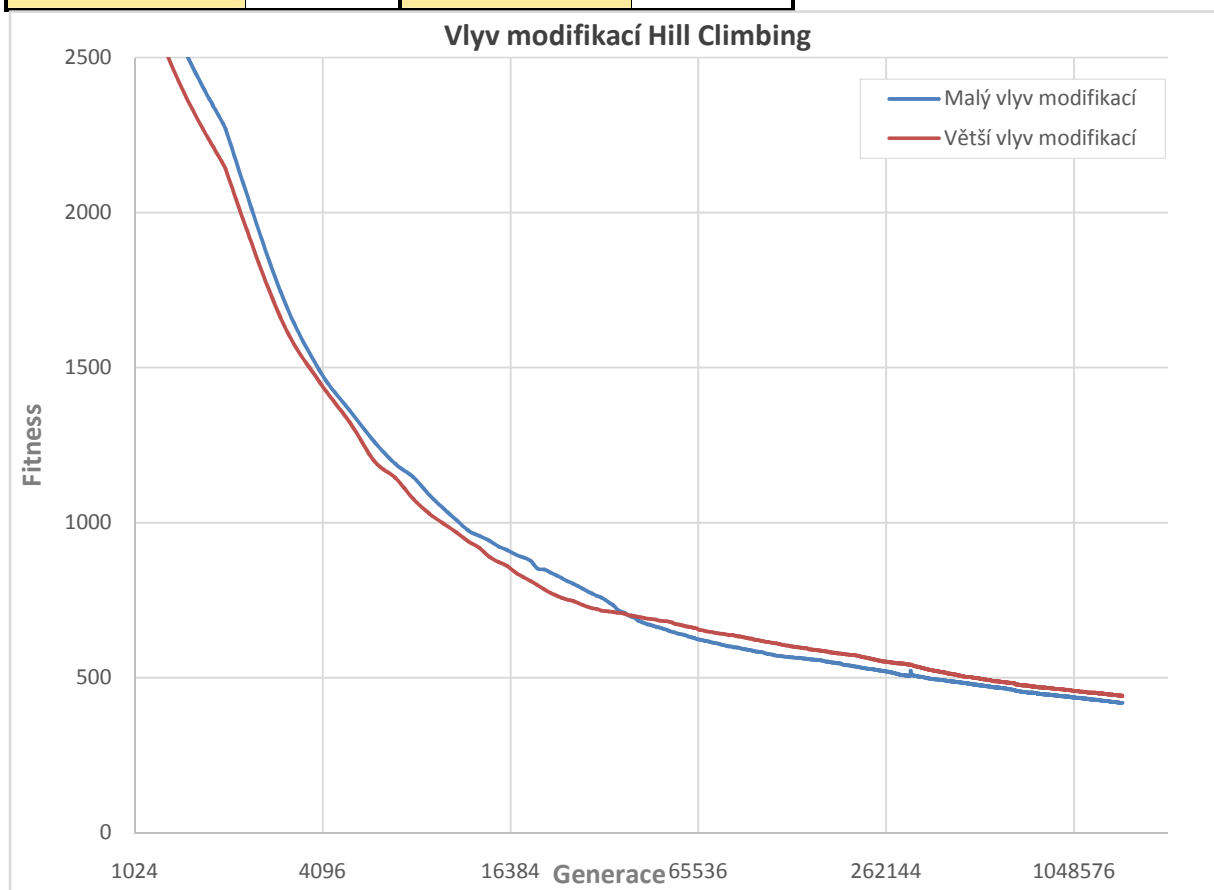
"MUTATE_POS": 0.1

"REMOVE": 0.25

"SWAP": 0.1

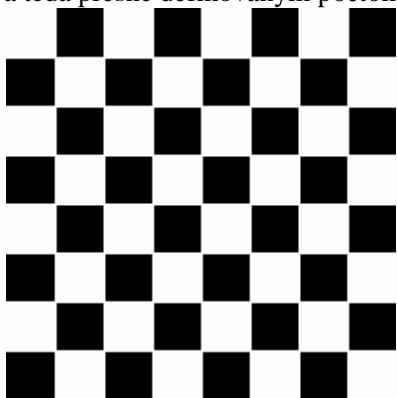
"SCALE": 0.3

Malý vplyv		Väčší vplyv	
			
FITNESS	442	FITNESS	419
SELECTED	12741	SELECTED	14191
N POLYGONS	129	N POLYGONS	178
ADD	145	ADD	252
REMOVE	17	REMOVE	75
MUTATE_POS	460	MUTATE_POS	506
MUTATE_POINT	1309	MUTATE_POINT	1267
MUTATE_COLOR	6387	MUTATE_COLOR	6942
MUTATE_ALPHA	4029	MUTATE_ALPHA	4566
SWAP	394	SWAP	583



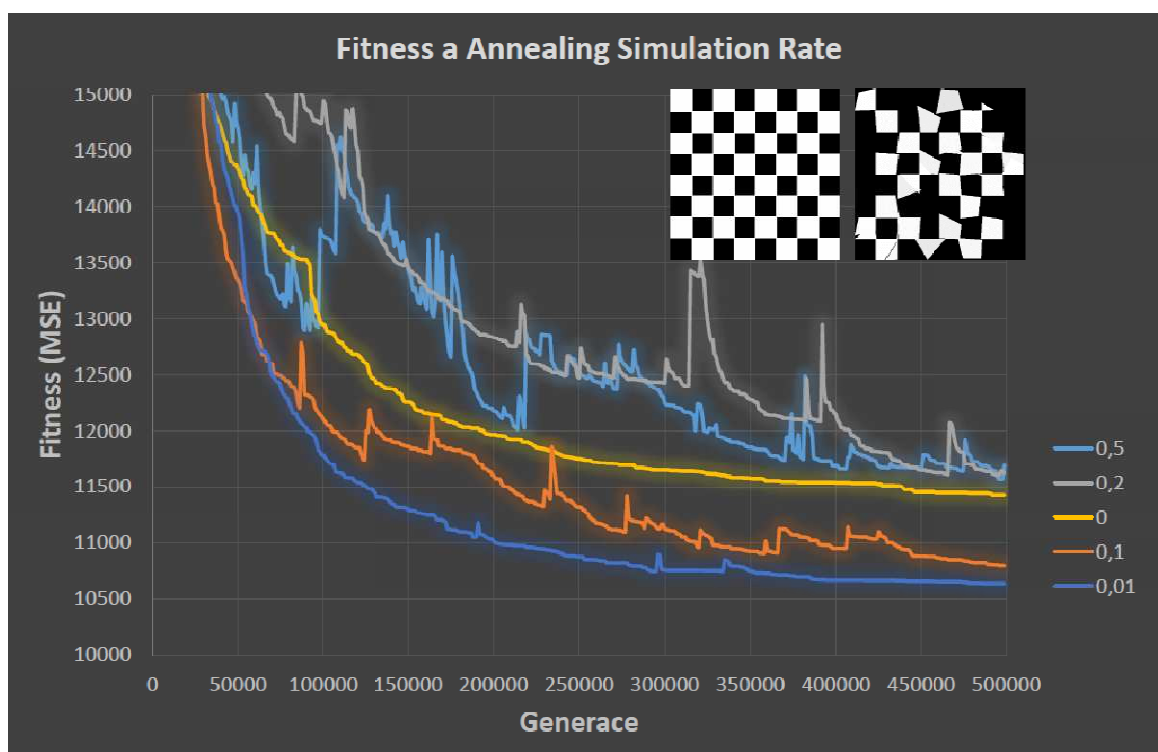
Overenie vplyvu pridaných modifikácií Hill Climbingu

Pre lepšiu prehľadnosť sme si rozšírili množinu testovacích obrazov o ďalší s šachovnicovým vzorom a teda presne definovaným počtom mnohouholníkov:



Zmeny v nastavení konštánt pri malom vplyve modifikácií:

```
"N_GENERATIONS": 500000  
"MAX_N_POLYGONS": 32  
"ANNEALING_SIMULATION_RATE": (0 až 0.5)  
"REMOVING_POLYGON": false  
"ADD": 0.05  
"MUTATE_ALPHA": 0.05  
"MUTATE_COLOR": 0.05  
"MUTATE_POINT": 0.4  
"MUTATE_POS": 0.4  
"REMOVE": 0.05  
"SWAP": 0  
"DNA_MODE": "Polar"  
"N_VERTICES": 4
```



