Aplikace algoritmů počítačové grafiky

Jan Štalmach

Bakalářská práce
2004

Univerzita Tomáše Bati ve Zlíně
Fakulta technologická
Institut řízení procesů a aplikované informatiky

Vložit oficiální zadání bakalářské práce

Děkuji M. Ševčíkové za pečlivé pročtení tohoto textu.

Souhlasím s tím, že s výsledky mé práce může být naloženo podle uvážení vedoucího bakalářské práce a ředitele ústavu a institutu. V případě publikace budu uveden jako spoluautor.

Prohlašuji, že jsem na celé bakalářské práci pracoval samostatně a použitou literaturu jsem citoval.

Ve Zlíně, 21. 10. 2004 .....

podpis

RESUMÉ

Tato práce se zabývá aplikací rasterizačních algoritmů v oblasti počítačové grafiky nazvané počítačová geometrie. V teoretické části je vypracována literární rešerše 21 algoritmů určených pro rasterizaci základních geometrických primitiv, parametrických polynomiálních křivek a ploch. V praktické části je naprogramována pod objektovým C++ knihovna obsahující všechny teoreticky popsané algoritmy. Krátce je pojednáno o vhodnosti a použití těchto algoritmů. Nad touto vyvinutou knihovnou a grafických rozhraním OpenGL je dále naprogramována softwarová aplikace Rasterizer sloužící pro jednoduché interaktivní intuitivní modelování 2D a 3D počítačové grafiky. Výstup tohoto programu tvoří obrazovou dokumentaci práce.

This essay deals with the application of rasterization algorithms in the field of computer graphics called computational geometry. In the theoretical part there are described 21 algorithms designed for rasterization of 2D output primitives, parametrical polynomial splines and surfaces. Whereas in the practical part there is in a C++ programmed library containing all theoretically described algorithms. Shortly is also mentioned the suitability and the usage of this algorithms. Above this developed library and graphical interface OpenGL is programmed a software application Rasterizer that serves for easy interactive intuitive model of 2D and 3D computer graphics. The output of this program forms a screen documentation of the essay.

OBSAH

ÚVOD

V 50. a 60. letech minulého století se zrodilo odvětví výpočetní techniky, které mělo pomoci inženýrům při konstrukci a návrhu zařízení co nejlépe využít nástup rozvoje výpočetní kapacity počítačů. Pro tento vědní obor se vžil název ***počítačová grafika*** - soubor metod a prostředků pro zpracování grafické informace pomocí počítače. Základní problém, který byl pomocí počítačové grafiky úspěšně vyřešen se datuje právě do 50. let dvacátého století, kdy bylo potřeba vyřešit funkci lineárních vektorových interpolátorů pro zobrazování a archivaci liniových rovinných dat (grafy, výkresy, atd.) pomocí počítače. Počítačoví inženýři přitom vyšli z klasické diferenciální rovnice přímky.

Rozvoj počítačové grafiky dále vedl ke stále hlubšímu porozumění základních teoretických poznatků. Algoritmy a postupy vypracované týmy vědců a inženýrů postupně položily základy dnes všeobecně známého pojmu CAD (*Computer Aided Design*). Významným milníkem vývoje se stala možnost digitální identifikace grafického elementu a jeho následné vyhodnocení uživatelem. Tímto byl položen princip základního komunikačního nástroje pro ovládání veškerých grafických aplikací současnosti – princip interaktivní grafiky. Další zlom způsobil přechod od vektorového popisu grafické informace k popisu grafických elementů jako množiny diskrétních bodů (rastrová grafika). Tento způsob vyjádření umožnil realistickou interpretaci modelovaných objektů nebo i syntetickou tvorbu obrazů. Další uplatnění našla počítačová grafika i v problematice detekce grafických elementů v obraze. Spojíme-li více obrazových elementů pomocí časového prostoru můžeme analyzovat, jak se v daném časovém úseku tyto objekty pohybují. Detekci pohybu v obrazech využívají především algoritmy určené pro digitální kompresi videa.

V současné době je již poměrně obtížné přesně určit hranici, vymezující striktně pojem počítačová grafika. Z definičního hlediska mají největší význam dva pojmy: *objektový* a *obrazový prostor*. Z hlediska způsobu zobrazení mezi těmito dvěmi množinami můžeme základní vědní disciplíny počítačové grafiky rozdělit do čtyř následujících kategorií:

- **Počítačová grafika** (*computer graphics*) – nejobecnější pojem definující transformaci dat z prostoru objektového do prostoru obrazového (digitálního)

- **Počítačová geometrie** (*computational geometry*) – návrh a analýza matematických algoritmů pro zpracování grafických dat nad objektovým prostorem. Matematická disciplína zkoumající teoretický základ počítačové geometrie, především problémy spojené se spojitostí a vlastnostmi diskrétního prostoru se nazývá *digitální topologie*.
- **Zpracování obrazu** (*image processing*) – návrh algoritmů zabývajících se především problematikou vyhodnocování skupin diskrétních pixelů a jejich následná analýza (detekce hran, eliminace šumu a aliasu, tvorba konvolučních maticových filtrů)
- **Počítačové vidění** (*computer vision*) – zpracování vlastností objektů a jejich vztahů (detekce kolizí, virtuální realita)

Technika, jež se používá pro zobrazování dat na rastrových zařízeních se nazývá *rasterizace*. Jejím základem je transformace dat z prostoru objektového do prostoru obrazového. Využívá se přitom postupného generování množiny diskrétních bodů vzorkováním grafického prvku s krokem odpovídajícím velikosti jednoho pixelu (*pixel = picture element*). Rastrová data ukládáme jako souřadnice bodů, které jsou v souladu s euklidovskou geometrií modelovány jako bezrozměrné objekty. Zobrazovací plocha je však fyzické zařízení a proto se v rastrových zařízeních bod transformuje na pixel. Pixel chápeme jako nejmenší zobrazitelný element grafického rastrového zařízení. Tímto zařízením je monitor, tiskárna, atd. Použijeme-li atribut velikosti bodu v měřítku *m*, pak se bod na rastrovém zařízení zobrazí jako čtverec o [*m*]*x*[*m*] pixelech, případně jako kruh o průměru [*m*], je-li povoleno vyhlazování scény pomocí *antialiasingu*. Dále je třeba rozlišovat *fyzické* a *logické pixely*. Za fyzický pixel se považuje konkrétní obrazový bod zobrazovacího zařízení. Při zobrazení informace na fyzický pixel se odkazujeme pomocí souřadnic a zde je velmi důležité, na který euklidovský bod fyzického pixelu se odkazujeme – zda na střed, vrchol nebo na nějakou jinou část. Body na které se odkazujeme pomocí souřadnic se nazývají *logické pixely*. Tento krok se nazývá *mapování* a musí odpovídat skutečné velikosti fyzických pixelů. Na logický pixel se odkazujeme pomocí tzv. *světových souřadnic*. Tyto souřadnice jsou většinou celá nezáporná čísla s počátkem v levém horním nebo dolním rohu zobrazovacího zařízení, souřadnice sousedních pixelů se liší o jedničku.

1 CÍLE BAKALÁŘSKÉ PRÁCE

Bakalářská diplomová práce se zabývá aplikací matematických algoritmů v oblasti počítačové grafiky nazvané *počítačová geometrie*. Řešená problematika je rozdělena podle jednotlivých hlavních témat do několika ucelených kapitol, které se zabývají konkrétními problémy specifikovanými v zadání této práce.

Nejprve je věnována pozornost teoretickému aparátu pro modelování 2D a 3D počítačové grafiky. V rámci této práce je podrobně popsáno celkem 20 rasterizačních algoritmů, které jsou rozděleny do tří základních skupin:

1. Generování grafických elementů
2. Modelování parametrických křivek
3. Modelování parametrických ploch

V části o generování elementárních grafických objektů je vypracována literární rešerše algoritmů pro rasterizaci úseček, kružnic a elips. Další část je velmi podrobně věnována studiu modelování parametrických polynomiálních křivek. Zabývá se modelovacími křivkami aproximačními i interpolačními, které se uplatňují v počítačové animaci. Dále je pojednáno o tématu spojitosti a navazování křivek po částech polynomiálních. V třetí části je zpracována teorie trojrozměrných parametrických polynomiálních ploch.

V další kapitole následuje praktická realizace knihovny. Veškeré algoritmy popsané v teoretické úvodní kapitole jsou zde vyřešeny programově. Výsledná knihovna je napsána v programovacím jazyku C++ dle mezinárodní normy ANSI/ISO. Knihovna je postavena na objektovém přístupu, kdy od společného předka zajišťujícího spolupráci s grafickým aplikačním rozhraním, je odvozena rodina potomků reprezentující jednotlivé algoritmické celky (třídy pro rasterizaci grafických elementů, křivek a ploch). Zásadní výhodou knihovny je nezávislost na grafickém prostředí. Knihovnu lze použít nejenom pod OpenGL, ale i v prostředí standardního Microsoft WinApi nebo DirectX, popřípadě na jakémkoli jiném rozhraní.

V následující kapitole se věnuji srovnání algoritmů, jejich rasterizačních rychlostí a vhodnosti použití.

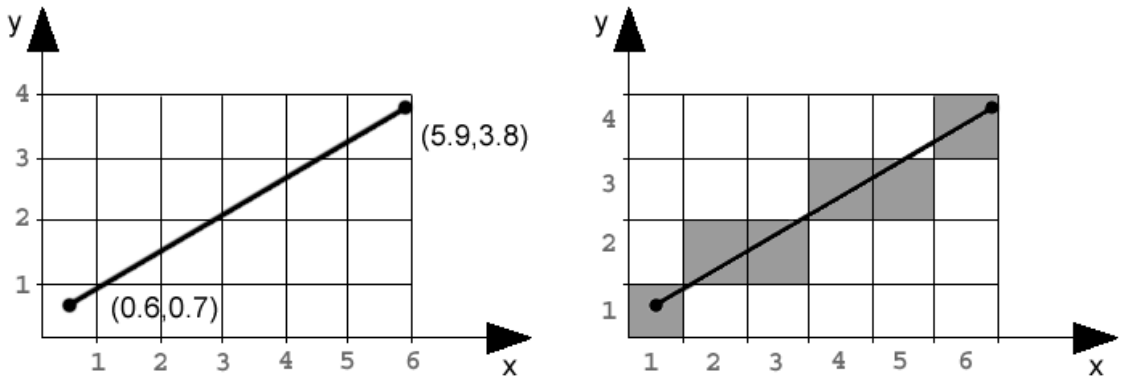
Závěrečná část popisuje softwarovou aplikaci postavenou na vyvinuté rasterizační knihovně a grafickém X-Window OpenGL rozhraní GLUT (7.4.2). Aplikace nazvaná *Rasterizer* slouží pro intuitivní interaktivní modelování grafických elementů, křivek a ploch. Zásadní výhoda je přenositelnost zdrojového kódu a tím i možnost provozovat aplikaci stejně dobře v prostředí Windows32 jako v prostředí Linux, UNIX, MacOS, atd. Veškerý obrazový výstup v této prácí byl vymodelován tímto programem. Program bude dále využíván jako výuková aplikace pro předmět *Počítačová grafika*. Využití lze najít i v návrhu součástek a grafickém designu výrobků.

2 GENEROVÁNÍ GRAFICKÝCH ELEMENTŮ

Mezi základní obrazové dvourozměrné grafické elementy (*2D output primitives*) patří úsečka, kružnice, elipsa, kruhový a eliptický oblouk. Podle typu zobrazovacího zařízení je výsledkem aplikace algoritmů pro kresbu grafických prvků buď diskrétní posloupnost pixelů nebo spojitá posloupnost pixelů nebo úseček. V prvním případě vytváříme rastrový obraz, který můžeme přímo zobrazovat na rastrových zobrazovacích zařízením. Ve druhém případě algoritmy generují reprezentaci obrazu ve vektorové podobě, který můžeme přímo zobrazovat na vektorových zařízeních (vektorový display, plotter, atd.) nebo musíme obraz pomocí *rasterizace* převést do diskrétní rastrové podoby.

2.1 Rasterizace úsečky

Historicky nejstarší problém je rasterizace úsečky. Přestože je úsečka poměrně jednoduchý grafický prvek bylo způsobu její rasterizace věnováno mimořádné úsilí, jehož výsledkem je velká rodina rasterizačních algoritmů, které můžeme charakterizovat společným rysem, a to snahou o co nejrychlejší rasterizační algoritmus. Pomocí napojování úseček bývají vytvářeny složitější lomené čáry, mnohoúhelníky, úsečka bývá také používána pro aproximaci křivek.

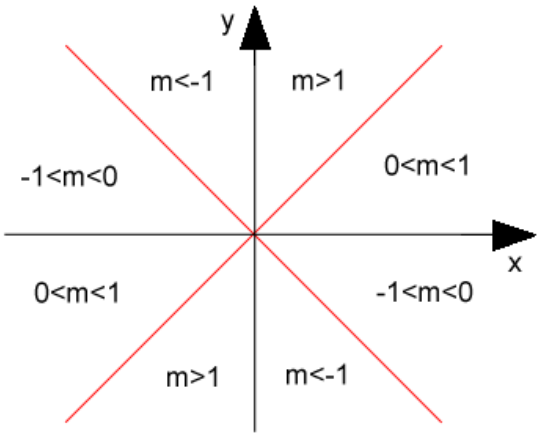


Obr. 1. Vektorové a rastrové zobrazení úsečky

Úsečku nejčastěji popisujeme počátečním $[x_1,y_1]$ a koncovým bodem $[x_2,y_2]$. Jiné vyjádření vychází z parametrické rovnice přímky, kdy je úsečka zadána svým počátečním bodem $[x_1,y_1]$ a vektorem rozdílu souřadnic $(\Delta x,\Delta y) = (x_2-x_1,y_2-y_1)$. Obecně jsou tyto

hodnoty neceločíselné a při vykreslení v diskrétním rastru musí být zaokrouhleny. Charakter úsečky můžeme dále ovlivňovat pomocí atributu *tloušťky* a způsobu *přerušování* úseků z kterých se úsečka skládá. Vzniká tak např. čerchovaná úsečka atd.

Rasterizace úsečky probíhá vzorkováním s konstantním krokem vždy v jedné ze souřadnicových os *x* a *y*. Určení ve směru jaké osy se bude provádět konstantní inkrementace velikosti jednoho pixelu určíme ze *směrnice m*. Tato hodnota je dána podílem rozdílů koncového a počátečního bodu úsečky.



Obr. 2. Velikost směrnice *m* pro různé sklony úsečky

Jestliže je směrnice *m* v absolutní hodnotě $|m| < 1$, pak se úsečka více blíží k ose *x* a proto bude vzorkována v ose *x* s konstantním krokem jednoho pixelu. V opačném případě se úsečka více přibližuje ose *y* a bude tedy vzorkována jedním pixelem v ose *y*. Osa ve které se provádí vzorkování se nazývá *řídící*, druhé ose se říká *vedlejší*. Tento způsob rasterizace podle 8-mi kvadrantů se nazývá *osmi-spojitá rasterizace* úsečky. Následující soused $[x_{i+1},y_{i+1}]$ úsečky $[x_i,y_i]$ může ležet v jednom z osmi směrů – vodorovně, svisle nebo diagonálně.

2.1.1 Algoritmus DDA

Algoritmus DDA (*digital differencial analyser*) je nejstarším způsobem provádění rasterizace úsečky. Podstata algoritmu spočívá v postupném inkrementálním zvětšováním souřadnic úsečky.

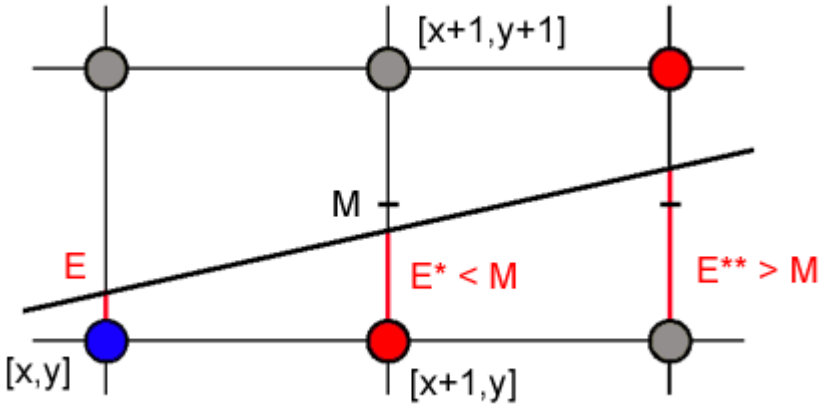
Výchozím bodem tohoto algoritmu je klasická rovnice přímky

$$m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}. \tag{1}$$

Algoritmus můžeme rozdělit do několika kroků. Nejprve určíme řídící a vedlejší osu. V řídící ose zvolíme konstantní krok odpovídající velikosti jednoho pixelu. Krok ve vedlejší ose položíme roven směrnici *m* úsečky. Algoritmus dále pracuje v jednom jediném cyklu, který provádí postupnou inkrementaci v řídící a vedlejší ose. Cyklus končí při dosažení koncového bodu úsečky. Je zřejmé, že při inkrementálním přičítání směrnice je výsledek obecně neceločíselný. Tento fakt, tedy nutnost použít *reálnou aritmetiku* a následné *zaokrouhlení*, je nejvíce limitujícím faktorem rychlosti tohoto algoritmu. Při výpočtu nových souřadnic se používá už vypočtených souřadnic z předchozího kroku. Tomuto postupu říkáme iterační.

2.1.2 Bresenhamův algoritmus

V roce 1965 uveřejnil Jack E. Bresenham v *IBM Systems Journal* velmi efektivní algoritmus pro rasterizaci úsečky pracující pouze v celočíselné aritmetice. Vstupní parametry algoritmu jsou celočíselné počáteční a koncové body úsečky. Stejně jako DDA algoritmus se v tom osovém směru, kde je větší přírůstek, volí krok odpovídají jednomu pixelu.



Obr. 3. Výběr odpovídajících pixelů úsečky

Pro celočíselné odvození budeme předpokládat rozložení koncových bodů podle Obr. 3. a rasterizaci budeme provádět pro řídící osu *x*.

Rozdíl polohy dvou po sobě následujících souřadnic vyjádříme jako

$$dx = x_2 - x_1, \tag{2}$$

$$dy = y_2 - y_1. \tag{3}$$

Výchozí rovnici sestavíme podle Obr. 3. v následujícím tvaru

$$E^* = E + \frac{dy}{dx} \leq M = 0.5. \tag{4}$$

Vynásobením celé rovnice *2dx* dostaneme

$$2dxE^* = 2dxE + 2dy \leq dx. \tag{5}$$

Odečtením *–dx* získáme

$$dx(2E^* - 1) = dx(2E - 1) + 2dy \leq 0. \tag{6}$$

Označme pomocí substituce jako *predikaci* ***D***

$$D^* = dx(2E^* - 1), \tag{7}$$

$$D = dx(2E - 1). \tag{8}$$

Potom

$$\text{pokud } D \leq 0 \Rightarrow D^* = D + 2dy,\ y^* = y, \tag{9}$$

$$\text{pokud } D > 0 \Rightarrow D^* = D + 2dy - 2dx,\ y^* = y + 1. \tag{10}$$

Počáteční hodnota

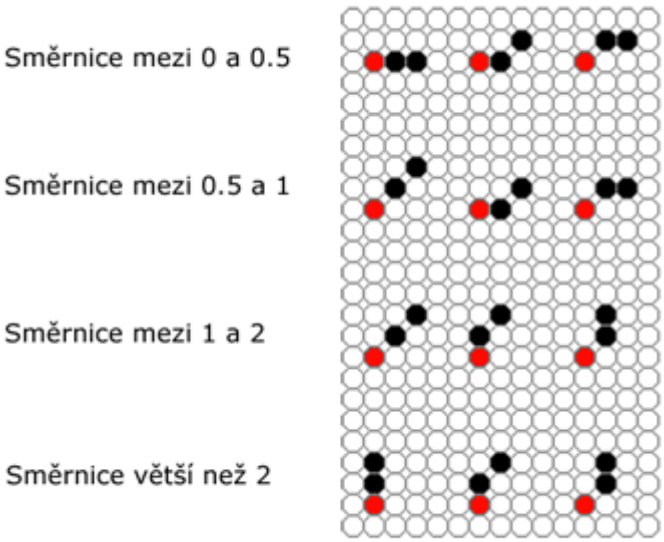
$$D_0 = 2dy - dx. \tag{11}$$

Predikace *D* je základním kritériem pro výběr pixelů tvořících rasterizovaný obraz úsečky. Hodnotu *D* aktualizujeme pro každý pixel pouze sčítáním. Pokud je její hodnota záporná, souřadnice *y* následujícího pixelu se nemění. Při kladném znaménku *D* má následují pixel souřadnici *y* zvětšenou o jedničku.

2.1.3 Line Two-Step algoritmus

Algoritmus Line Two-Step navrhl pan Xiaolin Wu. Tento algoritmus vznikl až po slavném Bresenhamově algoritmu a i když nedosahuje takové rasterizační rychlosti je velmi zajímavým způsobem řešení. Předchozí algoritmy (Bresenhamův, DDA) jsou příkladem

tzv. *jednokrokových algoritmů*, kdy je hledám právě jeden následují pixel v rastru. Naproti tomu *vícekrokové algoritmy* zkoumají způsob umístění více po sobě jdoucích pixelů v rastru. Takovýmto příkladem je i dvoukrokový Line Two-Step algoritmus.

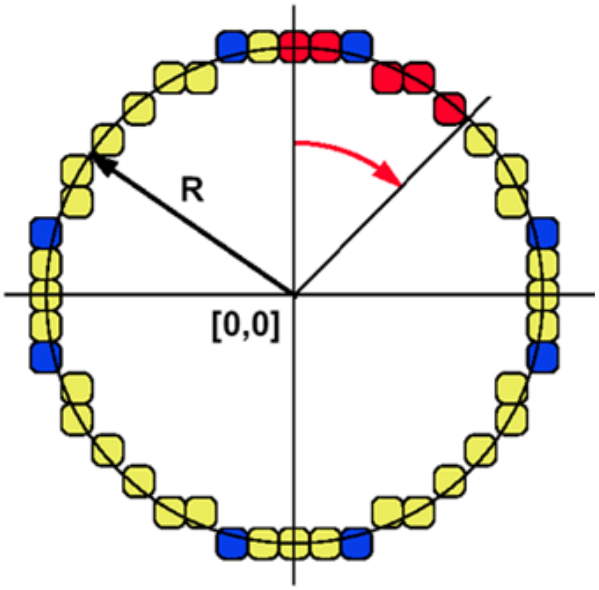


Obr. 4. Možné konfigurace umístění následujících 2 pixelů

Podstata tohoto algoritmu je v určení tzv. vzoru (*pattern*), který popisuje chování pixelů v rastru. Bylo zjištěno, že dva následující pixely ležící v rastru mají podle velikosti směrnice v předcházejícím kroku konečně mnoho variant umístění.

2.2 Rasterizace kružnice

Kružnice bývá nejčastěji zadána svým středem [*x,y*] a poloměrem *r* nebo třemi různými body neležícími na přímce. Rasterizace se většinou provádí pro kružnici se středem v počátku a získané body v rastru jsou posunuty do cílové polohy až před vykreslením. Rasterizace se provádí pouze pro jeden *oktant*, zbylé body jsou odvozeny na základě symetrií.



Obr. 5. Symetrické rozdělení kružnice

2.2.1 Algoritmus rasterizace pomocí parametrického vyjádření

Nejjednodušší implementace rasterizačního algoritmu vychází z klasické parametrické rovnice kružnice

$$\begin{aligned} x &= x + r\cos\alpha \\ y &= y + r\sin\alpha \end{aligned} . \tag{12}$$

Nové souřadnice lze jednoduše pomocí parametru $\alpha \in (0, \pi/4)$ vyčíslovat a tím získat posloupnost bodů pravidelně rozmístěných na jednom oktantu kružnice. Zbylé oktanty se určí pomocí symetrií. Velikost přírůstku úhlu α_p by měla odpovídat velikosti kružnice. Přírůstkový úhel je proto definován jako

$$\alpha_p = \frac{\pi}{4r}. \tag{13}$$

Algoritmus v jediném cyklu postupně vyčísluje reálné souřadnice dosazováním přírůstkového úhlu α_p vynásobeného o aktuální krok do (12). Cyklus končí pokud celková velikost přírůstku α_p^* je rovna úhlu vymezující jednu osminu kružnice. Tento stav je docílen při takovém počtu kroků, které odpovídají velikosti poloměru této kružnice.

$$\alpha_p^* = \sum \alpha_p = \frac{\pi}{4r} r = \frac{\pi}{4} . \tag{14}$$

2.2.2 Algoritmus rotace úsečky

Předchozí algoritmus aproximoval kružnici pomocí bodů ležících v rastru. Jiný způsob rasterizace kružnice vychází z myšlenky aproximovat kružnici pomocí rotace úsečky kolem středu této kružnice.

Rotace úsečky v rovině (koncových bodů) kolem počátku je dána následující transformační maticí

$$A_r = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{15}$$

Toto maticové vyjádření můžeme zapsat jako

$$\begin{aligned} X^* &= X\cos\alpha - Y\sin\alpha \\ Y^* &= X\sin\alpha + Y\cos\alpha \end{aligned}. \tag{16}$$

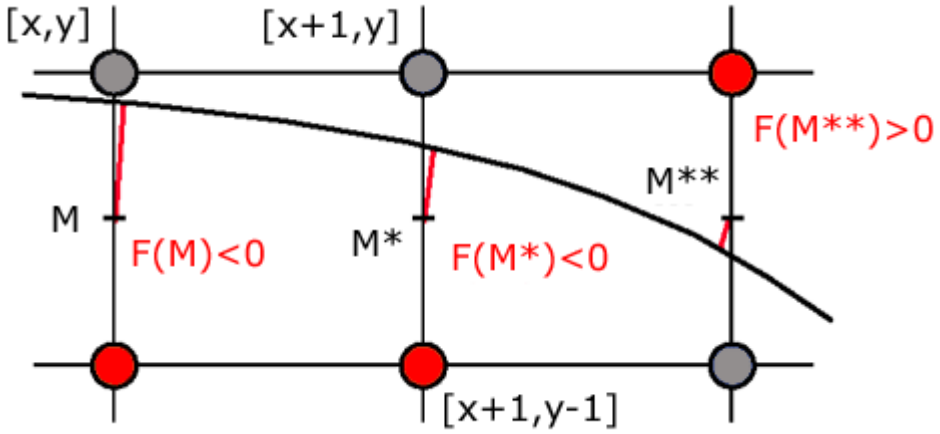
Výpočet goniometrické funkce můžeme provádět pouze jednou a v cyklu používat pouze sčítání a násobení. Rasterizujeme opět pouze jeden oktant a počet kroků a velikost přírůstkového úhlu α volíme v závislosti na velikosti kružnice. Je žádoucí docílit přesně hranice jednoho oktantu, aby se při symetrickém odvození zbylé části setkali koncové a počáteční body.

Na základě studia způsobu této rasterizace bylo odvozeno následující doporučení pro volbu kroku a přírůstkového úhlu α v závislosti na poloměru r

$$\begin{aligned} &r \le 30 \Rightarrow krok = 3 \wedge \alpha = 50 \\ &r \in \langle 30, 200 \rangle \Rightarrow krok = 6 \wedge \alpha = 25 \\ &r > 200 \Rightarrow krok = 12 \wedge \alpha = 12.5 \end{aligned}. \tag{17}$$

2.2.3 Bresenhamův algoritmus

Stejně tak jako u rasterizace úsečky (2.1.2) odvodil Jack E.Bresenham velmi efektivní algoritmus pro generování bodů na kružnici pracující pouze v celočíselné aritmetice. Algoritmus využívá také symetrie v kružnici a proto se rasterizace provádí pouze pro jeden oktant. Tento algoritmus se někdy v anglické literatuře nazývá také jako *midpoint algorithm*.



Obr. 6. Výběr odpovídajících pixelů kružnice

Bresenhamův algoritmus předpokládá řídící osu x, která má konstantní krok výpočtu. Pro body na kružnici platí implicitní rovnice, která se zapíše jako funkce

$$F(x, y) : x^2 + y^2 - r^2 = 0. \tag{18}$$

Znaménko funkce určuje polohu bodu $[x,y]$ vůči kružnici. Funkční hodnota pro body uvnitř kružnice je záporná, pro body vně kružnici je kladná. Funkce F je základní kritérium pro zavedení predikace.

Podle Obr. 6. je souřadnice aktuálního bodu $[x,y]$. Následující bod může mít buď souřadnice $[x+1,y]$, nebo $[x+1,y\text{-}1]$. Bod M^* ležící mezi těmito dvěmi možnými novými souřadnicemi se nazývá *středový bod* (*midpoint)* a určuje, který z bodů bude pro rasterizaci použit. Dosadíme-li do funkce (18) souřadnice tohoto bodu $M^*=[x+1,y\text{-}0.5]$, tak nám znaménko této funkce určí, zda-li tento bod leží uvnitř, resp. vně kružnice. Výsledná hodnota se nazývá *predikace*

$$F(M^*) = (x+1)^2 + (y-0.5)^2 - r^2 < 0, \tag{19}$$

$$F(M^{**}) = (x+2)^2 + (y-0.5)^2 - r^2 = F(M^*) + 2x + 3 < 0, \tag{20}$$

$$F(M^{**}) = (x+2)^2 + (y-0.5)^2 - r^2 = F(M^*) + 2x + 2y - 5 \geq 0. \tag{21}$$

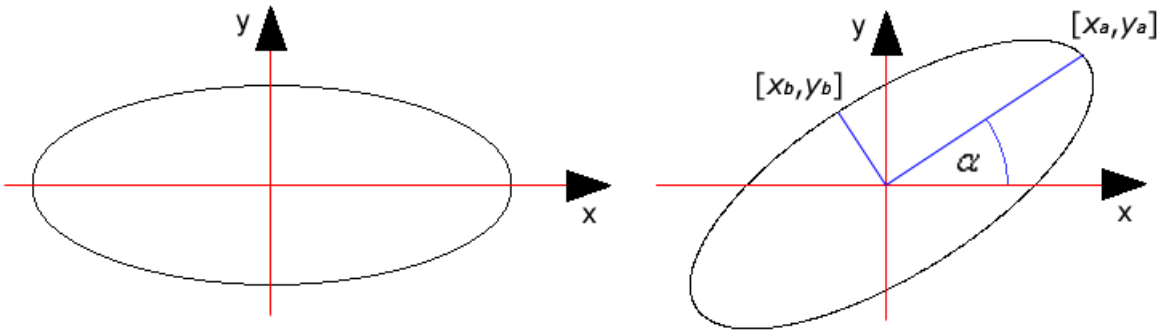
Je-li znaménko této funkce (20) záporné, tak bude vybrán bod se stejnou souřadnicí y, v druhém případě (21) bude vykreslen bod ležící o jeden pixel níže.

Označíme-li $D=F(M)$, potom můžeme podle znaménka predikace tento vzorec zapsat ve výsledné podobě jako

$$\begin{aligned} &D<0 \Rightarrow D^* = D + 2x + 3, \quad y^* = y \\ &D \geq 0 \Rightarrow D^* = D + 2x + 2y + 5, \quad y^* = y - 1. \\ &D_0 = 1 - r \end{aligned} \tag{22}$$

2.3 Rasterizace elipsy

Rasterizace elipsy je velmi podobná rasterizaci kružnice. Osově orientovaná elipsa je charakterizována středem $[x,y]$ a délkami dvou poloos a, b. Pro obecně zadanou elipsu, která není osově orientovaná je nutné definovat elipsu středem a dvěma body určujícími hlavní a vedlejší poloosu. Jiná možnost je předpoklad umístění elipsy jako osově orientované a odchylku od řídící osy definovat jako úhel natočení elipsy. Na rozdíl od kružnice lze symetrii využít pouze pro tři další body a algoritmus musí tedy vygenerovat body elipsy v jednom celém kvadrantu.



Obr. 7. Osově orientovaná a obecná elipsa

2.3.1 Algoritmus rasterizace pomocí parametrického vyjádření

Tento algoritmus vychází podobně jako jeho ekvivalent pro kružnici z parametrické rovnice osově orientované elipsy

$$\begin{aligned} x &= x + a\cos\alpha \\ y &= y + b\sin\alpha \end{aligned}. \tag{23}$$

Průběh rasterizace je totožný s algoritmickým postupem (2.2.1) uvedeným pro rasterizaci kružnice. Pouze je nutné si uvědomit, že rasterizace se provádí po kvadrantech a tak je

nutné zdvojnásobit počet kroků potřebných pro pokrytí elipsy. Celková hodnota přírůstkového úhlu α_p^* odpovídá úhlu vymezujícímu jeden kvadrant elipsy

$$\alpha_p^* = \sum \alpha_p = \frac{\pi}{4r} 2r = \frac{\pi}{2}. \tag{24}$$

2.3.2 Bresenhamův algoritmus

Nejefektivnějším algoritmem pro rasterizaci osově orientované elipsy je Bresenhamův algoritmus pracující pouze v celočíselné aritmetice a vycházející podobně jako u rasterizace kružnice (2.2.3) z implicitního vyjádření elipsy

$$F(x, y) : b^2 x^2 + a^2 y^2 - a^2 b^2 = 0. \tag{25}$$

Z této rovnice odvodíme pomocí středového bodu (*midpoint*) predikaci obdobně jako u kružnice. Při rasterizaci elipsy tímto způsobem se mění řídící osa. Jestliže budeme generovat body v prvním kvadrantu zleva doprava, je řídící osou nejprve osa *x* a poté osa *y*. Bod, ve kterém dochází k této změně má směrnici $m = -1$ a jeho tečna má tedy sklon $\alpha_m = -45^\circ$. Souřadnice tohoto bodu určíme jako první parciální derivace funkce (25).

$$\frac{\partial F}{\partial x} = 2b^2 x, \quad \frac{\partial F}{\partial \mathrm{y}} = 2a^2 y. \tag{26}$$

Řídící osa se změní právě tehdy, když nastane rovnost těchto parciálních derivací

$$\frac{\partial F}{\partial x} = \frac{\partial F}{\partial \mathrm{y}} \Rightarrow \alpha_m = 45^\circ \Rightarrow m = -1. \tag{27}$$

V první části pro rasterizaci podle řídící osy *x* použijeme následující kritérium pro výběr odpovídajících bodů v rastru

$$\begin{aligned} D \le 0 &\Rightarrow D^* = D + b^2(2x+1), \\ D > 0 &\Rightarrow D^* = D + b^2(2x+1) - 2a^2 y. \end{aligned} \tag{28}$$

V druhé části algoritmu pro rasterizaci podle řídící osy *y* jsou vztahy obdobné.

2.3.3 Algoritmus obecné polohy elipsy

V technické praxi je třeba často používat nejenom osově orientované elipsy, ale i elipsy v obecné poloze. Tento algoritmus reprezentuje nejjednodušší naivní možnou

implementaci řešení tohoto problému. Základem algoritmu je *algoritmus parametrického vyjádření osově orientované elipsy* (2.3.1) obohacený o transformaci otočení v rovině kolem středu této elipsy. Výpočet goniometrických funkcí pro transformaci lze provádět pouze jednou a tím zvýšit efektivitu tohoto postupu. Vstupem tohoto algoritmu je proto nejenom střed $[x,y]$ a délky poloos a,b, ale i úhel α určující velikost natočení této elipsy.

Poznamenejme, že existují i sofistikovanější algoritmy pro rasterizaci elipsy v obecné poloze založené např. na modifikaci Bresenhamova postupu a že i tyto algoritmy pracují v celočíselné aritmetice.

3 MODELOVÁNÍ PARAMETRICKÝCH KŘIVEK

Křivky patří v počítačové grafice a mnoha souvisejících aplikacích k nejvyužívanějším grafickým prostředkům. Jejich uplatnění je nejenom v technické praxi velmi široké. Křivky jsou používány při modelování ve 2D i 3D prostoru, při definici fontů, při určování dráhy pohybu objektů, při počítačové animaci, při definování objektů pro šablonování a v neposlední řadě hlavně pro návrh v CAD/CAM systémech. Nejvíce používané jsou křivky polynomiální, jejichž tvar lze interaktivně měnit polohou řídících bodů, které určují průběh křivky. Konstruktér se tak může odpoutat od ryze matematických teorií křivek a pracovat s tvarem křivky podle návrhu zcela volně. Matematický aparát v pozadí zajišťuje přesnou interpretaci takto modelované křivky.

3.1 Vyjádření a základní vlastnosti parametrických křivek

Křivku můžeme fyzikálně chápat jako dráhu pohybujícího se bodu. Křivky lze obecně vyjádřit třemi způsoby:

- implicitně : $F(x, y, z) = 0$
- explicitně : $y = f(x, y, z)$
- parametricky : $x = x(t), y = y(t), z = z(t)$

Právě *parametrické vyjádření* těchto křivek se pro rasterizaci a práci s křivkami v počítačové grafice hodí nejvíce. Parametr $t = \langle t_{\min}, t_{\max} \rangle = \langle 0,1 \rangle$ představuje časový průběh křivky a jeho postupným inkrementálním dosazováním do parametrické rovnice křivky získáme posloupnost bodů, které poté spojíme lomenou úsečkou a tím získáme obrazovou reprezentaci této parametrické křivky. Vhodnou volbou vzdáleností těchto časových úseků určujeme počet aproximovaných bodů a tedy i úseček, tvořících výslednou kvalitu aproximované křivky.

Každý bod je tedy jednoznačně určen svou *bodovou rovnicí parametrické křivky*

$$Q(t) = [x(t), y(t), z(t)], \tag{29}$$

nebo *vektorovou rovnicí*

$$\vec{q}(t) = (x(t), y(t), z(t)). \tag{30}$$

Vektor $\vec{q}(t) = Q(t) - [0,0,0]$ se nazývá *polohový* (*směrový*) *vektor*, jeho velikost je rovna vzdálenosti bodu $Q(t)$ od počátku souřadnicového systému.

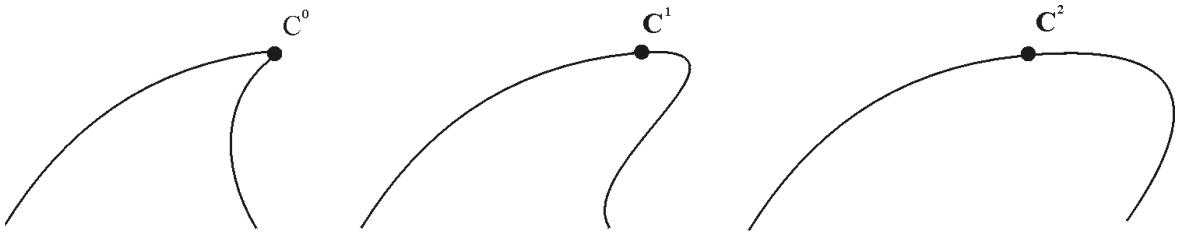
Derivace parametricky vyjádřené křivky se provádí po jejích složkách. Tečný vektor v v bodě $Q(t_0)$ se určí jako první parciální derivace polohového vektoru podle času

$$\overrightarrow{q'=}(t_0) = \left(\frac{\partial x(t_0)}{\partial t}, \frac{\partial y(t_0)}{\partial t}, \frac{\partial z(t_0)}{\partial t} \right). \tag{31}$$

Analogicky lze vyjádřit i druhou derivaci. Právě tento fakt, a to *snadná diferencovatelnost,* je nejvýznamnější předností parametrického vyjádření. Ze znalostí tečných vektorů můžeme jednoduše zajistit hladké navazování segmentů křivek a vytvářet tak libovolně složité struktury, které mohou být dokonce reprezentovány i odlišnými rovnicemi.

3.2 Spojitost

Pokud při navazování segmentů křivek budeme sledovat i kvalitu výsledného spojení těchto segmentů dojdeme k závěru, že je velmi důležité znát nejenom koncový bod prvního a počáteční bod druhého segmentu, ale i způsob jejich napojení, zejména tzv. *spojitost* (*continuity*) v dotýkajících se bodech.



Obr. 8. Parametrická spojitost třídy C^0, C^1 a C^2

Rozlišujeme dva základní typy spojitosti: Silnější podmínka je *parametrická spojitost*, slabší podmínkou je *geometrická spojitost*. Křivka $Q(t)$ je třídy C^n, má-li ve všech bodech spojité derivace do řádu n. Označení C^n říkáme *parametrická spojitost třídy* C^n. Dva segmenty $Q_1(t)$ a $Q_2(t)$ jsou spojité třídy C^0, pokud je koncový bod prvního segmentu totožný s počátečním bodem druhého segmentu. Dva segmenty jsou spojité třídy C^1, když

jsou si rovné první derivace (tečné vektory) těchto segmentů v napojených bodech. Analogicky pokud zaručíme rovnost i druhé derivace (vektor zrychlení) jsou segmenty v napojovaných bodech C^2 spojité. Čím vyšší požadujeme spojitost, tím více se k sobě oba segmenty přimykají. Jestliže nahlédneme na tento problém z hlediska pohybujícího se bodu po křivce, tak zjistíme, že spojitě se pohybující bod v místě napojení dvou segmentů $Q_1(t)$ a $Q_2(t)$ při spojitosti třídy C^0 může v tomto bodě skokově změnit nejenom směr, ale i rychlost a zrychlení. Rychlost bodu se nezmění pokud jsou uvažované segmenty C^1 spojité a zrychlení je stejné při napojení C^2 . Při využití křivek pro počítačovou animaci je nutno dle požadavků zajistit odpovídající spojitost.

Podmínka parametrické spojitosti je v určitých aplikacích zbytečně silná a z hlediska implementace je daleko snazší zaručit tzv. *geometrickou spojitost* G^n. V určitých aplikacích je vyžadování rovnosti tečných vektorů dokonce zbytečné, např. dvě napojené nestejně dlouhé úsečky tvořící přímku, by v bodě napojení měnily skokem rychlost, ale směr pohybu myšleného bodu by byl zachován. Právě tyto případy vedly k zavedení slabé podmínky spojitosti.

Jestliže jsou dva segmenty $Q_1(t)$ a $Q_2(t)$ totožné v počátečním bodu druhého segmentu a koncovém bodu prvního segmentu, tak jsou tyto segmenty G^0 spojité. Dva segmenty jsou G^1 spojité, pokud tečné vektory $\overrightarrow{q_1}$ segmentu $Q_1(t)$ a vektory $\overrightarrow{q_2}$ segmentu $Q_2(t)$ lze vyjádřit jako lineární kombinaci, tj. jestliže pro tyto vektory platí

$$\overrightarrow{q_1} = k\overrightarrow{q_2} \wedge k > 0\,. \tag{32}$$

Tato spojitost zaručuje totožnost tečen, nikoli rovnost tečných vektorů. Pohubující se bod nemůže skokem změnit směr, ale může skokově měnit rychlost. Tento typ spojitosti je nejčastějším typem geometrické spojitosti.

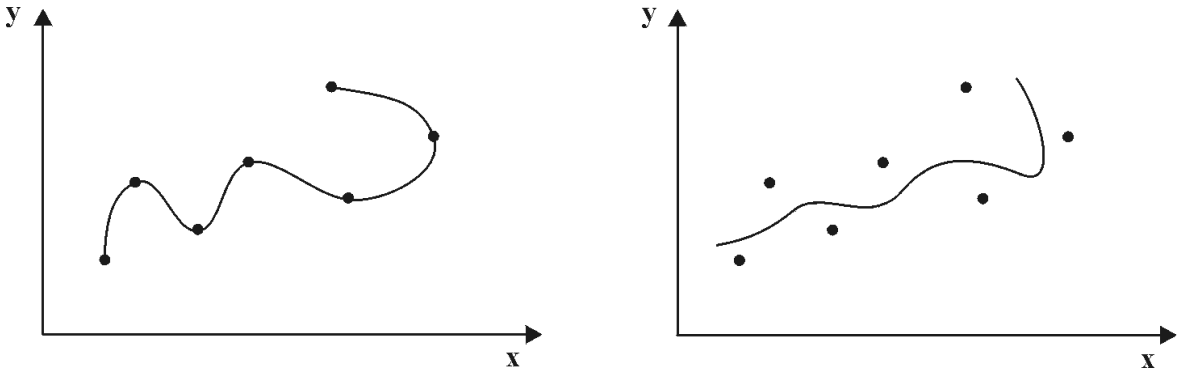
Spojitost G^2 je zaručena, pokud existuje shoda prvních křivostí v napojených bodech obou segmentů. V praxi se používá velmi málo.

3.3 Modelování parametrických polynomiálních křivek

V počítačové grafice se nejvíce prosadilo modelování parametrických křivek jejíchž zápis je matematicky veden za použití *polynomiálních vektorových rovnic*

$$p_n = a_0 + a_1 t + a_2 t^2 + \ldots + a_n t^n . \tag{33}$$

Tato forma zápisu má několik velmi podstatných výhod. Polynomiální křivky můžeme rychle vypočítávat a jsou také velmi snadno diferencovatelné a tudíž je lze lehce napojovat. Nejčastěji používané jsou křivky třetího stupně – *kubiky*, jejich výpočet je poměrně jednoduchý a tvary které lze kubikami reprezentovat poskytují dostatečně širokou modelovací škálu. Lze u nich zajistit spojitost třídy C^2.



Obr. 9. Interpolační a aproximační způsob modelování křivek

Samotné modelování spočívá ve vytvoření posloupnosti řídících (opěrných) bodů tvořících tzv. *řídící polygon*. Křivka těmito body může nebo nemusí procházet, mluvíme tedy o *interpolačním* resp. *aproximačním* způsobu modelování. Chování křivky mezi těmito body je dáno právě pomocí polynomiálních parametrických rovnic, které nazýváme *mísícími* (*blending*) funkcemi.

Parametricky zadanou kubiku $Q(t)$ zapisujeme

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y \\ z(t) &= a_z t^3 + b_z t^2 + c_z t + d_z \end{aligned} . \tag{34}$$

Tuto rovnici můžeme vyjádřit zkráceně v maticovém tvaru

$$Q(t) = \mathbf{TC} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}. \tag{35}$$

Matici konstant $\mathbf{C}$ lze rozepsat jako součin

$$\mathbf{C} = \mathbf{BP}, \tag{36}$$

kde matice $\mathbf{B}$ je typu [4x4] a nazýváme ji *bázová matice* a čtyřprvkový vektor $\mathbf{P}$ se nazývá *geometrický vektor* a obsahuje řídící body křivky. Bázová matice určuje výpočetní metodu použitou pro aproximaci nebo interpolaci řídících bodů.

Výsledná podoba rovnice pro výpočet polynomiálních parametrických křivek je ve tvaru

$$Q(t) = [x(t), y(t), z(t)] = \mathbf{TC} = \mathbf{TBP} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} & \mathrm{m}_{14} \\ m_{21} & m_{22} & m_{23} & \mathrm{m}_{24} \\ m_{31} & m_{32} & m_{33} & \mathrm{m}_{34} \\ m_{41} & m_{42} & m_{43} & \mathrm{m}_{44} \end{bmatrix} \begin{bmatrix} \mathrm{P}_1 \\ \mathrm{P}_2 \\ \mathrm{P}_3 \\ \mathrm{P}_4 \end{bmatrix}. \tag{37}$$

Nejčastější požadované vlastnosti křivek:

- Invariance k lineárním transformacím a projekcím
- Poloha křivky uvnitř konvexní obálky
- Lokalita editační změny
- Křivka může procházet krajními body

Křivky, které používáme v počítačové grafice mohou být *racionální* nebo *neracionální*. Racionální křivky vyjadřujeme v homogenních souřadnicích a pro každý řídící bod je tedy určena racionální váha konkrétního řídícího bodu. Neracionální křivky nejsou invariantní k perspektivní projekci a jejich váhy jsou rovny jedné.

Poznamenejme, že v následujícím textu jsou výsledné křivky konstruovány ze segmentů třetího stupně – *kubik*. Výsledná konkrétní křivka vznikne jako C^2 spojité napojení těchto segmentů.

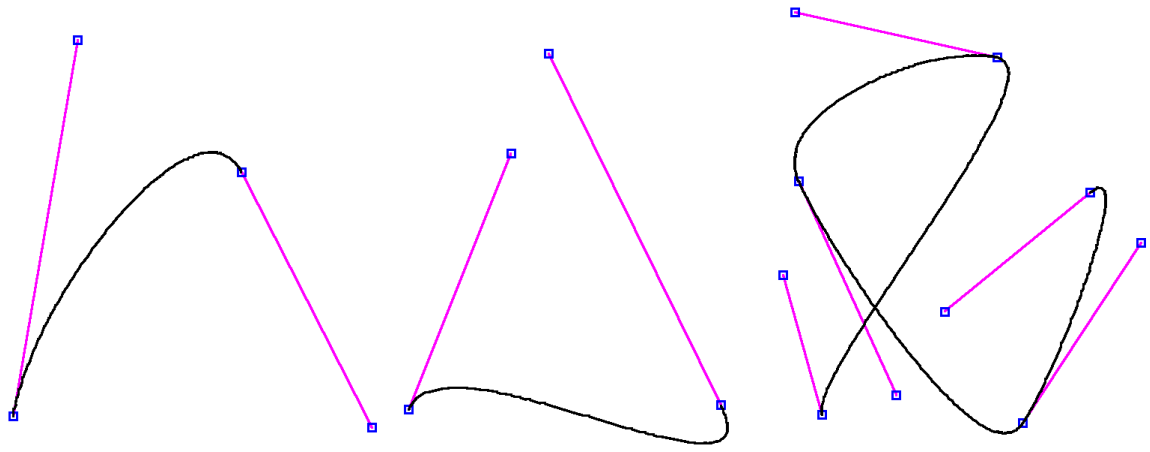
3.4 Interpolační křivky

Interpolační křivky mají v počítačové grafice významné uplatnění. Jejich konstrukce se využívá zejména v počítačové animaci, kde je pohyb objektu po křivce často definován právě pomocí interpolačních křivek. Interpolace je chápána tak, že je požadováno, aby modelovaná křivka procházela (všemi) řídícími body.

3.4.1 Fergusonova křivka

Nejznámější představitelem interpolačních křivek je tzv. *Fergusonova kubika*, kterou popsal v roce 1964 J. C. Ferguson. Spojením více kubik vznikne *Fergusonova křivka*. Tyto kubiky jsou určeny dvěma řídícími body P_0, P_1 a dvěma tečnými vektory $\overrightarrow{P'_0}$ a $\overrightarrow{P'_1}$ v těchto bodech.

Hovoříme o tzv. *Hermitově kubické interpolaci*. Orientace a velikost vektorů určuje výsledný tvar modelované křivky. Zvětšujeme-li velikost těchto vektorů, stále více se výsledná křivka k těmto vektorům přimyká. V opačném případě, je-li velikost obou tečných vektorů nulová, degraduje Fergusonova kubika na úsečku.



Obr. 10. Fergusonovy kubiky a Fergusonova křivka

Rovnice pro výpočet Fergusonovy kubiky

$$Q(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathrm{P}_0 \\ \mathrm{P}_0^{'} \\ \mathrm{P}_1 \\ \mathrm{P}_1^{'} \end{bmatrix}. \tag{38}$$

Tento vztah lze interpretovat následující rovnicí

$$Q(t) = H_0^3(t)P_0 + H_1^3(t)P_0' + H_2^3(t)P_1 + H_3^3(t)P_1', \tag{39}$$

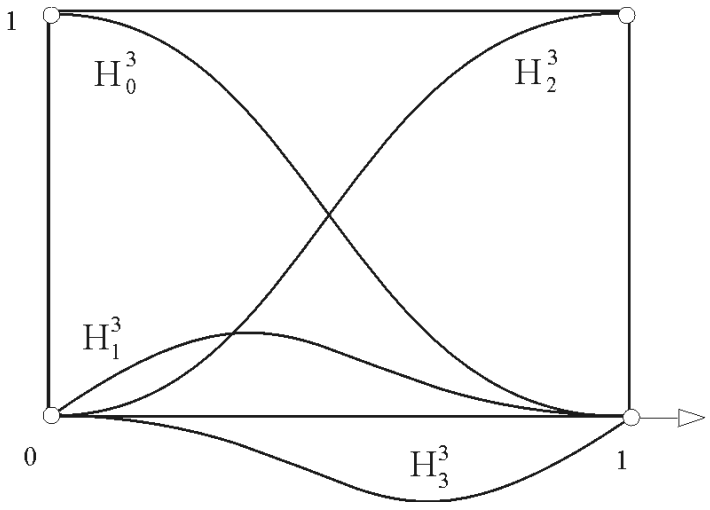
kde

parametr $t \in \langle 0, 1 \rangle$

P_0, P_1, P_0', P_1' jsou tzv. *geometrické koeficienty*,

$H_i^3(t)$ jsou kubické *Hermitovy polynomy* tvaru

$$\begin{aligned} H_0^3(t) &= 2t^3 - 3t^2 + 1 \\ H_1^3(t) &= t^3 - 2t^2 + t \\ H_2^3(t) &= -2t^3 + 3t^2 \\ H_3^3(t) &= t^3 - t^2 \end{aligned} \quad . \tag{40}$$



Obr. 11. Hermitovy polynomy $H_0^3, H_1^3, H_2^3, H_3^3$

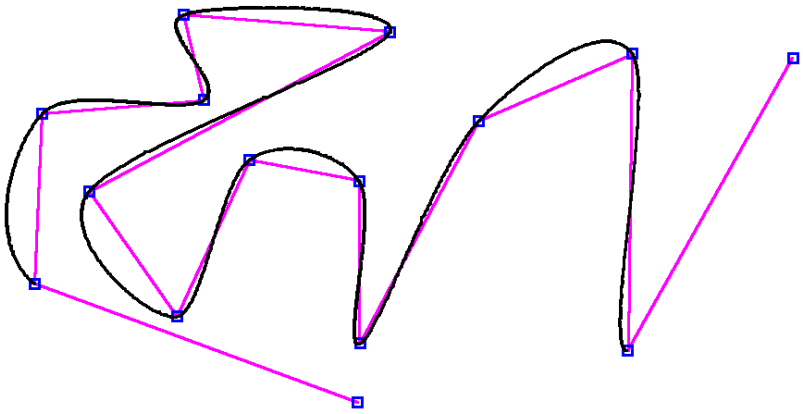
Největší předností těchto křivek je jejich navazování. Ze znalostí tečných vektorů lze napojovat tyto kubiky velmi lehce a zajistit tak bez problémů C^2 spojitost navazujících segmentů.

3.4.2 Catmul-Rom křivka

Tato křivka patří mezi *spline křivky* (3.6) a je tvořena pomocí interpolace mezi jejími řídícími body. Křivka je C^2 spojitá a nemění tedy skokem rychlost ani zrychlení. Catmul-Rom křivka je používána především v *počítačové animaci*.

Je zadána posloupností bodů $P_0, P_1, \ldots, P_n$. Její počátek je v bodě P_1 a končí v bodě P_{n-1}. Neprochází tedy prvním a koncovým bodem řídícího polygonu. Její výpočet je dán následující rovnicí

$$Q(t) = \frac{1}{2}\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{bmatrix}. \qquad (41)$$



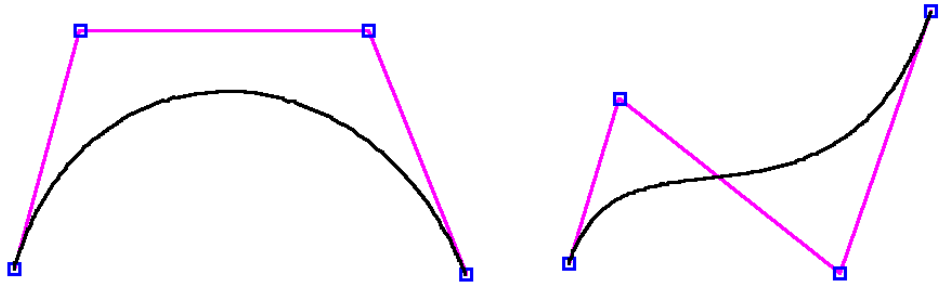
Obr. 12. Animační interpolační Catmul-Rom křivka

Výpočet probíhá postupně po kubických segmentech. Nevýhodou těchto křivek je, že obecně neleží ve své konvexní obálce. Tento aspekt může způsobit jisté potíže při detekcí kolizí objektů, jež se po těchto křivkách pohybují.

3.5 Aproximační křivky

3.5.1 Beziérova křivka

Nejznámější a nejpopulárnější aproximační křivky používané pro modelování ve dvourozměrném i třírozměrném prostoru jsou *Beziérovy křivky*. Tyto křivky se často používají také pro definici písma. Tato metoda umožňuje interaktivní vytváření a modifikaci křivek. Teorii Bézierových křivek vyvinuli nezávisle na sobě P. E. Bézier a P. de Casteljau v letech 1959 – 1962, kteří vyvíjeli Bézierovy křivky a plochy pro CAD/CAM systémy pro navrhování automobilů. Spojení mezi oběma pracemi objevil teprve v roce 1970 R. Forrest.



Obr. 13. Bézierovy kubiky

Bézierova křivka stupně n je určena rovnicí

$$Q(t) = \sum_{i=0}^{n} P_i B_i^n(t), \tag{42}$$

kde

parametr $t \in \langle 0, 1 \rangle$

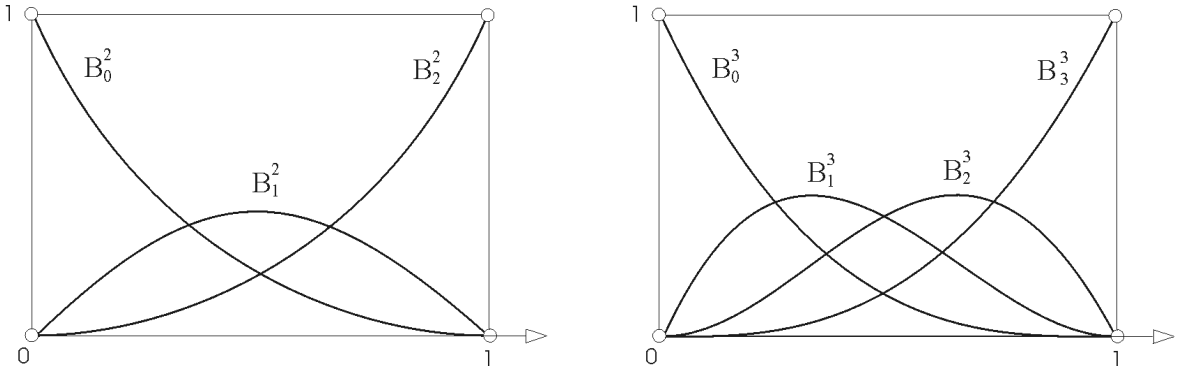
P_i jsou polohové vektory vrcholů řídícího polygonu (*Bézierovy body*),

$B_i^n(t)$ jsou *Bernsteinovy polynomy* n-tého stupně, pro které platí

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}, \quad i = 0, 1, \ldots n. \tag{43}$$

Tyto Bernsteinovy polynomy mají několik důležitých analytických vlastností jako nezápornost, polohu výsledné křivky v konvexní obálce nebo rekurentní charakter pro kombinační čísla

$$\binom{n}{i} = \binom{n-1}{i} + \binom{n-1}{i-1}. \tag{44}$$



Obr. 14. Beziérova kvadrika a kubika

Bézierova křivka prochází počátečním a koncovým bodem svého řídícího polygonu

$$Q(0) = P_0 \text{ a } Q(1) = P_n . \tag{45}$$

V počátečním bodě se Bézierova křivka dotýká strany P_0P_1 řídícího polygonu a v koncovém bodě se dotýká strany $P_{n-1}P_n$. Pro tečné vektory v koncových bodech řídícího polygonu platí

$$\overrightarrow{P'}(0) = n \cdot (P_1 - P_0), \quad \overrightarrow{P'}(1) = n \cdot (P_n - P_{n-1}). \tag{46}$$

Beziérovy křivky leží v konvexní obálce svého řídícího polygonu. Nejsou ivariatní k perspektivní projekci, jsou proto neracionální. Nevýhodou těchto křivek je fakt, že změnou polohy jednoho řídícího bodu křivky dojde ke změně tvaru celé křivky. Zejména tato nepříjemná vlastnost a také narůstající složitost vyjadřovaní Bernsteinových polynomů pro složitější křivky vedla k napojování těchto křivek z jednodušších segmentů a to především *kubik*.

Beziérova kubika

Beziérova kubika je dána čtyřmi řídícími body P_0, P_1, P_2 a P_3 . Prochází prvním a koncovým bodem svého řídícího polygonu a je určena následujícím vztahem

$$Q(t) = B_0^3(t)P_0 + B_1^3(t)P_1 + B_2^3(t)P_2 + B_3^3(t)P_3 = \sum_{i=0}^{3} P_i B_i^3 , \tag{47}$$

Bernsteinovy polynomy stupně tři mají tento tvar

$$\begin{aligned} B_0^3(t) &= (1-t)^3 \\ B_1^3(t) &= 3t(1-t)^2 \\ B_2^3(t) &= 3t^2(1-t) \\ B_3^3(t) &= t^3 \end{aligned} . \tag{48}$$

Maticově můžeme tento zápis vyjádřit

$$Q(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathrm{P}_0 \\ \mathrm{P}_1 \\ \mathrm{P}_2 \\ \mathrm{P}_3 \end{bmatrix} . \tag{49}$$

3.5.2 de Casteljau algoritmus

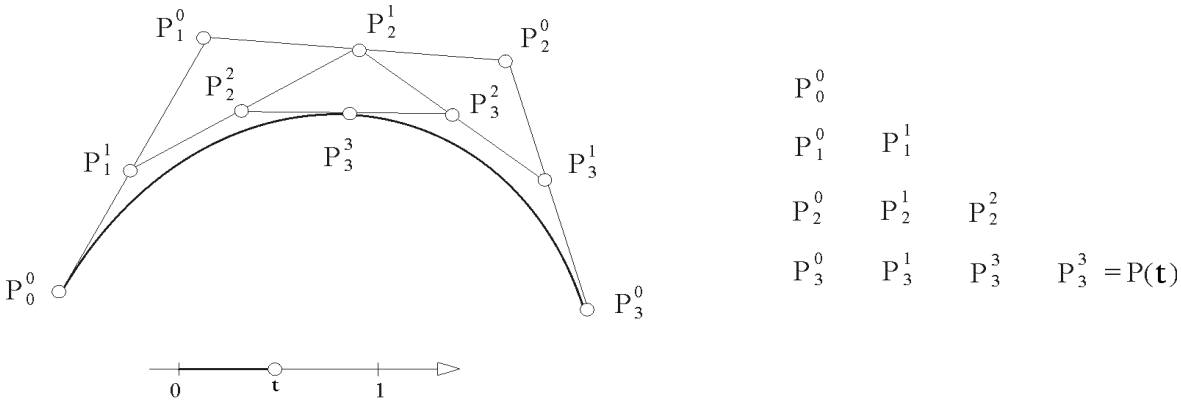
Předchozí popis Bezpérových křivek používá pro jejich vykreslování většinou ekvidistantní dělení časového intervalu $t \in \langle 0,1 \rangle$. V těchto bodech je vypočítávána hodnota křivky. Výsledná křivka se poté aproximuje posloupností úseček. Jiná možnost výpočtu křivky je použít *rekurentní algoritmus de Casteljau*

$$P_i^k(t) = (1-t) \cdot P_{i-1}^{k-1} + t \cdot P_i^{k-1} , \tag{50}$$

kde

k=1,…n a i=0,…,n-k

Vstupem tohoto algoritmu jsou body řídícího polygonu. Dosazujeme za $P_i^0 = P_i$ a rekurentní vztah postupně vyčísluje body P_i^k podle následujícího schématu

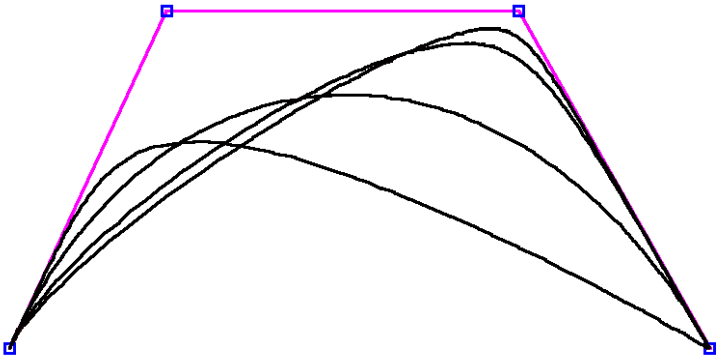


Obr. 15. Algoritmus de Casteljau

Aplikací tohoto algoritmu na Beziérovu kubiku obdržíme tedy *dva nové řídící polygony*, které poté dále rekurentně dělíme. Každé dva nové segmenty můžeme považovat za přesnější aproximace Beziérovy křivky. Kritérium konce dělení můžeme zvolit buď podle počtu iterací, podle vzdálenosti bodů v rastru nebo podle plochy jakou zabírá výsledný polygon. Tohoto faktu se využívá především při adaptivním dělení Beziérovy křivky.

3.5.3 Beziérova racionální křivka

Předchozí popis Beziérových křivek se zabýval jejich neracionální variantou, toto vyjádření má jednu společnou nevýhodu: řídící body jsou určeny jednoznačně. Ke změně tvaru křivky je třeba změnit řídící body. To může být problematické, zvláště v případě, kdy chceme měnit tvar modelované křivky, nikoli však tečny v krajních bodech. Řešení tohoto problému nám nabízejí *racionální Beziérovy křivky*, které jsou reprezentovány nejenom řídícím polygonem, ale také svými homogenními souřadnicemi. Každý bod nese tedy informaci o své *racionální váze* w_i, která určuje jak se konkrétní bod projeví na výsledném tvaru křivky. Položíme-li váhu některého bodu nule, nebude se tento bod podílet na tvaru křivky a bod se stane neaktivním. Racionální váhu si lze také představit jako „gravitaci“, která k sobě přitahuje modelovanou křivku.



Obr. 16. Změna racionální váhy třetího bodu (0,1,5,10)

Hodnotu racionální váhy volíme v intervalu $w_i \in \langle 0, \infty)$. Teoreticky lze váhu zadat i zápornou, poté by křivka měla snahu „odtáhnout se“ od tohoto řídícího bodu.

Matematicky lze tuto křivku vyjádřit

$$Q(t) = \sum_{i=0}^{n} P_i \cdot R_i^n(t) \,. \tag{51}$$

$R_i^n(t)$ jsou tzv. *racionální Bernsteinovy polynomy*, pro které platí

$$R_i^n(t) = \frac{\omega_i \cdot B_i^n(t)}{\sum_{j=0}^{n} \omega_j \cdot B_j^n(t)} \,. \tag{52}$$

3.6 Aproximační Spline křivky

Spline křivka stupně *n* je po částech polynomiální křivka zaručující napojení třídy C^n. Rozlišujeme *přirozenou* (interpolační) a *nepřirozenou* (aproximační) *spline křivku*. V počítačové grafice, kvůli poměrně složitému vyčíslování přirozených spline křivek, se nejvíce uplatnila druhá – nepřirozená – varianta. Spline křivky jsou obecně velmi propracovaná a složitá matematická teorie přesahující rozsah a pojetí této práce. Dále tedy velmi stručně. Podrobnější informace lze nalézt v [1, 2] a především v [4].

3.6.1 Coonsova B-Spline křivka

Coonsova B-Spline křivka (*Uniformní neracionální kubický B-Spline*) je nejznámějším a nejprimitivnějším zástupcem z velké rodiny aproximačních *spline křivek*. Tato křivka vznikne jako C^2 spojité napojení *Coonsových kubických segmentů*. Tyto křivky jsou spojené s prací S.A. Coonse, který tyto křivky začal používat v technické praxi. Coonsova kubika je určena čtyřmi řídícími body P_0, P_1, P_2 a P_3 a kubickými polynomy třetího stupně C_0^3, C_1^3, C_2^3 a C_3^3. Je vyjádřena následující rovnicí

$$Q(t) = \frac{1}{6}\left(C_0^3(t)P_0 + C_1^3(t)P_1 + C_2^3(t)P_2 + C_3^3(t)P_3\right) = \frac{1}{6}\sum_{i=0}^{3} P_i C_i^3 , \tag{53}$$

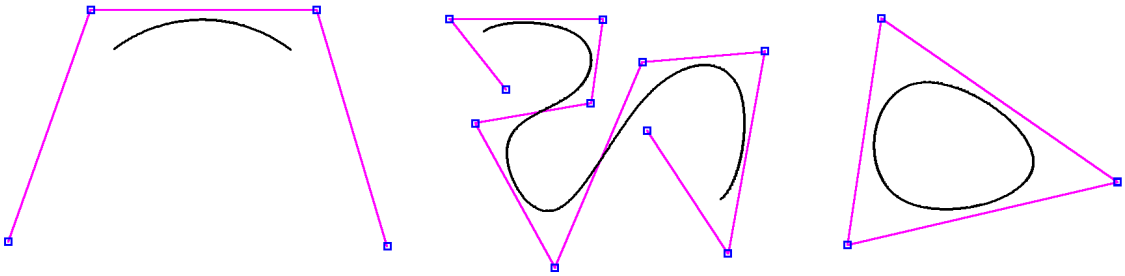
kde

$C_i^3(t)$ jsou *kubické polynomy* tvaru

$$\begin{aligned} C_0^3(t) &= \left(1 - t^3\right) \\ C_1^3(t) &= 3t^3 - 6t^2 + 4 \\ C_2^3(t) &= 3t^3 + 3t^2 + 3t + 1 \\ C_3^3(t) &= t^3 \end{aligned} . \tag{54}$$

Maticově můžeme rovnici Coonsova kubiky vyjádřit

$$Q(t) = \frac{1}{6}\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}\begin{bmatrix} \mathrm{P}_0 \\ \mathrm{P}_1 \\ \mathrm{P}_2 \\ \mathrm{P}_3 \end{bmatrix}. \tag{55}$$



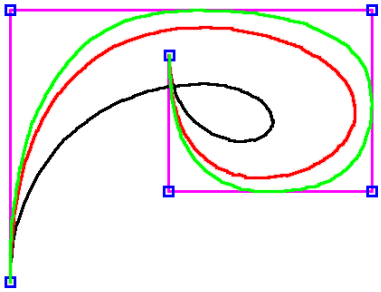
Obr. 17. Coonsova kubika, Coonsova křivka a Coonsova uzavřená křivka

Coonsova křivka obecně *neprochází* prvním a posledním bodem svého řídícího polygonu. Opakováním posledních tří bodů na konci řídícího polygonu obdržíme tzv. *Uzavřený* (periodický) *Coonsův B-Spline*. Při editaci máme možnost zadávat vícenásobné body prostým opakováním následujících bodů řídícího polygonu, vznikají tak *dvojnásobné*,

trojnásobné body, atd. Důležitým aspektem je také *lokalita změny*, která znamená, že pokud změníme polohu některého z řídících bodů, změní se tvar pouze té části Coonsova B-Splinu, který je tímto bodem určen.

3.6.2 Obecná B-Spline křivka

Jestliže se předchozí Coonsova B-Spline křivka pouze dotkla teorie spline křivek, tak u obecných B-Spline křivek je matematické svázání se spline teorií na podstatně komplikovanější úrovni. Coonsovy B-Spliny jsme považovali pouze jako jakési speciální případy B-Spline křivek. Při studiu obecných B-Spline křivek je nutný již poněkud fundovanější přístup k této teorii. M. Cox a C. de Boor objevili nezávisle na sobě v roce 1972 rekurentní vztah pro výpočet B-spline křivek. Tato teorie byla poté použita pro výpočet parametrických B-spline křivek.



Obr. 18. B-Spline, násobnost v uzlovém vektoru (Z=3,Č=4,Černá=6)

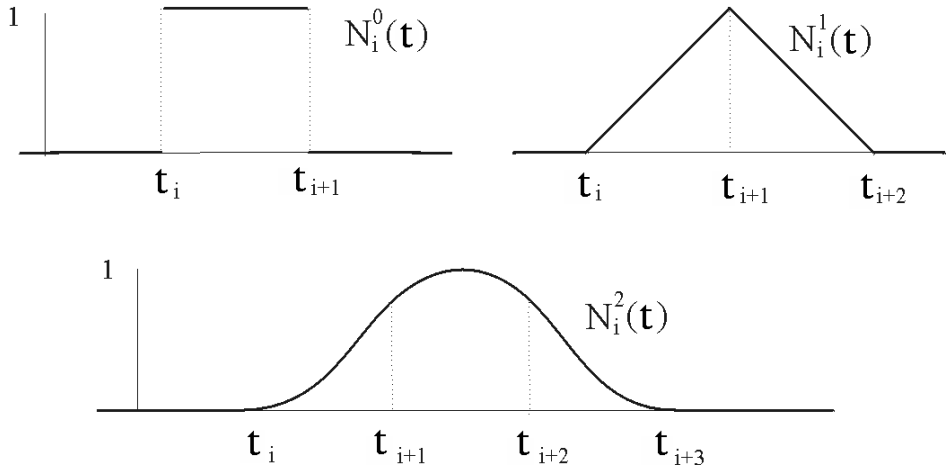
Obecná B-spline křivka stupně k je určena rovnicí

$$Q(t) = \sum_{i=0}^{n} P_i N_i^k(t), \tag{56}$$

kde

P_i jsou řídící body polygonu (*de Boor body*),

$N_i^k(t)$ jsou normalizované bázové funkce stupně k (*de Boor funkce*).



Obr. 19. Příklady bázových funkcí $N_i^k(t)$ stupně 0, 1, 2

Analytické vlastnosti de Boor bázových funkcí

Pro de Boor bázové funkce (normalizované B-Spline báze) platí následující rekurentní vztah

$$N_i^k(t) = \frac{t - t_i}{t_{i+k} - t_i} N_i^{k-1}(t) + \frac{t_{i+k+1} - t}{t_{i+k+1} - t_{i+1}} N_{i+1}^{k-1}(t), \tag{57}$$

kde

$$N_i^0(t) = \begin{cases} 1 \text{ pro } t \in \langle t_i, t_{i+1}) \\ 0 \text{ pro } t \notin \langle t_i, t_{i+1}) \end{cases}. \tag{58}$$

Hodnoty parametrů u_j se nazývají *uzly* (*knots*) . Hovoříme o tzv. *uzlovém vektoru* parametrů:

$$U = (t_0, t_1, \ldots, t_k, t_{k+1}, \ldots, t_{m-k}, \ldots, t_m) , \tag{59}$$

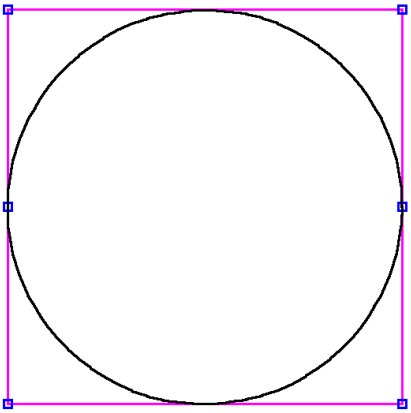
kde $t_i \le t_{i+1}$, $m = n + k + 1$.

de Boor funkce mají i další důležité vlastnosti jako nezápornost, jednotkový součet, lokální vlastnost a okrajový uzlový vektor.

Obecná B-Spline křivka je stejně jako Coonsova B-Spline křivka neracionální uniformní křivka. Neracionalita znamená, že není vyjádřena svými homogenními souřadnicemi a uniformnost je spojena s uzlovým vektorem v tom smyslu, že vzdálenosti jednotlivých uzlů jsou konstantní. Volbou násobností v počátku a konci uzlového vektoru můžeme řídit průběh modelované spline křivky.

3.6.3 NURBS

Neuniformní racionální B-Spline křivky (*Nonuniform Racional B-Splines*) jsou dvojím zobecněním B-Spline křivek. Neuniformnost je odvozena od vzdáleností uzlů ve smyslu parametru t, která nemusí být obecně ekvidistantní. Racionální vyjádření znamená, že body řídícího polygonu jsou vyjádřeny ve svých *homogenních souřadnicích*, každý řídící bod má proto definovanou svou vlastní *racionální váhu* w_i.



Obr. 20. NURBS kružnice

NURBS křivka je definována $n+1$ řídícími body, uzlovým vektorem délky $m+1$ a stupněm křivky k. Uzlový vektor má tvar

$$U = (0,0,\ldots 0,t_{k+1},\ldots,t_k,t_{m-k-1},\beta,\beta,\ldots,\beta)\,. \tag{60}$$

Hodnoty uzlů musí být zadány neklesající $t_i \le t_{i+1}$. Násobnost nul na začátku a násobnost β (obvykle $\beta=1$) na konci je $k+1$. Prostředních hodnot v uzlovém vektoru je $m+1$. Dále platí vztah $m = n + k + 1$.

NURBS křivka je definována

$$Q(t) = \frac{\sum_{i=0}^{n} w_i P_i N_i^k(t)}{\sum_{i=0}^{n} w_i N_i^k(t)}, \tag{61}$$

kde

w_i je váha i-tého bodu řídícího polygonu,

P_i jsou řídící body polygonu (*de Boor body*),

$N_i^k(t)$ jsou normalizované bázové funkce stupně k (*de Boor funkce*).

NURBS křivky jsou v současné době nejpoužívanějším modelovacím nástrojem v CAD/CAM a jejich uplatnění je i v ostatních technických disciplínách nezastupitelné.

NURBS mají několik velmi důležitých vlastností:

1. Procházejí prvním a posledním bodem řídícího polygonu
2. Leží v konvexní obálce svého řídícího polygonu
3. Umožňují vyjádření (Racionálních)Beziérových křivek
4. Jsou invariantní vůči transformacím a především vůči rovnoběžnému a středovému promítání
5. Umožňují přesné vyjádření ***kuželoseček***

Zejména poslední vlastnost představuje nejdůležitější význam NURBS, všechny předchozí křivky totiž nebyly schopny přesně popsat tyto nejpoužívanější konstrukční primitiva. Jedinou nevýhodou je fakt, že pro popis kuželoseček je potřeba složitější popis. Poznamenejme, že NURBS nemají společnou bázi a proto je nemá smysl vyjadřovat maticově.

4 MODELOVÁNÍ PARAMETRICKÝCH PLOCH

Stejně tak jako křivky, mají plochy v počítačové grafice a především v technické praxi nezastupitelné místo. Jejich použití je především při návrhu a konstrukci v CAD/CAM systémech. Nelze opomenout fakt, že většina ploch (křivek) byla vyvinuta právě pro průmyslové využití. Stěžejní význam při vývoji hrály především velké letecké (Boeing) a automobilové (Citroën) koncerny. Motivy, které vedly pro zavedení a rozvoji teorie těchto ploch, jsou charakterizovány snahou oprostit konstruktéra od matematického základu a dovolit modelovat tyto plochy s maximální geometrickou názorností. Modelování probíhá, stejně tak jako u křivek, zadáváním řídících bodů.

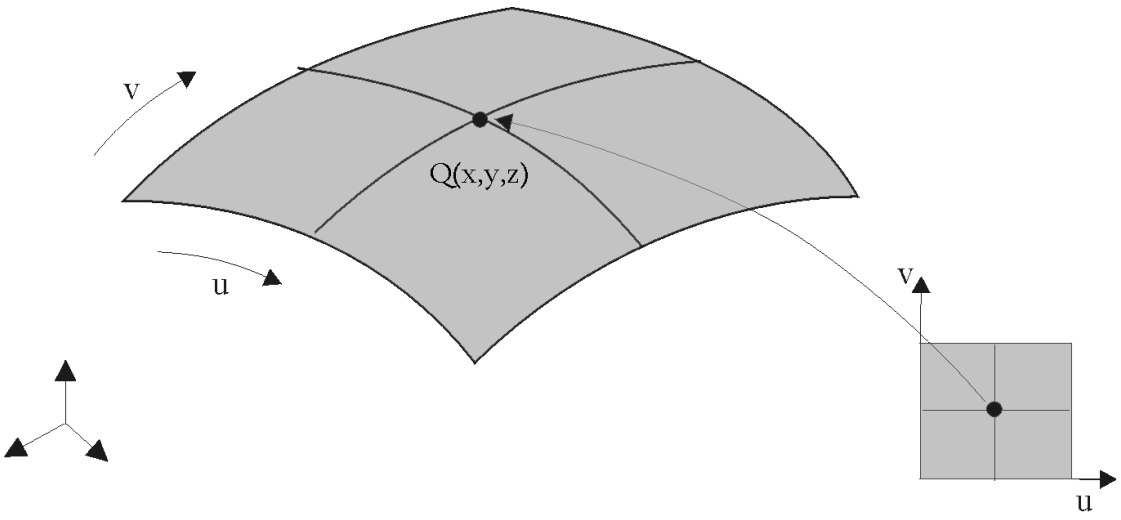
4.1 Vyjádření a základní vlastnosti parametrických ploch

Na parametrické plochy můžeme nahlížet jako na zobecnění teorie křivek. Bodovou rovnicí parametrické plochy $Q(u,v)$ dvou parametrů *u* a *v* vyjádříme

$$Q(u,v) = \left[x(u,v), y(u,v), z(u,v)\right], \tag{62}$$

kde

$x(u,v)$, $y(u,v)$ a $z(u,v)$ jsou polynomiální funkce dvou parametrů $u \in \langle 0,1 \rangle$ a $v \in \langle 0,1 \rangle$.



Obr. 21. Transformace $Q(u,v) \rightarrow Q(x,y,z)$

Bod $Q(x, y, z)$ v trojrozměrném kartézském prostoru má souřadnice $Q(u,v)$ v prostoru parametrickém. Rovnice tečného vektoru $\overrightarrow{q_u}(u,v)$ ve směru parametru u k ploše $Q(u,v)$ je dána první parciální derivací

$$\overrightarrow{q_u}(u,v) = \left(\frac{\partial x(u,v)}{\partial u}, \frac{\partial y(u,v)}{\partial u}, \frac{\partial z(u,v)}{\partial u} \right). \tag{63}$$

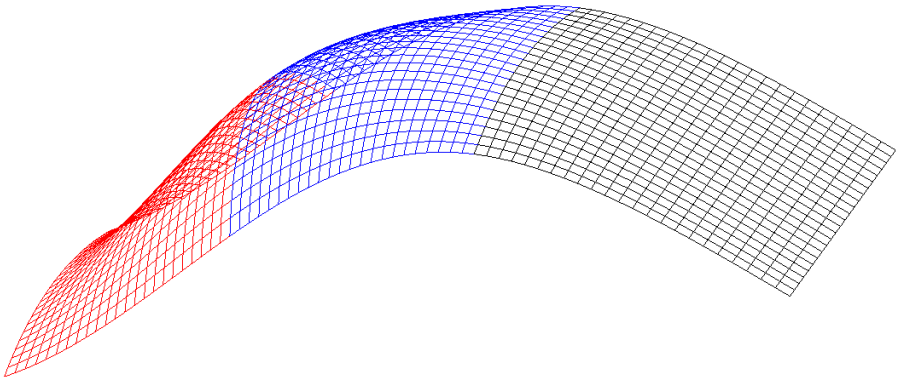
Analogicky vektor $\overrightarrow{q_v}(u,v)$ ve směru parametru v k ploše $Q(u,v)$

$$\overrightarrow{q_v}(u,v) = \left(\frac{\partial x(u,v)}{\partial v}, \frac{\partial y(u,v)}{\partial v}, \frac{\partial z(u,v)}{\partial v} \right). \tag{64}$$

Podobně jako u křivek výslednou plochu skládáme ze segmentů, kterým říkáme *pláty*. Tyto segmenty se nejčastěji popisují jako polynomy třetího stupně ve směrech obou parametrů u a v. Vzniká tak *plátovaná bikubická plocha*.

4.2 Napojování bikubických ploch

Při napojování bikubických plátů platí stejné podmínky jako u napojování kubických křivek Používá se parametrická C a geometrická G spojitost.



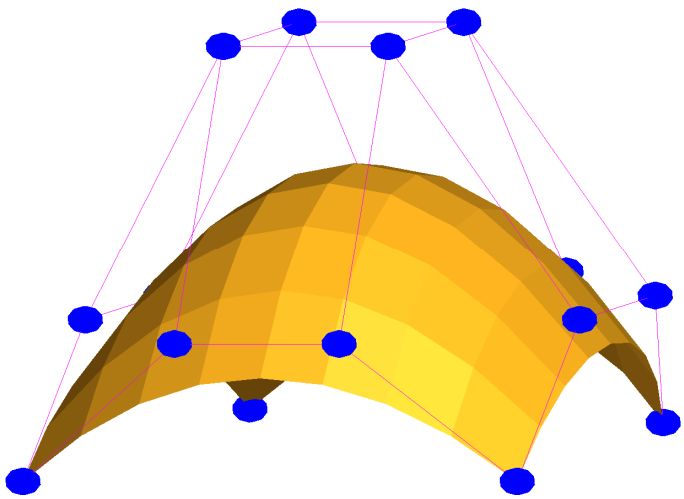
Obr. 22. C^2 spojité napojení bikubických plátů

Dva pláty mají napojení třídy C^0, mají-li společnou stranu, která je křivkou třídy alespoň třídy C^0. Dva pláty mají napojení třídy C^1, mají-li společnou stranu a shodují-li se jejich první parciální derivace ve všech bodech společné strany prvního i druhého plátu. Pro spojitost třídy C^2 platí i rovnost druhých parciálních derivací. Dva pláty jsou G^1 spojité

pokud mají společnou stranu, která je alespoň G^1 spojitou křivkou a jsou-li první parciální derivace podél této strany ve směru napojení lineárně závislé s koeficientem $k > 0$, který se spojitě mění podél této společné strany.

4.3 Modelování parametrických polynomiálních ploch

Plochy zadáváme *řídícími body* a *bázovými funkcemi*. Stejně jako u křivek se pro modelování ploch volí nejčastěji polynomy, protože jsou lehce diferencovatelné a jejich vyčíslování je poměrně rychlé. Vzhledem k dostatečné tvarové modifikovatelnosti a snadnosti výpočtu a napojování se používají převážně polynomy třetího stupně – bikubiky. Navazováním těchto segmentů vzniká *bikubická plátovaná plocha*.



Obr. 23. Řídící polygon bikubické plochy

Reprezentace bikubické plochy se maticově zapisuje

$$Q(u,v) = \mathbf{U}\mathbf{M}_\mathbf{B}\mathbf{P}\mathbf{M}_\mathbf{B}^\mathbf{T}\mathbf{V}_\mathbf{T}, \tag{65}$$

kde

U, $\mathbf{V}_\mathbf{T}$ jsou matice parametrů plochy [3x1] a [1x3],

$\mathbf{M}_\mathbf{B}$ je matice bázových funkcí [4x4],

P je matice řídících bodů [4x4].

Rovnici můžeme rozepsat po složkách do tvaru

$$Q(u,v)=\begin{bmatrix}u^3 & u^2 & u & 1\end{bmatrix}\begin{bmatrix}m_{00} & m_{01} & m_{02} & \mathrm{m}_{03}\\ m_{10} & m_{11} & m_{12} & \mathrm{m}_{13}\\ m_{20} & m_{21} & m_{22} & \mathrm{m}_{23}\\ m_{30} & m_{31} & m_{32} & \mathrm{m}_{33}\end{bmatrix}\begin{bmatrix}p_{00} & p_{01} & p_{02} & p_{03}\\ p_{10} & p_{11} & p_{12} & p_{13}\\ p_{20} & p_{21} & p_{22} & p_{23}\\ p_{30} & p_{31} & p_{32} & p_{33}\end{bmatrix}\begin{bmatrix}m_{00} & m_{10} & m_{20} & \mathrm{m}_{30}\\ m_{01} & m_{11} & m_{21} & \mathrm{m}_{31}\\ m_{02} & m_{12} & m_{22} & \mathrm{m}_{32}\\ m_{03} & m_{13} & m_{23} & \mathrm{m}_{33}\end{bmatrix}\begin{bmatrix}v^3\\ v^2\\ v\\ 1\end{bmatrix}. \quad (66)$$

Modelování probíhá interaktivním zadáváním sítě řídících bodů, která určuje chování modelované plochy. Pro zobrazování na monitoru se používá převodu výsledné plochy na aproximační trojúhelníkovou nebo čtvercovou polygonální síť.

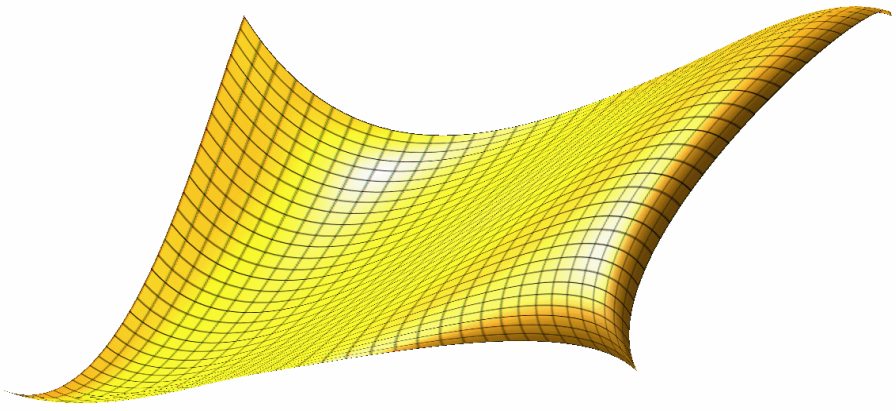
Nejčastější požadované vlastnosti ploch:

- Invariance k lineárním transformacím a projekcím
- Vlastnost konvexní obálky
- Lokalita změny
- Plocha může procházet svými krajními body

4.4 Interpolační plochy

4.4.1 Fergusonova bikubická plocha

Tato plocha vznikla v roce 1964 jako nástroj pro konstruování letadel ve firmě Boeing a je prostým zobecněním Fergusonových křivek (3.4.1) o další prostorový parametr.



Obr. 24. Fergusonova bikubická plocha

Fergusonova bikubika je určena maticovým zápisem

$$Q(u,v) = \mathbf{U} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \mathbf{V}^{\mathrm{T}}. \tag{67}$$

nebo rovnicí

$$Q(u,v) = \sum_{i=0}^{3} \sum_{j=0}^{3} P_{i,j} H_i^3(u) H_j^3(v)\,, \tag{68}$$

kde

parametry $u \in \langle 0,1 \rangle$ a $v \in \langle 0,1 \rangle$,

$P_{i,j}$ je matice řídících bodů,

$H_i^3(u)$ a $H_j^3(v)$ jsou *kubické Hermitovy polynomy* ve směru parametrů *u* a *v*.

Na matici řídících bodů nelze nahlížet jako na prosté pole 16-ti řídících bodů. Zde si je nutno uvědomit analogii v definici Fergusonovy křivky (3.4.1), která je určena řídícími body a tečnými vektory. Ve skutečnosti je tedy Fergusonova plocha zadána 4-mi řídícími body $P_{i,j}$, 8-mi tečnými vektory $v_{i,j}$ a 4-mi zkruty $z_{i,j}$ v rozích a tyto koeficienty tvoří matici řídících bodů podle následujícího schématu:

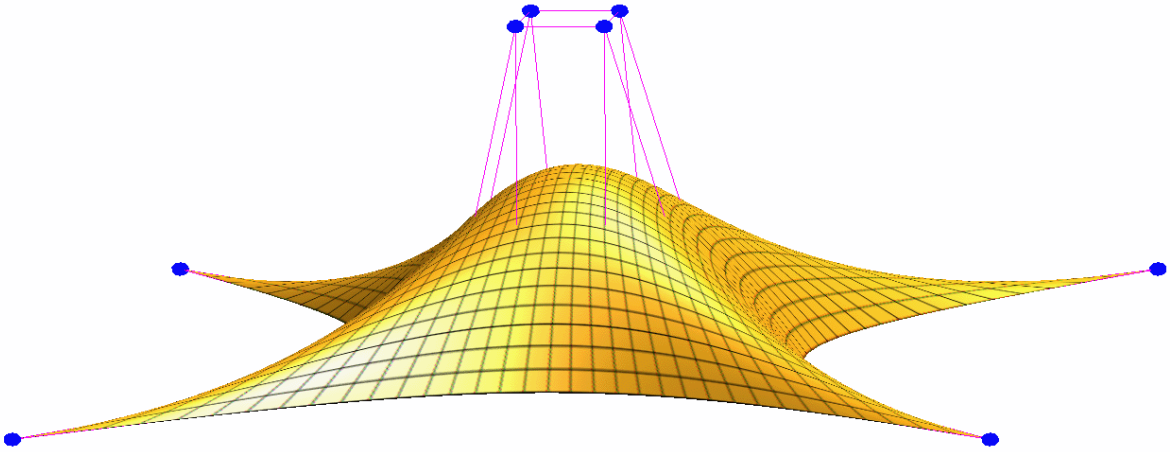
$$\begin{bmatrix} z_{00} & v_{01} & v_{02} & z_{03} \\ v_{10} & P_{11} & P_{12} & v_{13} \\ v_{20} & P_{21} & P_{22} & v_{23} \\ z_{30} & v_{31} & v_{32} & z_{33} \end{bmatrix} \Leftrightarrow \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{bmatrix}. \tag{69}$$

Jsou-li zkruty nulové vektory, jedná se o plochu *dvanáctivektorovou*, která je určena čtyřmi rohy a 8-mi tečnými vektory, jinak hovoříme o ploše *šestnáctivektorové*. Podrobněji rozepsáno v [14].

4.5 Aproximační plochy

4.5.1 Beziérova bikubická plocha

Velmi oblíbeným typem polynomiálních ploch jsou Beziérovy plochy. Tyto plochy lze jednoduše intuitivně modelovat a vyčíslovat. Nevýhodou Beziérových ploch je jejich obtížnější navazování při plátování výsledné plochy z bikubických segmentů a nemožnost vyjádřit některé jednoduché konstrukční prvky (plochy založené na kuželosečkách).



Obr. 25. Beziérova bikubická plocha

Beziérova plocha *n,m-tého* stupně je obecně určena $(n+1)(m+1)$ řídícími body $P_{i,j}$ a rovnicí

$$Q(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m}P_{i,j}B_i^n(u)B_j^m(v)\,, \tag{70}$$

kde

parametry $u\in\langle 0,1\rangle$ a $v\in\langle 0,1\rangle$,

$P_{i,j}$ jsou polohové vektory vrcholů řídícího polygonu (*Bézierovy body*),

$B_{i,j}^{n,m}(u,v)$ jsou *Bernsteinovy polynomy* n,m-tého stupně.

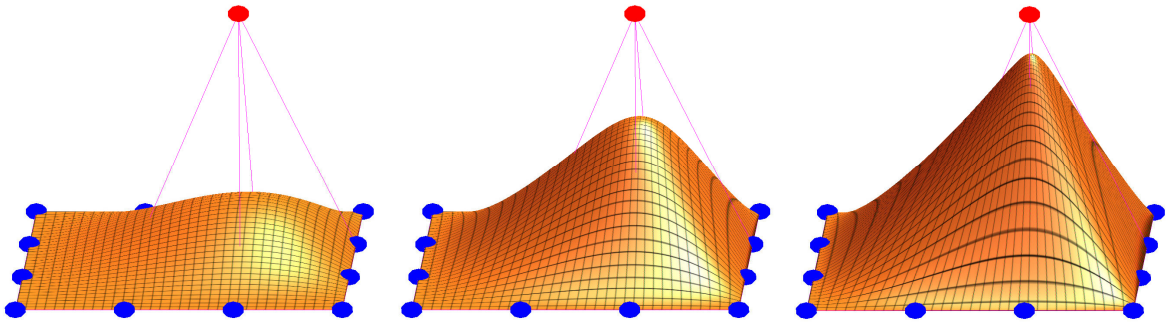
Speciálním a nejčastěji používaným případem Beziérových ploch je *Beziérova bikubika*, kterou obdržíme dosazením za $n=m=3$ do (70).

Maticově je Beziérova bikubická plocha vyjádřena

$$Q(u,v) = \mathbf{U} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \mathbf{V}^{\mathrm{T}}. \tag{71}$$

4.5.2 Beziérova racionální bikubická plocha

Racionální Beziérova křivka představuje trojrozměrnou plošnou analogii racionálních Beziérových křivek (3.5.3). Každý bod řídícího polygonu má definovanou svou váhu představující sílu s jakou se tento řídící bod podepíše na výsledném tvaru plochy. Jsou-li všechny váhy rovny jedné dostáváme klasickou Beziérovu plochu (4.5.1).



Obr. 26. Racionální Beziérova bikubická plocha (váha 1,5,20)

Plocha je určena rovnicí

$$Q(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{i,j} R_{i,j}^{m,n}(u,v), \tag{72}$$

kde

parametry $u \in \langle 0,1 \rangle$ a $v \in \langle 0,1 \rangle$,

$P_{i,j}$ jsou polohové vektory vrcholů řídícího polygonu (*Bézierovy body*),

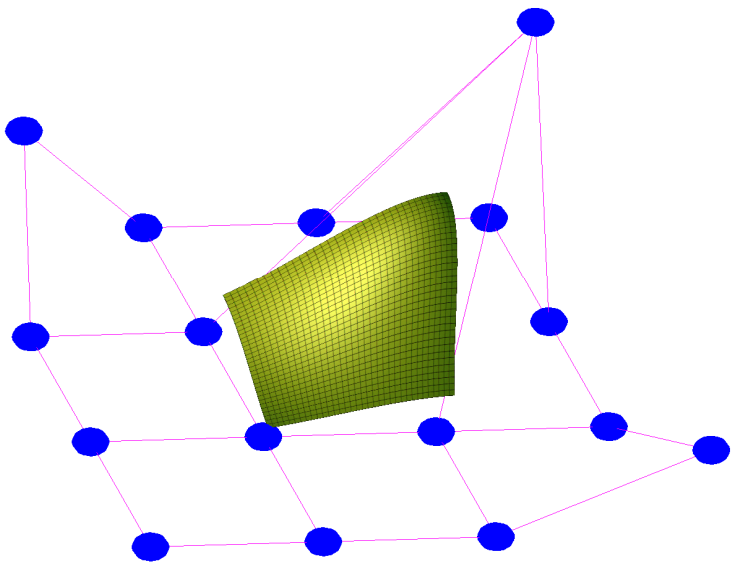
$R_{i,j}^{m,n}(u,v)$ jsou *racionální Bernsteinovy polynomy* n,m-tého stupně ve tvaru

$$R_{i,j}^{m,n}(u,v) = \frac{B_i^n(u) B_j^m(v) w_{i,j}}{\sum_{i=0}^{m} \sum_{j=0}^{n} B_i^m(u) B_j^n(v) w_{i,j}}. \tag{73}$$

4.5.3 Coonsova B-Spline bikubická plocha

Nevýhodou Beziérových ploch (4.5.1) je jejich obtížné navazování. B-Spline plochy, které jsou zobecněním B-Spline křivek (3.6.1) se velice snadno navazují a jsou proto velmi vhodným nástrojem pro modelování. Lze u nich jednoduše zajistit C^n spojitost ve všech bodech a změnou polohy řídícího bodu měníme pouze tvar části této plochy.

B-Spline plochy patří v počítačové grafice mezi velmi důležitou skupinu ploch a jejich obecné vlastnosti a především jejich racionální neuniformní varianta – *NURBS plochy* jsou dnes nejmodernějším a nejsilnějším modelovacím nástrojem vůbec. Tyto partie již však překračují rozsah a pojetí této práce. Více lze nalézt v [15].



Obr. 27. Coonsova bikubická plocha

Nejjednodušší z B-Spline ploch jsou *bikubické Coonsovy plochy*. Tyto plochy jsou na rozdíl od Beziérových ploch C^2 spojité bez nutnosti zadávat nějaké další omezení pro polohu řídících bodů. Tyto plochy nejsou invariantní k perspektivní projekci.

Plocha je určena 16 řídícími body $P_{i,j}$ a rovnicí

$$Q(u,v) = \frac{1}{36}\sum_{i=0}^{3}\sum_{j=0}^{3} P_{i,j} C_i^3(u) C_j^3(v), \tag{74}$$

kde

parametry $u \in \langle 0,1 \rangle$ a $v \in \langle 0,1 \rangle$,

$P_{i,j}$ je matice řídících bodů,

$C_i^3(u)$ a $C_j^3(v)$ jsou *kubické polynomy* (viz 54) ve směru parametrů *u* a *v*.

Maticově je Coonsova bikubická plocha vyjádřena

$$Q(u,v) = \mathbf{U} \frac{1}{36} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 0 & 4 \\ -3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \mathbf{V}^{\mathbf{T}}. \tag{75}$$

5 REALIZACE ALGORITMICKÉ KNIHOVNY

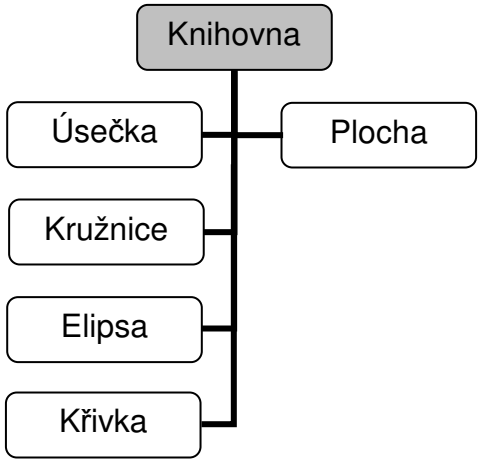
Všechny algoritmy teoreticky popsané v kapitolách 2,3,4 byly vyřešeny programově a lze je najít na přiloženém CD-ROM v adresáři *Knihovna*.

5.1 Základní specifikace

Knihovna algoritmů pro 2D a 3D počítačovou grafiku je napsána v C++ dle mezinárodního standartu ANSI/ISO. Knihovna je postavena na objektovém přístupu, který umožňuje použitelnost algoritmů nejen pod domovským grafickým prostředím *OpenGL*, ale i ve *Windows API* atd. Programátor jednoduchou modifikací knihovny docílí korektní práci knihovny v těchto rozhraních.

5.2 Struktura knihovny

Knihovna zpracovává celkem 21 rasterizačních algoritmů popsaných v teoretické části. Skládá se z objektů reprezentujících jednotlivé skupiny algoritmů. Strukturu knihovny popisuje následující schéma:

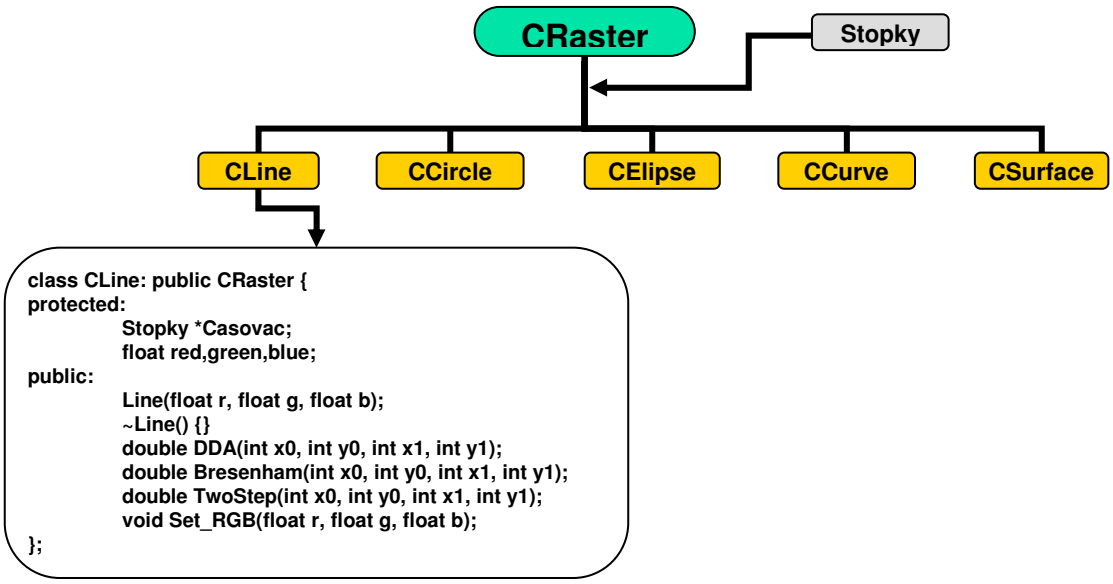


Obr. 28. Sestavení knihovny

5.3 Programové řešení knihovny

Knihovna je založena na objektovém přístupu. Hlavní jádro knihovny je objekt *CRaster* nacházející se v souboru *CRaster.h* který je společný předek všech algoritmických objektů. Řídící direktivy v tomto souboru ovládají použité modifikace objektu *CRaster*. Tyto modifikace umožňují použít pro rasterizaci buď *OpenGL* nebo standardní *WinApi*. Pro použití knihovny pod jiným grafickým rozhraním stačí doprogramovat příslušnou variantu objektu CRaster, do struktury knihovny nemusí být dále zasahováno.

Knihovna dále obsahuje 5 objektů *CLine*, *CCircle*, *CElipse*, *CCurve* a *CSurface* reprezentující jednotlivé rasterizační algoritmy. Společným prvkem všech algoritmů je knihovna *Stopky*, která zajišťuje změření potřebného rasterizačního času algoritmů. Každý algoritmus je volán svou *metodou* vracející dobu rasterizace v milisekundách.



Obr. 29. Objektová struktura knihovny

Struktura všech objektů je velmi podobná, popíšeme tady pouze objekt pro rasterizaci *Úseček – CLine*. Ostatní objekty jsou principiálně totožné. Objekt *CLine* je nejprve deklarován v hlavičkovém souboru CLine.h (viz Obr. 29.). Soubor *CLine.cpp* pak obsahuje implementaci metod tohoto objektu.

Pro měření rasterizačního času algoritmů slouží objekt *Stopky*. Pro Win32 verzi knihovny se využívá knihovna *windows.h.* V jiných operačních systémech je využíváno měření času pomocí multiplatformní knihovny SDL (Simple DirectMedia Layer – knihovna pro vytváření multiplatformních multimediálních aplikací) [12]. Knihovna SDL měří pouze s přesností *ms*, Win32 verze dokáže určit tisíciny *ms*.

5.4 Vzorové použití knihovny

Na přiloženém CD-ROM lze najít v adresáři *Knihovna* dva další adresáře (*OpenGL* a *Win32API*) sloužící jako ukázkové aplikace využívající rasterizační knihovny.

V adresáři *OpenGL* je naprogramována demonstrace použití knihovny v prostředí Visual Studia 6.0 C++ pod OpenGL (7.2.1). Pro inicializaci a správu oken demo využívá knihovnu GLUT (7.2.2) a samotný zdrojový kód může být přeložen na libovolné X-Window platformě.

V adresáři *Win32API* je příklad použití knihovny v prostředí Visual Studia 6.0 C++ pod standardním Windows API GUI rozhraním.

V příloze P1 a P2 lze nalézt dvě ukázky algoritmů. V prvním případě se jedná o *Bresenhamův lineární interpolátor* a ve druhém o *Beziérovu kubickou křivku*. Ostatní algoritmy jsou již součástí CD-ROM.

6 SROVNÁNÍ A POUŽITELNOST ALGORITMŮ

Rasterizační algoritmy mají v počítačové grafice a mnoha souvisejících aplikacích nezastupitelné místo. Jejich uplatnění sahá od jednoduchých grafických skic až po velmi složité modelovací průmyslové systémy. Historie vývoje těchto algoritmů vždy šla společně s historií počítačového konstruování a zobrazování. Sledování rasterizační rychlosti, rasterizační kvality, modelovací schopnosti a použitelnosti jsou velmi důležité ukazatele, které umožňují efektivně využívat těchto algoritmů.

6.1 Rasterizace základních grafických elementů

Při rasterizaci základních grafických elementů (úsečka, kružnice, elipsa) je nejdůležitějším sledovaným parametrem *rasterizační rychlost*. I v dnešní době je žádoucí rasterizovat tyto objekty co možná nejrychleji. Napojováním těchto elementů lze vytvářet i velmi složitou liniovou strukturu. V modelovacích systémech typu AutoCAD může při návrhu a konstrukci ve výkresu existovat i několik tisíc těchto primitiv.

Nejrychlejšími a také nejdoporučovanějšími rasterizačními algoritmy jsou algoritmy založené na práci v digitální rovině – *Bresenhamovy algoritmy* pracují pouze v celočíselné aritmetice a jejich rasterizační rychlost je ve všech směrech nepřekonatelná.

6.2 Vhodnost použití parametrických křivek

Parametrické křivky patří mezi velmi využívanou skupinu grafických prostředků. Lze se s nimi setkat téměř ve všech grafických programech. Jednodušší modelovací a kreslící programy pracují především s *Beziérovými křivkami* (3.5.1). Práce s těmito křivkami je velmi jednoduchá a intuitivní. Nevýhodou těchto křivek je jejich obtížné navazování. Situaci částečně řeší *spline křivky* (3.6), které se navazují velmi lehce.

Společnou nevýhodou výše uvedených křivek je nemožnost vyjádření velmi jednoduchých konstrukčních primitiv jako např. kružnice. Moderní modelovací aplikace jsou proto založeny na technologii *NURBS křivek* (3.6.3), které jsou racionálním neuniformním

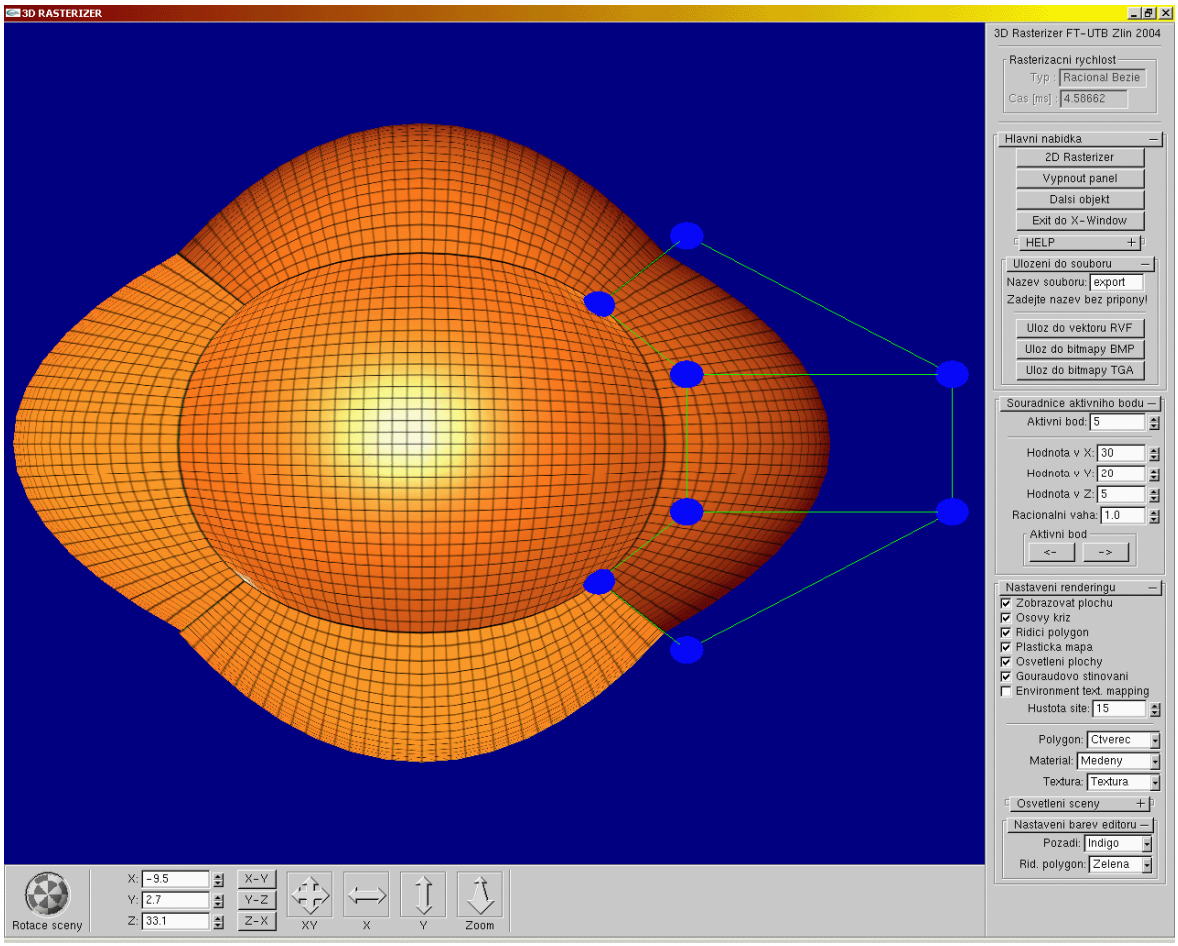
zobecněním spline křivek. Pomocí těchto křivek lze popsat a vyjádřit libovolnou kuželosečku, tyto křivky jsou invariantní k afinním transformacím

V počítačové animaci se uplatňují interpolační *Fergusonovy křivky* (3.4.1), *Catmul-Rom křivky* (3.4.2) a *Kochanek-Bartels křivky* [15].

6.3 Vhodnost použití parametrických ploch

Při studiu parametrických ploch dojdeme ke stejným závěrům jako při posuzování parametrických křivek. Nejjednodušší variantou jsou *Beziérovy plochy* (4.5.1), obtížně se však navazuji a modelování v 3D prostoru již potřebuje poměrně dobrou konstruktérskou představivost. *Coonsovy plochy* (4.5.3) se plátují velmi lehce. *Fergusonova plocha* (4.4.1) se dříve používala ke konstrukci letadel ve firmě Boeing a byla ve své době pokrokovým modelovacím nástrojem. Všechny dnešní moderní CAD/CAM systémy používají pro modelování technologii *NURBS ploch* [4,15]. Tyto plochy představují dvouparametrickou analogii NURBS křivek (3.6.3). Studium těchto ploch již nebylo součástí této práce.

7 SOFTWAROVÁ APLIKACE RASTERIZER



Obr. 30. Modelovací prostředí ve 3D

7.1 Charakteristika aplikace

Rasterizer je X-Window softwarová aplikace vyvinutá pro interaktivní intuitivní modelování 2D a 3D počítačové grafiky. Jeho základem je vytvořená knihovna rasterizačních algoritmů. Software je určený pro výuku předmětu *Počítačová grafika*, realizovaném kabinetem aplikované informatiky. Využití však nelze spatřovat pouze jako výukový software. Rasterizer lze dále využít *pro návrh* a *design výrobků*, tvorbu liniové grafiky, atd. Ukázková grafika modelovaná tímto programem je zobrazena v příloze P3-P6. Spustitelný program pro Win32 je umístěn na CD-ROM v adresáři *Rasterizer/bin*. Kompletní zdrojový kód je umístěn v adresáři *Rasterizer/source*. Rasterizer umožňuje

export modelované grafiky do interního vektorového formátu *rvf* nebo do klasické bitmapy BMP nebo Truevision Targa TGA.

7.1.1 Hardwarové nároky

Aplikace pro svůj spolehlivý chod vyžaduje počítač s hardwarovým grafickým akcelerátorem s podporou OpenGL. Rozlišení pracovní plochy pro grafické aplikace je obecně požadováno co možná nejvyšší (min 800x600, optimum 1280x1024 a více). Doporučeno je zejména použití *kvalitního grafického akcelerátoru*. Aplikace nemusí korektně a spolehlivě pracovat na tzv. „on-board" grafických kartách, které jsou součástí základní desky počítače.

Minimální a doporučená konfigurace počítače pro platformu IBM PC

Minimální konfigurace:

Pentium II (Duron 600), 32 MB RAM, OpenGL grafická karta 16 MB VRAM, rozlišení SVGA 800x600x16bpp.

Doporučená konfigurace:

Pentium IV (Athlon XP 1600+), 256 MB RAM, OpenGL grafická karta 64 MB VRAM, rozlišení SVGA 1280x1024x32bpp a více.

7.2 Zobrazovací a modelovací možnosti

Rasterizer umožňuje ve 2D modelovat pomocí úseček, kružnic, elips a parametrických křivek. Ve 3D módu používá parametrické plochy. Celkem je možno použít 21 různých modelovacích algoritmů, tzn. všechny algoritmy obsažené ve vyvinuté rasterizační knihovně (kapitola 5).

Při modelování ve 2D i 3D lze interaktivně měnit polohu řídících bodů a docílit tak požadovaného průběhu křivky. Ve 2D je možné *řídit kvalitu* renderovaných křivek změnou hustoty ekvidistantní úsečkové aproximace těchto křivek. Ve 3D módu lze měnit hustotu

polygonální sítě, plasticky stínovat objekty *konstantním* nebo *Gouraudovým* stínováním. Měnit polohu a barvu osvětlení, texturovat objekty a libovolně s nimi rotovat, posunovat a zoomovat.

Lze nastavovat barvu pozadí editoru, barvu aktivních bodů, řídícího polygonu a měnit mnoho dalších uživatelských nastavení.

7.3 Ovládání a uživatelské rozhraní

Základem ovládání je práce s rozbalovacím panelem, který je umístěn v pravé části pracovní plochy. Dále se program ovládá pomocí kaskádovitého pop-up menu, které se aktivuje stiskem *pravého tlačítka myši* a pomocí klávesnice. Program pracuje ve dvou modelovacích módech – 2D a 3D. Podrobní informace o ovládání a práci s programem jsou k dispozici také v nápovědě *help.htm*, která je umístěna v adresáři programu.

7.3.1 Spuštění programu a nahrání uloženého souboru

Program se spouští ve svém adresáři příkazem *rasterizace.exe*. Chceme-li nahrát již existující uložený soubor zadáme jeho název jako řádkový parametr (i s příponou).

Př.

rasterizace.exe plocha.rvf - nahraje soubor *plocha.rvf* z aktuálního adresáře
rasterizace.exe rvf/cajník.rvf - nahraje soubor *cajnik.rvf* z adresáře *rvf/*

7.3.2 Ovládání

Aplikace se ovládá pomocí myši a klávesnice. Podrobný návod k ovládání programu je umístěn na CD-ROM v adresáři programu (*/Rasterizer/bin*). *Help* je vytvořen formou jednoduché internetové stránky a jeho spuštění provedete příkazem *help.htm*. Uživatel může dále využít ovládací panel s mnoha možnostmi nastavení.

7.3.3 Konfigurace textur

Program ve 3D módu umožňuje jednoduchou vizualizaci drátového modelu pomocí texturované stínované trojúhelníkové nebo čtvercové polygonální sítě. Textury, které chceme v programu použít musí být ve formátu *bmp* a jejich jméno a přístupová cesta musí být uvedena v konfiguračním souboru textur – *textury.cfg*.

7.4 Použité softwarové prostředky

Rasterizer je naprogramován, stejně tak jako rasterizační knihovna, v jazyku C++. Zdrojový kód není vázaný na žádný operační systém a může být jednoduše exportován do libovolného X-Window prostředí. Mateřská Windows verze aplikace je napsána v prostředí *Microsoft Visual C++ 6.0*. Pro zobrazování je využíváno grafické rozhraní *OpenGL*, které je v současné době průmyslovým standardem pro grafické aplikace. Správu oken a spolupráci s X-Window operačním systémem zajišťuje knihovna *GLUT* a ovládací prvky jsou řešeny za použití knihovny *GLUI*.

7.4.1 OpenGL

OpenGL je grafické rozhraní pro definici a zobrazování 3D objektů. Knihovna OpenGL je na platformě nezávislá a lze ji provozovat na téměř jakémkoli operačním systému a počítačové architektuře současnosti. (Windows, Linux, UNIX, MacOS, SGI, …). Knihovna sama o sobě vznikla před více než dvaceti lety na počítačích Silicon Graphics (SGI) a od té doby se stále vyvíjí. Původní název na SGI byl IrisGL a pojmenování OpenGL dostala knihovna na počátku devadesátých let, kdy byla přenesena na platformu IBM PC.

OpenGL pracuje na architektuře *klient-server* a využívá v maximální možné míře podporu akcelerátorů pracujících na principu *proudového zpracování dat* známého např. z architektury procesorů RISC. Proudové zpracování spočívá v rozdělení posloupnosti grafických operací do menších celků, které řeší speciálně navržený hardwarový obvod akcelerátoru. Výsledek je poté složen z těchto dílčích operací. Mezi typickou zpracovávací pipeline patří např. následují sekvence: *modelovací transformace* →

zobrazovací transformace → projekce a ořezání → transformace pracoviště → výsledný obraz.

Mezi specializované hardwarové obvody akcelerátorů patří např. Depth-Buffer, Stencil Buffer, texturovací jednotka, vyhlazovací jednotka, ořezávací jednotka atd.

V současné době je OpenGL ve verzi 1.5 a ve vývoji je revoluční verze číslo 2.0, která by měla do standardu zapracovat velkou množinu nových grafických funkcí jako *pixel* a *vertex shadering* nebo *dynamic shadows* . Kvalita zobrazované scény by na dostatečně rychlém akcelerátoru měla dosahovat úrovně filmových triků, renderovaných ovšem v reálném čase. V současnosti je největší využití OpenGL především v oblasti grafického software a herního průmyslu. OpenGL využívá např. *3D Studio MAX*, *Alias*, *Lightwave 3D, Maya, Softimage*. Z počítačových her dominují 3D enginy od id Software nebo Valve.

Programově se s OpenGL pracuje velmi jednoduše pomocí neobjektového jazyka C. Práce se, velmi zjednodušeně, skládá z definice skupin vrcholů, textur a normál. Tyto skupiny poté posíláme do renderovacího řetězce. Oproti konkurenční knihovně DirectX je zdrojový kód potřebný pro realizaci grafické operace podstatně jednodušší a přehlednější.

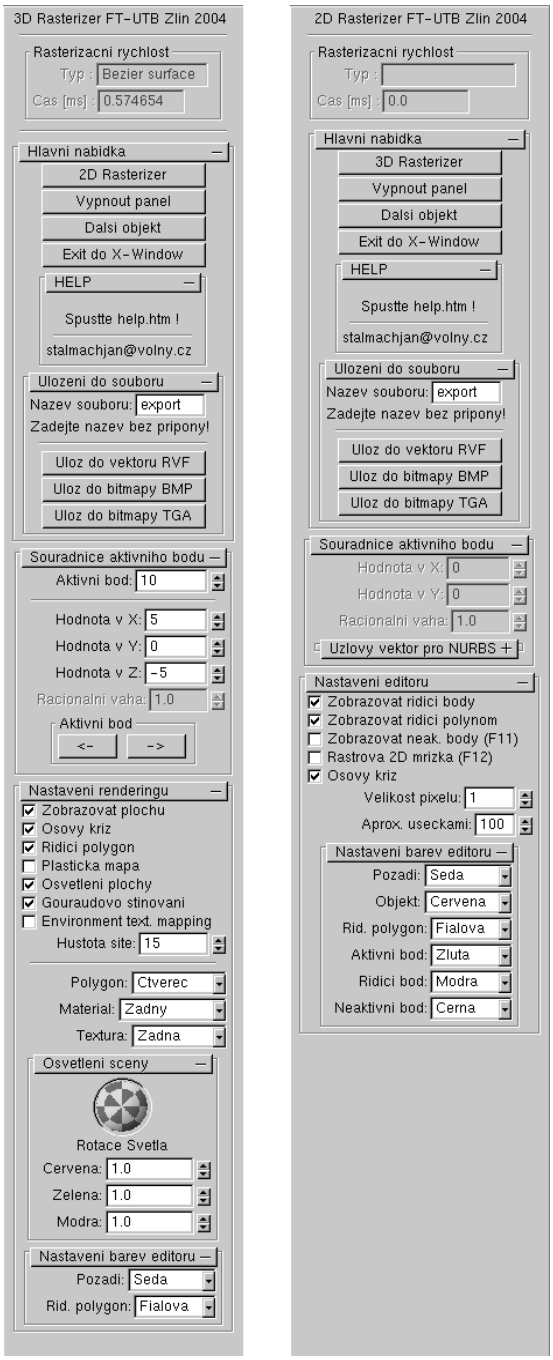
7.4.2 The OpenGL utility toolkit

The OpenGL utility toolkit (GLUT) je jednoduché a velmi výkonné na platformě nezávislé programové rozhraní (knihovna) určené pro vytváření jednoduchých uživatelských rozhraní (GUI) pro OpenGL aplikace. Glut má tyto vlastnosti a schopnosti:

- Jednoduché vytváření oken pro OpenGL rendering
- Zpracování událostí založené na Callback funkcích
- Víceúrovňová kaskádová pop-up menu
- Možnost renderování základních geometrických objektů
- