

Programová podpora výuky předmětu Počítačová grafika

Kamil Stokláška



Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2013/2014

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Kamil STOKLÁSKA**

Osobní číslo: **A10803**

Studijní program: **B3902 Inženýrská informatika**

Studijní obor: **Informační a řídicí technologie**

Forma studia: **prezenční**

Téma práce: **Programová podpora výuky předmětu Počítačová grafika**

Zásady pro vypracování:

- 1. Seznamte se s algoritmy vyplňování rastrových a vektorových oblastí. V práci je blíže specifikujte a shrňte jejich vlastnosti.**
- 2. Seznamte se s algoritmy anti-aliasingu. V práci je blíže specifikujte a shrňte jejich vlastnosti.**
- 3. Seznamte se s jednorozměrným a dvourozměrným prokládacím schématem u grafických formátů GIF a PNG. V práci je blíže specifikujte a shrňte jejich vlastnosti.**
- 4. Navrhněte a realizujte programy, které budou implementovat výše zmíněné algoritmy a budou provádět jejich názornou vizualizaci.**
- 5. Koncepti těchto programů vytvořte tak, aby mohly být použity ve výuce předmětu Počítačová grafika.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. Jiří ŽÁRA, Bedřich BENEŠ, Petr FELKEL a Jiří SOCHOR. Moderní počítačová grafika. Vyd 2. Brno: Computer Press, 2005, 608 s. EAN 9788025104545.
2. MARTIŠEK, Dalibor. Matematické principy grafických systémů. Vyd. 1. Brno: Litera, 2002, 278 s. ISBN 80-857-6319-2.
3. FOLEY, J. Computer Graphics. Boston, 1997, 1175 s. ISBN 02-018-4840-6
4. JURAJ, Štugel. Netgraphics [online]. [cit. 2014-02-05]. Dostupné z: <http://pg.netgraphics.sk/>
5. ADOBE SYSTEMS INCORPORATED. ActionScript 3.0 Reference for the Adobe Flash Platform [online]. [cit. 2014-01-24]. Dostupné z: http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/

Vedoucí bakalářské práce:

Ing. Pavel Pokorný, Ph.D.

Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce:

28. února 2014

Termín odevzdání bakalářské práce:

13. června 2014

Ve Zlíně dne 28. února 2014



prof. Ing. Vladimír Vašek, CSc.
děkan



prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu


Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně 11. 6. 2014


.....
podpis diplomanta

ABSTRAKT

Tato bakalářská práce se zabývá algoritmy pro vyplňování vektorových a rastrových oblastí, potlačení efektu aliasingu a pro jednorozměrné a dvourozměrné prokládací schéma u grafických formátů GIF a PNG. V teoretické části jsou tyto algoritmy obecně shrnuty a blíže specifikovány vybrané z nich. Obsahem praktické části je zhotovení programů v prostředí Flash, které implementují vybrané algoritmy a provádí jejich názornou vizualizaci. Tyto programy jsou exportovány v podobě appletů, které jsou přehledně uspořádány do struktury webové stránky. Vytvořené programy budou dále sloužit jako podpora pro výuku předmětu Počítačová grafika.

Klíčová slova: počítačová grafika, vyplňování oblastí, anti-aliasing, prokládací schéma, Flash, ActionScript

ABSTRACT

This Bachelor thesis is focused on algorithms for filling vector and raster areas, alias effect suppression, and for one and two-dimensional interlacing scheme in PNG and GIF graphic formats. In the theoretical part, these algorithms are generally summarized and few of them are further specified. Practical part is represented by a programs created in Flash environment which implements the selected algorithms and make their illustrative visualization. These programs are exported as applets that are arranged in a structure of website. Created programs will be used as a software support of Computer Graphics tuition.

Keywords: Compute graphics, filling areas, anti-aliasing, interlacing scheme, Flash, ActionScript

Tímto bych chtěl poděkovat vedoucímu bakalářské práce Ing. Pavlu Pokornému, Ph.D za odborný dohled, ochotu, spolupráci a čas, který mi po dobu realizace této práce věnoval.

Děkuji rodině za podporu a motivaci po celou dobu psaní této práce.

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST.....	10
1 VYPLŇOVÁNÍ OBLASTÍ.....	11
1.1 VYPLŇOVÁNÍ GEOMETRICKY URČENÉ HRANICE	11
1.1.1 Metoda řádkového rozkladu.....	12
1.1.2 Vyplňování trojúhelníků	13
1.1.3 Šrafování	14
1.1.4 Vyplňování vzorem.....	15
1.1.5 Inverzní vyplňování	16
1.2 VYPLŇOVÁNÍ HRANICE NAKRESLENÉ V RASTRU.....	17
1.2.1 Semínkové vyplňování.....	18
1.2.2 Řádkové semínkové vyplňování	19
1.2.3 Vyplňování vzorem.....	20
2 ANTI-ALIASING.....	21
2.1 PRAVIDELNÉ VZORKOVÁNÍ S VYŠŠÍ FREKVENCÍ	21
2.1.1 Vzorkování s otočenou mřížkou	22
2.1.2 Quincunx	23
2.2 STOCHASTICKÉ VZORKOVÁNÍ.....	24
2.2.1 Poisson disc.....	24
2.2.2 Roztřesení.....	24
2.2.3 Metoda N-věží.....	25
3 PROKLÁDÁNÍ.....	26
3.1 PROKLÁDÁNÍ U GRAFICKÉHO FORMÁTU GIF	26
3.2 PROKLÁDÁNÍ U GRAFICKÉHO FORMÁTU PNG	27
4 TECHNOLOGIE FLASH	29
4.1 ADOBE FLASH PROFESSIONAL.....	29
4.1.1 Grafické nástroje	30
4.1.2 Nástroje pro animaci	31
4.1.3 Nástroje pro psaní kódu	32
4.2 ACTIONSCRIPT	33
4.2.1 Objekty pro reprezentaci obsahu.....	33
II PRAKTICKÁ ČÁST	35
5 VYTVOŘENÉ PROGRAMY	36
5.1 STRUKTURA PROGRAMŮ.....	36
5.2 APLIKACE VYPLŇOVÁNÍ OBLASTÍ	36
5.2.1 Popis vytvořených tříd	37
5.2.2 Uživatelské rozhraní.....	39
5.2.3 Ovládání	40
5.3 APLIKACE ANTI-ALIASING	41
5.3.1 Popis vytvořených tříd	41
5.3.2 Uživatelské rozhraní.....	43
5.3.3 Ovládání	43

5.4	APLIKACE PROKLÁDÁNÍ.....	44
5.4.1	Popis vytvořených tříd	44
5.4.2	Uživatelské rozhraní.....	45
5.4.3	Ovládání	46
6	WEBOVÁ PREZENTACE	47
6.1	STRUKTURA WEBU	47
	ZÁVĚR	49
	SEZNAM POUŽITÉ LITERATURY	50
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	52
	SEZNAM OBRÁZKŮ	53
	SEZNAM PŘÍLOH.....	54

ÚVOD

Počítačová grafika je jednou z nejrychleji se rozvíjejících oblastí informačních technologií. Při práci s počítačovou grafikou se řeší hned několik problémů současně, které spadají do různých vědních disciplín. Jedná se zejména o metody pro získávání, reprezentaci a zobrazování dat.

Podle způsobu reprezentace obrazových informací a jejich uchovávání dělíme počítačovou grafiku na rastrovou a vektorovou. Zatímco v rastrové grafice je obraz popsán pomocí jednotlivých barevných bodů, vektorový obraz je definován základními geometrickými útvary, z kterých se skládá. Obě varianty využívají zcela odlišných metod pro nejrozličnější grafické operace, s kterými se uživatel při práci s počítačem běžně setkává, aniž by o nich tušil.

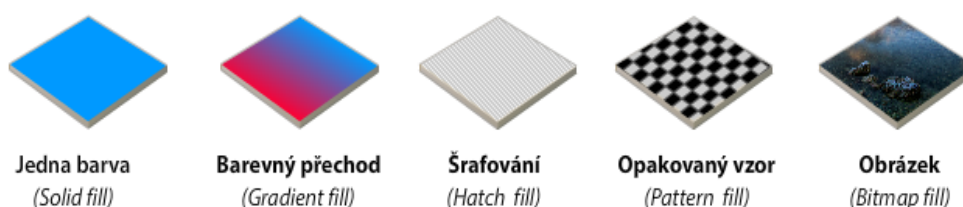
Cílem této práce je podrobně popsat základní principy vybraných algoritmů pro vyplňování oblastí, potlačení aliasingu a zobrazování dat s využitím prokládacího schématu, které byly následně využity při tvorbě interaktivních aplikací pro jejich názornou vizualizaci. Všechny aplikace jsou navrženy co nejpřehledněji tak, aby bylo možné z implementace vybrané techniky co nejlépe dané problematice porozumět, a aplikace se tak mohly stát součástí výuky předmětu Počítačová grafika.

I. TEORETICKÁ ČÁST

1 VYPLŇOVÁNÍ OBLASTÍ

Oblast je uzavřená, rovinná plocha, definovaná popisem její hranice a metodou vyplňování vnitřních bodů. V počítačové grafice se jedná o obecný prvek, který je využíván pro zobrazení libovolného plošného obrazce. Tvar oblasti je přitom určen její hranicí, která vzájemně odděluje jednu oblast od druhé. Tato hranice může být popsána dvěma základními způsoby – prvním je geometrické vyjádření (viz kapitola 1.1), kde je hranice nejčastěji určena posloupností bodů mnohoúhelníku, případně posloupností geometricky popsaných primitiv (křivky, přímky, úsečky). Druhou možností je hranice nakreslená v rastru (viz kapitola 1.2), tedy hranice určená jednotlivými obrazovými body. Při takto definované hranici je třeba znát její barvu, případně barvu bodů uvnitř, nebo vně oblasti. Před vyplňováním je také v tomto případě nutné zadat souřadnice libovolného vnitřního bodu.

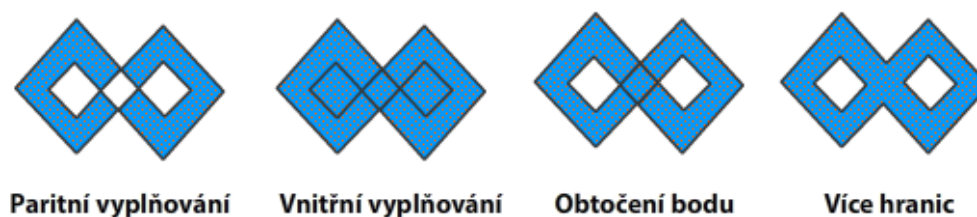
Vyplňováním oblasti se rozumí především nalezení její hranice, respektive nalezení vnitřních bodů a definice jejich barev – tedy přesné vymezení plochy této oblasti. Ve většině případů je samotná definice barev jednotlivých bodů oblasti algoritmicky mnohem jednodušší, než nalezení její hranice. Oblast může být vyplněna jednou barvou, barevným přechodem, šrafováním, opakováním zadaného vzoru, nebo rastrovým obrázkem.[1]



Obr. 1 Varianty barevných výplní

1.1 Vyplňování geometricky určené hranice

Nejčastějším případem geometricky určené hranice je mnohoúhelník, který je definován posloupností bodů, určující vrcholy polygonu. Při takto definované hranici často dochází k jejímu křížení, kdy uvnitř jedné oblasti může vzniknout oblast nová. K řešení této situace lze využít několik typů algoritmu – algoritmus využívající vnitřní vyplňování, metodu obtočení bodu, vyplňování více hranic a nejčastěji používané paritní vyplňování, jehož varianty jsou v této kapitole dále popsány.[2]



Obr. 2 Možnosti řešení výplně při křížení hranic

1.1.1 Metoda řádkového rozkladu

Jedná se o základní metodu pro paritní vyplňování geometricky určené hranice, kdy každým řádkem hranice je vedena vodorovná přímka, jejíž průsečíky s hranicí tvoří po dvojicích koncové body úseček. Tyto úsečky pak představují výplň oblasti.

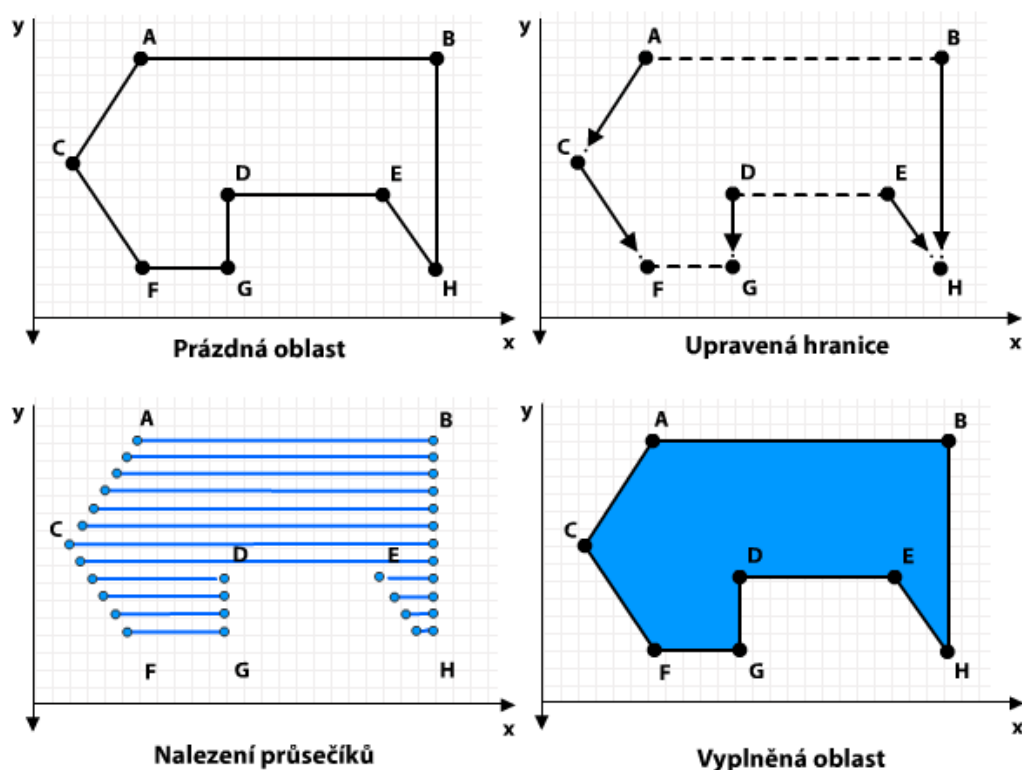
Před samotným vyplňováním je nejprve třeba hranici upravit pomyslným odstraněním všech vodorovných úseček, které pro algoritmus v daném řádku představují nekonečně mnoho průsečíků s vedenou přímkou. Poté je třeba veškeré hrany oblasti zkrátit o 1 pixel (obrazový bod) ve směru osy Y. Touto úpravou docílíme zdánlivého rozpojení hranice, které zajistí na každém řádku sudý počet průsečíků s vedenou přímkou. Algoritmus dále postupuje od nejvyššího vrcholu vyplňované oblasti po nejnižší s krokem -1. Na každém řádku nalezne průsečíky s vedenou přímkou, které seřadí podle souřadnice osy X a na konci každého kroku spojí sudé a liché průsečíky, čímž se vytvoří výplň daného úseku. Z tohoto důvodu je také nutné dodržet podmínku sudosti průsečíků. [3]

Při hledání průsečíků se vychází z parametrického vyjádření přímky

$$x = x_1 + t \cdot (x_2 - x_1), \quad (1)$$

$$y = y_1 + t \cdot (y_2 - y_1), \quad (2)$$

kde x_1, y_1 jsou souřadnice počátečního bodu a x_2, y_2 jsou souřadnice koncového bodu.

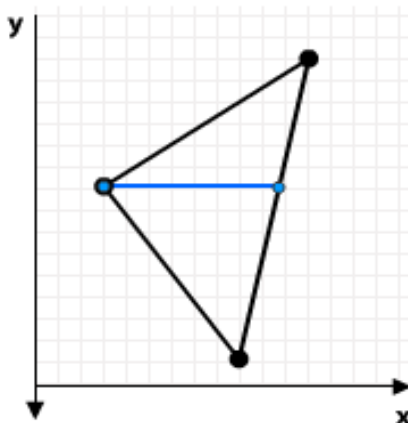


Obr. 3 Zjednodušený průběh algoritmu řádkového rozkladu

Pro zrychlení metody lze v praxi uplatit tzv. seznam aktivních hran, který představuje tabulka, do níž se ukládají právě aktivní hrany rozkladového řádku. Při přechodu na další řádek se vychází z této tabulky a část výpočtů pro nalezení a seřazení průsečíků je tak ušetřena. Na každém řádku je dále provedena aktualizace seznamu, při které jsou dále neaktivní hrany z tabulky odstraněny a případně přidány hrany nové. [1]

1.1.2 Vyplňování trojúhelníků

Při vyplňování trojúhelníků lze využít speciální modifikace metody řádkového rozkladu, kdy se namísto obecného algoritmu pro vyplňování polygonu využije zjednodušený postup. Trojúhelník se rozdělí vodorovným řezem, který je veden jeho prostředním vrcholem ve směru osy Y (viz obr. 4). Touto úpravou se trojúhelník rozdělí na dva jednoduché subtrojúhelníky, které mají společnou vodorovnou hranu. Tuto hranu je možné v algoritmu řádkového rozkladu vynechat a souběžně postupovat po dvou zbylých hranách, mezi kterými se vykreslí spojnice. Tento postup je oproti klasické metodě řádkového rozkladu mnohem efektivnější a je vhodný především v trojrozměrné grafice, kde jsou plochy reprezentovány právě navazujícími trojúhelníky. [1]



Obr. 4 Rozdělení trojúhelníku pro modifikaci algoritmu

1.1.3 Šrafování

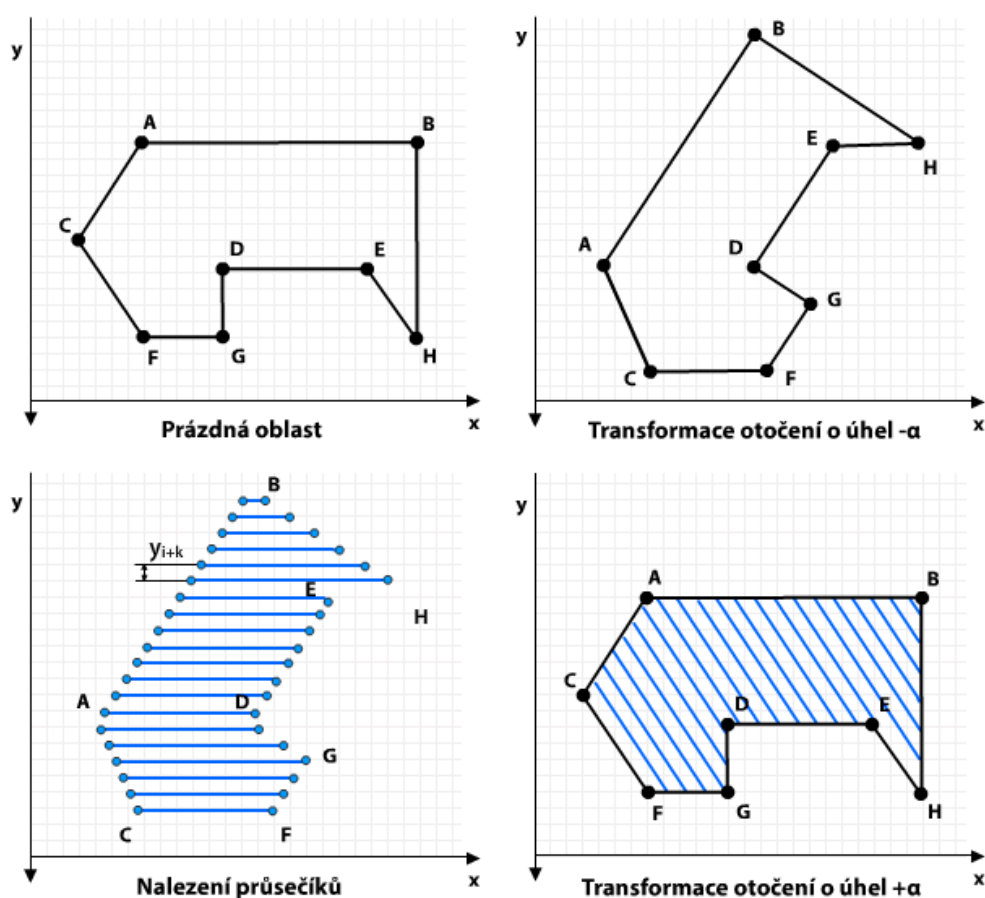
Jedná se o velmi jednoduché rozšíření algoritmu řádkového vyplňování, kdy v základě stačí upravit krok souřadnice Y. Tímto postupem lze docílit nevyplněné oblasti mezi jednotlivými kroky. Volbou kroku je možné jednoduše nastavit šířku nevyplněné oblasti, tedy šířku samotného šrafování. U šrafování pod úhlem se nejprve na celou hranici aplikuje transformace otočení, po které je možné využít stejný postup jako u šrafování vodorovného. Před vykreslením jednotlivých vnitřních úseček se na každou z nich aplikuje opět transformace otočení, která zajistí požadovaný sklon. [4]

Transformace otočení se aplikuje vztahy

$$x' = x \cdot \cos \alpha - y \cdot \sin \alpha, \quad (3)$$

$$y' = x \cdot \sin \alpha + y \cdot \cos \alpha, \quad (4)$$

kde x, y jsou souřadnice otáčeného bodu a α je úhel otočení.

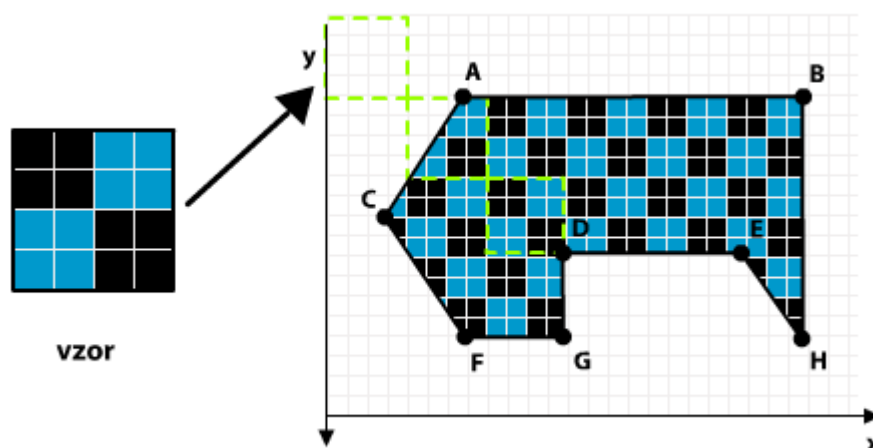


Obr. 5 Zjednodušený průběh algoritmu při vyplňování šrafováním

1.1.4 Vyplňování vzorem

Pro vyplňování vzorem lze využít zobecněný algoritmus řádkového vyplňování. Při vyplňování vzorem je nutné definovat barvu každého pixelu nalezené úsečky mezi dvěma průsečíky hranice zvlášť. Vzor je popsán maticí o rozměrech $m \times n$, kde jednotlivé prvky matice představují barvu pixelu na dané pozici. Pro výpočet pozice pixelu v matici se využívá operace celočíselného dělení. Celočíselný zbytek po vydělení souřadnice daného pixelu a rozměru matice vzoru představuje pozici uvnitř této matice, která určí barevnou hodnotu pixelu. Tímto postupem se barevný vzor opakuje uvnitř celé oblasti. Stejný postup se využívá i při vyplňování obrázkem (texturou).

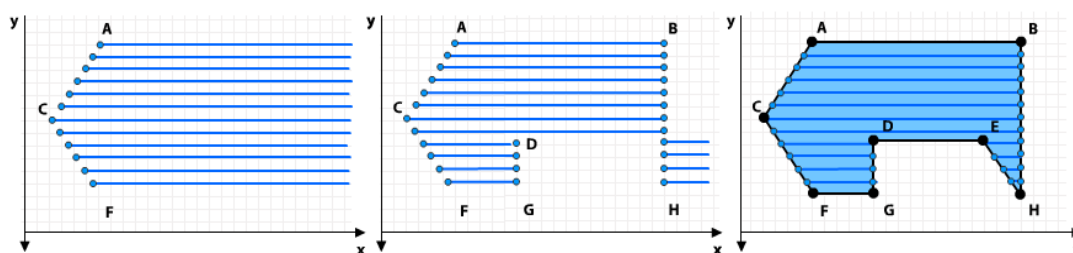
V některých případech se do algoritmu zadává také vztažný bod, který představuje počáteční posunutí v matici vzoru. Díky tomu je možné ošetřit, aby vzor začínal vždy spolu s hranicí oblasti. [1]



Obr. 6 Aplikace vzorové výplně

1.1.5 Inverzní vyplňování

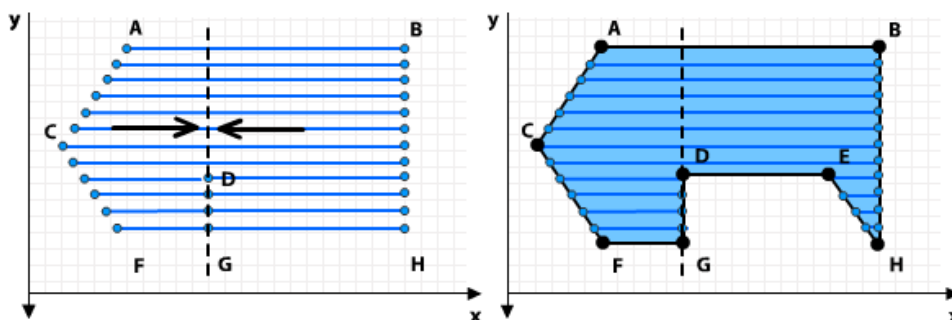
Inverzní vyplňování využívá efektu funkce XOR, který při prvním aplikování invertuje barvu daného pixelu a při druhém aplikování ji vrátí do původní hodnoty. Obecný algoritmus vychází z původní metody řádkového rozkladu. Postupně prochází všechny hraniční úsečky ve směru osy Y a na každém řádku, kterým daná úsečka prochází, algoritmus invertuje pixely ležící vpravo od nalezeného průsečíku. Pixely, které budou invertovány dvakrát, se vrátí na svou původní hodnotu a zůstanou invertovány pouze pixely ležící vně oblasti. [1]



Obr. 7 Zjednodušený průběh algoritmu inverzního vyplňování

Efektivnější modifikací této varianty je plotové inverzní vyplňování - ve zvoleném vrcholu hranice, který je co nejblíže středu oblasti, se vytvoří vertikální přímka, tzv. plot. Algoritmus poté postupuje obdobně jako u obecné varianty, ale při inverzi zohledňuje pouze ob-

last, která leží mezi průsečíkem hranice na daném řádku a plotem. Tím se výrazně zmenší invertovaný prostor a celý algoritmus se značně urychlí. [5]



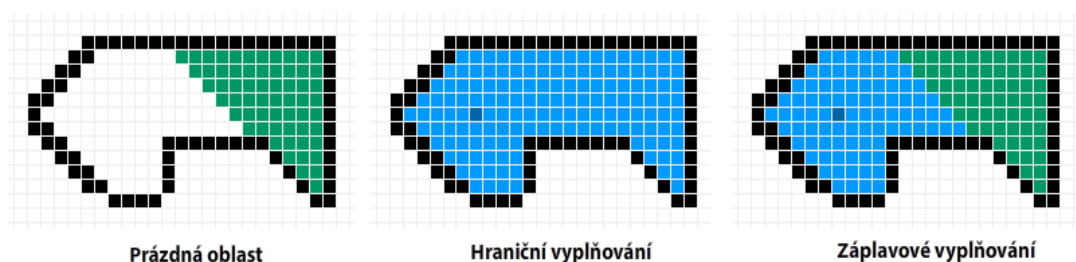
Obr. 8 Plotové inverzní vyplňování

Pro inverzní vyplňování vzorem (šrafováním, texturou...) je nutné využít šablonu. Šablona je část paměti, ve které se během vyplňování provádí jednotlivé operace. Velikost šablony musí být stejně velká jako vyplňovaná oblast, proto bývá tato metoda velmi často náročná na paměť. Každý pixel je v šabloně reprezentován jedním bitem, který reprezentuje stav vyplnění (1 - vyplněno/ 0 - nevyplněno). Algoritmus postupuje stejně jako v předešlém případě, ale invertují se pouze údaje v šabloně. Na konci algoritmu jsou tedy všechny pixely uvnitř vyplňované oblasti reprezentovány log. 1. Pozice údaje v šabloně se shoduje s pozicí daného pixelu a je možné jej vyplnit libovolnou barvou. Pro zrychlení celého postupu je možné vyplněné části oblasti zaznamenat do šablony pouze jejich koncovými body. Tím je zajištěn minimální přístup do šablony. Šablona se díky své efektivitě využívá v grafických procesorech a akcelerátorech, kde má stejný rozměr jako obrazovka a slouží jako univerzální pomocná paměť.

1.2 Vyplňování hranice nakreslené v rastru

Hranice nakreslená v rastru je oproti vektorové hranici definovaná jednotlivými obrazovými body stejné barvy, které spolu sousedí. Při vyplňování rastrové oblasti se testuje každý bod zvlášť a všechny potřebné informace jsou získávány přímo z obrazové paměti. Podle způsobu testování jednotlivých bodů lze rastrové vyplňování rozdělit do dvou základních variant – hraniční a záplavové. U hraničního vyplňování se určí barva hranice a testovaný bod patří do oblasti pouze tehdy, je-li jeho barva odlišná. U záplavového vyplňová-

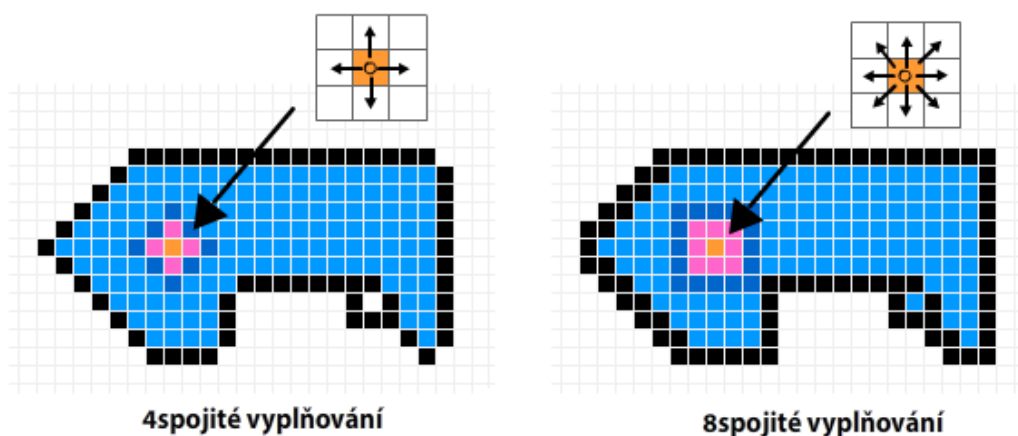
ní se naopak definuje počáteční barva vnitřního bodu oblasti a testovaný bod do oblasti patří, pokud se jeho barva s původním bodem shoduje. [5]



Obr. 9 Varianty vyplňování hranice nakreslené v rastru

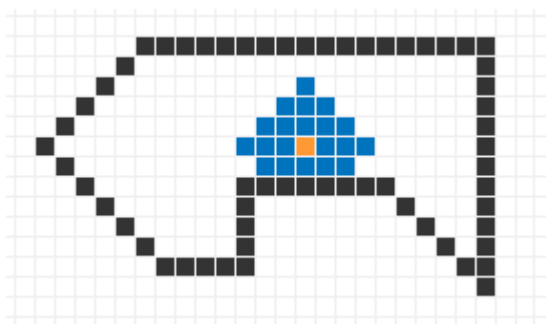
1.2.1 Semínkové vyplňování

Pod tento typ vyplňování spadají obecně všechny algoritmy pro vyplňování rastrových oblastí. Před začátkem vyplňování je nutné znát alespoň jeden vnitřní bod (tzv. semínko). Tento bod značí počátek, od kterého se budou postupně testovat další (sousední) body. U každého bodu se poté vždy testuje, zda leží v dané oblasti a zda nebyl již vyplněn dříve. Směr prohledávání oblasti je dán strukturou její hranice, na základě které rozlišujeme vyplňování 4spojité a 8spojité. U 4spojitého vyplňování se za sousední body považují pouze body sousedící ve vodorovném a svislém směru, zatímco u 8spojitého vyplňování se uvažují i obě diagonály. Pro vyplňování 8spojité hranice je vhodné využít 4spojité vyplňování a naopak. V opačném případě by mohlo dojít k proniknutí semínka mimo oblast, respektive nevyplnění celé oblasti.



Obr. 10 Typy semínkového vyplňování

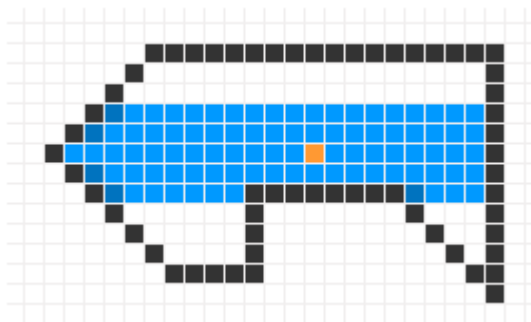
Základní variantou semínkového vyplňování je rekurzivní algoritmus, jehož vstupním parametrem je testovaný bod. Pokud tento bod spadá do oblasti a ještě nebyl vybarven, vybarví se a jeho sousední body jsou vstupními body pro další úroveň rekurze. Tento algoritmus je však i přes svou jednoduchost značně neefektivní a má vysoké nároky na paměť. V praxi se proto využívají nerekurzivní modifikace tohoto algoritmu. [6]



Obr. 11 Základní varianta semínkového vyplňování

1.2.2 Řádkové semínkové vyplňování

Tato modifikace semínkového vyplňování minimalizuje počet přístupů do obrazové paměti a tím se stává mnohem efektivnější a rychlejší. Algoritmus si v průběhu vyplňování ukládá veškeré informace o semínkách na zásobník. Tato informace je tvořena souřadnicemi daného semínka a směry, kterými je možné pokračovat. Pro každý bod v zásobníku se vypočítají koncové body řádku, na kterém se daný bod nachází. Celý řádek je poté vyplněn najednou. Do každého prázdného úseku nad i pod aktuálně vyplněným řádkem je zasazeno nové semínko, které je opět uloženo na zásobník. [6]

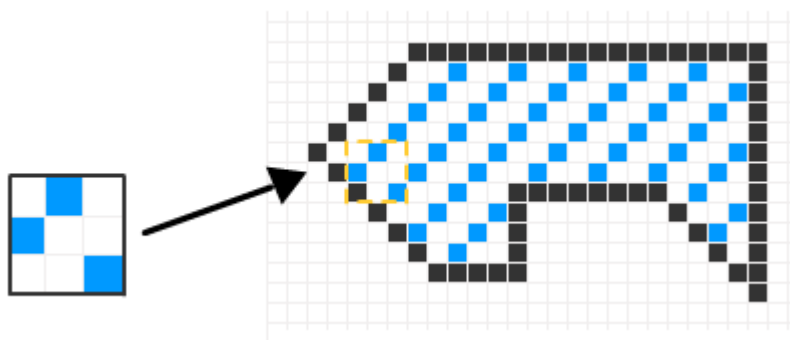


Obr. 12 Řádkové semínkové vyplňování

1.2.3 Vyplňování vzorem

Při vyplňování rastrové oblasti vzorem se postupuje obdobně jako při vyplňování oblasti určené geometricky. Vzor je popsán maticí o rozměrech $m \times n$, kde jednotlivé prvky matice představují barvu pixelu. Při řádkovém semínkovém vyplňování je v tomto případě nutné určit barvu každého bodu na řádku zvlášť. V tomto případě však může dojít k situaci, kdy bude mít bod vzoru stejnou barvu jako prázdná oblast. Takový bod by pak i po vyplnění mohl být považován za nevyplněný a došlo by k zacyklení algoritmu. K ošetření této situace lze využít šablonu, uvnitř které je zaznamenán stav vyplnění daného bodu.

Tento postup lze využít i pro šrafování, které je možné napodobit vzorem. [1]

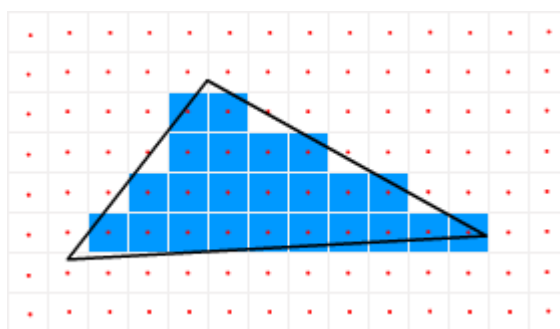


Obr. 13 Šrafování pomocí rastrového vzoru

2 ANTI-ALIASING

Anti-aliasing je metoda vedoucí k potlačení aliasingu. Ten vzniká při přechodu signálu do diskrétního prostředí a projevuje se jako nová, nežádoucí informace. V počítačové grafice se alias vyskytuje například při vzorkování textur, nebo při rasterizaci vektorového objektu, jehož hrana může procházet i částí pixelu. K vykreslení objektu je však v rastrovém zařízení možné využít pouze celé pixely a efekt aliasingu se tak může projevit v podobě ozubené hrany na vykreslovaném objektu, či dalších vad.

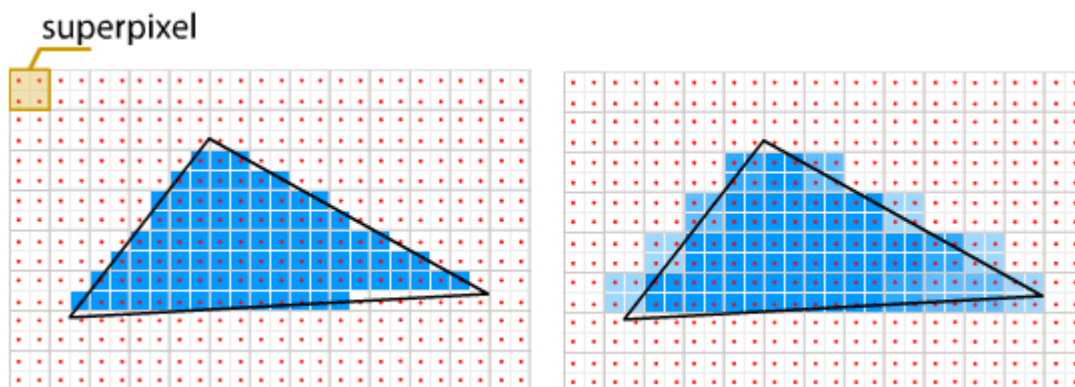
Právě efekt ozubených hran (jaggies) je v počítačové grafice jeden z nejčastějších projevů aliasingu. Při rasterizaci se každý pixel vzorkuje ve svém středu a na základě barvy tohoto vzorku se vybarví celý pixel. Mezi další, velmi časté projevy aliasingu v počítačové grafice, patří například efekt pohybujících se hran při změně úhlu objektu (crawling), nebo efekt přerušených čar, kdy při vzorkování velmi tenkých objektů může vlivem aliasingu dojít k jejich rozpadu, či úplnému zániknutí (unconnected lines, flickering). Typicky se v počítačové grafice problém aliasingu řeší zvýšením vzorkovací frekvence, nebo převedením aliasu na šum. [7]



Obr. 14 Alias vzniklý při rasterizaci (jaggies)

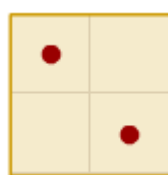
2.1 Pravidelné vzorkování s vyšší frekvencí

Jedná se o plošnou metodu, aplikovanou na celou obrazovou strukturu. Celý obraz se nejprve převzorkuje podle předem určeného měřítka. Každý pixel v obrazu je tak nahrazen $n \times n$ subpixely, které spolu tvoří jeden superpixel. V praxi to znamená, že je nutné obraz vyrenderovat v n -násobném rozlišení, abychom získali potřebný počet vzorků. Tento postup se také nazývá jako supersampling. Barva daného superpixelu je pak určena jako průměr barev všech jeho subpixelů. Po převzorkování obrazu zpátky do původního rozlišení se pixely s novým odstínem projeví jako rozmazání a efekt aliasu je tak potlačen. Tato metoda projevy aliasu neodstraní, pouze je potlačí a přesměruje do vyšších frekvencí. [1]



Obr. 15 Pravidelné převzorkování s dvojnásobnou frekvencí

Kvůli nutnosti vzorkovat obraz v několikanásobně vyšším rozlišení má však tato metoda velmi vysoké nároky na výkon a v praxi se využívají méně náročné modifikace. Zásadní optimalizací celého procesu je zmenšení počtu subpixelů, ke kterému je možné využít mnoho různých způsobů. Přesnějším vzorkováním pouze ve směru jedné z os lze zredukovat počet pixelů na polovinu a tím algoritmus znatelně urychlit. Tímto zásahem však nelze dosáhnout zdaleka tak dobrých výsledků jako u použití původního algoritmu. Poměrně dobrých výsledků je možné dosáhnout rozmístěním subpixelů diagonálně tak, aby v každém řádku i sloupci zůstal pouze jeden subpixel. Takto lze optimalizovat vzorkování s libovolnou frekvencí a snížit tak jeho nároky při výpočtu výsledné barvy pixelu na polovinu při méně znatelném zhoršení kvality anti-aliasingu. [8]

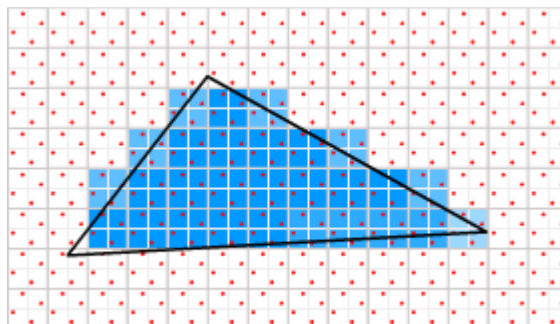


Obr. 16 Diagonální redukce subpixelů

2.1.1 Vzorkování s otočenou mřížkou

U pravidelného vzorkování jsou pozice vzorků v každém superpixelu rozloženy v pravidelné mřížce tak, aby vzorek ležel vždy ve středu subpixelu. Nevýhodou tohoto uspořádání je však zhoršená kvalita vyhlazování u hran s výrazným horizontálním, či vertikálním sklonem. Účinného vyhlazení v těchto kritických úhlech je možné dosáhnout pootočením vzorkovací mřížky, které vede zároveň k potlačení flickeringu (mizení tenkých objektů).

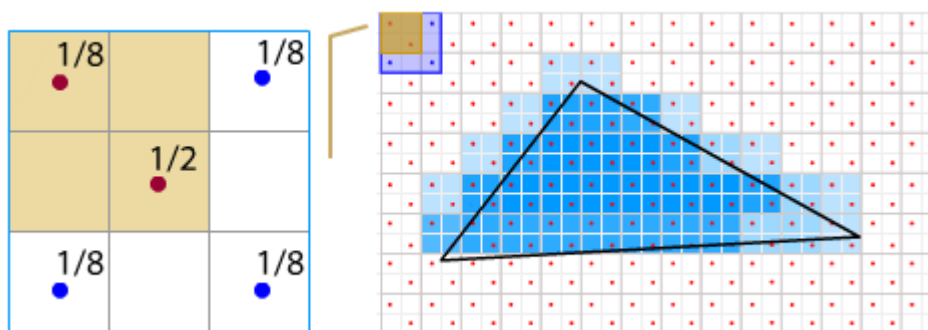
V případech, kdy je zapotřebí vzorek umístit mimo střed pixelu, se využívá slučovací paměti, ve které je možné načítat obrazy s mírným posunutím a provádět jednoduché výpočetní operace. [9]



Obr. 17 Vzorkování s otočenou mřížkou

2.1.2 Quincunx

Technika Quincunx zohledňuje při výpočtu odstínu výsledného pixelu nejen jeho vlastní subpixely, ale bere v úvahu i okrajové subpixely okolních bodů, které se mezi pixely sdílejí. Pokud je tedy obraz převzorkován s dvojnásobným rozlišením, pro výpočet pixelu je využito pět vzorků, přičemž tři vzorky z okolí jsou znovu využity i při výpočtu sousedních bodů. Pro výpočet výsledné barvy pixelu má vzorek v jeho středu váhový koeficient $1/2$ a ostatní vzorky v rozích pixelu pak $1/8$. Touto metodou lze při vzorkování ve dvojnásobném rozlišení dosáhnout přibližně stejné kvality, jako u standardního výpočtu při vzorkování v rozlišení čtyřnásobném. Díky těmto výsledkům se technika Quincunx nazývá také jako anti-aliasing s vysokým rozlišením (HRAA). Tuto metodu využívají některé grafické čipy od společnosti nVidia. [10]



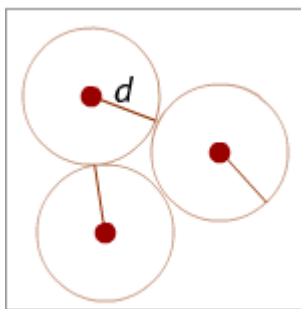
Obr. 18 Vzorkování technologií Quincunx

2.2 Stochastické vzorkování

U stochastického vzorkování není na rozdíl od pravidelného konstantní vzdálenost mezi jednotlivými vzorky. Ty jsou náhodně rozmístěny a působí tak jako uměle přidaný šum. Tato metoda vychází z vnímání lidského oka. Alias je převeden na šum, které lidské oko, na rozdíl od pravidelné chyby, za chybu nepovažuje. Oproti pravidelnému vzorkování je tak tento typ anti-aliasingu efektnější, nicméně v některých případech může být náročnější na výpočet. U tohoto typu vzorkování je důležité, aby byl výsledný efekt šumu co nejméně rušivý. Z tohoto důvodu existuje několik variant algoritmu, které upravují pseudonáhodné rozložení jednotlivých vzorků.

2.2.1 Poisson disc

Rozmístění jednotlivých vzorků u tohoto typu algoritmu vychází z Poissonova rozložení, tedy rozložení s minimální vzdáleností mezi dvěma vzorky. Nejprve je definována minimální vzdálenost mezi dvěma vzorky, kterou je nutné dodržet. Při každé vygenerované pozici nového vzorku je pak otestováno, zda se v této vzdálenosti od vzorku nenachází již vzorek jiný. V případě, že se v okolí nového vzorku již nějaký vzorek nachází, je vygenerována nová pozice pro nový vzorek a cyklus se opakuje. Při tomto postupu však může nastat situace, kdy špatně zvolená minimální vzdálenost způsobí zacyklení algoritmu. Vzhledem k výpočetní náročnosti při testování každého vzorku by pak řešení podobné situace bylo velmi obtížné. Z tohoto důvodu se obvykle využívají tabulky, do kterých se uloží předem vypočítané pozice vzorků. Ty jsou poté volány náhodně. [1]

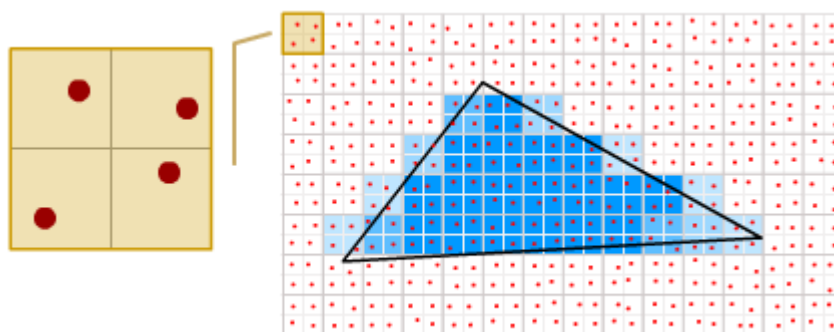


Obr. 19 Rozmístění vzorků s využitím Poisson disc

2.2.2 Roztřesení

Jedná se o nejčastěji používaný typ stochastického anti-aliasingu, který je používán společně se supersamplingem. Algoritmus postupuje stejně jako u pravidelného vzorkování, kde jsou vzorky umístěny do středu každého pixelu. U roztřesení jsou však tyto vzorky

vychýleny ze své původní polohy přičtením náhodného vektoru. Při tomto kroku musí být zaručeno, že pozice vzorku neopustí prostor daného subpixelu a zůstane tak zachováno rovnoměrné zastoupení každé části pixelu. Náhodné posunutí jednotlivých vzorků se na výstupu projeví jako šum, který je lidským okem vnímán lépe, než pravidelný vzor. V některých případech jsou pravidla pro posun vzorku specificky upravovány tak, aby bylo dosaženo co nejlepších výsledků pro odstranění konkrétního typu aliasu. [9]

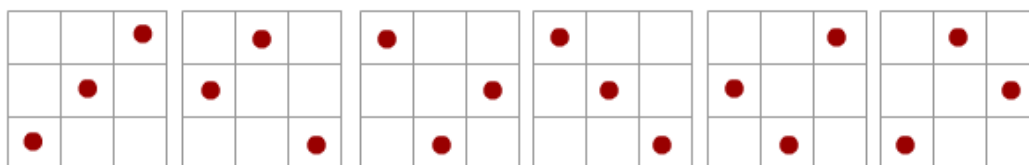


Obr. 20 Náhodné roztřesení vzorků (jittering)

2.2.3 Metoda N-věží

Alternativou k náhodnému generování, či posunu vzorků jsou algoritmy, které využívají předem definovaných vzorkovacích schémat. Tyto schémata definují přesné rozložení vzorků pro daný pixel, které je možné aplikovat jako filtr. Pomocí generátoru náhodných čísel je pak pro každý pixel vybráno náhodné schéma.

Metoda N-věží vychází z úlohy rozmístění n věží na šachovnici tak, aby se vzájemně neohrožovaly. Ve výsledku mají všechna schémata v každém řádku i sloupci zastoupen jen jeden vzorek a počet všech takto definovaných možností je roven permutaci n . [1]



Obr. 21 Tří věžová vzorkovací schémata

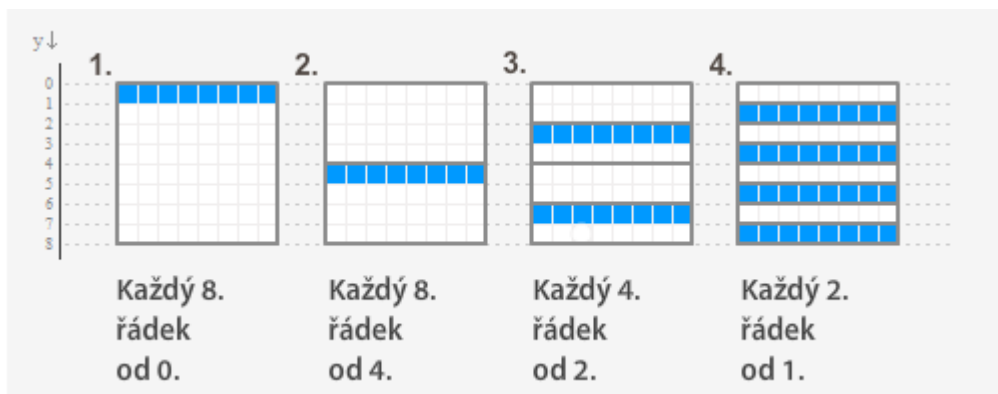
3 PROKLÁDÁNÍ

Prokládání je způsob kódování grafického obrazu tak, aby jej bylo možné rozpoznat ještě před úplným načtením všech dat. Při načítání prokládaného obrazu lze nejprve pozorovat zhoršenou kopii původního obrazu, která se s velikostí načtených dat zaostřuje do výsledné kvality. Díky tomu lze pozorovat informaci v obraze i při částečném načtení a v některých případech pak není nutné načítat obraz celý. Tento způsob kódování byl v minulosti využíván kvůli pomalým komunikačním metodám především v oblasti webového designu. V současnosti jsou výhody prokládání kompenzovány rychlým standardním připojením, které uživateli umožní zobrazit obrázek velice rychle a prokládání se tak stává značně neefektivní. Z tohoto důvodu se dnes využívá již jen velmi zřídka.

3.1 Prokládání u grafického formátu GIF

GIF (Graphics Interchange Format) je rozšířený grafický formát s využitím komprese LZW. Formát GIF slouží pro ukládání rastrových obrazů s podporou nejvýše 256 barev. Mimo ukládání samostatných obrázků je možné tento formát využít také pro animaci, kdy se v časových intervalech zobrazují jednotlivé obrazy za sebou. V případě animace je možné pro každý obraz využít samostatnou barevnou paletu, opět však s maximálním rozsahem pouze 256 barev. Výhodou tohoto formátu je podpora bitového alpha kanálu a možnost využít prokládání.

Prokládání je u formátu GIF realizováno jako jednorozměrné (řádkové). Algoritmus ukládá rastrový obraz po řádcích s rozestupy, které se postupně zmenšují. Při zobrazování je nejprve vykreslen každý osmý řádek, počínaje řádkem nultým. Poté každý osmý řádek, počínaje řádkem čtvrtým. Dále každý čtvrtý řádek, počínaje řádkem druhým a nakonec každý druhý řádek, počínaje řádkem prvním. Takto je postupně vykreslen celý obraz. Pixelů na řádcích, které ještě nebyly vykresleny, přebírají v každém kroku vlastnosti od vykreslených pixelů, které jsou v řádku nad nimi. Díky tomuto postupu je možné výsledný obraz obvykle rozpoznat již po přenesení $\frac{1}{4}$ objemu dat. Tento postup je vhodný především při přenášení obrázku po síti, kdy postupné vykreslování obrazu vzbuzuje dojem jeho zaostřování. Nevýhodou tohoto prokládání je však nárůst velikosti výsledného souboru. [11]

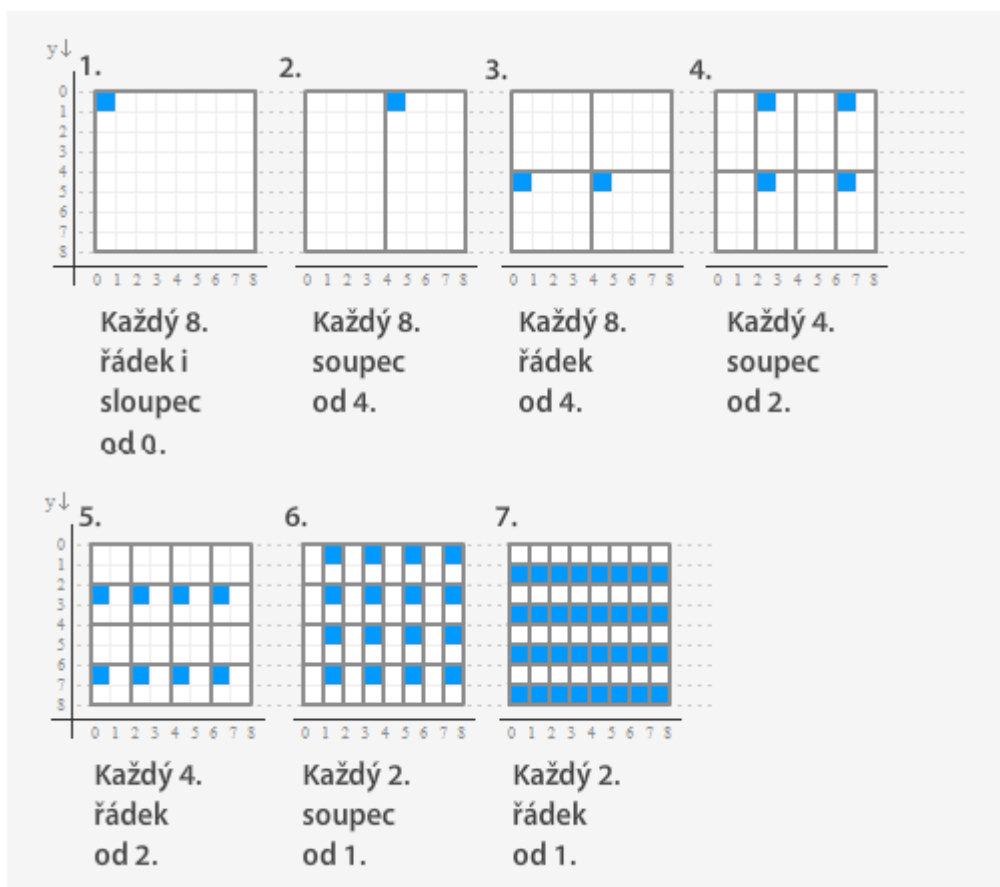


Obr. 22 Jednorozměrné prokládací schéma

3.2 Prokládání u grafického formátu PNG

PNG (Portable Network Graphics) je grafický formát určený pro rastrovou grafiku s bezztrátovou kompresí. Vychází z vlastností formátu GIF, zdokonaluje je a přináší nové možnosti. Oproti formátu GIF disponuje vylepšenou kompresní metodou a nabízí podporu 24 bitové barevné hloubky (true color). Před kompresí může být navíc každý pixel předzpracován, čímž se výsledky komprese ještě zlepší. Podporu alpha kanálu rozšiřuje na 8 bitů, díky čemuž je možné vykreslit i částečnou průhlednost. Nevýhodou je však scházející podpora více obrazů v jednom souboru a není tak možná jednoduchá animace.

Při prokládání využívá formát PNG dvourozměrné prokládací schéma. Obraz je v tomto případě vykreslován po blocích, které se postupně zjemňují. Při zobrazování je na začátku algoritmu dekódován pouze každý osmý pixel ve sloupci i řádku, čímž se obraz rozdělí na jednotlivé bloky. V každém dalším kroku jsou všechny oblasti opět rozděleny a vzniká tak vždy 2x více bloků. Každý blok je přitom kompletně vyplněn barvou pixelu, který je pro daný blok dekódován. Podobně jako u formátu GIF tento postup při vykreslování navozuje dojem postupného vyostřování obrazu. Díky dvourozměrnému schématu je však možné informaci v obraze rozpoznat již po přenesení 1/8 celkového objemu dat. Dvourozměrné prokládací schéma je také vhodnější pro zobrazování textu. Stejně jako u jednorozměrného prokládání povolením této vlastnosti narůstá výsledná velikost souboru. [11]



Obr. 23 Dvourozměrné prokládací schéma

4 TECHNOLOGIE FLASH

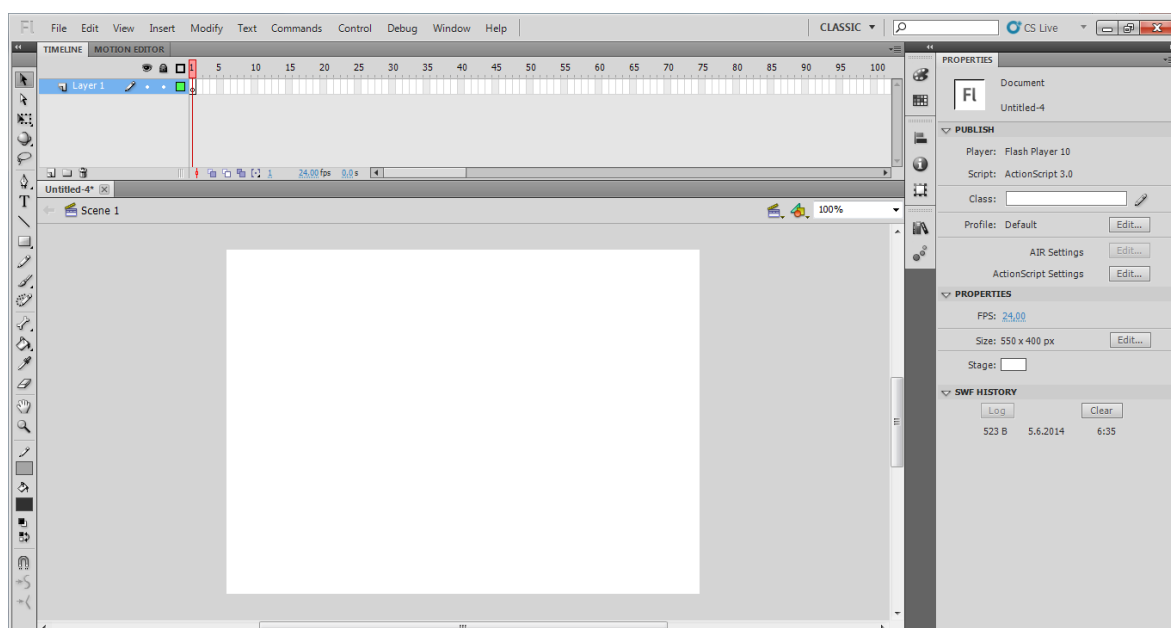
Adobe Flash je světový, multiplatformní standard pro tvorbu interaktivních animací, prezentací a aplikací s multimediálním obsahem. Jeho největší výhodou je práce s vektorovým grafickým obsahem, čímž je možné snížit velikost výstupního souboru na minimum. Díky této vlastnosti má Flash velké využití například v oblasti reklamních bannerů, které mohou obsahovat i velmi složité animace s minimálními nároky na systémové prostředky. Využití Flashe je však mnohem komplexnější a díky podpoře multimediálního obsahu má také zásadní uplatnění pro přehrávače videí na internetu. Primárním výstupem flashové aplikace je soubor ve formátu SWF – ten je přehráván zásuvným modulem Flash Player, který je dnes běžným doplňkem většiny webových prohlížečů. Soubor SWF je možné vložit jako objekt na webové stránky, kde je schopný komunikovat s dalšími technologiemi, jako například PHP, či JavaScript. Prostřednictvím technologie Flash je možné vyvíjet aplikace a hry nejen pro webový prohlížeč, ale také přímo pro stolní počítače, či mobilní zařízení. Pro Flash vzniklo také velké množství externích nástrojů a knihoven, díky kterým je například možné velice jednoduše pracovat s trojrozměrnými objekty, nebo vytvořit simulaci prostředí reálné fyziky. V současné době je Flash kvůli omezené podpoře pro webové rozhraní na mobilních zařízeních v některých oblastech nahrazován aplikacemi, které využívají nové specifikace HTML5. Flash se tomuto trendu však velice rychle přizpůsobuje a je průběžně doplňován o nástroje pro interakci a převod mezi jednotlivými technologiemi. [12]

Historie technologie Flash sahá do roku 1996, kdy byl společností Macromedia představen základní koncept obsahující jednoduché grafické nástroje a časovou osu pro animace. Postupně byl koncept vylepšen o podporu řízení animací pomocí jednoduchých scriptů na základě technologie JavaScript a následně byl pro stejný účel vyvinut vlastní scriptovací jazyk nazvaný jako ActionScript. V dalších verzích byl Flash rozšířen o práci s multimediálním obsahem a přibýly nové nástroje pro tvorbu grafických efektů a animací. Od roku 2005 je Flash dále vyvíjen pod společností Adobe Systems, která plně využila jeho potenciál a rozvinula jej do současné podoby.