Záver

V rámci tohto projektu sa nám úspešne podarilo navrhnuť program realizujúci rekonštrukciu obrazu s využitím evolučného algoritmu, pričom sme využili programovacieho jazyka C++ a knižnicu OpenCV. Náš prvý návrh tohto programu využíval jednoduché genetické algoritmy spočívajúce v prekladaní mnohouholníkov, veľkú množinu jedincov (obrázkov), 2 druhy selekcie, elitárstvo, jednoduché kríženie a mutácie. Výsledky tohto algoritmu však neboli uspokojujúce, preto sme sa po konzultácii s vedúcim projektu rozhodli prejsť z tohto návrhu na program fungujúci na báze Hill Climbingu.

Po prekonaní úvodných problémov s použitím priehľadných objektov v kombinácii s využívaním knižnice OpenCV a zrýchlení najpomalšej časti algoritmu – vykresľovania mnohouholníkov, sme sa zamerali na vylepšenie samotného algoritmu Hill Climbing. Implementovali sme modifikácie zabraňujúce zasekávaniu algoritmu v lokálnych maximách fitness funkcie (Simulated annealing) a znižujúce počet mnohouholníkov, čím sme urýchlili konvergenciu k zadanému obrazu.

Program (viz Obr. 2) automaticky rozlišuje farebné a čiernobiele zadané obrazy na vstupe a odpovedajúco upravuje algoritmus v snahe o zmenšenie potrebného výpočtového času. Program tiež umožňuje použitie dvoch reprezentácií mnohouholníkov (viz Obr. 1), troch typov fitness funkcie, 6 druhov mutácií, nastavenie veľkosti generovaného obrazu alebo zmenu rozlíšenia. Tieto a množstvo ďalších parametrov je možné nastaviť pomocou externého súboru vo formáte .json. Dôležitou súčasťou nášho programu je tiež zaznamenávanie aktuálnych hodnôt zvolených premenných počas behu programu v textovom formáte .csv a aktuálneho stavu rekonštruovaného obrazu v dvoch grafických formátoch .png a .svg.

Na záver, ako aj počas vývoja programu, sme uskutočnili veľké množstvo testov. Najzaujímavejšie výsledky týchto testov, ako napríklad porovnanie výsledkov pri rovnakom počte generácií a použití rôznych fitness funkcií, ich priebeh pri väčšom počte generácií alebo vplyv modifikácií algoritmu Hill Climbing, sme zobrazili graficky (viz kapitola Výsledky testovania).

Použitá literatúra

- [1] Honzík, P.: *Strojové učení*, elektronická skripta VUT Brno, 2006.
- [2] Kubalík, J.: *Evolutionary Algorithms: Introduction*, elektronická skripta CTU Prague, 2015.
- [3] Genetic Programming: Evolution of Mona Lisa. *Roger Johansson Blog* [online]. Sweden: WordPress.com, 2008 [cit. 2016-05-03]. Dostupné z: <https://rogersaling.com/2008/12/07/genetic-programming-evolution-of-mona-lisa/>
- [4] *OpenCV* [online]. San Francisco: Itseez, 2016 [cit. 2016-05-03]. Dostupné z: <http://opencv.org/>

Prílohy

Príloha 1 – Niektoré výsledky testov

Test 1

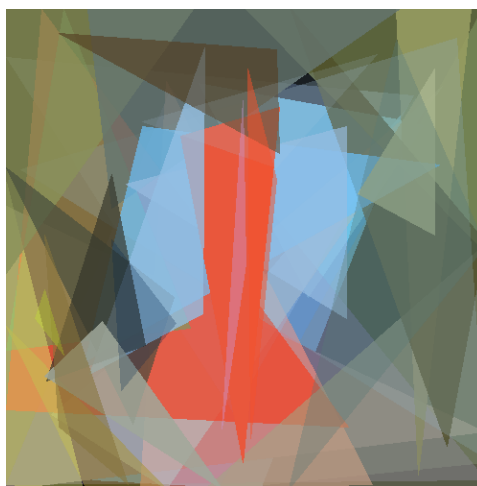
Zmeny v nastavení konštánt:

"N_GENERATIONS": 300000

"REMOVING_POLYGON_TOLERANCE": 0.5

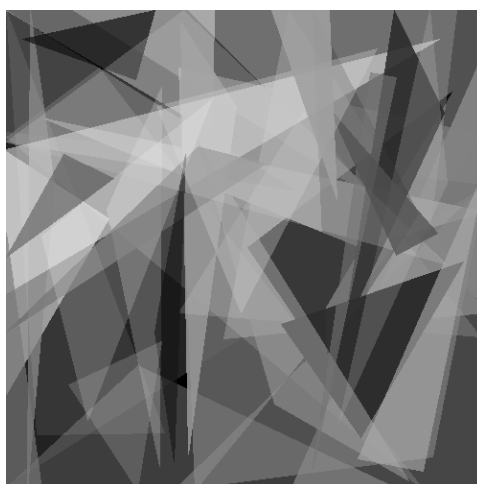
EXAMPLE 1

FITNESS	749.502
SELECTED	4139
POLYGONS	34
ADD	249
REMOVE	216
MUTATE_POS	219
MUTATE_POINT	522
MUTATE_COLOR	1645
MUTATE_ALPHA	1184
SWAP	104



EXAMPLE 2

FITNESS	731.445
SELECTED	4618
POLYGONS	38
ADD	543
REMOVE	506
MUTATE_POS	217
MUTATE_POINT	536
MUTATE_COLOR	942
MUTATE_ALPHA	1639
SWAP	235



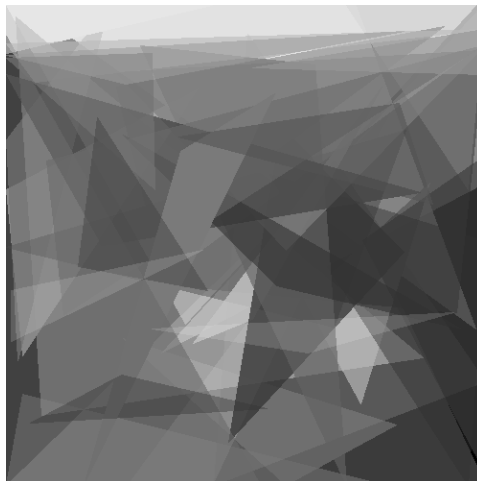
EXAMPLE 3

FITNESS	657.618
SELECTED	3831
POLYGONS	30
ADD	263
REMOVE	234
MUTATE_POS	236
MUTATE_POINT	602
MUTATE_COLOR	1395
MUTATE_ALPHA	989
SWAP	112



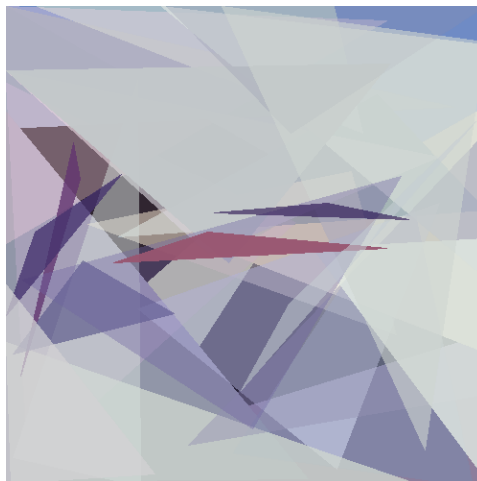
EXAMPLE 4

FITNESS	428.305
SELECTED	3880
POLYGONS	37
ADD	526
REMOVE	490
MUTATE_POS	221
MUTATE_POINT	502
MUTATE_COLOR	734
MUTATE_ALPHA	1274
SWAP	133



EXAMPLE 5

FITNESS	666.902
SELECTED	4051
POLYGONS	31
ADD	218
REMOVE	188
MUTATE_POS	234
MUTATE_POINT	543
MUTATE_COLOR	1559
MUTATE_ALPHA	1181
SWAP	128



Test 2

Zmeny v nastavení konstant:

