VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

INTERAKTIVNÍ WEBOVÉ VÝUKOVÉ APLIKACE Z OBLASTI VEKTOROVÉ GRAFIKY
INTERACTIVE APPS FOR EDUCATION OF VECTOR GRAPHICS THEORY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Matúš Lipa

VEDOUCÍ PRÁCE
SUPERVISOR
doc. Mgr. Pavel Rajmic, Ph.D.

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Matúš Lipa ***ID:*** 167861
Ročník: 2 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Interaktivní webové výukové aplikace z oblasti vektorové grafiky

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte problematiku vztahující se k výukovým apletům, navrhněte jejich funkcionalitu a ovládání, implementujte a odlaďte je. Aplety budou zaměřeny na interaktivní podání algoritmů a konstrukcí používaných v oblasti vektorové grafiky, konkrétně budou aplety prezentovat postupy při vytváření vektorových objektů na základě uživatelského vstupu a jejich přenos na obrazovku počítače: 1/ Bézierova křivka (a její parametrická konstrukce), 2/ Racionální Bézierova křivka (obdobně), 3/ Algoritmus de Casteljau pro výpočet polohy bodu na křivce, 4/ Rasterizace úsečky a 5/ Rasterizace kružnice. Aplety implementujte pomocí HTML a JavaScriptu tak, aby demonstrovaly příslušnou teorii a vybízely studenty k interakci.

DOPORUČENÁ LITERATURA:

[1] Beneš, B.; Sochor, J.; Felkel, P.; Žára, J.: Moderní počítačová grafika. Computer Press, Brno, 2005.

[2] Piegl, L.,; Tiller, W.: The NURBS Book. Druhé vydání. Springer, 1997

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 16.5.2019

Vedoucí práce: doc. Mgr. Pavel Rajmic, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Táto práca sa zaoberá vytvorením niekoľkých interaktívnych webových apletov pre výuku vektorovej grafiky. Popísaný je obrazový signál, jeho diskretizácia, vektorový a bitmapový typ záznamu obrazu. Ďalej sú popísané vybrané vektorové krivky, ich vlastnosti, algoritmy pre ich konštrukciu a použitie. Vysvetlené sú princípy rasterizácie základných vektorových objektov. Sú navrhnuté grafické užívateľské rozhrania pre jednotlivé aplety. Tieto aplety sú zároveň implementované pomocou jazyka HTML a Javascript. Implementované aplety sú umiestnené do webových stránok, ktoré sa používajú pri výuke počítačovej grafiky.

KĽÚČOVÉ SLOVÁ

aproximačné krivky, bézierova krivka, bitmapová grafika, DDA algoritmus, digitalizácia obrazu, figma, interpolačné krivky, obrazový signál, rasterizácia, reprezentácia obrazu, vektorová grafika, vektorové krivky, webové aplety,

ABSTRACT

This work deals with the creation of several interactive web applets for education of vector graphics. Described is the image signal, his discretization, vector and bitmap types of image record. Further they describe selected vector curves, their properties, algorithms for their construction and usage. The principles of rasterization of basic vector objects are explained. Using the Figma tool, graphical user interfaces for each applet are designed. These applets are implemented using HTML and Javascript. The implemented applets are placed on web pages that are used in computer graphics education.

KEYWORDS

aproximation curves, bézier curve, bitmap graphic, DDA algorithm, figma, image digitalization, image representation, image signal, rasterization, interpolation curves, vector graphic, vector curves, web applets,

LIPA, Matúš. *Interaktivní webové výukové aplikace z oblasti vektorové grafiky*. Brno, 2019, 55 s. Diplomová práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: doc. Mgr. Pavel Rajmic, Ph.D.

VYHLÁSENIE

Vyhlasujem, že som svoju diplomovú prácu na tému „Interaktivní webové výukové aplikace z oblasti vektorové grafiky“ vypracoval samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákoníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovéj práce pánovi doc. Mgr. Pavlovi Rajmicovi Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Brno ....

podpis autora

Obsah

Zoznam obrázkov

Úvod

Táto práca sa zaoberá dvojrozmernou počítačovou grafikou, predovšetkým však vektorovým krivkám používaným v 2D priestore. Počítačová grafika je jednou z najrýchlejšie sa rozvíjajúcich oblastí v informatike. Grafické objekty môžeme zobrazovať bitmapovo alebo vektorovo. Práve vektorový zápis má v dnešnej dobe veľký význam. Pri vytváraní moderných dizajnov či už aplikácií alebo internetových stránok sa využívajú nástroje kde sú jednotlivé prvky dizajnu reprezentované vektorovo. Cieľom tejto práce je vytvoriť niekoľko apletov, ktoré budú slúžiť ako pomôcka pri výuke počítačovej grafiky. Tieto aplety umožnujú študentom priblížiť vytváranie jednotlivých vektorových objektov a tak doplniť ich teoretické vedomosti. Pre tieto aplety bolo navrhnuté grafické užívateľské prostredie, ktoré umožňuje meniť parametre na základe ktorých môžu užívatelia vidieť zmeny vytvorených objektov. Teoretická časť práce sa venuje obrazu, jeho matematickému popisu a jeho diskretizácií. Ďalej sú vysvetlené princípy vektorovej a bitmapovej grafiky, ich výhody a nevýhody a oblasti použitia. Druhá kapitola sa venuje 2D vektorovej grafike. Popísané sú vektorové krivky, ich typy, vyjadrenie, vlastnosti. Veľká časť kapitoly sa venuje Bézierovej krivke, jednotlivým variantám krivky, ich vlastnostiam a použitiu. Ďalej je popísaný algoritmus de Casteljau používaný k zisteniu polohy bodu na krivke v určitom časovom okamžiku a NURBS krivky. V tretej kapitole sa práca venuje problému zobrazenia vektorových objektov na rastrových zobrazovacích zariadeniach. Sú popísané definície jednotlivých základných objektov, algoritmy na ich rasterizáciu, a postup rasterizácie. V praktickej časti sa práca venuje návrhu grafických užívateľských rozhraní apletov a ich implementáciou. Navrhnuté a implementované sú aplety na výuku: Bézierovej krivky, Racionálnej Bézierovej krivky, algoritmu de Casteljau, rasterizáciu úsečky a rasterizáciu kružnice.

1 Reprezentácia obrazového signálu

Číslicové spracovanie signálov má v súčasnej dobe vo vede a technike veľmi dôležitú roľu. Problematika počítačovej grafiky nie je obmedzená len na popis dát, s ktorými pracujeme, ale je potreba zaistiť aj ich zobrazenie na príslušnom zobrazovacom zariadení. Spracovávané číslicové postupnosti sú vo väčšine prípadov získané digitalizáciou analógového signálu. V tejto kapitole je popísaná reprezentácia obrazu, jeho matematický popis a spôsob samotnej digitalizácie. Ďalej sú popísané dva základné druhy reprezentácie obrazu bitmapový a vektorový.

1.1 Matematický popis signálu

Definícia obrazu sa líši podľa svojej aplikačnej oblasti a väčšina disciplín používa definíciu „ad hoc“, ktoré najlepšie vyhovujú ich potrebám. Pri formálnom vyjadrení obrazu v modeli šedej je spojitá funkcia:

$$z = f(x, y). \tag{1.1}$$

Pri obraze predpokladáme obmedzené rozmery, môžeme teda definičný obor obrazovej funkcie zapísať ako kartézsky súčin dvoch spojitých intervalov z oboru reálnych čísel. Obrazová funkcia realizuje zobrazenie

$$f : (\langle x_{\min}, x_{\max}\rangle \times \langle y_{\min}, y_{\max}\rangle) \to H \tag{1.2}$$

Reálne čísla x, y kde $x : x_{\min} \geq x \geq x_{\max}$ a $y : y_{\min} \geq y \geq x_{\max}$ sú súradnice bodu v dvoch rozmeroch, v ktorých funkcia naberá nejakú hodnotu z z oboru hodnôt to je $z \in H$. Hodnotou obrazovej funkcie môže byť jediné reálne číslo napríklad jas, v počítačovej grafike je to ale obvykle viac hodnôt, typicky trojica červená, zelená, modrá zložka obrazovej informácií v modeli RGB. Funkcia môže nadobúdať hodnoty ako usporiadaná n-tica údajov $z = [z_1, z_2, \ldots, z_n]$ čo môžeme zapísať ako

$$f : (\langle x_{min}, x_{max}\rangle \times \langle y_{min}, y_{max}\rangle) \to (H_1 \times H_2 \times \cdots \times H_n) \tag{1.3}$$

Pri práci s obrazom v počítačovej grafike máme zriedka k dispozícií spojitý definičný obor funkcie. Omnoho častejšie pracujeme s rastrom, ktorý je zložený s obrazových bodov nazývaných pixely. V praxi je obor hodnôt funkcie 1.1 rôzne obmedzený, napríklad použitou aplikáciou alebo vlastnosťami technického zariadenia ktoré obraz vytvárajú. [1].

1.2 Digitalizácia obrazu

Pri digitálnom spracovaní obrazu je často cieľom spracovávať obrazy, ktoré vznikajú ako obrazy spojité. Spojitá reprezentácia môže byť žiadaná tiež pre výstup

spracovania. Napríklad ak má byť výsledok pozorovaný človekom. Pre samotné digitálne spracovanie obrazu v počítači nie je však spojitá reprezentácia vhodná, pretože obraz na obrazovke je diskrétny. Vhodnejšou je diskrétna reprezentácia kedy je obraz reprezentovaný konečnou množinou hodnôt. Jedným zo základných dejov, ktorý je spojený so získavaním počítačového obrazu, je prechod od spojitej funkcie $f(x, y)$ ku diskrétnej funkcií I_{ij} a to ako v definičnom obore funkcie $f(x, y)$, tak aj v jej obore hodnôt H. Z uvedeného dôvodu vykonávame pred digitálnym spracovaním prevod obrazu do diskrétnej podoby. Tento proces prechodu od spojitého obrazu k diskrétnemu sa nazýva digitalizácia a odohráva sa v dvoch krokoch ktorými sú kvantovanie a vzorkovanie. Po spracovaní v digitalizovanej podobe môže byť naopak požadovaná rekonštrukcia obrazu do podoby spojitej [2], [1].

1.2.1 Vzorkovanie

Vzorkovanie je najbežnejší spôsob prevodu signálu spojitého v čase na diskrétny signál. Je to zaznamenávanie hodnôt – vzoriek, vo vopred daných intervaloch. Vzdialenosť medzi vzorkami sa nazýva perióda vzorkovacieho signálu. Jej prevrátená hodnota sa nazýva vzorkovacia frekvencia f_{vz}. Najčastejšie sa používa vzorkovanie s pevnou vzorkovacou frekvenciou. Signál je správne vzorkovaný vtedy, keď zo získaných vzoriek dokážeme spätne zrekonštruovať pôvodný signál. Tento proces sa nazýva rekonštrukcia signálu.

Nastáva otázka, aká môže byť minimálna frekvencia, aby bol signál zachytený dostatočne a bolo možné bezchybnú rekonštrukciu obrazu z jeho vzoriek. Odpoveď na túto otázku udáva Shannonov teorém. Chyba vzorkovania sa nazýva aliasing. Alias je jedným z najdôležitejších pojmov v počítačovej grafike. Vzniká pri rekonštrukcií signálu vzorkovaného pod Nyquistovým limitom a prejavuje sa ako nová nízkofrekvenčná informácia, ktorá nebola v pôvodnom signále prítomná.
Vzniká v dvoch prípadoch:

- Pokiaľ je pôvodná informácia frekvenčne neobmedzená, teda ak neexistuje žiadna maximálna frekvencia a funkciu nie je možné v diskrétnej mriežke rastru reprezentovať presne.
- Ak je pôvodná funkcia frekvenčne obmedzená, tj. v jej Furierovóm spektre existuje určitá maximálna frekvencia $f_{\max}$, a túto funkciu vzorkujeme s frekvenciou menšou ako $2f_{\max}$, teda pod Nyquistovým limitom.

Príkladom aliasu je napríklad televízna obrazovka snímaná kamerou. Vzhľadom k tomu, že kamera sníma obraz v diskrétnych časových intervaloch a obrazovka premieta po polsnímkoch, dochádza k interferencií frekvencií a následnému aliasu, ktorý sa prejavuje ako pohybujúce sa pruhy či blikanie. Odstránenie či zmenšenie aliasu sa nazýva antialiasing. Najjednoduchšou cestou je odstrániť z obrazu informáciu, ktorú

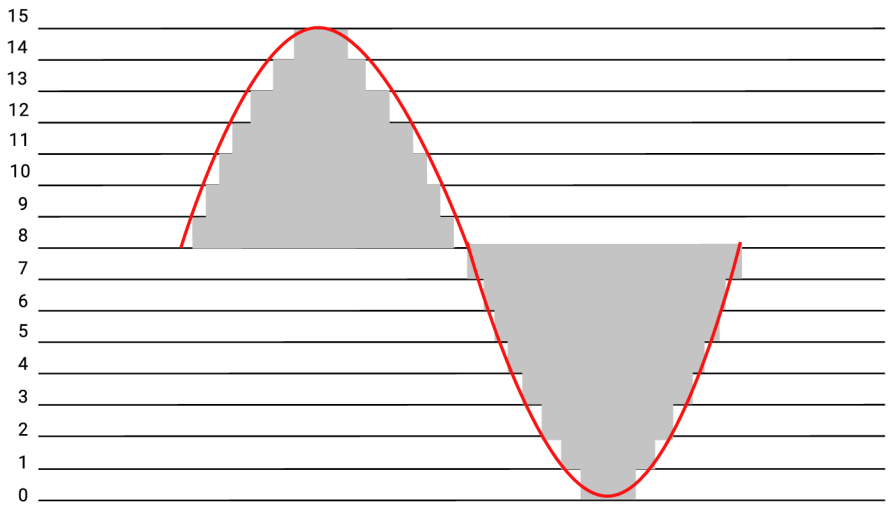
nie je možné vzorkovať, teda odstrániť vysoké frekvencie. Ochranou proti aliasingu je antialiasingový filter, ktorý má za úlohu odfiltrovať frekvencie vyššie ako odpovedajú Shannonovmu teorému. Je to dolná prepusť realizovaná ako analógový frekvenčný filter [1], [3].

1.2.2 Kvantovanie

Vzorkovaním získame konečný počet diskrétnych vzoriek pôvodného analógového signálu. Tieto vzorky sú však pre ďalšie spracovanie stále nevhodné. Kvantovanie je proces jednorozmernej diskretizácií a jeho podstatou je zaokrúhlenie hodnôt signálu získaného pri vzorkovaní na predom definované kvantizačné hladiny. Celý proces sa riadi tzv. rozhodovacími úrovňami, ktoré sa nachádzajú v polovičnej vzdialenosti medzi jednotlivými kvantizačnými hladinami.

Podľa spôsobu rozdelenia kvantovanej veličiny rozdeľujeme kvantovanie na uniformné a neuniformné. Uniformné kvantovanie používa konštantnú dĺžku intervalu zatiaľ čo neuniformné kvantovanie používa premennú dĺžku intervalu. Neuniformné kvantovanie sa používa menej často, napríklad pri prevode obrázku s veľkým počtom farieb na obraz s nižším počtom farieb.

Hodnoty pôvodného analógového signálu sa teda neprenášajú presne, ale vždy ako celočíselné hodnoty dané kvantizačnými hladinami. To má za následok že pri prevode digitálneho signálu na analógový sa objaví kvantizačný šum. Ten je ale v porovnaní s hodnotami výsledného signálu veľmi malý [4].



Obr. 1.1: Digitalizácia analógového signálu

1.3 Bitmapová grafika

Bitmapová grafika má využitie v mnohých počítačových odboroch, od drobných prvkov na internetových stránkach až po textúry aplikované na 3D objekty. Vyniká hlavne tam kde by vektorová grafika príliš komplexná, napríklad pri fotografiách alebo dátach kde nie je možné urobiť ich vektorizáciu. Bitmapová grafika je zložená z bitmapy, akejsi matice bodov, kde každý bod má definovanú farbu a jas. Pri určitom množstve a jemnosti tejto bitmapy začnú body opticky splývať a vytvárajú obraz.

Obraz je definovaný pomocou rastru pixelov alebo bodov, každý z nich nesie informáciu o vlastnom vzhľade. Pri obrázkoch v obrazovom priestore RGB má každý pixel aspoň tri bajty, kde je pre každú farbu definovaná jej intenzita. Čím je hlbší farebný priestor, tým je informácia o každom bode objemnejšia. Najmenšiu farebnú hĺbku ma čiernobiely obraz často nazývaný ako monochromatický obraz alebo binárny obraz, kde každý pixel môže mať iba dve hodnoty bielu alebo čiernu a na toto vyjadrenie nám stačí jeden bit. Pojem monochromatický má svoje historické dôvody a je čiastočne zavádzajúci pretože, binárna informácia nemusí reprezentovať práve čiernu a bielu farbu, ale ľubovoľné dve farby.

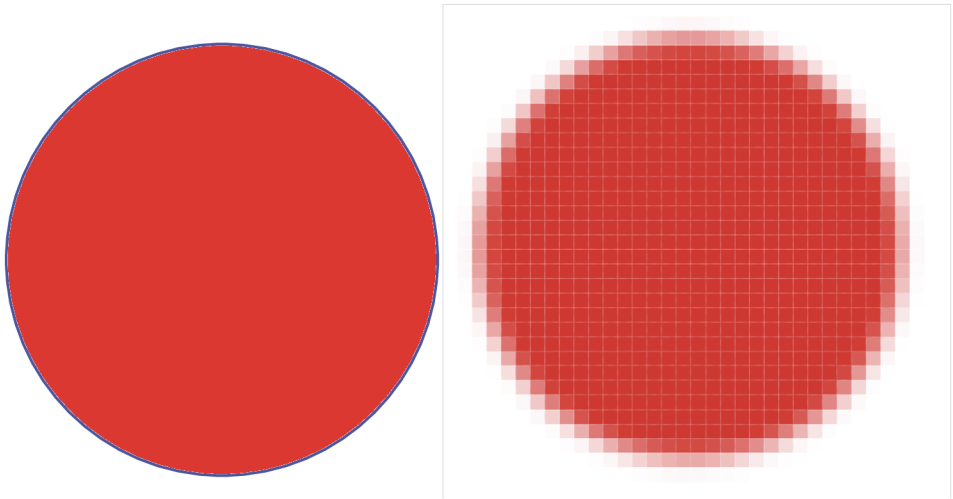
Ďalším spôsobom na určenie hodnoty jednotlivých pixelov je indexový mód, ktorý je spojený s používaním tzv. farebnej palety (*palette*), alebo mapy farieb (*colormap*). Pri takomto obraze nereprezentuje hodnota pixelu priamo farbu, ale ukazuje do tabuľky – farebnej palety.

Okrem výhod ako široká podpora a existencia množstva filtrov má bitmapová grafika aj radu nevýhod. Hlavnou nevýhodou je pamäťová náročnosť, keďže každý bod obrazu musí niesť informáciu o svojom jase, farbe, poprípade priehľadnosti a táto náročnosť rastie kvadraticky s rozlíšením. Preto je kompresií obrazov venovaná značná pozornosť a existuje mnoho kompresných metód a formátov ukladania bitmapových obrázkov medzi najznámejšie patria GIF (Graphics lnterchange Format), PNG (Portable Graphics Network), JPEG (Joint Photographic Experts Group). Druhou nevýhodou je nemožnosť veľkosť obrazu zmeniť bez zníženia kvality, pretože rozlíšenie bitmapovej matice je dopredu dané a nie je možné ho jednoducho meniť. Kedže pixely sú zarovnané do mriežky, rozlíšenie udáva počet pixelov na šírku a výšku obrázku. Jedinou možnosťou zmeny rozlíšenia je prevzorkovanie, ktoré síce zmení fyzické rozlíšenie, ale chýbajúcu informáciu nepridá. Rozlíšenie obrazu sa pri tomto kroku buď zhorší (pod vzorkovaním) alebo zlepší (interpoláciou). Informácia o obraze je teda uložená s konečnou úrovňou detailu [5], [6], [7].

1.4 Vektorová grafika

Vektorová grafika poskytuje elegantný spôsob konštrukcie digitálnych obrázkov pre jeho kompaktnosť, nezávislosť na rozlíšení a jednoduchú možnosť úpravy. Na rozdiel od bitmapovej grafiky, pri ktorej sa obrázok skladá s jednotlivých bodov. Vektorová grafika využíva k popisu obrázku základné geometrické primitívny, ktoré je možné jednoducho matematicky popísať. V dvojrozmernom priestore sú to: bod, úsečka, priamka, krivka a mnohouholník ktorými je možné jednoducho popísať akýkoľvek tvar [7]. Programy, ktoré pracujú s vektorovou grafikou, ukladajú grafickú informáciu pomocou matematického zápisu. Tento zápis definuje tvar, hrúbku, výplň a prípadné ďalšie parametre krivky. Pretože obrazy založené na vektoroch nie sú vytvorené zo špecifického počtu bodov, môžu byť zmenené na väčšiu veľkosť a nestratia žiadnu kvalitu obrazu.

Vďaka tomu je vektorová grafika ideálna pre logá, ktoré môžu byť dostatočne malé na to, aby sa zobrazili na vizitke, ale môžu byť tiež prispôsobené na vyplnenie billboardu. Vektorový obraz môže byť konvertovaný na bitmapový obraz v ľubovoľnej veľkosti. Medzi najpoužívanejšie formáty ukladania vektorových obrazov patria AI (Adobe Illustrator Artwork), EPS (Encapsulated PostScript) [6], [8], [9].



Obr. 1.2: Porovnanie vektorového a bitmapového obrazu po zväčšení

2 Krivky

Krivky sa používajú v počítačovej grafike na mnohých miestach. Stretávame sa s nimi napríklad pri modelovaní v dvoch dimenziách, pri definíciách fontov, pri určovaní dráhy objektov v animáciách. Rôzne aplikácie použitia majú aj rôzne požiadavky a preto je táto oblasť pomerne široká. Už v roku 1959 používal P. de Casteljau matematický model kriviek, ktorý mu ich umožňoval jednoducho zadávať. Metódy tvarovania kriviek sa postupom času zdokonaľovali a dnes je k dispozícií široká škála kriviek pre rôzne použitie. Za rozvojom tejto oblasti stojí kvalitný matematický aparát a komerčný efekt, ktorý sa prejavuje hlavne v priemyselnom dizajne. V tejto kapitole sú popísané najčastejšie používané vektorové krivky ich vlastnosti a konštrukcia a algoritmy s nimi súvisiace.

2.1 Vektorové krivky

V počítačovej grafike sú krivky rozdelené podľa niekoľkých kritérií do rôznych skupín. V počítačovej grafike sú to typ krivky, spôsob vyjadrenia krivky a algoritmus akým je krivka modelovaná. Základné rozdelenie kriviek je na interpolačné a aproximačné krivky. Ďalšia vlastnosť pri vytváraní je racionalita a neracionalita krivky. Neracionálna krivka pracuje iba s pozíciou bodu.
Racionálna krivka používa nielen pozíciu bodu ale aj jeho váhu. Čím vyššiu má bod váhu oproti ostatným, tým viac sa k nemu modelovaná krivka približuje. V počítačovej grafike sú krivky obvykle vyjadrené ako sústava parametrov nejakej rovnice. Toto vyjadrenie môže byť v podstate trojitého druhu – explicitné, implicitné a parametrické.

- Explicitne kde je krivka zadaná ako spojitá funkcia v tvare $y = f(x)$ a býva orientovaná v smere rastúcej hodnoty x. Toto vyjadrenie nie je vhodné pre modelovanie kriviek, pretože je možné pracovať iba s krivkami, ktoré sú zároveň funkciami.
- Implicitne kedy je krivka zadaná v tvare $F(x, y) = 0$. Toto vyjadrenie nie je vhodné pre potrebu počítačovej grafiky, pretože neumožňuje postupný výpočet krivky.
- Parametricky je krivka vyjadrená $x = x(t), y = y(t)$ pričom parameter t sa volí z intervalu $< 0, 1 >$ kde krajné body intervalu sú vektorovým zápisom vyjadrené krajné body. Výhodou je závislosť iba na jednom parametri, nevýhodou je nerovnomerné rozloženie bodov na krivke. Krivku môžeme chápať ako dráhu pohybujúceho sa bodu, ktorého súradnice sú funkciami parametru t.

2.1.1 Modelovanie kriviek

Základným druhom parametrických kriviek používaných v počítačovej grafike sú krivky polynomiálne, ktoré môžme vyjadriť ako:

$$Q_n(t) = a_0 t + a_1 t + \cdots + a_n t^n. \tag{2.1}$$

Polynomiálne krivky môžeme takto jednoducho vyčísliť a sú jednoducho diferencovateľné. Z polynomiálnych kriviek je možné skladať krivky po častiach polynomiálne, čo sú krivky ktorých časti sú polynomiálnymi krivkami. Najčastejšie sa používajú krivky tretieho stupňa teda kubiky. Poskytujú širokú škálu tvarov, ich výpočet je nenáročný a je možné s nimi jednoducho manipulovať. Pri modelovaní je definovaných niekoľko riadiacich bodov, ktoré určujú riadiaci polygón a matematicky aparát z ich polohy určí priebeh krivky. Existujú dva základné spôsoby interpretácií riadiacich bodov, interpoláciou a aproximáciou.

Pri modelovanej krivke často sledujeme požadované vlastnosti.

- Invariancia k lineárnym transformáciám – pokiaľ prevedieme rotáciu riadiacich bodov, nebude to mať vplyv na tvar krivky.
- Konvexnosť obálky – sledujeme či vygenerovaná krivka leží v konvexnej obálke svojich riadiacich bodov.
- Lokalita zmien – sledujeme či pri zmene jedného riadiaceho bodu dôjde k zmene tvaru celej krivky, alebo len jej časti.
- Interpolácia krajných bodov – sledujeme či krivka prechádza alebo neprechádza počiatočným a koncovým bodom riadiaceho útvaru.

O tom aké vlastnosti vytváraná krivka má rozhoduje predovšetkým algoritmus pre vykreslenie krivky [1], [10].

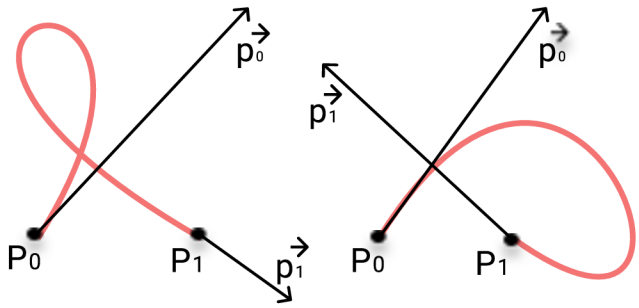
2.2 Interpolačné krivky

Zostrojenie interpolačnej krivky spočíva v konštrukcii krivky, ktorá prechádza danými uzlovými bodmi. V numerickej matematike sa na výpočet určitého integrálu používajú interpolácie pomocou polynómov. V počítačovej grafike sa interpolácia používa na zostrojovanie kriviek pre zadané riadiace body, ktorými interpolačná krivka prechádza. Problémom je určenie hodnoty parametra, pre ktorý získame daný riadiaci bod interpolačnej krivky. Najznámejšia interpolačná krivka je Fergusonova kubika [11].

2.2.1 Fergusonová kubika

Fergusonova kubika je často označovaná aj ako Hermitovská kubika. Tieto krivky sú určené dvomi riadiacimi bodmi P_0, P_1 a dvomi vektormi $\vec{p_0}$ a $\vec{p_1}$. Body P_0 a P_1

určujú polohu krivky v týchto bodoch krivka začína a končí. Smer a veľkosť vektorov určuje určuje mieru jej zahnutia. Čím je veľkosť vektoru väčšia, tým viac sa k nemu krivka priťahuje. Ak sú obidva vektory nulové, krivka sa stane úsečkou.



Obr. 2.1: Zmena tvaru Fergusonovej kubiky

Predpis pre výpočet Fergusonovej kubiky má tvar:

$$Q(t) = [t^3 t^2 t 1] \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \\ \vec{p_0} \\ \vec{p_1} \end{pmatrix} \tag{2.2}$$

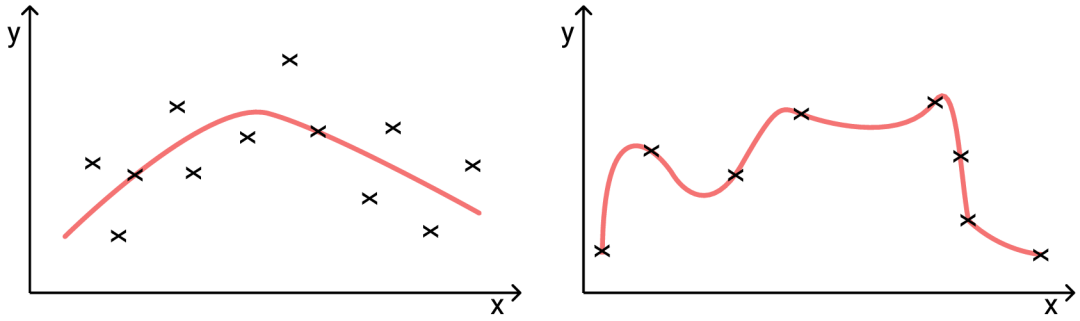
Ak rozpíšeme predchádzajúci vzťah do jednej rovnice dostaneme 5.12 vzťah

$$Q(t) = P_0 F_1(t) + P_1 F_2(t) + \vec{p_0} F_3(t) + \vec{p_1} F_4(t) \tag{2.3}$$

Najväčšia prednosť týchto kriviek je pri ich spájaní, keďže súčasťou definície kriviek sú vektory v jej koncových bodoch, môžeme spájanie realizovať jednoducho. Spojitosť pri nadväzovaní dvoch Fergusonových kubík docielime totožnosťou posledného bodu jedného segmentu a prvého bodu nasledujúceho segmentu. Naopak nevýhodou týchto kriviek je pomerne náročná zmena vektoru v troch rozmeroch [1].

2.3 Aproximačné krivky

Pre aproximačnú krivku máme daných niekoľko bodov, ale väčšinou nepožadujeme aby nimi prechádzala. Základ konštruovania takýchto kriviek položili Bézier a Casteljau v rokoch 1959–62 a spočíva v aproximácii danej krivky pomocou riadiaceho polygónu (lomenej čiary), ktorá určuje výsledný tvar krivky [11].



Obr. 2.2: Aproximačná a interpolačná krivka a ich riadiace body

2.3.1 Bézierova krivka

Bézierova krivka je pravdepodobne najpopulárnejšou krivkou používanou pre modelovanie v dvoch rozmeroch, mimo iné používaná aj na návrh počítačových fontov. Základ tejto krivky vytvoril na začiatku 60. rokov P. E. Bézier pri návrhu programu na návrh kriviek a plôch vo firme Renault. Je to parametrická krivka definovaná bodmi P_0 až P_n, kde body P_0 a P_n sú koncové body.
Ostatné body sa nazývajú riadiace a udávajú tvar krivky. Zakrivenie krivky sa chová ako keby bolo priťahované k riadiacim bodom. Bézierova krivka stupňa n je definovaná $n+1$ riadiacimi bodmi usporiadanými do riadiaceho polygónu. Krivka krajnými bodmi riadiaceho polygónu prechádza a všetky vnútorné body riadiaceho polygónu aproximuje. Je definovaná vzťahom:

$$Q(t) = \sum_{i=0}^{n} P_i B_i^n(t). \tag{2.4}$$

Kde B_i^n sú Bernsteinove polynómy n-tého stupňa.

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}; t \in <0,1> \; i = 0, 1, ..., n. \tag{2.5}$$

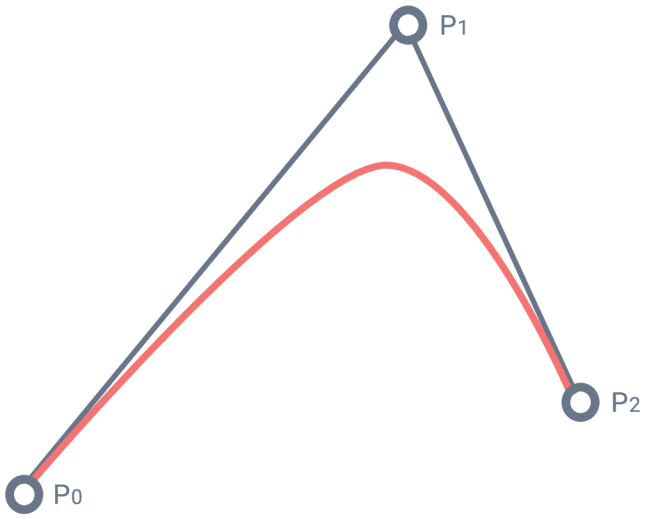
Horný index polynómu určuje stupeň a dolný index určuje poradie riadiaceho bodu s ktorým je polynóm asociovaný. Pre $n = 0$ je jediný polynóm B_0^0 rovný 1. Pre $n = 3$ sú polynómy štyri:

$$\begin{aligned} B_0^3(t) &= (1-3)^3 \\ B_1^3(t) &= 3t(1-3)^2 \\ B_2^3(t) &= 3t^2(1-3) \\ B_3^3(t) &= t^3. \end{aligned}$$

O Bernsteinových polynómoch platí:

- Na intervale [0,1] sú ich hodnoty kladné.
- Súčet všetkých polynómov dokopy sa rovná 1.
- Na intervale [0,1] majú maximum v čase $t = i/n$.

Nakoľko Bézierové krivky ležia v konvexnej obálke svojho riadiaceho polygónu, pri zmene polohy niektorého z riadiacich bodov P je celá krivka pozmenená čo je problém pri krivkách vyššieho stupňa. Preto sa na vytváranie zložitejších objektov využíva spájanie viacerých kriviek nižších stupňov. Pri spájaní segmentov zložených s Bézierových kriviek zaručíme spojitosť C^0 identickým posledným bodom riadiaceho polygónu segmentu Q_1 a prvým bodom segmentu Q_2. Pri dosadení $n = 1$ dostaneme lineárnu Bézierovú krivku prvého rádu, ktorá je úsečka určená bodmi P_0, P_1. Ak ležia všetky riadiace body Bézierovéj krivky v jednej priamke, zredukuje sa krivka na úsečku. Táto vlastnosť sa nazýva lineárna presnosť. Najčastejšie je používaná Bézierová krivka stupňa tri, teda Bézierová kubika [12], [13].



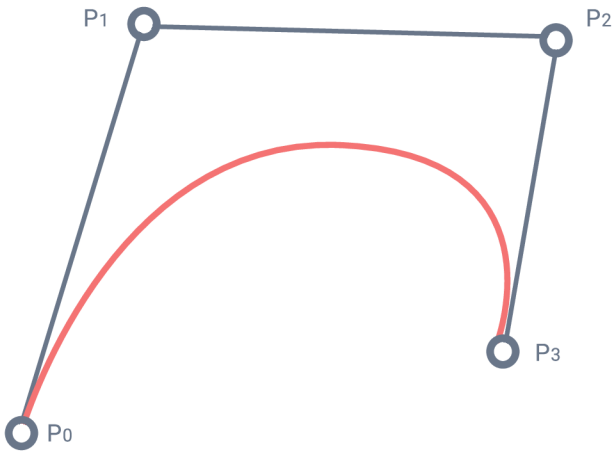
Obr. 2.3: Bézierova krivka

2.3.2 Bézierová kubika

Bézierova kubika je v počítačovej grafike najčastejšie používanou variantov Bézierovej krivky v praxi. Bézierová kubika je Bézierova krivka tretieho rádu teda n=3, je zadaná štyrmi bodmi P_0, P_1, P_2, P_3 pričom modelovaná krivka vychádza z bodu P_0 a končí v P_3 zvyšné body určujú zakrivenie [13]. Stupeň polynómu tri umožňuje rýchly výpočet výslednej krivky, čo je výhodné pre jej interaktívnu tvorbu. Pri vytváraná zložitejších tvarov sa používa spájanie viacerých kriviek nižších stupňou.

Je určená vzťahom

$$Q(t) = \sum_{i=0}^{3} P_i B_i^n(t). \tag{2.6}$$



Obr. 2.4: Bézierova kubika

2.3.3 Algoritmus de Casteljau

Algoritmus de Casteljau je rekurzívny algoritmus vytvorený Paulom de Casteljau. Určuje postup na zistenie hodnoty Bézierovej krivky v určenom časovom okamžiku. Vstupom tohto algoritmu sú riadiace body a čas t v ktorom chceme získať bod Bézierovej krivky. Je definovaný rovnicou

$$P_i^k = (1-t)P_{i-1}^{k-1} + tP_i^{k-1}, k = 1, \ldots, n, i = 0, \ldots, n-k, \tag{2.7}$$

pričom n značí počet riadiach bodov.

Princíp algoritmu je v podstate postupné delenie úsečiek riadiaceho polygónu na dve časti v pomere t a $1-t$. Počet nových vytvorených bodov sa v každom kroku zmenšuje o jedna až do chvíle pokiaľ neostane jediný bod ktorý je hľadaný.

Algoritmus priebieha v niekoľkých krokoch:

- Zvolíme $t \in [0, 1]$.
- Všetky ramená riadiaceho polygónu delíme v pomere $t : 1-t$, týmto krokom dostaneme body A_i.
- Body A_i spojíme úsečkami. Vznikne tak nový riadiaci polygón, ktorý má o jedno rameno menej ako pôvodný riadiaci polygón.
- Ramená vzniknutého polygónu znovu delíme a obdržíme body B_i.
- Predchádzajúce dva body opakujeme pokiaľ nie je riadiaci polygón tvorený jedným ramenom.

- Posledným delením získame výsledný bod.

Z popísaných krokov algoritmu je zrejmé, že pre nájdenie bodu na Bézierovej krivke stupňa n pomocou algoritmu de Casteljau je nutné urobiť celkom n delení. Algoritmus de Casteljau je možné taktiež využiť aj na rozdelenie Bézierovej krivky na dve rovnakého stupňa ako pôvodná [1].

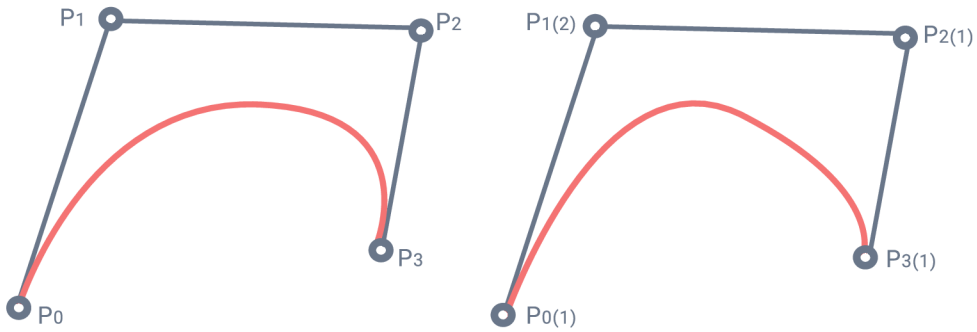
2.3.4 Racionálna Bézierova krivka

Pomocou Bézierových kriviek nie sme schopný zostrojiť tvary ako kružnica a elipsa. Pri zmene tvaru Bézierovej krivky je nutné zmeniť polohu bodu, čo nie je vždy žiadané. Tento problém sa prejavuje hlavne pri práci v trojrozmernom priestore. Racionálna Bézierová krivka je určená riadiacimi bodmi P_0 až P_n a naviac je každému bodu priradené číslo ω, čo je váha bodu ktorá určuje ako veľmi je krivka k tomu bodu priťahovaná vďaka čomu môžme zostrojiť aj zložitejšie tvary. [15]. Bézierova krivka stupňa n je potom určená vzťahom:

$$P(t) = \frac{\sum_{i=0}^{n} \omega_i P_i B_i^n(t)}{\sum_{i=0}^{n} \omega_i B_i^n(t)} = \sum_{i=0}^{n} P_i R_i^n \tag{2.8}$$

Kde R_i^n sú racionálne Bernsteinove polynomy.

$$R_i^n(t) = \frac{\omega_i B_i^n(t)}{\sum_{i=0}^{n} \omega_i B_i^n(t)} \tag{2.9}$$



Obr. 2.5: Bézierova krivka a Racionálna Bézierova krivka

2.3.5 Coonsová kubika

Coonsova kubika zaujíma vedľa Bézierových kriviek významné postavenie v histórií parametrických kriviek a plôch. Je bežne používaná pri tvorbe po častiach

skladaných polynomiálnych kriviek, úvahy o uniformnej a neuniformnej parametrizácií výrazne prispeli k zobecnenému pohľadu na vlastnosti a tvorbu polynomiálných bází a v neposlednej rade sa stala základom obecného aparátu NURBS.
Coonsová kubika je daná štyrmi riadiacimi bodmi usporiadanými do riadiaceho polygónu a je tvorená jediným segmentom, ktorý neprechádza žiadnym zo zadaných riadiacich bodov. Nech sú dané riadiace body P_0, P_1, P_2, P_3. Potom je vektorová rovnica Coonsovy kubiky

$$P(t) = C_0(t)P_0 + C_1(t)P_1 + C_2(t)P_2 + C_3(t)P_3, t \in [0, 1], \tag{2.10}$$

Coonsová kubika má zaujímavé vlastnosti, ktoré hrajú zásadnú roľu pri jej širokom uplatnení v modelovaní kriviek. Coonsová kubika neinterpoluje žiadny zadaný riadiaci bod. Je do dôsledok skutočnosti, že pre ľubovolnú hodnotu parametru $t \in [0, 1]$ nedosahuje žiadny Coonsonov polynóm hodnoty jedna. Najväčšia výhoda Coonsových kubík je zrejmá až v okamžiku, keď ich použijeme pre skladanie aproximačných kriviek [12].

2.3.6 Spline krivky

Spline krivka stupňa n je po častiach polynomiálna krivka triedy C^n. Termín spline pochádza z pružného pravítka, ktoré tieto krivky modelujú. Prirodzený spline je spline, ktorý interpoluje svoje riadiace body. Prirodzený kubický spline je interpolačná krivka skladajúca sa z polynomiálnych kriviek tretieho stupňa. Výpočet prirodzeného splinu vedie k výpočtu inverznej matice k matici $(m + 1) \times (m + 1)$, kde m je počet bodov riadiaceho polygonu.
Nevýhodou týchto kriviek je, že zmenou polohy jediného bodu sa zmení tvar celej krivky. V počítačovej grafike sú najčastejšie používané B–spline kubiky, ktoré nie sú prirodzenými spline krivkami, pretože sú aproximačné krivky. V nasledujúcich kapitolách sú vysvetlené jednoduchšie uniformné neracionálne B–spline a neuniformné neracionálne B–spline skrátene NURBS [1].

2.3.7 Uniformný kubický B-spline

Uniformný kubický B–spline sa tiež nazýva Coonsnov kubický B–spline. Vznikne naviazaním Coonsnových kubík týmto spôsobom: segment Q_i je určený bodmi $P_{i-2}, P_{i-1}, P_i, P_{i+1}$, teda teda tromi poslednými bodmi segmentu a jedným bodom nasledujúceho segmentu. Coonsov B–spline je určený $n \geq 4$ bodmi a skladá sa z $n - 3$ segmentov a je v uzloch C^2 spojitý.
Uzavretý Coonsov B–spline je možné vytvoriť opakovaním troch bodov riadiaceho polygónu na jeho konci. Riadiacim polygónom uzavretej krivky je teda postupnosť

bodov $(P_0, P_1, \ldots, P_n, P_0, P_1, P_2)$.
Medzi dôležité vlastnosti B–spline kriviek patrí invariancia voči otáčaniu, posunutiu a zmene mierky, B–spline krivka leží celá vo svojej konvexnej obálke a jej segmenty ležia v konvexných obálkach svojich riadiacich polygónov. Ak zmeníme polohu niektorého z riadiacich bodov, zmení sa len časť krivky.
Pri tvorbe B–spline môžeme niektoré riadiace body zopakovať a vytvoriť násobné body. Ak definujeme krivku postupnosťou $P_0, P_0, P_0, P_1, \ldots, P_{n-1}, P_n, P_n, P_n$, zaistíme, že výsledný kubický spline bude prechádzať krajnými bodmi svojho riadiaceho polygónu. Prvý segment bude úsečkou spájajúcou bod P_0 s bodom ležiacim v jednej šestine úsečky P_0P_1a analogicky posledný segment na konci riadiaceho polygónu. Toto je dôsledkom vlastností Coonsových kubík [1].

2.3.8 NURBS krivky

Neuniformné racionálne B–spline krivky patria medzi najpoužívanejšie krivky parametrických kriviek. Názov neuniformné je odvodený od vzdialenosti uzlov, ktorá nemusí byť pri týchto krivkách konštantná. NURBS krivky sa stali populárnymi hlavne vďaka podpore pre matematicky presné modelovanie kužeľosečiek, ktoré pomocou klasických Bézierových plôch nebolo možné zostrojiť.
Krivka je tvorená $n+1$ bodmi riadiaceho polygónu, k a uzlovým vektorom $\boldsymbol{U}$ dĺžky $n+k+1$. Uzlový vektor je vytvorený postupnosťou neklesajúcich čísel – uzlových hodnôt. Dôležitá je veľká flexibilita pri tvorbe NURBS, pretože okrem polôh samotných bodov je možné meniť aj ich váhu a globálny uzlový vektor ktorým sa určuje vzdialenosť riadiacich bodov v parametrickom priestore. Postupom času sa NURBS krivky stali štandardom pre systémy CAD, CAM a pre pokročilé vykresľovacie programy [14], [15]. NURBS krivka je určená vzťahom:

$$Q(t) = \frac{\sum_{i=0}^{n} \omega_i P_i P_{i,k}(t)}{\sum_{i=0}^{n} \omega_i N_{i,k}(t)}, \tag{2.11}$$

Behom výpočtu sa vyskytujú aj výrazy ktoré majú v menovateli nulu a tak je možné pri zápise využiť B-spline bázy

$$R(t) = \frac{\omega_i N_{i,k}(t)}{\sum_{j=0}^{n} \omega_j N_{j,k}(t)} \tag{2.12}$$

a krivku je potom možné zapísať jednoduchším sposôbom

$$Q(t) = \sum_{i=0}^{n} P_i R_{j,k}(t). \tag{2.13}$$

3 Rasterizácia objektov

V prvej kapitole boli vysvetlené obidva spôsoby grafického záznamu ich princípy a rozdiely. Vektorový záznam je flexibilnejší a z hľadiska úložného priestoru aj šetrnejší. Problém nastáva v zobrazení tohto záznamu na rastrových zobrazovacích zariadeniach, a preto je rasterizácia nevyhnutným krokom.

Rasterizácia je proces pri ktorom sa vektorové objekty prevádzajú do bitmapovej podoby opakom rasterizácie je vektorizácia. Výsledkom rasterizácie je nájsť všetky body reprezentujúce prevádzaný obraz a určiť ich správne parametre. Vektorový obraz je zostavený s množstva objektov ktoré ale predstavuje len niekoľko základných útvarov. Preto je postačujúcich len pár algoritmov na prevod týchto objektov a získame komplexný nástroj na prevod vektorového obrazu do bitmapovej podoby. Podrobný popis prevodu jednotlivých útvarov je popísaný v nasledujúcich kapitolách.

3.1 Rasterizácia úsečky

Úsečka je základným grafickým prvkom. Jej skladaním je možné vytvárať lomené čiary a zložitejšie útvary. Pretože je úsečka často používaná jej vykreslenie úsečky by malo byť čo najrýchlejšie a najefektívnejšie. Úsečka je obecne zadaná neceločíselnými súradnicami a smernica tiež nebýva celé číslo.
Aby algoritmy pre rasterizáciu pracovali rýchlo, je potrebné súradnice úsečky zaokrúhliť. Chyba ktorá vznikne pri zaokrúhlení súradníc úsečky nie je považovaná za významnú, pokiaľ nie je porušená náväznosť ťahu pri lomenej čiare. Úsečka je vzorkovaná konštantným krokom vždy v jednej ose x alebo y podľa sklonu určeného smernicou m.
Úsečku je možné zapísať dvoma spôsobmi:

1. súradnicami počiatočného a koncového bodu $[x_1, y_1]$, $[x_2, y_2]$
2. pomocou počiatočného bodu a smerového vektoru.

Smerová rovnica úsečky má tvar:

$$y = mx + 1 \tag{3.1}$$

Podľa veľkosti smernice m sa určí ku ktorej ose sa bude vzorkovať. Pre $|m| < 1$ je úsečka bližšie k ose x a bude vzorkovaná vzhľadom k nej. Pre $|m| > 1$ bude úsečka vzorkovaná k ose y. Pri tomto spôsobe rasterizácie je úsečka prevedená na takú postupnosť pixelov, ktorá je spojitá v ôsmich smeroch. Pre ľubovoľný pixel v tejto postupnosti platí, že jeho najbližšie susedné body môžu ležať v ôsmich smeroch. Táto vlastnosť je dôležitá pre algoritmy vyplňovania oblastí. Osa v ktorej smere sa bude vzorkovať sa volá riadiaca osa, druhá osa sa nazýva vedľajšia osa. Pre určenie

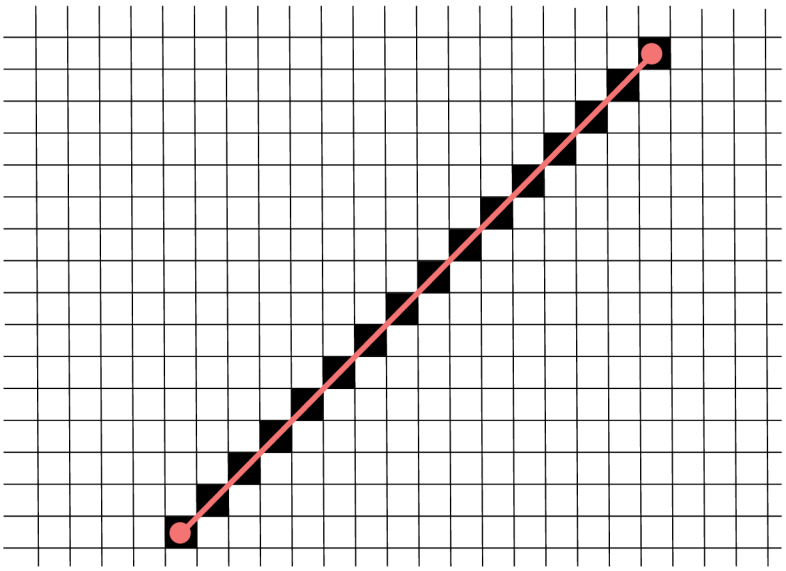
ktoré pixely budú vykreslené existuje niekoľko algoritmov. V nasledujúcich častiach sú popísané dva najpoužívanejšie. [1]

3.1.1 Algoritmus DDA

DDA (Digital Differential Analyzer) je jeden z prvých algoritmov používaných v počítačovej grafike. Je to prírastkový algoritmus pre výpočet bodov úsečky. Prírastok môže mať ľubovoľnú hodnotu, vzhľadom ku vzorkovaniu v rastri býva rovný jednému pixelu. Algoritmus je tvorený jediným cyklom, v ktorom je postupne k riadiacej ose pripočítaná jednotka a k zvyšnej súradnici pripočítaný neceločíselný prírastok. Pre každý vykreslený pixel je potrené súradnicu na vedľajšej ose zaokrúhliť. Postup spočíva v postupnom zvyšovaní o jeden pixel na súradnicovej ose x a dopočítaní súradnice sa ose y. Keďže sú súradnice pixelov celé čísla, musí sa vykonať zaokrúhlenie hodnôt, a pixel sa vykreslí. Postup algoritmu pre riadiacu os x je možné popísať nasledujúcim spôsobom:

1. Z koncových bodov $[x_1, y_1]$, $[x_2, y_2]$ urči smernicu m.
2. Nastav bod $[x, y]$ na hodnotu $[x_1, y_1]$.
3. Pokiaľ je $x \leq x_2$
   (a) Vykresli bod x
   (b) $x = x + 1$
   (c) $y = y + m$

Algoritmus pri výpočte nových súradníc pixelu využíva znalosť predchádzajúceho vzorku úsečky. Algoritmus je jednoduchý ale nevýhodou je že pracuje s reálnymi číslami a preto je relatívne pomalý. [1]



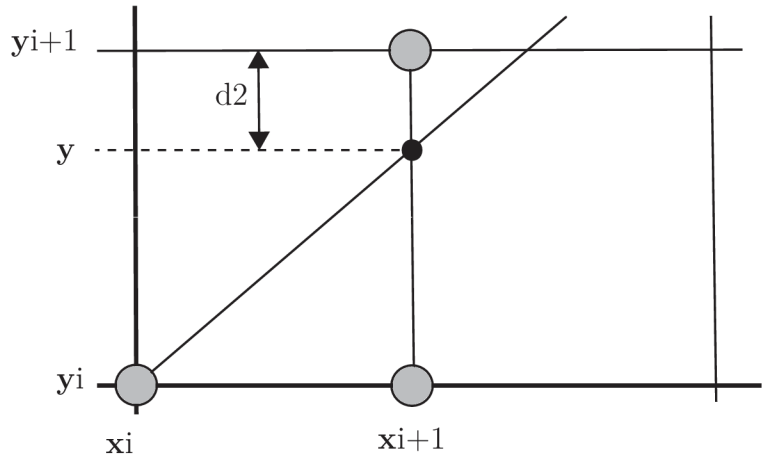
Obr. 3.1: Rasterizacia úsečky

3.1.2 Bresenhamov algoritmus pre kresbu úsečky

J. E. Bresenham vyvinul efektívnejší algoritmus na vykreslenie úsečky. Ako bolo spomenuté v predošlej kapitole v DDA algoritme sa počítajú súradnice v reálnej aritmetike a musíme ju zaokrúhliť. Bressenhamov algoritmus postupuje tak, aby sa operácie v desatinou čiarkou nepoužívali a preto je oproti DDA algoritmu rýchlejší. Pri rasterizácií po ose x môžeme umiestniť pixel iba na dve pozície – $[x + 1\ ,\ y]$, $[x + 1,\ y + 1]$. Rozdiely medzi súradnicou y stredu uvedených pixelov a skutočnou súradnicou y označíme d_1 a d_2 a môžeme ich celočíselne vypočítať z parametrickej rovnice úsečky. Ich rozdiel vyhodnotíme a pokiaľ je kladný $d_1 < d_2$ bližším pixelom je ten vo vzdialenosti d_2. Počiatočné súradnice sa zaokrúhľujú a vzniknutá nepresnosť nemá veľký vplyv na výsledok [1].

Postup Bressenhamovho algoritmu pre riadiacu os x vyzerá nasledovne:

- Z koncových bodov $[x_1, y_1]$ a $[x_2, y_2]$ urči konštanty
  $k1 = 2\Delta y$
  $k2 = 2(\Delta y - \Delta x)$
- Inicializuj rozhodovací člen p na hodnotu $2\Delta y - \Delta x$
- Inicializuj $[x, y]$ ako $[x_1, y_1]$
- Vykresli bod $[x, y]$
- Pokiaľ je $x \leq x_2$ opakuj:
  1. $x = x + 1$
  2. Ak je kladné p, tak $y = y + 1$ a $p = p + k2$
  3. Ak nie je kladné p, tak $p = p + k1$
  4. Vykresli bod $[x, y]$



Obr. 3.2: Princíp Bresenhamova algoritmu

3.2 Rasterizácia prerušovanej čiary

Algoritmy pre vykreslenie prerušovanej čiary pracujú dvomi spôsobmi. V prvom jednoduchšom spôsobe, sú postupne vypočítané súradnice všetkých pixelov napríklad Bressenhamovým algoritmom, a ich kresba je povolená alebo zakázaná podľa toho, či pixel patrí do plného alebo prázdneho úseku. Informácie o aktuálnom úseku sú uložené do dvoch premenných. Jedna obsahuje index aktuálneho úseku, druhá počet zvyšných pixelov v tomto úseku. Druhá premenná je zmenšená o jedno pri každom určení súradnice pixelu. Ak klesne jej hodnota na nulu, do prvej premennej sa uloží ukazateľ na ďalší úsek a do druhej sa uloží dĺžka úseku.
Tento spôsob generovania prerušovaných čiar je rýchly, pretože pracuje v celočíselnej aritmetike, ale vzhľad výslednej úsečky je ovplyvňovaný jej sklonom. Preto sa v praxi častejšie používa presnejší postup založený na výpočte dĺžok. Pomocou geometrických výpočtov sa určia koncové body jednotlivých úsekov a takto získané elementárne úsečky sa vykreslia bežným algoritmom. Pri kresbe prerušovanej čiary je žiadúce aby v oboch koncových bodoch úsečky boli zobrazené plné úseky. Jednoduché riešenie spočíva v tom, že posledný úsek sa nakreslí plnou čiarou bez ohľadu na definíciu vzoru [1].

3.3 Rasterizácia hrubej čiary

Podobne ako pri prerušovanej čiare rozlišujeme dva typy algoritmov pre kresbu hrubej čiary. Prvý, jednoduchý algoritmus je rýchly, ale vykazuje nepresnosti. Druhý presnejší algoritmus vyžaduje náročnejšie výpočty. Do skupiny jednoduchých algoritmov patrí upravený Bressenhamov algoritmus, pri ktorom je namiesto jedného pixelu vykreslených v každom kroku niekoľko pixelov nad sebou alebo vedľa seba. Výsledok tohto postupu však obsahuje dva negatívne javy.

- Hrúbka čiary sa mení podľa sklonu úsečky.
- Zakončenie čiar je neprirodzené. Je ostré a rovnobežné z niektorou súradnicovou osou.

Prvý negativny jav je možné čiastočne odstrániť prepočítaním požadovanej hrúbky čiary na počet pixelov v závislosti na sklone úsečky. Druhý negatívny jav pri malej hrúbke čiar nemusí pôsobiť rušivo, no pri veľkej hrúbke je výrazný. Kvalitnejšie programy pracujú s hrubou čiarou ako s vyplnenou plochou, vykresľovaniu tejto plochy sa venujú algoritmy pre vypĺňanie oblastí, ktorým sa táto práca nevenuje. Náročnejšie na výpočet ale vzhľadovo lepšie je zakončenie do oblúku. Toto ukončenie zároveň rieši problém vzhľadu pri nadväzovaní úsečiek [1].

3.4 Rasterizácia kružnice

Kružnica je ďalším útvarom ktorý je často využívaný na zápis objektu vo vektorovej grafike. Kružnica je presne definovaná stredom a polomerom. Stred kružnice je označený ako $S[s_x,s_y]$ a polomer ako R. Na rasterizáciu kružnice využíva Bresenhamov algoritmus. Matematický popis kružnice vyzerá takto:

$$F(x, y) : x^2 + y^2 - r^2 \tag{3.2}$$

Bresenhamov algoritmus pre kresbu kružnice

Rovnako ako pri rasterizácií úsečky môžeme získať body vykresľovanej kružnice iba pomocou celočíselnej aritmetiky. Skutočnosť že kružnica je stredovo súmerná nám umožňuje z jediného vypočítaného bodu kružnice odvodiť ďalších sedem bodov zámenou súradníc a zmenou ich znamienka. Pre rasterizáciu celej kružnice teda stačí vypočítať hodnoty súradníc bodov ležiacich v jednom oktante.
J. E. Bresenham vytvoril algoritmus, v ktorom vyriešil generovanie bodov na kružnici podobným spôsobom ako pri úsečke. Často s tento algoritmus nazýva midpoint algoritmus. Znamienko funkcie kružnice 3.2 určuje polohu bodu $[x, y]$ voči kružnici. Funkčná hodnota pre body vo vnútri kružnice je záporná, pre body mimo kružnicu je kladná. Funkcia 3.2 je vhodným kritériom pri zavedení rozhodovacieho členu. Obrázok 3.4 zobrazuje situáciu kedy bol bod $[x_i, y_i]$ určený ako bod skutočnej kružnice. Nasledujúci bod môže mať súradnice $[x_i + 1, y_i]$ alebo $[x_i + 1, y_i - 1]$. Na obrázku je naznačený bod ležiaci v polovici medzi uvedenými dvomi možnými bodmi tzv. midpoint. Dosadením do funkcie 3.2 zistíme podľa znamienka výsledku, či tento bod leží vo vnútri alebo mimo kružnice.
Výslednú hodnotu nazývame rozhodovací člen p_i:

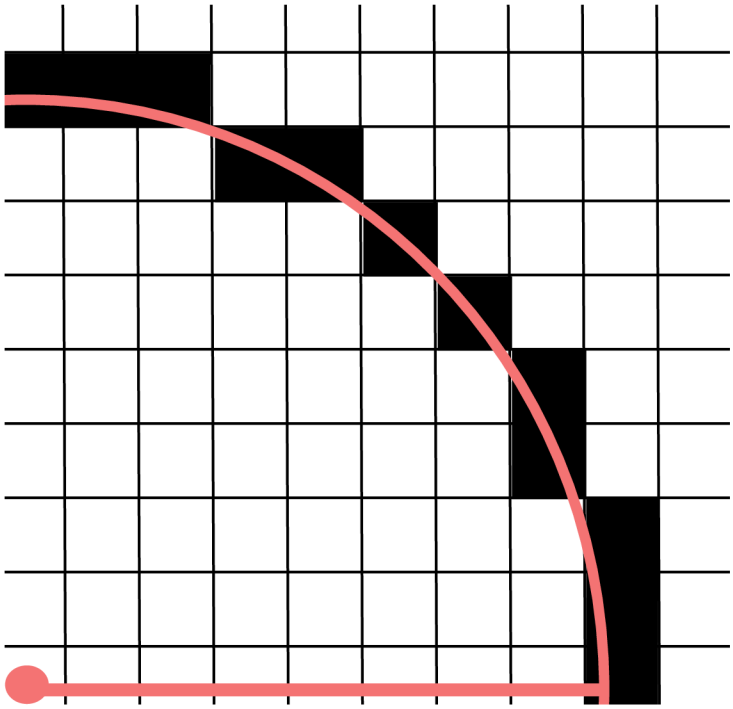
$$p_i = F(x_i + 1, y_i - \frac{1}{2}). \tag{3.3}$$

Ak je znamienko p_i záporné, bude pre ďalšiu kresbu vybratý bod s rovnakou súradnicou y_i inak bude nakreslený bod ležiaci o jeden pixel nižšie [1]. Algoritmus pracuje v niekoľkých krokoch:

1. Nastavíme dve pomocné premenné $dx = 3$ a $dy = 2r - 2$.
2. Nastavíme rozhodovací člen $p_i = 1 - r$.
3. Nastavíme $[x, y]$ ako $[0, r]$.
4. Vykreslíme bod $[x, y]$.
5. Pokiaľ je x $\leq$ y, opakujeme postup:
    - Vykreslíme 8 bodov symetrických s bodom $[x, y]$
    - Ak je $p > 0$ tak:

(a) $p = p - dy$
(b) $dy = dy - 2$
(c) $y = y - 1$

- $p = p + dx$
- $dx = dx + 2$
- $x = x + 1$



Obr. 3.3: Princíp rasterizácie kružnice

3.5 Rasterizácia elipsy

Elipsa je v mnohom podobná kružnici. Na zrýchlenie výpočtov pixelov pozdĺž elipsy môžeme využiť súmernosti elipsy. Je súmerná v štyroch kvadrantoch a teda stačí určiť polohu pixelov v jednom kvadrante. Zámenou súradníc alebo ich znamienok dosiahneme body v ostatných troch kvadrantoch. Ak počítame prvý kvadrant musíme ho rozdeliť na dve časti. V bode kde $2b^2x = 2a^2y$. V prvej časti v každom cykle narastá súradnica x o 1 a súradnica y o 1 podľa prediktoru. V druhej časti tomu je naopak, ale prediktor sa znovu inicializuje. Rozhodovanie pri výbere dvoch možných bodov záleží na tom či ich stred patrí do elipsy alebo nie. To odvodíme z funkcie 3.4 podobne ako pri kružnici [1]. Matematický predpis funkcie elipsy je nasledovný:

$$F(x, y) : b^2x^2 + a^2y^2 - a^2b^2 = 0. \tag{3.4}$$

3.6 Rasterizácia Bézierovej krivky

Rasterizácia Bézierovej krivky môže byť vykonaná dvoma spôsobmi. Jednoduchšia neadaptívna metóda spočíva v dosadzovaní parametru t do 2.6 a v spojovaní vypočítaných bodov úsečkami. Zmena parametru Δt je obyčajne konštantná. Tento spôsob nie je príliš efektívny, pretože nerešpektuje veľkosť zakrivenia. Naopak jeho veľkou výhodou je že generuje vopred známy počet úsečiek.
Ďalšou metódou ako vypočítať Bézierovú krivku stupňa n je použiť rekurzívny algoritmus de Casteljau ktorý je popísaný v kapitole 2.3.3. Bézierova krivka môže byť pomocou algoritmu de Casteljau rozdelená na dve časti v ľubovoľnom mieste tzn. pre zvolenú hodnotu parametru $t \in < 0, 1 >$. Najjednoduchšie je delenie v strede krivky teda v $t = 0{,}5$. Každé takéto rozdelenie generuje dva nové riadiace body. Pokiaľ tento postup opakujeme, konverguje polygón určený riadiacimi bodmi k Bézierovej krivke.
Tento rekurzívny proces zastavíme v okamihu kedy je vzdialenosť dvoch susedných bodov v riadiacom polygóne menšia ako dĺžka uhlopriečky pixelu pretože je ďalšie delenie zbytočné. Výhodou tejto metódy je že produkuje menší objem dát. Rovné úseky sú reprezentované ako úsečka, zatiaľ čo v krivých častiach je použité jemnejšie rozdelenie. [1]

4 Návrh grafických užívateľských rozhraní

Praktická časť práce je rozdelená na dve časti. Prvou častou je navrhnutie grafického užívateľského rozhrania pre dané aplety. Grafické užívateľské rozhranie je rozhranie ktorým užívateľ komunikuje s aplikáciou pomocou vizuálnych ovládacích prvkov. Je to rozhranie kde jednotlivé objekty sú reprezentované graficky a užívateľ s týmito prvkami interaguje pomocou vstupných zariadení, napríklad myši, touchpadu či klávesnice. Na vykonávanie akcií s objektami používame rôzne prvky ako tlačidlá, polia pre zadanie vstupných dát, a podľa potreby rôzne objekty pre manipuláciu s objektom. Návrhy grafických rozhraní majú jednotný vzhľad rozdelený na dve skupiny, prvou sú aplety pre rasterizáciu a druhou skupinou sú aplety pre krivky. V apletoch je využívaná priama manipulácia a užívateľ na základe jeho vstupov okamžite vidí zmeny. Pre návrh grafických rozhraní bol použitý nástroj Figma.

Figma

Figma je vektorový editor a návrhársky nástroj na báze cloudu, pracuje na akomkoľvek operačnom systéme, ktorý spúšťa webový prehliadač. Online charakter Figmy poskytuje niektoré zo silných stránok tohto nástroja. Jednou z týchto silných stránok je, že Figma umožňuje spoluprácu v reálnom čase. Všetci členovia tímu sa môžu prihlásiť do dizajnu v jeden moment a zároveň vykonať zmeny. Figma je v súčastnosti jeden s najpoužívanejších nástrojov pre vytváranie grafických návrhov s robustnými funkciami [16].

4.1 Návrh grafického užívateľského rozhrania apletu pre výuku Bézierovej krivky

Úlohou tohto apletu je priblížiť študentom ako vzniká Bézierová krivka tretieho stupňa. Bézierovú krivku tretieho stupňa nazývame Bézierova kubika. Hlavným ovládacím prvkom apletu je interaktívna plocha do ktorej pridávame štyri riadiace body krivky. Po určení štvrtého bodu sa vykreslí krivka. Riadiacimi bodmi krivky je možné manipulovať a tým meniť výsledný tvar krivky. Ovládanie apletu je navrhnuté tak aby bolo intuitívne a ovládacie prvky boli zrozumiteľné.
Ďalším ovládacím prvkom je tlačidlo na zobrazenie bodu na krivke ktorý slúži na zobrazenie bodu pre aktuálnu hodnotu t. K tomuto bodu patrí aj posuvný jazdec ktorým je možné nastavovať hodnotu t pre zobrazený bod. Ďalším ovládacím prvkom je prepínač zobrazenia Bernsteinových polynomov, ktorých súčtom dostávame funkčné hodnoty priebehov $x(t)$ a $y(t)$. Posledným ovládacím prvkom je tlačidlo pre

vynulovanie apletu, ktorým si užívateľ môže vymazať zvolené riadiace body a navrhnúť novú Beziérovu krivku. Návrh grafického užívateľského rozhrania pre tento aplet je na obrázku 4.1.

4.2 Návrh grafického užívateľského rozhrania apletu pre výuku Racionálnej Bézierovej krivky

Navrhnuté grafické rozhranie je takmer totožné s apletom pre Bézierovu krivku. Ako bolo spomenuté v kapitole o Racionálnej Beziérovej krivke 2.3.4 táto krivka sa od Bézierovej krivky líši tým že každý riadiaci bod má pridelene číslo, ktoré udáva váhu daného bodu. Čím väčšie je toto číslo, tým viac je krivka k tomuto bodu priťahovaná. Z toho vyplýva že grafické užívateľské rozhranie musí obsahovať všetky ovládacie prvky ako pri Bézierovej krivke čo je popísané v kapitole 4.1. Navyše pod ovládacím panelom sú vstupné polia pre zadanie váhy jednotlivých riadiacich bodov. ktorým užívateľ môže meniť váhu jednotlivých bodov. Tlačidlom reset je možné vymazať riadiace body a navrhnúť nové. Návrh grafického užívateľského prostredia pre tento aplet je zobrazený na obrázku 4.2.

4.3 Návrh grafického užívateľského rozhrania apletu pre výuku algoritmu de Casteljau

Grafické užívateľské prostredie pre aplet na výuku algoritmu de Casteljau je rozdelené na tri hlavné časti. Prvou častou je interaktívna kresliaca plocha. Keďže je aplet vytvorený pre Bézierová kubiku je potrebné zadať štyri riadiace body. Po zadaní týchto bodov je vykreslená Bézierova kubika.
Druhou častou je panel s ovládacími prvkami. Posuvným jazdcom je možné meniť hodnotu parametru t čo je pomocná premenná používaná na výpočet. Zmenou tohto parametru sa vykonáva nový výpočet pre hľadaný bod. Prvým tlačidlom je možné spustiť animáciu, ktorá užívateľovi zobrazuje ako pracuje algoritmus de Casteljau pri hľadaní bodu od $t = 0$ po $t = 1$. Druhým tlačidlom má užívateľ možnosť zobraziť algoritmus, užívateľ pohybom posuvného jazdca môže meniť hodnotu t a sledovať ako pracuje algoritmus. Poslednou častou užívateľského rozhrania je výpočet bodov. V tejto časti sú zobrazené funkčné hodnoty bodov ktoré sa používajú na výpočet hľadaného bodu. V prvom riadku sú zobrazené súradnice riadiacich bodov. V druhom riadku sú zobrazené súradnice čiernych bodov. V tretom riadku sú súradnice modrých bodov. V poslednom riadku sú súradnice hľadaného bodu v zadanom čase

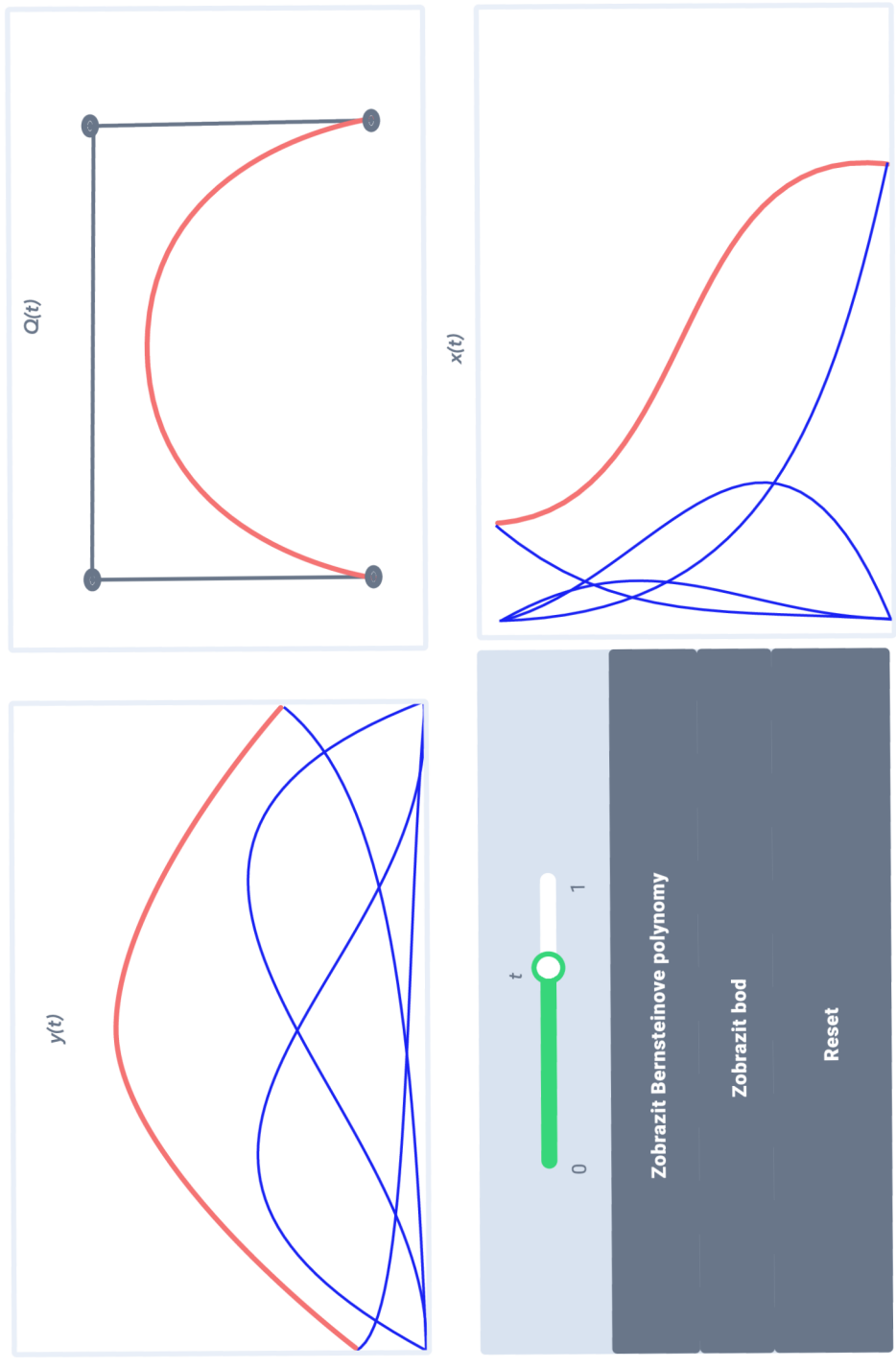
t. Posledným prvkom je tlačidlo na vynulovanie riadiacich bodov čo umožňuje zadať nové riadiace body. Návrh grafického užívateľského rozhrania je zobrazený na obrázku 4.3.

4.4 Návrh grafického užívateľského rozhrania apletu pre výuku Rasterizácie úsečky

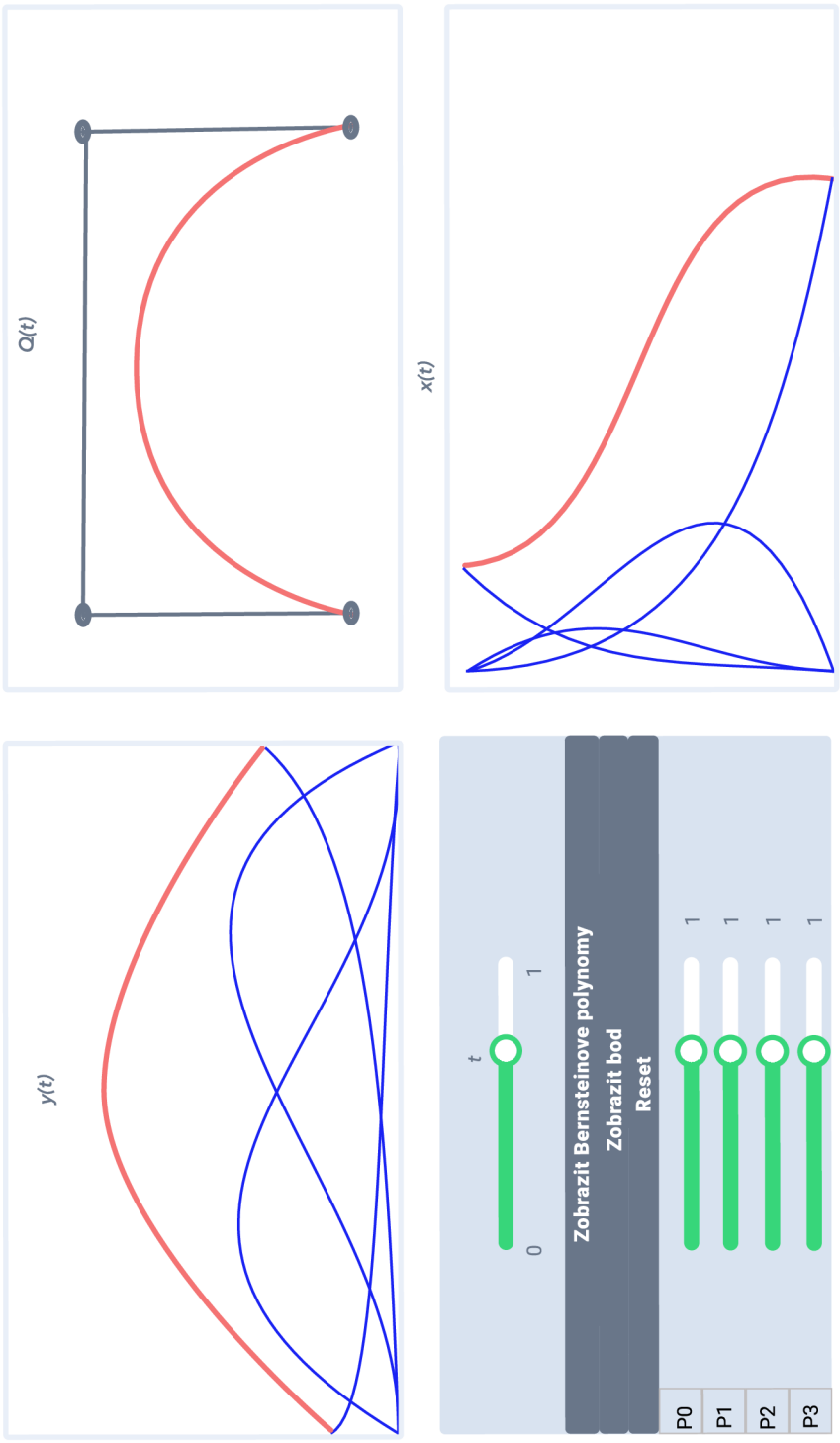
Tento aplet užívateľovi umožňuje priblížiť princíp rasterizácie úsečky. Pri rasterizácií úsečky sa snažíme o čo najlepší vizuálny výsledok. Dôležité je rozhodnúť ktorý pixel leží bližšie k teoretickej úsečke. Existujú dva algoritmy na rasterizáciu úsečky. Jednoduchší algoritmus DDA a rýchlejší Bresenhamov algoritmus, obidva sú popísané v kapitole 3.1 o rasterizácií úsečky. Návrh grafického užívateľského rozhrania je v tomto prípade v celku jednoduchý.
Hlavnou častou je interaktívna kresliaca plocha ktorá je založená na pomyselnej sieti, ktorá zobrazuje jednotlivé pixely. Pixely sú niekoľko krát zväčšené aby postup rasterizácie bol viditeľný. Užívateľ zvolí dva body, prvý označuje začiatok úsečky a druhy označuje koniec úsečky. Pomocou Bresenhamovho algoritmu je vypočítaný výsledný tvar úsečky po pixeloch. Dvojica tlačidiel umožnuje užívateľovi zväčšiť alebo zmenšiť veľkosť pixelov pre vhodnejšie zobrazenie postupu rasterizácie. Tlačidlo reset umožňuje užívateľovi vymazať zadané body a určiť nové body. Návrh grafického užívateľského rozhrania je zobrazený na obrázku 4.4.

4.5 Návrh grafického užívateľského rozhrania apletu pre výuku Rasterizácie kružnice

Aplet užívateľovi približuje postup rasterizácie kružnice. Podobne ako pri rasterizácií úsečky je použitý Bresenhamov algoritmus, ktorý je veľmi efektívny. Návrh grafického užívateľského rozhrania je pomerné jednoduchý a veľmi podobný ako to bolo pri úsečke. Hlavnou časťou je interaktívna kresliaca plocha s pomyselnou sieťou na kreslenie ktorá zobrazuje pixely mnoho krát zväčšené aby užívateľ videl postup rasterizácie.
Užívateľ zadá dva body, prvý určuje stred kružnice a druhy vzdialenosť od stredu teda polomer. Po zadaní týchto bodov je vykreslená kružnica. Dva prepínače umožňujú zobraziť teoretickú kružnicu a spojnicu. Posuvný jazdec ponúka možnosť určiť veľkosť pixelov kresliacej plochy. Tlačidlo reset umožňuje vymazať zadané body a určiť nové pre Ďalšiu kružnicu. Návrh grafického užívateľského prostredia je na zobrazený na obrázku 4.5.

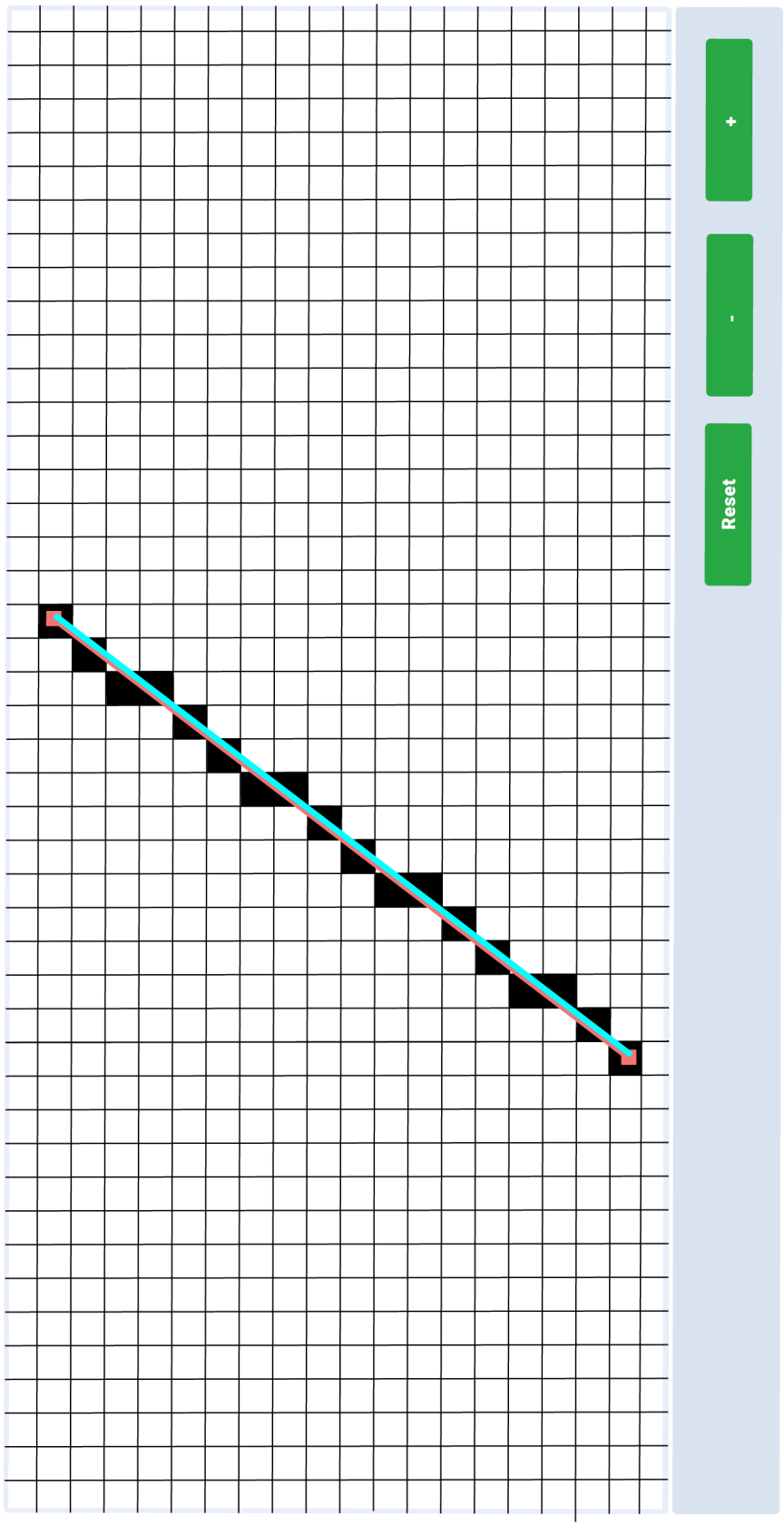


Obr. 4.1: GUI pre aplet Bézierova krivka

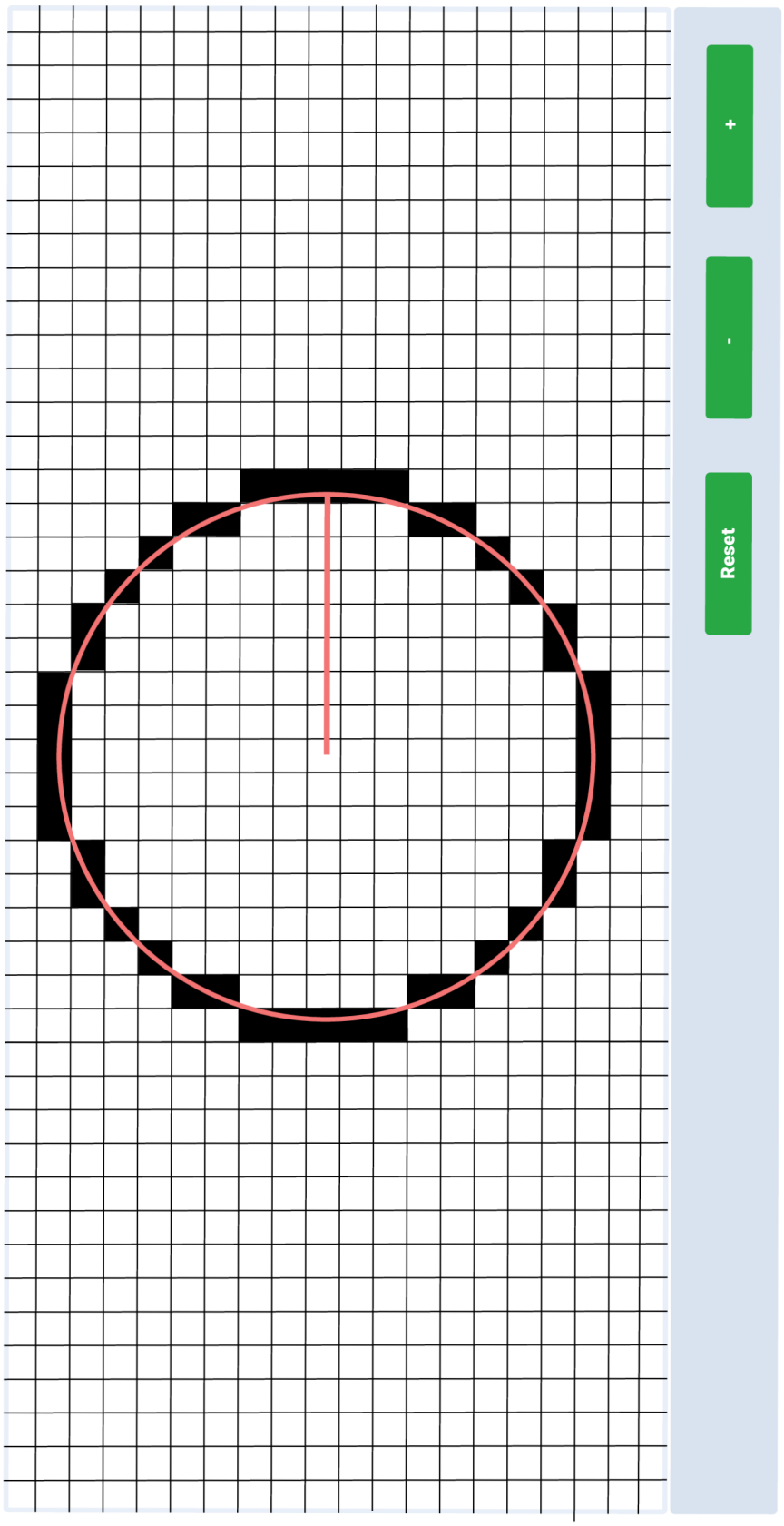


Obr. 4.2: GUI pre aplet Racionálna Bézierova krivka

Obr. 4.3: GUI pre aplet de Casteljau



Obr. 4.4: GUI pre aplet Rasterizácia úsečky



Obr. 4.5: GUI pre aplet Rasterizácia kružnice

5 Implementácia apletov

Aplety sú implementované pomocou jazykov HTML a Javascript, a pre ich používanie stačí užívateľovi iba obyčajný webový prehliadač. Pomocou jazyka CSS sú jednotlivým HTML elementom pridané štýly, aby výsledné aplety naozaj vyzerali tak ako boli v predchádzajúcej kapitole vytvorené ich GUI. Taktiež bol použitý Bootstrap čo je bezplatný a open-source CSS framework. Obsahuje CSS a JavaScriptove šablóny pre typografiu, formuláre, tlačidlá, navigáciu a ďalšie komponenty rozhrania. Na prácu s dátami bola použitá Javascriptová knižnica D3 ktorá je popísaná neskôr v tejto kapitole. Ovládanie jednotlivých apletov je veľmi intuitívne, a pri každom je na stránke popísané jeho ovládanie. V nasledujúcich kapitolách je základne popísaná implementácia jednotlivých apletov. Vytvorené aplety boli zároveň vložené do web stránok vedúceho práce, ktoré používa na výuku počítačovej grafiky. Pre veľkosť zdrojových súborov sú v popísané len časti, ktoré priamo súvisia s výpočtom potrebných údajov. Ďalšie časti ktoré slúžia na zobrazenie dát a celé zdrojové súbory sú na priloženom CD. Aplety sú zároveň dostupné na http://dp.matuslipa.sk/.

HTML 5

Hypertext Markup Language verzie 5 je značkovací jazyk pre štruktúru a prezentáciu obsahu www stránok. HTML5 je najnovší štandard, ktorý definuje HTML. Tento pojem predstavuje dva rôzne koncepty. Je to nová verzia jazyka HTML s novými prvkami, atribútmi a správaním a väčšia sada technológií, ktoré umožňujú vytváranie rôznorodejších a výkonnejších webových lokalít a aplikácií. S HTML5, prehliadače ako Firefox, Chrome, Explorer, Safari a ďalšie, môžu vedieť, ako zobraziť konkrétnu webovú stránku, vedieť, kde sú prvky, kam umiestniť obrázky a kam umiestniť text [17].

Javascript

Javascript je multiplatformový objektovo orientovaný skriptovací jazyk. Dnes sa používa hlavne ako interpretovaný jazyk pre WWW stránky, ktorý je často vkladaný priamo do HTML kódu web stránky. Najčastejšie sú ním ovládané ovládacie prvky grafického užívateľského prostredia a tvorené animácie spojené s týmito grafickými prvkami. Program v Javascripte sa spúšťa na strane klienta až po stiahnutí WWW stránky z internetu, ale je možné ho použiť aj na strane serveru. Keďže pracuje na strane klienta plynú z toho aj bezpečnostné obmedzenia ako to, že nemôže pracovať so súbormi aby tak neohrozil bezpečnosť užívateľa [18].

D3.js

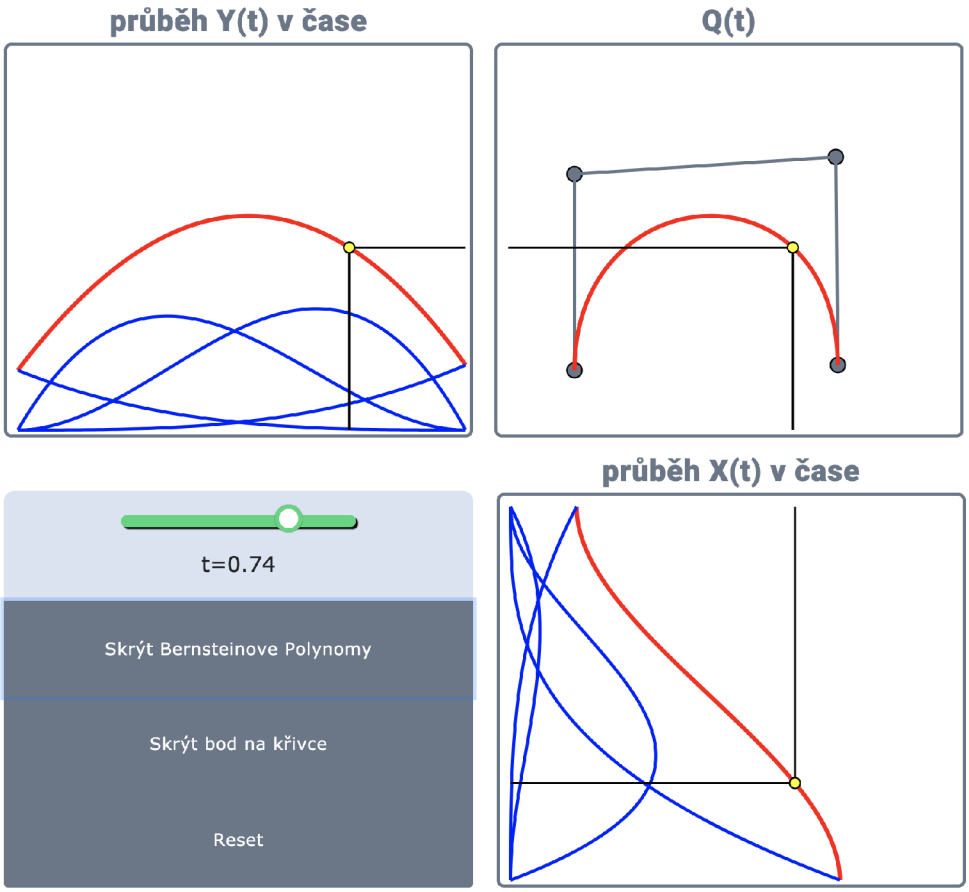
D3 je Javascriptová knižnica na manipuláciu s dokumentmi založených na údajoch. Je to extrémne výkonný vizualizačný nástroj na vytváranie interaktívnych vizualizácií dát. Využíva moderné webové štandardy: SVG, HTML a CSS na vytvorenie vizualizácie dát. D3 umožňuje manipulovať s DOM (Document Object Model) na základe údajov. Umožňuje dátam dynamicky generovať elementy a aplikovať štýly na elementy. S minimálnou réžiou je D3 extrémne rýchly, podporuje veľké súbory údajov a dynamické správanie pre interakciu a animáciu [19].

5.1 Aplet pre Bézierovú krivku

Tento aplet umožňuje vykreslenie Bézierovej krivky tretieho stupňa. Teória k tejto krivke pola vysvetlená v kapitole 2.3.1. Aplet za skladá zo štyroch častí, interaktívnej kresliacej plochy v pravo hore, na ľavo a dole od tejto plochy su zobrazené priebehy zadanej krivky. Poslednou časťou je ovládací panel. Kliknutím do plochy v pravo hore pridávame postupne jednotlivé riadiace body. Ťahaním za jednotlivé riadiace body je možné editovať ich polohu. Po zadaní riadiacich bodov je vykreslená vypočítaná krivka $Q(t)$. V ľavo a dole vykreslenej krivky je môžme vidieť priebehy $Y(t)$ a $X(t)$ z ktorých sa výsledná krivka skladá. Posuvným jazdcom je možné meniť hodnotu parametru t.

Tlačidlom Zapnout bod na křivce sa zapne bod v zadanom čase t a zmenou tohto parametru je možné sledovať v grafoch že sa naozaj jedná o rozloženie krivky. Prvým tlačidlom je možné zobraziť Bernsteinove polynomy, ktorých súčtom získame krivku. Tlačidlom reset je možné vynulovať zadané riadiace body a zadať nové. Základ tohto apletu je v celku jednoduchá HTML šablóna do ktorej pomocou Javascriptu pridávame jednotlivé prvky potrebné k zobrazeniu dát. Do tejto základnej šablóny sa pri načítaní stránky pridajú svg elementy ktoré sa používajú na kreslenie manipuláciu s objektmi v nich a animácie. Implementovaný aplet je na obrázku 5.1.

Pri prvom výpočte krivky a pri každej úprave polohy riadiacich bodov je volaná funkcia `update()`, ktorá sa stará o obnovu údajov a zobrazenia. Jej hlavnou častou je však volanie funkcie na získanie novej krivky `getCurve()`. Vstupným parametrom tejto funkcie je počet riadiacich bodov. Funkcia vymaže predchádzajúcu krivku. Postupným inkrementovaním hodnoty parametru t získava hodnoty novej krivky. Pri každej inkrementácií t sa volá funkcia `getLevels()`, ktorej vstupom je počet riadiacich bodov a aktuálna hodnota zvyšujúceho sa parametru t. Táto funkcia vykoná výpočet podľa vzorcu 2.6 a vráti súradnice bodu na krivke pre aktuálnu hodnotu parametru t. Vypočítanú hodnotu dosadíme do výslednej krivky, jej časových priebehov a hodnoty Bernsteinových polynomov.



Obr. 5.1: Aplet pre Bézierovu krivku

```
function update() {
  interpolation = qpriebeh.selectAll("g")
    .data(function (d) {
      return getLevels(d, t);
    });
    }
function getCurve(d) {
 curve = bezier[d] = [];
 for (var t_ = 0; t_ <= 1.01; t_ += delta) {
   var x = getLevels(d, t_);
      curve.push(x[1]);
      ycurve.push([t_ * 420, x[1].y]);
      xcurve.push([x[1].x, t_ * 350]);
   b1Ycurve.push([t_ * w,h- (1-t_)**3 *
   (h-interpolation.data()[0][0].y)]);
 }return [curve];
```

```
 }

function getLevels(d, t_) {
  getLevelsX = [points.slice(0, d)];
  var resultX = (points[0].x*(1-t_)**3 + points[1].x*(3*t_
  *(1-t_)**2 )
  +points[2].x*(3*(t_**2)*(1-t_))+points[3].x*t_**3);
  var resultY = (points[0].y*(1-t_)**3 + points[1].y*(3*t_
  *(1-t_)**2 )
  +points[2].y*(3*(t_**2)*(1-t_))+points[3].y*t_**3);
  getLevelsX.push({x: resultX, y:resultY});
  return getLevelsX;
}
```

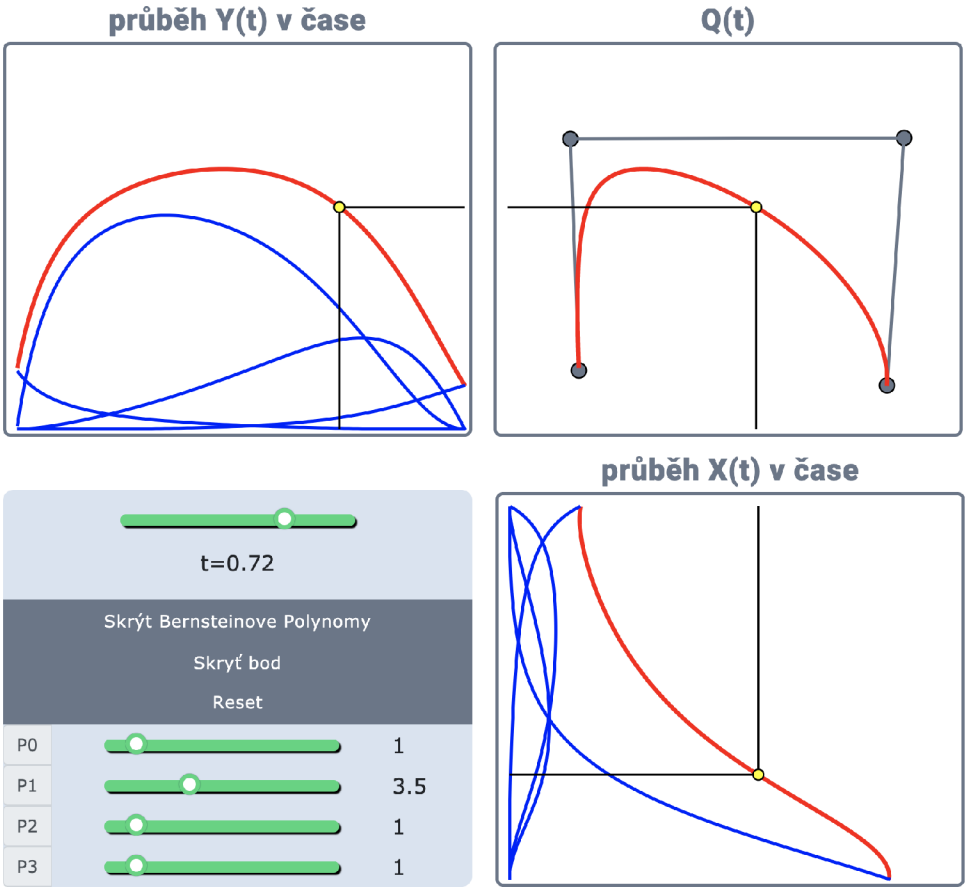
5.2 Aplet pre Racionálnu Bézierovú krivku

Aplet pre racionálnu Bézierovku krivku umožňuje študentom vyskúšať si vytváranie tejto krivky. Teória k tejto krivke je popísaná v kapitole 2.3.4. Štruktúra apletu je takmer identická so štruktúrou apletu pre Bézierovu krivku. Ako je popísané v teoretickej časti, riadiace body tejto krivky majú okrem súradníc pridelenú aj váhu. Aplet za skladá zo štyroch častí, interaktívnej kresliacej plochy v pravo hore, na ľavo a dole od tejto plochy sú zobrazené priebehy zadanej krivky v ľavo na dolnej strane je umiestnený ovládací panel. Kliknutím do plochy v pravo hore pridávame postupne jednotlivé riadiace body. Ťahaním za jednotlivé riadiace body je možné editovať ich polohu. Po zadaní riadiacich bodov je vykreslená vypočítaná krivka $Q(t)$. Vľavo a dole vykreslenej krivky je môžme vidieť priebehy $Y(t)$ a $X(t)$ z ktorých sa výsledná krivka skladá. Posuvným jazdcom je možné meniť hodnotu parametru t. Tlačidlom Zapnout bod na křivce sa zapne bod v zadanom čase t a zmenou tohto parametru je možné sledovať v grafoch že sa naozaj jedná o rozloženie krivky. Prvým tlačidlom je možné zobraziť Bernsteinove polynomy, ktorých súčtom získame krivku. Tlačidlom reset je možné vynulovať zadané riadiace body a zadať nové. Dôležitou súčasťou sú vstupné polia pre zadanie váh jednotlivých riadiacich bodov. Implementovaný aplet je na obrázku 5.2.

Po zadaní posledného riadiaceho bodu je vykreslená krivka. Prednastavené hodnoty váh jednotlivých riadiacich bodov sú 1. Pri každej zmene polohy riadiacich bodov alebo zmene váhy niektorého z bodov je vykonaný nový výpočet krivky. Výpočet začína zavolaním funkcie `update()`. Táto funkcia volá funkciu `getCurve()` a predá jej počet riadiacich bodov. Postupným inkrementovaním hodnoty parametru t získava hodnoty novej krivky. Pri každej inkrementácií t sa volá funkcia `getLevels()`.

Táto funkcia podľa vzorcu 2.8 vykoná výpočet bodu na krivke a vráti jeho súradnice. Funkcia `getCurve()` vrátené hodnoty dosadí do krivky, kriviek časových priebehov a Bernsteinovych polynomov.

```
function update() {
  interpolation = qpriebeh.selectAll("g")
    .data(function (d) {
      return getLevels(d, t);
    });}
function getCurve(d) {
curve = bezier[d] = [];
for (var t_ = 0; t_ <= 1.01; t_ += delta) {
  var x = getLevels(d, t_);
  curve.push(x[1]);
  ycurve.push([t_ * 420, x[1].y]);
  xcurve.push([x[1].x, t_ * 350]);
  b1Ycurve.push([t_ * w,h-
  ((1-t_)**3 *weight[0]*(h-interpolation.data()[0][0].y))
  /(weight[0]*(1-t_)**3 + weight[1]*3*t_*(1-t_)**2
  + weight[2]*(3*t_**2 * (1-t_))+ weight[3]*t_**3)]);
}}return [curve];}
function getLevels(d, t) {
  getLevelsX = [points.slice(0, d)];
  var t2 = t**2;
  var t3 = t**3;
  var s = 1 - t;
  var s2 = s**2;
  var s3 = s**3;
  var resultX =
  (points[0].x*weight[0]*s3 + points[1].x*weight[1]*3*t*s2
  + points[2].x*weight[2]*3*t2*s + points[3].x*weight[3]*t3)
  /(weight[0]*s3 + weight[1]*3*t*s2
  + weight[2]*3*t2*s + weight[3]*t3);
  var resultY =
  (points[0].y*weight[0]*s3 + points[1].y*weight[1]*3*t*s2
  + points[2].y*weight[2]*3*t2*s + points[3].y*weight[3]*t3)
  / (weight[0]*s3 + weight[1]*3*t*s2
  + weight[2]*3*t2*s + weight[3]*t3);
  getLevelsX.push({x: resultX, y:resultY});
  return getLevelsX;}
```



Obr. 5.2: Aplet pre Racionálnu Bézierovu krivku

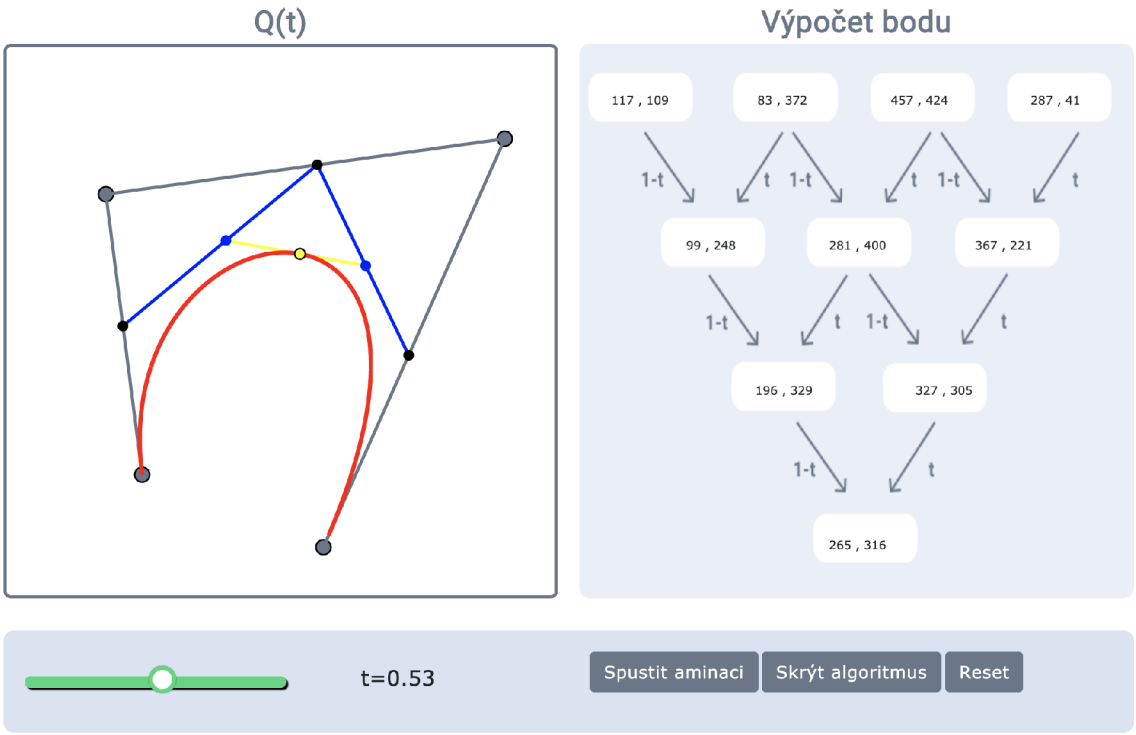
5.3 Aplet pre algoritmus deCasteljau

Tento aplet približuje fungovanie algoritmu de Casteljau pri hľadaní bodu na Bézierovej krivke. Algoritmus de Casteljau je popísaný v kapitole 2.3.3. Aplet sa skladá z troch častí. Prvou je interaktívna kresliaca plocha kde kliknutím vložíme štyri riadiace body Bézierovej krivky. Po vložení posledného je vykreslená krivka. V pravej časti apletu je zobrazená schéma výpočtu hľadaného bodu. V spodnej časti apletu sa nachádza ovládací panel posuvným jazdcom je možné meniť hodnotu parametru t a sledovať pohybujúci sa bod na krivke. Tlačidlom Zobraziť algoritmus sa nám pri krivke zobrazí grafický postup hľadania bodu pre lepšie pochopenie algoritmu. Kliknutím na tlačidlo Spustit animaci sa spustí animácia ktorá nám zobrazí priebeh algoritmu od $t = 0$ po $t = 1$. Posledným ovládacím prvkom je tlačidlo Reset ktoré vymaže zadané riadiace body a môžeme zadať nové.

Pri zmene polohy riadiacich bodov je volaná funkcia `update()`. Funkcia volá `getCurve()` a parameter je počet riadiacich bodov. Funkcia `getCurve()` volá funkciu `getLevels()`, ktorá pre riadiace body volá funkciu `interpolate()`. Táto fun-

kcia nám vráti súradnice bodu na krivke v zadanom čase t a súradnice bodov ktoré vznikajú v jednotlivých krokoch algoritmu. Vďaka týmto údajom môžme sledovať ako pracuje algoritmus v jednotlivých krokoch pri zmene parametru t. Ktoré môžeme vidieť na obrázku 5.3 ako čierne modré a žltý bod.

```
function update() {
 interpolation = qpriebeh.selectAll("g")
   .data(function (d) {
     return getLevels(d, t);
   });
function getCurve(d) {
  curve = bezier[d];
  if (!curve) {
    curve = bezier[d] = [];
    for (var t_ = 0; t_ <= 1; t_ += delta) {
      var x = getLevels(d, t_);
      curve.push(x[x.length - 1][0]);
    }}return [curve];
    }
var getLevelsX;
function getLevels(d, t_) {
  if (arguments.length < 2) t_ = t;
  getLevelsX = [points.slice(0, d)];
  for (var i = 1; i < d; i++) {
    getLevelsX.push(
    interpolate(getLevelsX[getLevelsX.length - 1], t_));
  }
  return getLevelsX;
}
function interpolate(d, p) {
  if (arguments.length < 2) p = t;
  var r = [];
  for (var i = 1; i < d.length; i++) {
    var d0 = d[i - 1], d1 = d[i];
    r.push({
    x: d0.x + (d1.x - d0.x) * p,
    y: d0.y + (d1.y - d0.y) * p});
  }
  return r;
}
```



Obr. 5.3: Aplet pre algoritmus de Casteljau

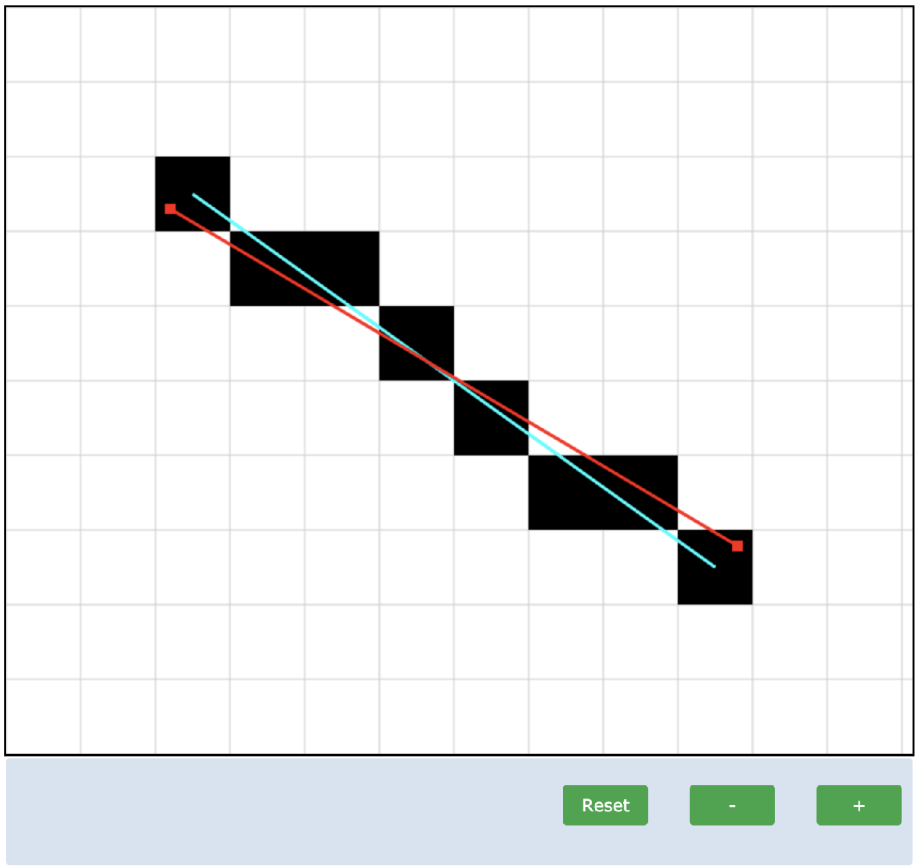
5.4 Aplet pre rasterizáciu úsečky

Aplet na rasterizáciu úsečky umožňuje študentom si otestovať pracovanie Bresenhamovho algoritmu popísaného v kapitole 3.1.2 v praxi. Aplet má jednoduchú štruktúru. Hlavnou častou je interaktívna kresliaca plocha v ktorej je vytvorená pomyselná sieť pixelov kde sa zobrazuje ktoré pixely sú vyfarbené. Pod touto plochou je ovládací panel. Kliknutím do plochy zadáme začiatok a koniec červenej skutočnej úsečky a je vykreslená aj svetlo modrá toretická úsečka. Úsečku je možné meniť v reálnom čase ťahaním za konce červenej čiary alebo kliknutím v blízkosti konca úsečky. Pri zmene súradníc niektorého bodu úsečky je hneď prepočítaná. Tlačidlami v ovládacom paneli môžme zmeniť veľkosť pixelov a preskúmať algoritmus detailnejšie. Tlačidlom reset vynulujeme zadanú úsečku a nastavíme preddefinovanú veľkosť pixelov. Implementovaný aplet je na obrázku 5.4.

Bressenhamov algoritmus zobrazuje nasledujúca ukážka kódu, konkrétne funkcia `draw_line()`. Vstupom sú súradnice začiatočného a konečného bodu úsečky. Po nastavení premenných nasleduje rozhodovanie o tom ktorý pixel bude vykreslený. Dôležitou je taktiež funkcia `put_pixel()`, ktorá vykreslí na základe výpočtu Bressenhamovho algoritmu pixel.

```
function draw_line(ctx, x0, y0, x1, y1) {
  var dx = x1 - x0;
```

```
  var dy = y1 - y0;
  var inc_x = (dx >= 0) ? +1 : -1;
  var inc_y = (dy >= 0) ? +1 : -1;
  dx = (dx < 0) ? -dx : dx;
  dy = (dy < 0) ? -dy : dy;
  if (dx >= dy) {
    var p = 2 * dy - dx;
    var x = 0;
    var y = 0;
    for (i = 0; i <= dx; i++) {
      put_pixel(ctx, x + x0, y + y0, "black");
      if (p < 0) {
        p += 2 * dy;
        x += inc_x;
      }
      else {
        p += 2 * dy - 2 * dx;
        x += inc_x;
        y += inc_y;
      }}}
  else {
    var p = 2 * dx - dy;
    var x = 0;
    var y = 0;
    for (i = 0; i <= dy; i++) {
      put_pixel(ctx, x + x0, y + y0, "black");
      if (p < 0) {
        p += 2 * dx;
        y += inc_y;}
      else {
        p += 2 * dx - 2 * dy;
        x += inc_x;
        y += inc_y;
      }}}}
function put_pixel(ctx, x, y, color) {
 ctx.save();
 ctx.fillStyle = color;
 ctx.fillRect(x*pixel_size,y*pixel_size,pixel_size,pixel_size);
 ctx.restore();}
```



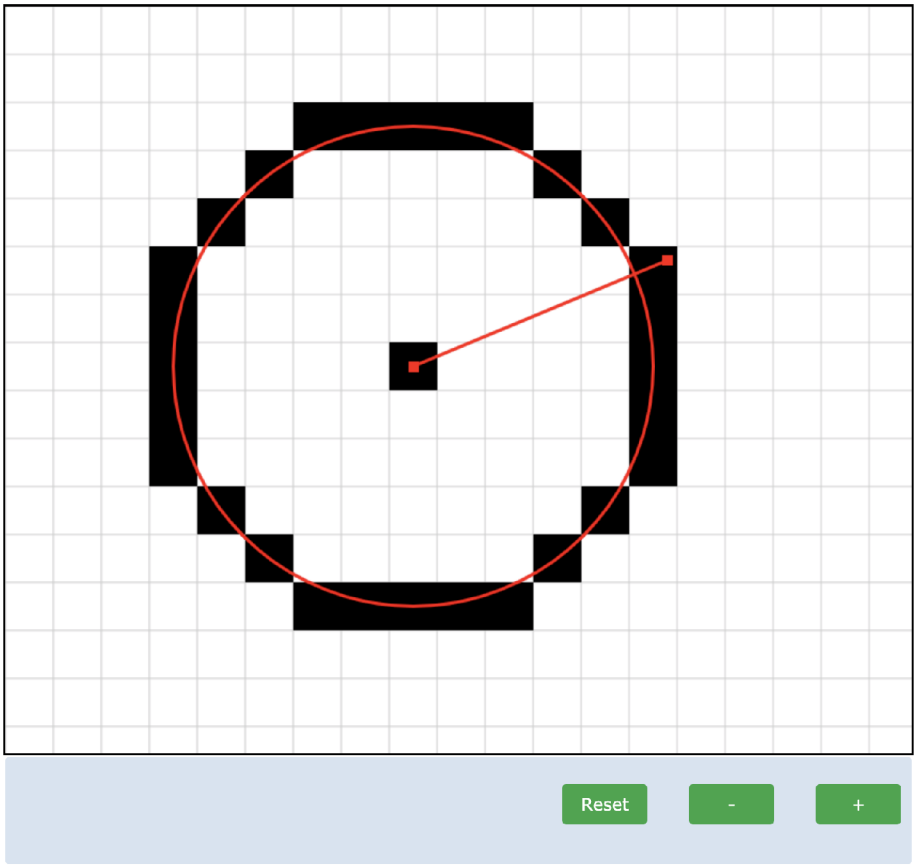
Obr. 5.4: Aplet pre Rasterizáciu úsečky

5.5 Aplet pre rasterizáciu kružnice

Aplet pre rasterizáciu kružnice približuje ako pracuje Bressenhamov algoritmus pre rasterizáciu kružnice algoritmus je popísaný v kapitole 3.4. Princíp algoritmu je založený na rozhodovaní či daný bod patrí do kružnice alebo von z kružnice. Štruktúra apletuje totožná z apletom pre rasterizáciu úsečky. Pri načítaní stránky je zobrazená preddefinovaná kružnica. Stred kružnice je pevný. Kliknutím do sieti pixelov je možné meniť polomer kružnice, zároveň je možné polomer meniť aj ťahaním za červený štvorec ktorý symbolizuje polomer kružnice. Pri zmene polomeru kružnice sa kružnica mení v reálnom čase. Tlačidlami v ovládacom paneli je možné meniť veľkosť pixelov a tak detailnejšie preskúmať algoritmus. Tlačidlom reset sa načíta preddefinovaná kružnica a nastavíme preddefinovanú veľkosť pixelov. Bressenhamov algoritmus pre rasterizáciu kružnice z-obecnuje nasledujúca ukážka kódu. Funkcia `draw_circle()`, ktorej vstupom je súradnica sredu kružnice a jej polomer, po nastavení potrebných premenných rozhoduje o tom ktoré pixely budú vykreslené. Funkcia `elipse_points()` na základe vstupných súradníc pomocou funkcie `put_pixel()`

vykreslí pixely najskôr v štyroch oktantoch a po zmene súradníc v ďalších štyroch oktantoch. Po tomto kroku je vykreslená celá kružnica, a nastavia sa potrebné premenné.

```
function draw_circle(ctx, x0, y0, r) {
  var p= 1-r;
  var x = 0;
  var y = r;
  var dx = 3;
  var dy = 2*r -2;
  while (x <= y) {
    ellipse_points(ctx, x0, y0, x, y, "black");
    ellipse_points(ctx, x0, y0, y, x, "black");
    if (p < 0) {
      p += dx;
      dx += 2;
      x += 1;}
    else {
      p -= dy;
      dy -= 2;
      y -= 1;}
  }}
function ellipse_points(ctx, x0, y0, x, y, color) {
  put_pixel(ctx, x0 + x, y0 + y, color);
  put_pixel(ctx, x0 - x, y0 + y, color);
  put_pixel(ctx, x0 + x, y0 - y, color);
  put_pixel(ctx, x0 - x, y0 - y, color);
}
function put_pixel(ctx, x, y, color) {
ctx.save();
ctx.fillStyle = color;
ctx.fillRect(x * pixel_size, y * pixel_size, pixel_size, pixel_size);
ctx.restore();
}
```



Obr. 5.5: Aplet pre Rasterizáciu kružnice

6 Záver

V teoretickej časti tejto práce bola popísaná problematika počítačovej grafiky. V prvej časti bol popísaný obrazový signál, jeho matematický model a spôsob jeho diskretizácie, ktorá je pri práci s obrazovým signálom nevyhnutná. Ďalej sú vysvetlené výhody, nevýhody a princípy dvoch typov záznamu obrazového signálu, bitmapového a vektorového. Ďalšia časť popisuje vektorové krivky, ich typy a vlastnosti, ktoré sú používané pri vytváraní jednoduchých ale aj zložitejších objektov. V tretej časti sa práca venuje problému zobrazenia vektorových objektov na rastrových zobrazovacích zariadeniach. Popisuje proces rasterizácie základných útvarov ako úsečka, krivka, kružnica, elipsa, ktoré sú často používané na vytvorenie zložitejšieho vektorového tvaru. Vysvetlené sú algoritmy ktoré sa používajú pri rasterizácií jednotlivých útvarov ich výhody a nevýhody.

Praktická časť tejto práce je rozdelená na dve časti. V prvej časti boli navrhnuté grafické užívateľské rozhrania apletov pre Bézierovu krivku, Racionálnu Bézierovu krivku, algoritmus de Casteljau, rasterizáciu úsečky a rasterizáciu kružnice. Grafické užívateľské rozhrania boli navrhnuté tak aby boli pre užívateľa prívetivé, zrozumiteľné a jednoducho sa ovládali. Tieto aplety boli zároveň implementované do funkčnej podoby a poskytujú priamu interakciu s používateľom a teda reagujú na zmenu parametrov v reálnom čase. Vzhľadom na veľkosť zdrojových súborov jednotlivých apletov sú v práci popísané základné, ale veľmi dôležité funkcie ktoré vykonávajú výpočty podľa teoretickej predlohy. Aplety slúžia študentom pre overenie teoretických znalostí v oblasti počítačovej grafiky. Študenti si tak teoretické znalosti môžu doplniť o praktické. Vytvorené aplety boli navyše vložené do web stránok vedúceho práce, ktoré využíva pri výuke počítačovej grafiky.

Literatúra

[1] ŽÁRA, Jiří, Bedřich BENEŠ, Jiří SOCHOR a Petr FELKEL. *Moderní počítačová grafika*. Druhé vydání. Brno: Computer Press, 2004. ISBN 80-251-0454-0.

[2] SOJKA, Eduard, Jan GAURA a Michal KRUMNIKL. *Matematické základy digitálního zpracování obrazu* [online]. Ostrava, 2011 [cit. 2018-12-04]. Dostupné z: http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit /digitalni_zpracovani_obrazu.pdf

[3] *Digitalizace rastrového obrazu* [online]. [cit. 2018-11-22]. Dostupné z: https://sites.google.com/site/xgrafika/digitalizace-rastroveho-obrazu

[4] Kvantovaní signálu. *Encyklopedie fyziky* [online]. [cit. 2018-11-15]. Dostupné z: http://fyzika.jreichl.com/main.article/view/1357-kvantovani-signalu

[5] Bitmapová grafika. *Bečvářová* [online]. [cit. 2018-11-22]. Dostupné z: http://becvarova.com/skoleni/inkscape/bitmapova-grafika/

[6] NAVRÁTIL, Pavel. *Počítačová grafika a multimédia*. Kralice na Hané: Computer Media, 2007. ISBN 80-866-8677-9.

[7] CHAPMAN, Nigel P. a Jenny CHAPMAN. *Digital multimedia*. 2nd ed. Chichester: Wiley, c2004. ISBN 04-708-5890-7.

[8] Vektorová grafika. *Bečvářová* [online]. [cit. 2018-11-22]. Dostupné z: http://becvarova.com/skoleni/inkscape/vektorova-grafika/

[9] What is vector graphics. *Techtarget* [online]. [cit. 2018-11-22]. Dostupné z: https://searchwindevelopment.techtarget.com/definition/vector-graphics

[10] Geometrie/Úvod do křivek. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-12-04]. Dostupné z: https://cs.wikibooks.org/wiki/Geometrie/%C3%9Avod_do_k%C5%99ivek

[11] ŠTUGEL, Juraj. Krivky a plochy. *Výuka počítačovej grafiky* [online]. [cit. 2019-03-23]. Dostupné z: http://doc.inver.sk/dip/index.html

[12] LINKEOVÁ, Ivana. *Základy počítačového modelování křivek a ploch*. V Praze: České vysoké učení technické, 2008. ISBN 978-80-01-04011-9.

[13] Bézier curve. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-12-04]. Dostupné z: https://en.wikipedia.org/wiki/B%C3%A9zier_curve

[14] Formát X3D a záhadná zkratka NURBS. *Root.cz* [online]. [cit. 2018-11-30]. Dostupné z: https://www.root.cz/clanky/format-x3d-a-zahadna-zkratka-nurbs/

[15] PIEGL, Les A a Wayne TILLER. *The NURBS book*. 2nd ed. New York: Springer, c1997. Monographs in visual communication. ISBN 978-3-540-61545-3.

[16] Figma. *Figma* [online]. [cit. 2019-05-10]. Dostupné z: https://www.figma.com/developers/docs

[17] *HTML5 Reference* [online]. [cit. 2019-05-10]. Dostupné z: https://dev.w3.org/html5/html-author/

[18] *Javascript Documentation* [online]. [cit. 2019-05-10]. Dostupné z: https://devdocs.io/javascript/

[19] *Data-Driven Documentation* [online]. [cit. 2019-05-10]. Dostupné z: https://d3js.org/

Zoznam symbolov, veličín a skratiek

WWW	World Wide Web
DDA	Digital Differential Analyzer
HTML	Hyper Text Markup Language
GIF	Graphics lnterchange Format
PNG	Portable Graphics Network
JPEG	Joint Photographic Experts Group
AI	Adobe Illustrator Artwork
EPS	Encapsulated PostScript
f_{vz}	vzorkovací kmitočet

A Obsah priloženého CD

Na priloženom CD sú všetky potrebné webové stránky a súbory na spustenie funkčných apletov a stránok do ktorých sú dosadené.

- `/` ..... korenový adresár priloženého CD
  - `bezier.html` ..... Webová apletom pre Bézierovu krivku
  - `bezier.js` ..... Javascript pre Bézierovu krivku
  - `racionalBezier.html` .... Webová stránkapre Racionálnu Bézierovu krivku
  - `rational.js` ..... Javascript pre Racionálnu Bézierovu krivku
  - `deCasteljau.html` ..... Webová stránka pre de Casteljau
  - `deCasteljau.js` ..... Javascript pre algoritmus de Casteljau
  - `line.html` ..... Webová stránka pre rasterizáciu kružnice
  - `line.js` ..... Javascript pre rasterizáciu úsečky
  - `circle.html` ..... Webová stránka pre rasterizáciu kružnice
  - `circle.js` ..... Javascript pre rasterizáciu úsečky
  - `d3.js` ..... Javascriptová knižnica D3
  - `bootstrap.min.js` ..... Javascriptová knižnica Bootstrap
  - `bootstrap.min.css` ..... CSS súbor pre knižnicu Bootstrap
  - `css.css` ..... CSS súbor so štýlmi stránok
  - `images` ..... Súbor s obrázkami
    - `Group4.png` ..... Obrázok
    - `obrazek1.png` ..... Obrázok
    - `rovnice21.png` ..... Obrázok
    - `rovnice22.png` ..... Obrázok
    - `rovnice23.png` ..... Obrázok
    - `rovnice24.png` ..... Obrázok
  - `media` ..... Súbor s obrázkami
    - `r1.png` ..... Obrázok
    - `r2.png` ..... Obrázok
    - `r3.png` ..... Obrázok
    - `r4.png` ..... Obrázok
    - `bezier.png` ..... Obrázok
    - `kruznice-r1.png` ..... Obrázok
    - `kruznice-r2.png` ..... Obrázok
    - `kruznice-r3.png` ..... Obrázok