4.1 Adobe Flash Professional

Software Adobe Flash Professional je hlavní vývojové prostředí pro vytváření Flash animací a multimediálních aplikací. Umožňuje vytváření a správu kompletních projektů s

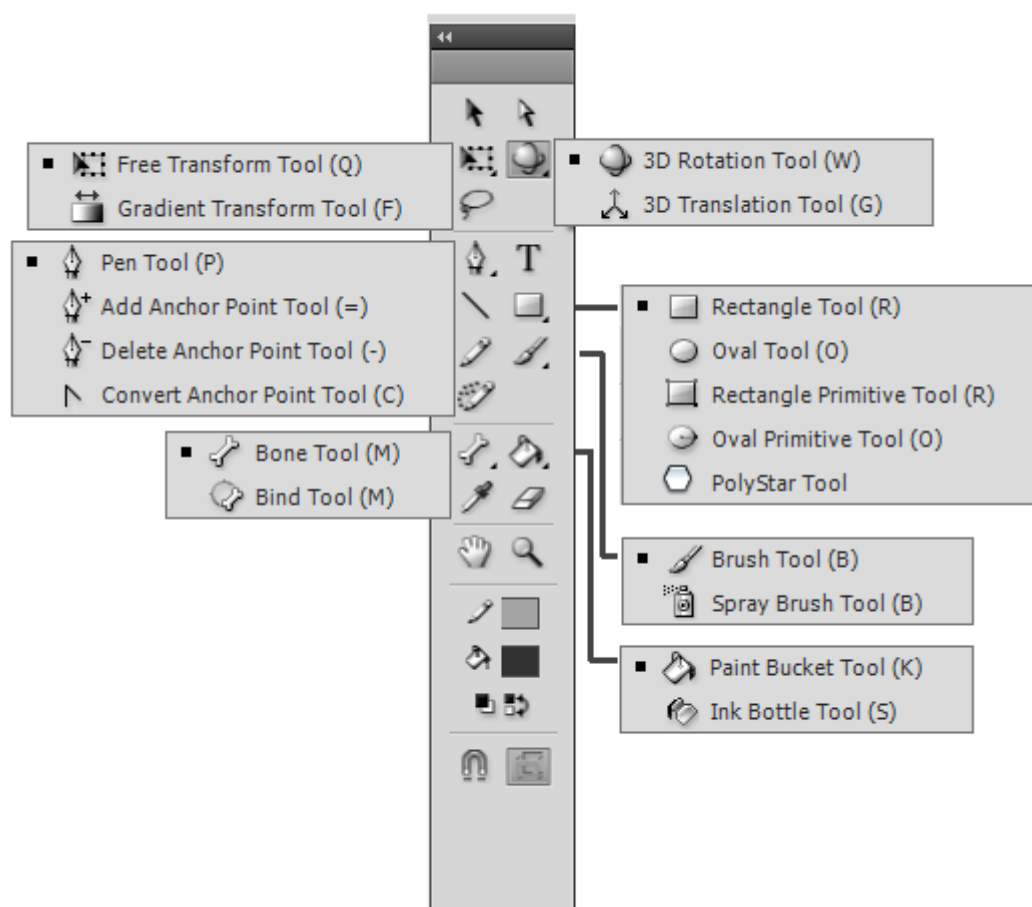
výstupem pro různé platformy, včetně prostředí pro vývoj mobilních aplikací. Kromě práce s celými projekty je možné vytvářet jednotlivé soubory tříd s předdefinovanou strukturou a nabízí také několik rozsáhlejších šablon pro tvorbu nejčastějších projektů. Každá vytvářená aplikace obsahuje hlavní scénu, která je reprezentována objektem třídy *Stage*. Tato scéna je zobrazena jako základní pracovní plocha s hlavní časovou osou, na kterou je možné vkládat další objekty. Samotné vývojové prostředí v sobě má mimo jiné zabudovaný grafický editor pro vytváření a editaci vektorové grafiky, kterou je možné vkládat přímo do jednotlivých objektů. V kombinaci s nástroji pro animace a editorem pro psaní kódu je tak možné vytvářet plnohodnotné aplikace bez nutnosti použití jiných programů.



Obr. 24 Pracovní prostředí Adobe Flash Professional CS5

4.1.1 Grafické nástroje

Ačkoliv tvorba grafiky není hlavní účel tohoto prostředí, zabudovaný grafický editor obsahuje celou škálu zajímavých nástrojů, díky kterým lze vytvářet grafiku a ilustrace na profesionální úrovni. Stejně jako u obdobných grafických editorů je možné grafický obsah přehledně řadit do jednotlivých vrstev a těm následně přidělovat různé vlastnosti. Díky interakci s ostatními programy společnosti Adobe Systems je navíc možné velmi jednoduše upravovat ilustrace a grafický obsah z programů Adobe Photoshop a Adobe Illustrator. Kromě standardních nástrojů na kreslení a transformaci základních tvarů jsou zde například také pokročilé nástroje pro tvorbu inverzní kinematiky, nebo nástroje pro vytváření 3D obsahu.

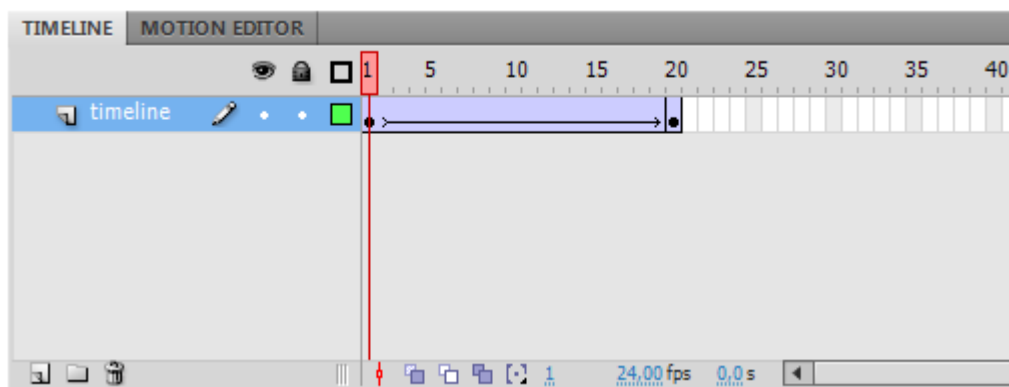


Obr. 25 Panel nástrojů

4.1.2 Nástroje pro animaci

Pro vytváření statických animací se využívá časové osy, která je součástí hlavní scény a dalších interaktivních objektů. Časové osy v různých objektech jsou na sobě přitom zcela nezávislé. Na každé časové ose je možné vytvářet snímky s libovolným obsahem, a pokud jsou dva snímky vloženy jako klíčové, lze mezi nimi vytvořit animovaný přechod. Pro tento účel nabízí Adobe Flash professional několik možností podle typu obsahu na jednotlivých snímcích.

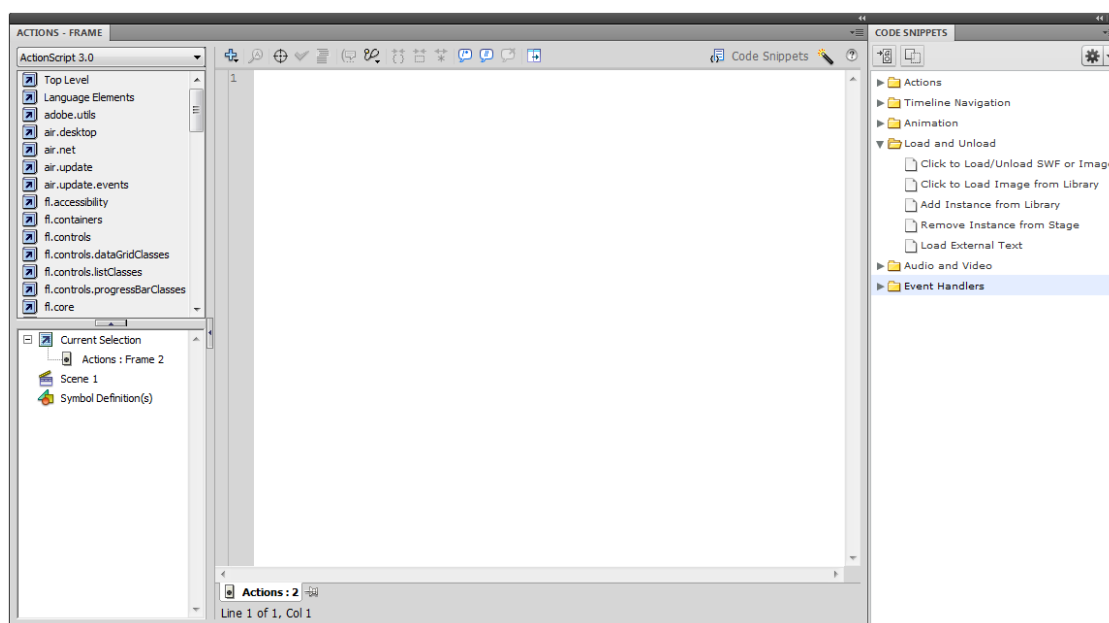
- Shape Tween – vytváří animovaný přechod mezi dvěma stavy grafického obsahu v křivkách
- Classic Tween – vytváří animovaný přechod mezi dvěma stavy vloženého objektu
- Motion Tween – vylepšená varianta Classic Tween, která mimo jiné umožňuje vytvářet i animovaný přechod s 3D rotací objektů. Pro editaci takto vytvářených přechodů je možné využít pomocný nástroj Motion Editor



Obr. 26 Ukázka časové osy

4.1.3 Nástroje pro psaní kódu

V prostředí Adobe Flash Professional je možné vkládat kód přímo na jednotlivé snímky. Vložený kód se pak spustí spolu s přehráním daného snímku a váže se pouze k jeho obsahu. Tento postup vkládání kódu je určený především pro řízení animací. Pro programování standardních aplikací se obdobně jako u jiných objektově orientovaných jazyků využívá externích tříd a jejich instancí. Pro psaní kódu se používá vestavěný editor Action Frame, který kromě běžných nástrojů pro kontrolu syntaxe nabízí také nástroj Code Snippets pro vkládání hotových šablon různorodých funkcí. [11]



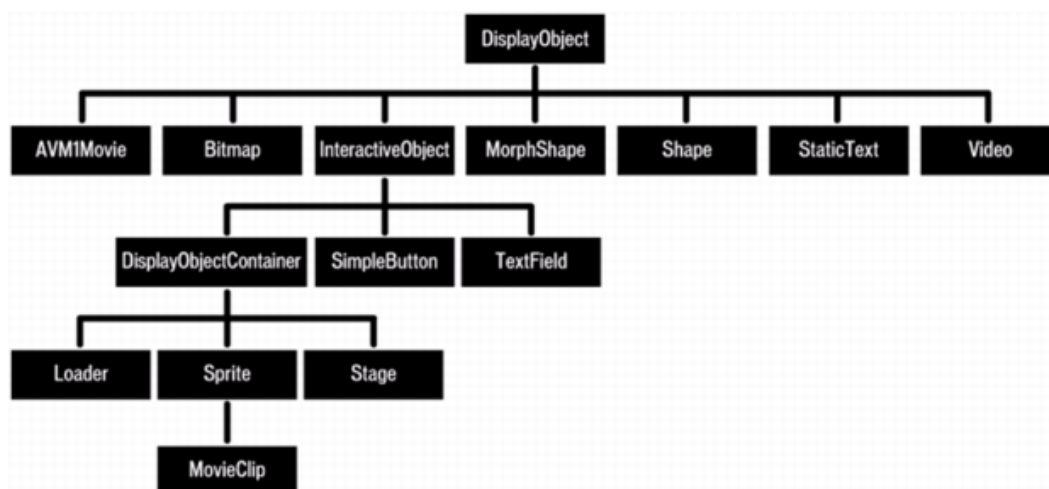
Obr. 27 Editor pro psaní kódu

4.2 ActionScript

ActionScript je programovací jazyk pro prostředí technologie Flash. Jeho první verze vycházela ze standardu jazyka JavaScript, kterému se svou syntaxí snažila co nejvíce přiblížit a byla vyvinuta především pro účely řízení animací. Postupně byl rozšiřován do verze 2.0 a následně kompletně přepracován do aktuální verze 3.0, která nabízí stabilní programovací model pro objektově orientované programování. Využívá standardu ECMA a svou syntaxí a strukturou se podobá jiným, objektově orientovaným jazykům z prostředí .NET, nebo Java. Kód jazyka ActionScript je kompilován do bytového kódu a následně přehráván v prostředí ActionScript Virtual Machine, které je součástí přehrávače Flash Player. [12]

4.2.1 Objekty pro reprezentaci obsahu

Pro reprezentaci zobrazovaného obsahu je u technologie Flash využíváno několik tříd objektů, které jsou zpracovávány odlišně a jejich správným používáním tak lze dosáhnout optimalizace a zlepšení výkonu aplikace.



Obr. 28 Rozdělení objektů pro reprezentaci obsahu [13]

- AVM1Movie – objekty pro zobrazení obsahu s podporou starší verze scriptů
- Bitmap – objekty pro zobrazení rastrové grafiky
- InteractiveObject – objekt, z kterého jsou odvozené další objekty pro zobrazení interaktivního obsahu
- MorphShape – objekty pro práci s animací vektorové grafiky
- Shape – objekty pro uchování jednoduché vektorové grafiky

- StaticText – objekty pro práci se statickým textem
- Video – objekty pro zobrazování videa
- DisplayObjectContainer – objekt, z kterého jsou odvozené další objekty pro uchování dynamického obsahu
- SimpleButton – objekt pro práci s jednoduchými tlačítky
- TextField – objekt pro práci s dynamickým textovým polem
- Loader – objekt pro načítání externího obsahu
- Sprite – objekt pro uchovávání a zobrazení dynamického obsahu
- MovieClip – objekt pro uchovávání dynamického obsahu s vlastní časovou osou
- Stage – objekt, který reprezentuje hlavní scénu aplikace

II. PRAKTICKÁ ČÁST

5 VYTVOŘENÉ PROGRAMY

Praktická část této práce je tvořena programy, které implementují výše zmíněné algoritmy a názorně vizualizují jejich průběh. Pro vytváření těchto programů byla zvolena technologie Flash - programovací jazyk ActionScript 3.0 a vývojové prostředí Adobe Flash Professional CS5, které je popsáno v teoretické části této práce. V tomto prostředí byla také vytvořena kompletní podoba všech grafických prvků, které jsou v aplikacích obsaženy.

5.1 Struktura programů

Pro co nejlepší přehlednost a možnost dalšího rozšíření, případně tvorby dalších programů, byla u všech již vytvořených programů použita obdobná struktura a členění kódu. Statické prvky a základní část kódu je umístěna přímo na první frame hlavní časové osy. Tento kód slouží pouze k načítání, inicializaci celé scény a odchyťování událostí na jednotlivých ovládacích prvcích.

Hlavní část programu pro vykonávání konkrétních operací je v každém programu umístěna ve specifické třídě, jejíž instance se vytváří při inicializaci základní scény. Všechny tyto třídy jsou potomky třídy *Sprite*, která je základním prvkem pro zobrazování grafiky a interaktivního obsahu. Metody všech hlavních tříd jsou popsány v kapitolách jednotlivých programů.

5.2 Aplikace Vyplňování oblastí

Tento program byl vytvořen pro vizualizaci průběhu algoritmů při vyplňování rastrových a vektorových oblastí. Program nabízí oddělenou scénu pro práci s rastrovou a vektorovou oblastí. Přehledně je tak možné zvolit si, v jakém prostředí chce uživatel pracovat.

V případě vektorové pracovní plochy je oblast definována posloupností bodů, které jsou vykresleny mezi orientované osy a spojeny v polygon určující oblast. Vektorová scéna nabízí vizualizaci dvou algoritmů pro vyplňování – metodu řádkového rozkladu a metodu inverzního vyplňování, které jsou popsány v teoretické části této práce (viz kapitola 1.1).

V případě rastrové pracovní plochy je oblast definována posloupností barevných hodnot uchovávaných ve dvourozměrném poli, které představuje plochu rastru. Jednotlivé barevné hodnoty jsou pak při vykreslování přiděleny pixelům na pracovní ploše. Stejně jako u vektorové scény je zde možnost vizualizace dvou algoritmů pro vyplňování. V tomto případě

se jedná o metodu semínkového vyplňování a její rozšířenou verzi řádkového semínkového vyplňování, které jsou opět popsány v teoretické části této práce (viz kapitola 1.2).

5.2.1 Popis vytvořených tříd

Třída **VectorScene**

Potomek třídy *Sprite*. Tato třída implementuje algoritmy pro vyplňování vektorových oblastí. Pro vykreslování grafických prvků využívá další objekty třídy *Sprite*, do kterých je možné vykreslovat grafiku ve vektorovém formátu.

Třída obsahuje tyto hlavní metody:

- **initCanvas()**
 - vykreslí pracovní plochu vektorové scény, včetně orientovaných os a jejich popisů
- **initScanLine()**
 - vykreslí skenovací linku, která označuje průběžnou pozici algoritmu
- **addPolygon(src:Array)**
 - přidá vektorovou oblast určenou polem bodů
- **drawPolygon()**
 - vykreslí zadaný polygon
- **checkIntersections()**
 - najde a seřadí průsečíky všech hran polygonu podle vzorců (1) a (2)
- **setIntersections()**
 - do pozice nalezených průsečíků vykreslí barevné body označující sudé a liché průsečíky
- **moveScanLine(y_:Number)**
 - posune skenovací linku do zadané pozice
- **scanLine()**
 - označí nalezené průsečíky na právě skenovaném řádku
- **fillLine()**
 - na skenovaném řádku vykreslení spojnice mezi nalezenými průsečíky, která simuluje výplň daného řádku
- **reset()**
 - vrátí skenovací linku do původní pozice a vykreslí prázdný polygon

- `stepNext()`
 - posune algoritmus o krok dopředu
- `stepPrev()`
 - posune algoritmus o krok zpátky

Třída **RasterScene**

Třída implementující algoritmy pro vyplňování rastrových oblastí. Tato třída je potomkem třídy *Sprite* a obsahuje bitmapu *_canvas*, do které je simulováno vykreslování rastru.

Třída obsahuje tyto hlavní metody:

- `clearRaster()`
 - naplní dvourozměrné pole představující rastr základní barvou
- `renderRaster()`
 - vykreslí obsah pole *_raster* do bitmapy *_canvas*
- `drawGrid()`
 - vykreslí mřížku ohraničující jednotlivé pixely v bitmapě *_canvas*
- `addArea(src:Array)`
 - přidá rastrovou oblast v podobě dvourozměrného pole barev
- `startSeed()`
 - zahájí zvolený algoritmus a určí počáteční polohu a barvu
- `reset()`
 - vrátí algoritmus na začátek a vykreslí prázdný rastr
- `stepNext()`
 - posune algoritmus o krok dopředu
- `stepPrev()`
 - posune algoritmus o krok zpátky
- `clearPixel(p:Point)`
 - vymaže pixel na pozici zadaného bodu
- `setSeed()`
 - vybere semínko pro další krok algoritmu podle zadané datové struktury
- `seedFill(p:Point)`
 - zkontroluje zadaný bod a rozhodne o jeho vyplnění podle algoritmu semínkového vyplňování

- `floodFill(p:Point)`
 - zkontroluje zadaný bod a rozhodne o jeho vyplnění podle algoritmu řádkového semínkového vyplňování
- `fillPixel(p:Point)`
 - vyplní zadaný bod definovanou barvou
- `getPixelAt(xpos:int,ypos:int)`
 - vrátí barevnou hodnotu v rastru podle zadaných souřadnic
- `isEmpty(p:Point)`
 - zkontroluje, zda byl pixel na zadané souřadnici bodu již vyplněn a vrátí hodnotu typu *Boolean*

Třída Seed

Slouží jako interaktivní grafický prvek pro určení výchozí pozice při vyplňování rastrové oblasti. Pomocí parametrů *size:Number* a *fullway:Boolean* lze za běhu měnit jeho velikost a grafickou podobu pro 4spojité, či 8spojité vyplňování.

Třída Pixel

Potomek třídy *Point*, který je určen pro uchování barevné hodnoty na dané souřadnici.

Třída RasterMaps

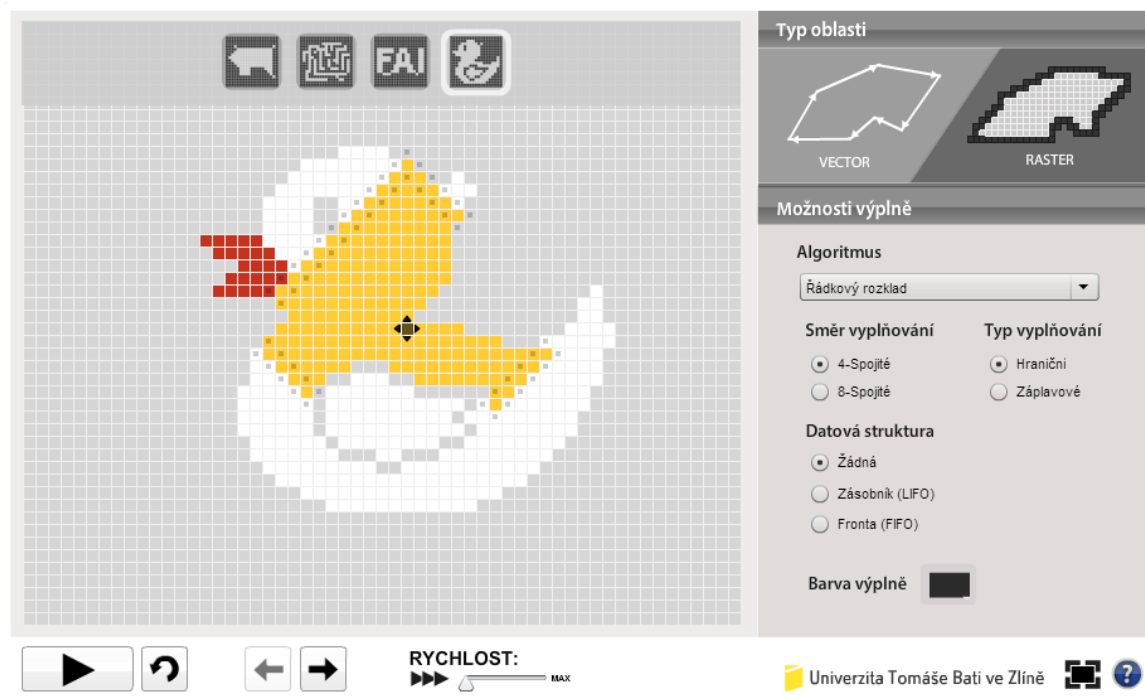
Statická třída, ve které jsou uloženy jednotlivé rastrové oblasti v podobě dvourozměrného pole barevných hodnot.

Třída VectorMaps

Statická třída, ve které jsou uloženy jednotlivé vektorové oblasti v podobě vektoru bodů třídy *Point*.

5.2.2 Uživatelské rozhraní

Uživatelské rozhraní se skládá ze tří hlavních částí. V levé části okna aplikace se nachází pracovní plocha pro zobrazení scény a vykreslování průběhů algoritmů, která je částečně překryta nabídkou pro výběr typu oblasti. V pravé části okna je umístěn panel s ovládacími prvky pro nastavení parametrů a typu algoritmu. V této části aplikace si také uživatel může přepínat mezi rastrovou a vektorovou scénou. Spodní část okna aplikace pak tvoří ovládací prvky pro samotnou animaci, krokování algoritmu, zobrazování nápovědy a tlačítko pro zapnutí, či vypnutí režimu zobrazení na celou obrazovku.



Obr. 29 Uživatelské rozhraní programu pro vizualizaci vyplňování oblastí

5.2.3 Ovládání

Uživatel má možnost spustit automatickou animaci průběhu algoritmu, u které je možné nastavit její rychlost a je možné ji kdykoliv přerušit a znovu spustit. Pokud je animace zastavena, uživatel má možnost algoritmus procházet po jednotlivých krocích tlačítky dopředu a zpět. Tlačítkem restart je možné algoritmus z kteréhokoliv kroku vrátit zpět do původní pozice. Aplikace dále disponuje tlačítkem „fullscreen“, kterým je možné aktivovat režim zobrazení přes celou plochu obrazovky. U všech typů algoritmu má uživatel dále možnost zvolit si ze čtyř typů oblastí, které jsou koncipovány tak, aby co nejlépe vystihly různé situace, do kterých se může algoritmus v praxi dostat.

U jednotlivých scén má uživatel možnost nastavit algoritmu tyto parametry:

Vektorová scéna

- *Zobrazit horizontální hrany* – zobrazí/skryje odstraněné hrany při vyplňování vektorových oblastí
- *Zachovat rozpojené body* - zobrazí/skryje rozpojení hran při vyplňování vektorových oblastí

Rastrová scéna

- *Směr vyplňování*
 - *4-spojité* – zvolí směr vyplňování ve čtyřech směrech od původního bodu
 - *8-spojité* – zvolí směr vyplňování v osmi směrech od původního bodu
- *Typ vyplňování*
 - *Hraniční* – bere v úvahu pouze hranici kolem oblasti
 - *Záplavové* – bere v úvahu jakoukoliv odlišnou barvu od původního bodu
- *Datová struktura*
 - *Žádná* – postupuje rekurzivně se všemi testovanými body najednou
 - *Zásobník* – testované body ukládá do pole a postupuje od posledního
 - *Fronta* – testované body ukládá do pole a postupuje od prvního
- *Barva* – nastaví barvu výplně oblasti

5.3 Aplikace Anti-aliasing

Tento program byl vytvořen pro vizualizaci metod anti-aliasingu. Program nabízí několik grafických objektů, které jsou uloženy ve vektorovém formátu, a je možné mezi nimi přepínat. Vybraný objekt se podle zadaného rozlišení převzorkuje a vykreslí do rastrové bitmapy. Následně je možné zvolit několik typů algoritmů, kterými se objekt znovu převzorkuje a vyhladí. Jedná se o algoritmy pro pravidelné vzorkování s vyšší frekvencí, pravidelné vzorkování s roztěsením, vzorkování s vysokým rozlišením a vzorkování metodou N-věží, které jsou popsány v teoretické části této práce (viz kapitola 2).

5.3.1 Popis vytvořených tříd

Třída AASStage

Potomek třídy *Sprite*. Tato třída implementuje algoritmy pro anti-aliasing. Obsahuje rastrovou bitmapu *_canvas*, do které je obrázek vzorkován, a objekt *_aMemoryData*, ve kterém je uložena vektorová předloha obrázku. Pro znázornění vzorkování a vykreslení ostatních grafických prvků jsou v třídě obsaženy další objekty typu *Sprite*.

Třída obsahuje tyto hlavní metody:

- *addShape(shape:Sprite)*
 - vloží vektorovou předlohu obrázku pro vzorkování

- `addOutLines(ol:Sprite)`
 - vloží obrys vektorové předlohy
- `regularSampling()`
 - převzorkuje obrázek z vektorové předlohy bez použití AA
- `regularFSAA()`
 - převzorkuje obrázek z vektorové předlohy za použití pravidelného vzorkování s vyšší frekvencí
- `quincunx()`
 - převzorkuje obrázek z vektorové předlohy podle algoritmu Quincunx
- `sochasticFSAA()`
 - převzorkuje obrázek z vektorové předlohy podle algoritmu pravidelného vzorkování s roztřesením
- `nRooksAA()`
 - převzorkuje obrázek podle algoritmu N-Věží
- `getPixel(x_:int, y_:int)`
 - vrátí barevnou hodnotu na zadaných souřadnicích vektorové předlohy
- `clearRaster()`
 - vymaže rastrovou bitmapu
- `drawAllGrids()`
 - nakreslí mřížku ohraničující pixely v rastru
- `p2s(value:Number = 1)`
 - vrátí zadanou hodnotu přepočítanou podle rozlišení scény

Třída **ColorMath**

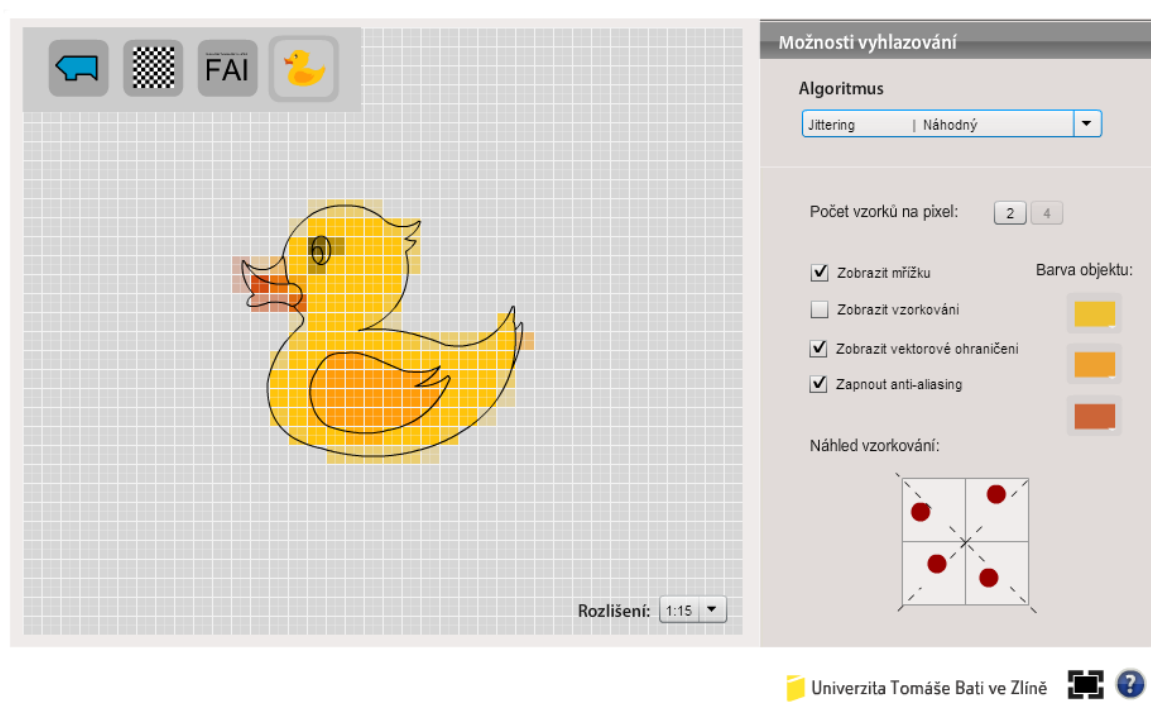
Statická třída, která slouží k veškerým výpočtům barev. Obsahuje metody pro výpočty průměrů a extrakci barevných složek.

Třída **ColorableObject**

Potomek třídy *MovieClip*, který slouží k uchovávání barevné vrstvy objektu pro vzorkování do rastru. Instance této třídy má parametr *color*, který označuje aktuální barvu vrstvy.

5.3.2 Uživatelské rozhraní

Uživatelské rozhraní je přehledně rozděleno do několika částí. Hlavní část okna je tvořena pracovní plochou, do které se vykresluje rastrová scéna se zvoleným obrázkem a všemi efekty. Tato část je částečně překryta nabídkou předloh obrázků, které je možné zvolit pro vizualizaci algoritmů. Níže se pak nachází rozbalovací nabídka pro výběr rozlišení scény. V pravé části okna aplikace se nachází panel, ve kterém jsou umístěny možnosti pro vykreslení scény, rozbalovací nabídka pro výběr algoritmu a ukázkový náhled vzorkování pixelu, který se mění podle zvolených parametrů a typu algoritmu. Ve spodní části okna se pak nachází tlačítko pro zapnutí, či vypnutí režimu zobrazení na celou obrazovku a nápovědy.



Obr. 30 Uživatelské rozhraní aplikace pro vizualizaci metod anti-aliasingu

5.3.3 Ovládání

Uživatel má v aplikaci možnost zvolit si ze čtyř předloh obrázků, které budou použity pro vizualizaci metod anti-aliasingu. Tato aplikace dále nabízí možnost zvolit poměr rozlišení scény, v kterém se bude zvolená předloha obrázku vzorkovat. Tlačítkem „fullscreen“ je možné aktivovat režim zobrazení aplikace přes celou obrazovku.

Uživatel má v aplikaci možnost nastavit tyto parametry zobrazení:

- *Počet vzorků na pixel*
 - 2 – pixel se vzorkuje diagonálně pouze ve dvou subpixelech
 - 4 – pixel se vzorkuje rovnoměrně ve 4 subpixelech
- *Zobrazit mřížku* – zobrazí/skryje ohraničení superpixelů a jeho částí
- *Zobrazit vzorkování* – zobrazí/skryje znázornění jednotlivých vzorků uvnitř pixelů
- *Zapnout vektorové ohraničení* – zobrazí/skryje ohraničení předlohy obrázku
- *Zapnout anti-aliasing* – převzorkuje obrázek podle zvoleného algoritmu
- *Barva objektu* – přebarví části obrázku zvolenou barvou

5.4 Aplikace Prokládání

Tento program byl vytvořen pro vizualizaci postupného načítání obrázků s použitím jedno-rozměrného a dvourozměrného prokládacího schématu popsány v teoretické části této práce. Do programu je možné nahrát libovolný obrázek, který je následně postupně zobrazován podle jednorozměrného, či dvourozměrného prokládacího schématu. Aplikace také nabízí několik předdefinovaných obrázků, které byly vybrány tak, aby byl efekt prokládání co nejzřetelnější.

5.4.1 Popis vytvořených tříd

Třída InterStage

Třída implementující algoritmy pro prokládací schémata. Tato třída je potomkem třídy *Sprite*. Obsahuje rastrovou bitmapu, do které se postupně vykresluje obrázek podle zvoleného algoritmu a objekt třídy *Sprite* pro vykreslení mřížky ohraničující jednotlivé pixely.

Třída obsahuje tyto hlavní metody:

- `renderRaster()`
 - zavolá metodu pro vykreslení obrázku podle zvoleného algoritmu
- `clearRaster()`
 - vymaže rastrovou bitmapu
- `PNGInterlacing()`
 - vykreslí obrázek za použití dvourozměrného prokládacího schématu
- `GIFInterlacing()`
 - vykreslí obrázek za použití jednorozměrného prokládacího schématu

- `drawAllGrids()`
 - nakreslí mřížku ohraničující pixely v rastru a oblast znázorňující prokládací schéma
- `doStep()`
 - podle aktuálního kroku určí velikost prokládací mřížky pro jednotlivé algoritmy
- `stepNext()`
 - posune algoritmus o krok dopředu
- `stepPrev()`
 - posune algoritmus o krok zpátky

Třída `ScrollBox`

Potomek třídy *Sprite*. Instance této třídy slouží jako komponenta pro posuvnou nabídku obrázků. Do komponenty je možné vložit libovolný počet položek, které se automaticky řadí pod sebe.

Třída `ScrollBoxItem`

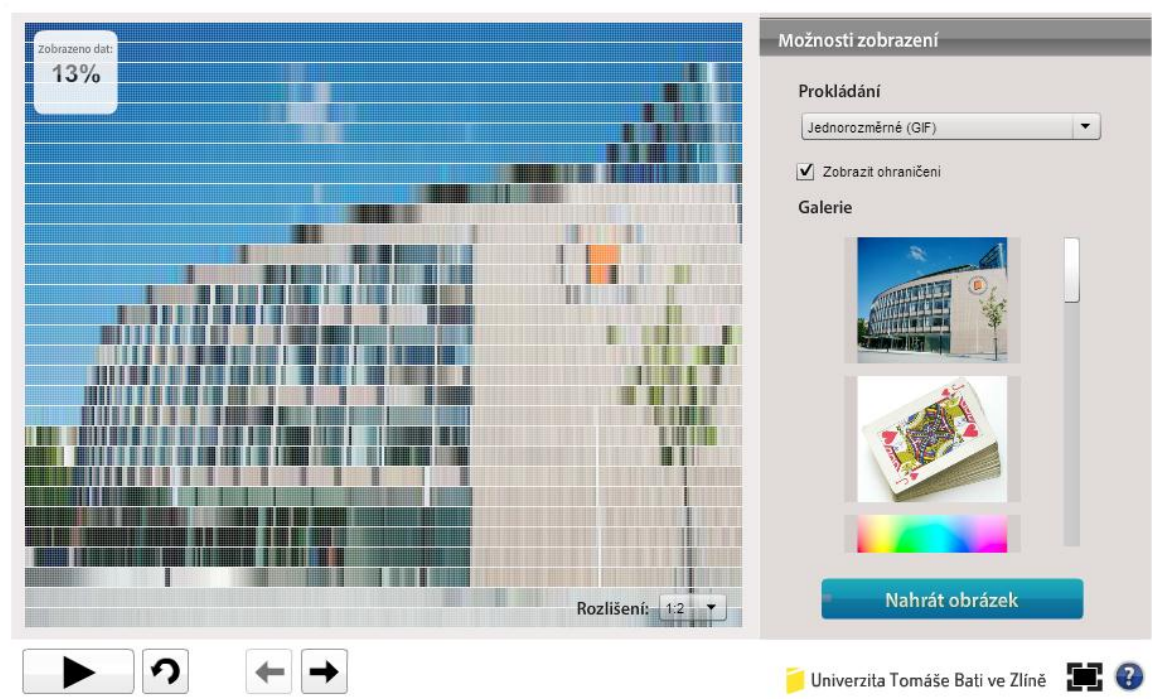
Potomek třídy *Sprite*, který slouží jako grafický kontejner pro položky komponenty *ScrollBox*.

Třída `ImageBox`

Třída pro uchování obrázku s ohledem na zadané maximální rozměry. Po vložení bitmapy se automaticky přepočítají její rozměry se zachováním původních proporcí.

5.4.2 Uživatelské rozhraní

Uživatelské rozhraní je obdobně, jako u ostatních vytvořených programů přehledně rozděleno do tří hlavních částí. V levé části okna je umístěna pracovní plocha pro postupné zobrazování obrázků, přes kterou je vložena informační ikonka zobrazující procentuální stav načtení obrazových dat a rozbalovací nabídka pro výběr rozlišení scény. V pravé části okna aplikace je umístěn panel s nabídkou algoritmů a posuvnou nabídkou obrázků, které jsou v aplikaci načteny. Pod touto nabídkou je také umístěno tlačítko pro načtení vlastního obrázku. Ve spodní části okna jsou pak umístěny ovládací prvky pro ovládání animace, krokování, zapnutí, či vypnutí režimu zobrazení na celou obrazovku a nápovědy.



Obr. 31 Uživatelské rozhraní aplikace pro vizualizaci prokládání

5.4.3 Ovládání

Uživatel má možnost spustit automatickou animaci průběhu načítání obrázku, kterou je možné kdykoliv přerušit a znovu ji spustit. Pokud je animace zastavena, uživatel má možnost načítání obrázku procházet po jednotlivých krocích tlačítka dopředu a zpět. Tlačítkem restart je možné zobrazení obrázku z kteréhokoliv kroku vrátit zpět do původní pozice. Stejně jako u předešlých aplikací disponuje i tato tlačítkem „fullscreen“, kterým je možné aktivovat režim zobrazení přes celou plochu obrazovky a tlačítko pro zobrazování, či zakázání nápovědy.

Obrázek pro vizualizaci prokládání si uživatel může vybrat z posuvné nabídky, v které je umístěno několik ukázkových motivů. Do této nabídky si uživatel také může kliknutím na tlačítko „nahrát obrázek“ přidat libovolný vlastní motiv. Uživatel má dále možnost zvolit rozlišení scény pro zobrazování obrázku a ve zvoleném rozlišení zapnout, či vypnout ohraničení jednotlivých pixelů.

6 WEBOVÁ PREZENTACE

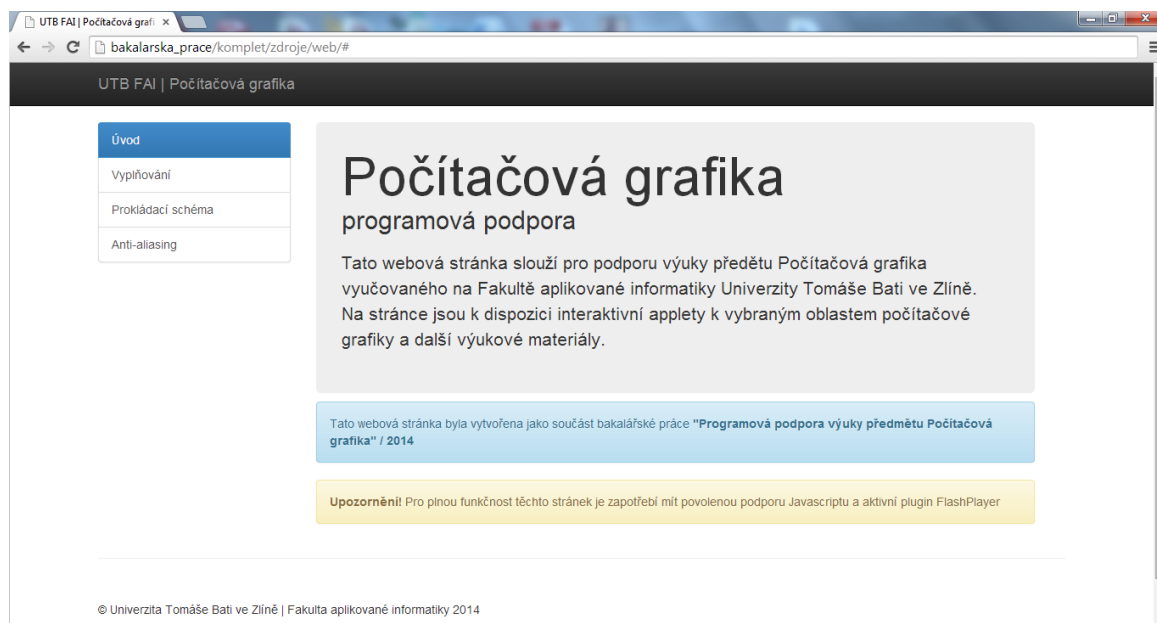
Kromě spustitelných exe souborů byly všechny aplikace vygenerovány také ve formátu swf, který je standardním výstupem technologie Flash. Tyto soubory mohou být vloženy jako objekt na webovou stránku, kde jsou pomocí zásuvného modulu FlashPlayer přehrávány přímo ve webovém prohlížeči. Pro tento účel byla vytvořena přehledná webová prezentace předmětu Počítačová grafika, ve které jsou všechny vytvořené aplikace vloženy jako applety a rozčleněny do jednotlivých kategorií.

Na vytváření webové prezentace byl použit responzivní framework Bootstrap, který obsahuje sadu nástrojů pro usnadnění vytváření uživatelského prostředí a ošetřuje správné zobrazení obsahu napříč prohlížeči.

6.1 Struktura webu

Webová prezentace je rozčleněna do úvodu a jednotlivých kategorií pro každou z problematik, kterými se tato práce zabývá. Pro každou kapitolu je vytvořena specifická podstránka, na které je vždy umístěn interaktivní applet a krátký úvod do problematiky s popisem funkcí appletu. Uživatel má také možnost stáhnout si applet jako spustitelný soubor v desktopovém prostředí.

Z kterékoliv podstránky je možné vrátit se kdykoliv zpět na úvod, případně přejít na kteroukoliv jinou kapitolu. Přejít mezi podstránkami je realizován technologií AJAX, díky které je možné měnit pouze obsahovou část stránky bez nutnosti obnovení celého okna. Pro plnou funkčnost stránek, včetně appletů, je proto zapotřebí mít povolenou podporu JavaScriptu a aktivní plugin FlashPlayer.



Obr. 32 Ukázka webové prezentace

ZÁVĚR

Bakalářská práce se zabývá metodami pro vyplňování oblastí, potlačení aliasingu a zobrazování dat s využitím prokládacího schématu. Cílem práce bylo podrobně popsat základní principy vybraných algoritmů a ve zvoleném prostředí vytvořit interaktivní aplikace pro jejich implementaci a názornou vizualizaci jejich průběhů.

Po seznámení s obecnými principy pro výše uvedené grafické operace byly vybrány konkrétní algoritmy, které jsou podrobně popsány v teoretické části této práce. Tyto algoritmy byly vybrány v souladu s tematickou náplní výuky předmětu Počítačová grafika a nejběžněji užívanými standardy v této oblasti. V práci jsou dále uvedeny i některé méně používané algoritmy pro srovnání vlastností a různých přístupů k řešení daných problémů. V teoretické části je dále popsána technologie Flash a vývojové prostředí Adobe Flash Professional, které bylo zvoleno pro návrh a tvorbu aplikací.

V rámci praktické části byly vytvořeny a popsány celkem tři aplikace, z nichž každá implementuje jednu z problematik počítačové grafiky, které jsou popsány v teoretické části této práce. Aplikace byly navrženy tak, aby bylo možné z implementace vybrané techniky co nejlépe dané problematice porozumět a aplikace se tak mohly stát součástí výuky předmětu Počítačová grafika. Pro přehlednější prezentaci vytvořených programů byly v rámci této práce také vytvořeny jednoduché webové stránky, na kterých jsou aplikace rozčleněny a umístěny ve formě interaktivních appletů spolu s popisem a krátkým úvodem do dané problematiky.

Díky zvolené technologii, přehledné struktuře kódu a multiplatformnímu výstupu je možné do budoucna aplikace velice jednoduše obohatit o další typy algoritmů a vlastností pro zobrazení. Stejně tak webová prezentace je koncipována tak, aby bylo průběžně možné přidávat další oblasti počítačové grafiky. Do budoucna tak může vzniknout kompletní sbírka materiálů pro programovou podporu výuky předmětu Počítačová grafika.

Veškeré součásti této práce, včetně zdrojových kódů aplikací a webové prezentace jsou nahrány na přiloženém CD.

SEZNAM POUŽITÉ LITERATURY

- [1] ŽÁRA, Jiří. Moderní počítačová grafika. 2., přeprac. a rozš. vyd. Brno: Computer Press, c2004, 609 s., 16 s. barev. obr. příl. ISBN 80-251-0454-0.
- [2] MARTIŠEK, Dalibor. Matematické principy grafických systémů. Vyd. 1. Brno: Litera, 2002, 278 s. ISBN 80-857-6319-2.
- [3] Geometrie/Vyplňování – Wikiknihy. [online]. [cit. 2014-04-29]. Dostupné z: <http://cs.wikibooks.org/wiki/Geometrie/Vypl%C5%88ov%C3%A1n%C3%AD>
- [4] ŠTUGEL, Juraj. Netgraphics. [online]. [cit. 2014-04-29]. Dostupné z: <http://pg.netgraphics.sk/>
- [5] Vyplňování oblastí [online]. 2010, [cit. 2014-05-03]. Dostupné z: http://temp.buchtic.net/skola/ipogr/prednasky/IPOGR_2010_P05_Alg-VyplnovaniOblasti_4s.pdf
- [6] ŠIMŮNEK, Milan. Vyplňování oblastí. [online]. [cit. 2014-05-09]. Dostupné z: <http://nb.vse.cz/~simunek/it418/2d/vypln2D/Vypln2D.html>
- [7] ŠULC, Tomáš. Antialiasing: teoretický i praktický test vyhlazování hran. [online]. 2012 [cit. 2014-05-15]. Dostupné z: <http://pctuning.tyden.cz/hardware/graficke-karty/23244-antialiasing-teoreticky-i-prakticky-test-vyhlazovani-hran>
- [8] ŠULC, Tomáš. Antialiasing: vyhlazování teoreticky i prakticky. [online]. 2009 [cit. 2014-05-17]. Dostupné z: <http://pctuning.tyden.cz/hardware/graficke-karty/14177-antialiasing-vyhlazovani-teoreticky-i-prakticky>
- [9] NO-X. Full-Scene AntiAliasing. [online]. 2004 [cit. 2014-05-17]. Dostupné z: <http://www.3dfx.cz/articles/fsaa.htm>
- [10] KABÁT, Zdeněk. 3D technologie: Anti-aliasing. [online]. 2003 [cit. 2014-05-19]. Dostupné z: <http://www.svethardware.cz/3d-technologie-anti-aliasing/8871>
- [11] DOLEŽAL, Jiří. Grafické formáty. [online]. [cit. 2014-05-10]. Dostupné z: <http://mdg.vsb.cz/jdolezal/Pgrafika/Prednaska/GrafickeFormaty.html>
- [12] BERNARD, Borek. Adobe Flash CS5 Professional: oficiální výukový kurz. Vyd. 1. Brno: Computer Press, 2010, 392 s. ISBN 978-80-251-3224-1.
- [13] ADOBE SYSTEMS INCORPORATED. ActionScript 3.0 Reference for the Adobe Flash Platform [online]. [cit. 2014-05-24]. Dostupné z: http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/

- [14] BRICHTA, Ondřej. ActionScript 3.0 – DisplayObject. [online]. 2008 [cit. 2014-05-28]. Dostupné z: <http://www.flash.cz/portal/clanek.aspx?id=1284>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

XOR	Exclusive OR
AA	Anti-Aliasing
HRAA	High Resolution Anti-Aliasing
FP	Flash Player
AS	ActionScript
JS	JavaScript
PHP	Hypertext Preprocessor (Personal Home Page)
ECMA	European Computer Manufacturers Association
PNG	Portable Network Graphic (grafický formát)
GIF	Graphics Interchange Format (grafický formát)
LZW	Lempel Ziv Welch
SWF	Shockwave Flash
AJAX	Asynchronous JavaScript and XML
HTML	HyperText Markup Language

SEZNAM OBRÁZKŮ

Obr. 1 Varianty barevných výplní	11
Obr. 2 Možnosti řešení výplně při křížení hranic	12
Obr. 3 Zjednodušený průběh algoritmu řádkového rozkladu	13
Obr. 4 Rozdělení trojúhelníku pro modifikaci algoritmu	14
Obr. 5 Zjednodušený průběh algoritmu při vyplňování šrafováním	15
Obr. 6 Aplikace vzorové výplně	16
Obr. 7 Zjednodušený průběh algoritmu inverzního vyplňování	16
Obr. 8 Plotové inverzní vyplňování	17
Obr. 9 Varianty vyplňování hranice nakreslené v rastru	18
Obr. 10 Typy semínkového vyplňování	18
Obr. 11 Základní varianta semínkového vyplňování	19
Obr. 12 Řádkové semínkové vyplňování	19
Obr. 13 Šrafování pomocí rastrového vzoru	20
Obr. 14 Alias vzniklý při rasterizaci (jaggies)	21
Obr. 15 Pravidelné převzorkování s dvojnásobnou frekvencí	22
Obr. 16 Diagonální redukce subpixelů	22
Obr. 17 Vzorkování s otočenou mřížkou	23
Obr. 18 Vzorkování technologií Quincunx	23
Obr. 19 Rozmístění vzorků s využitím Poisson disc	24
Obr. 20 Náhodné roztřesení vzorků (jittering)	25
Obr. 21 Tři věžová vzorkovací schémata	25
Obr. 22 Jednorozměrné prokládací schéma	27
Obr. 23 Dvourozměrné prokládací schéma	28
Obr. 24 Pracovní prostředí Adobe Flash Professional CS5	30
Obr. 25 Panel nástrojů	31
Obr. 26 Ukázka časové osy	32
Obr. 27 Editor pro psaní kódu	32
Obr. 28 Rozdělení objektů pro reprezentaci obsahu [13]	33
Obr. 29 Uživatelské rozhraní programu pro vizualizaci vyplňování oblastí	40
Obr. 30 Uživatelské rozhraní aplikace pro vizualizaci metod anti-aliasingu	43
Obr. 31 Uživatelské rozhraní aplikace pro vizualizaci prokládání	46
Obr. 32 Ukázka webové prezentace	48

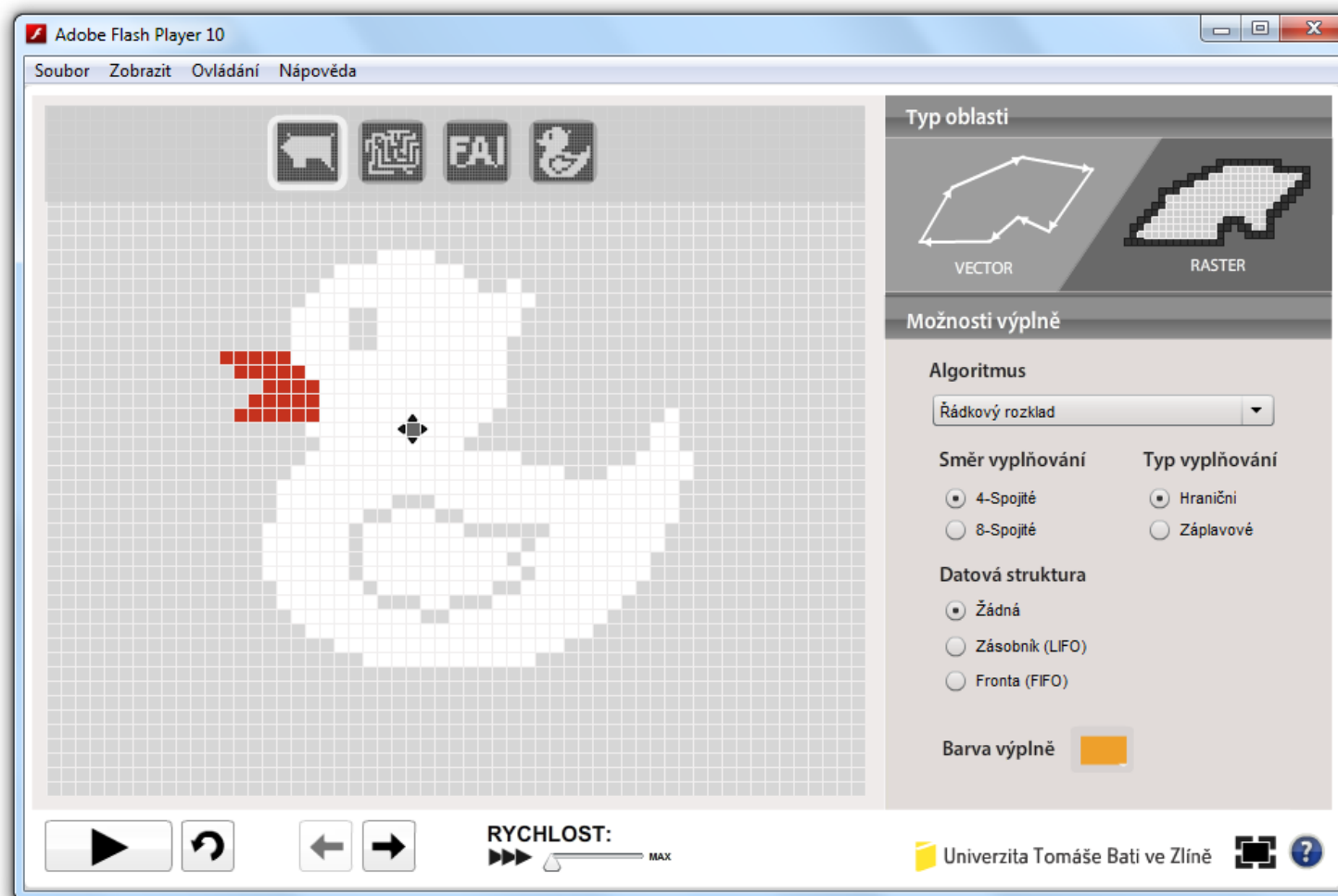
SEZNAM PŘÍLOH

- PI Obsah vloženého CD
- PII Náhled aplikace pro simulaci vyplňování oblastí
- PIII Náhled aplikace pro simulaci metod anti-aliasingu
- PIV Náhled aplikace pro simulaci načítání obrázků s prokládáním

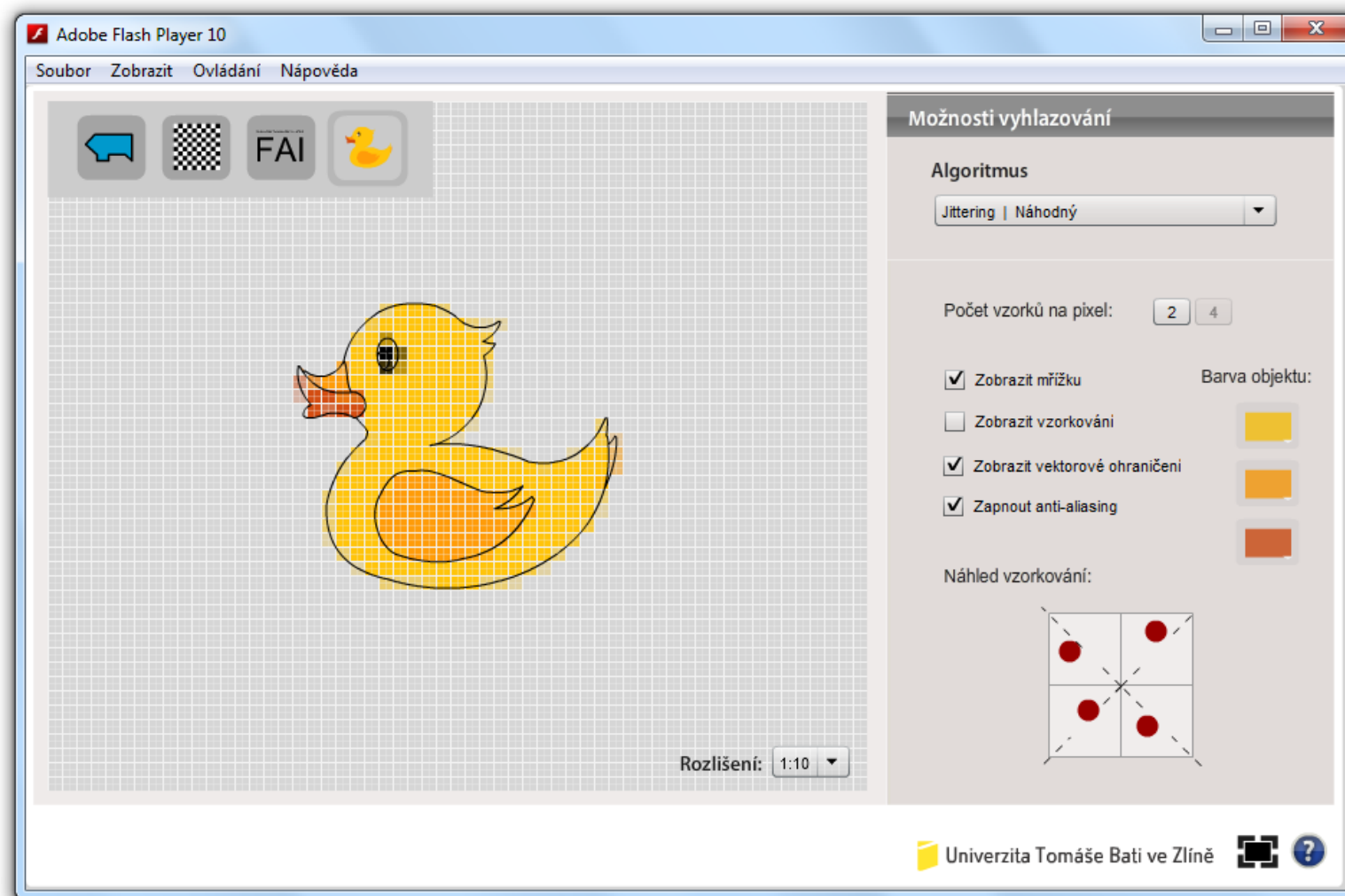
PŘÍLOHA P I: OBSAH VLOŽENÉHO CD

- Elektronická verze této práce
- Zdrojové kódy programů
- Spustitelné aplikace
- Webová prezentace

PŘÍLOHA P II: NÁHLED APLIKACE PRO SIMULACI VYPLŇOVÁNÍ OBLASTÍ



PŘÍLOHA P III: NÁHLED APLIKACE PRO SIMULACI METOD ANTI-ALIASINGU



PŘÍLOHA P IV: NÁHLED APLIKACE PRO SIMULACI NAČÍTÁNÍ OBRÁZKŮ S PROKLÁDÁNÍM