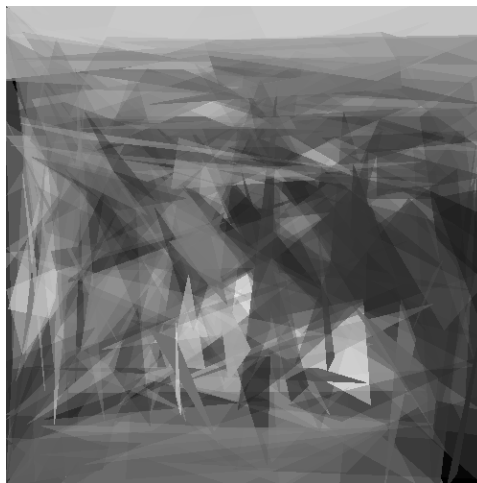
"N_GENERATIONS": 300000

"REMOVING_POLYGON_TOLERANCE": $1e-06$

"FITNESS_MODE": "SSIM"

EXAMPLE 2

FITNESS	0.373629
SELECTED	6164
POLYGONS	142
ADD	156
REMOVE	15
MUTATE_POS	244
MUTATE_POINT	761
MUTATE_COLOR	1675
MUTATE_ALPHA	2950
SWAP	363



EXAMPLE 3

FITNESS	0.467977
SELECTED	3091
POLYGONS	42
ADD	47
REMOVE	6
MUTATE_POS	133
MUTATE_POINT	421
MUTATE_COLOR	1470
MUTATE_ALPHA	851
SWAP	163



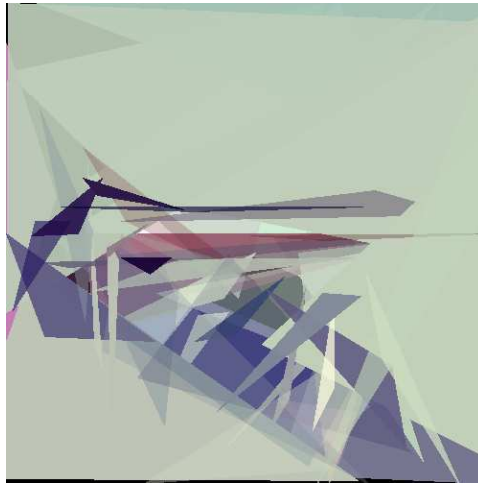
EXAMPLE 4

FITNESS	0.385789
SELECTED	6030
POLYGONS	181
ADD	197
REMOVE	17
MUTATE_POS	264
MUTATE_POINT	726
MUTATE_COLOR	1569
MUTATE_ALPHA	2849
SWAP	408



EXAMPLE 5

FITNESS	0.361632
SELECTED	4854
POLYGONS	48
ADD	55
REMOVE	8
MUTATE_POS	155
MUTATE_POINT	411
MUTATE_COLOR	2398
MUTATE_ALPHA	1654
SWAP	173



Test 3

Zmeny v nastavení konštánt:

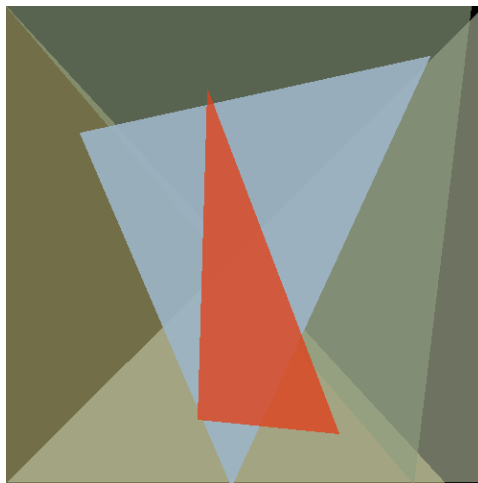
"N_GENERATIONS": 300000

"REMOVING_POLYGON_TOLERANCE": 0.5

"FITNESS_MODE": "SSIM"

EXAMPLE 1

FITNESS	0.756031
SELECTED	2988
POLYGONS	5
ADD	650
REMOVE	646
MUTATE_POS	73
MUTATE_POINT	171
MUTATE_COLOR	791
MUTATE_ALPHA	616
SWAP	41



EXAMPLE 2

FITNESS	0.630096
SELECTED	3846
POLYGONS	9
ADD	984
REMOVE	976
MUTATE_POS	144
MUTATE_POINT	317
MUTATE_COLOR	517
MUTATE_ALPHA	802
SWAP	106



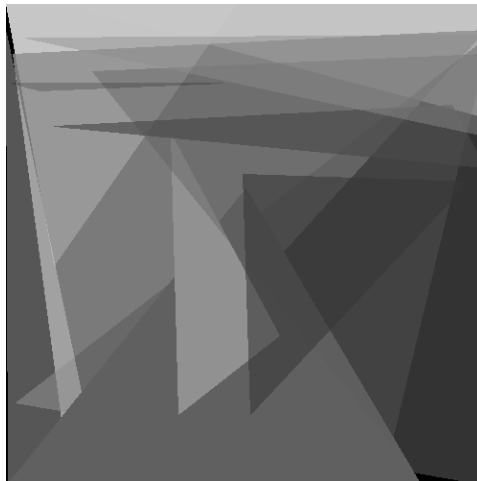
EXAMPLE 3

FITNESS	0.532368
SELECTED	1104
POLYGONS	6
ADD	102
REMOVE	97
MUTATE_POS	52
MUTATE_POINT	144
MUTATE_COLOR	385
MUTATE_ALPHA	285
SWAP	39



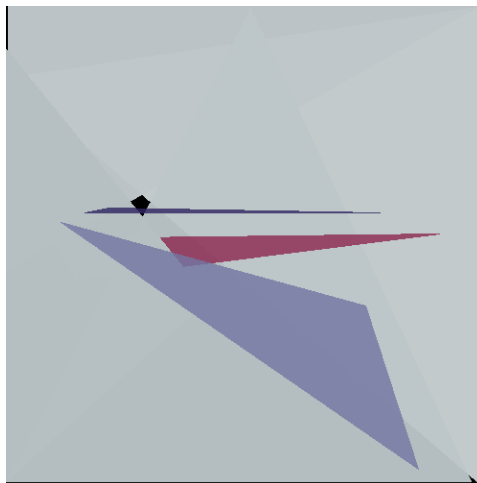
EXAMPLE 4

FITNESS	0.590839
SELECTED	5313
POLYGONS	12
ADD	1308
REMOVE	1297
MUTATE_POS	170
MUTATE_POINT	379
MUTATE_COLOR	852
MUTATE_ALPHA	1184
SWAP	123



EXAMPLE 5

FITNESS	0.444435
SELECTED	1683
POLYGONS	7
ADD	240
REMOVE	234
MUTATE_POS	57
MUTATE_POINT	154
MUTATE_COLOR	533
MUTATE_ALPHA	412
SWAP	53



Test 4

Zmeny v nastavení konštánt:

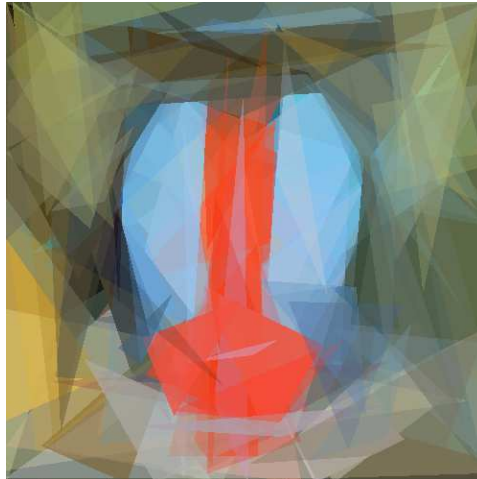
"N_GENERATIONS": 300000

"REMOVING_POLYGON_TOLERANCE": $1e-06$

"FITNESS_MODE": "MSE"

EXAMPLE 1

FITNESS	535.851
SELECTED	6670
POLYGONS	93
ADD	107
REMOVE	15
MUTATE_POS	340
MUTATE_POINT	812
MUTATE_COLOR	3075
MUTATE_ALPHA	2119
SWAP	202



...a mnoho dalších testů.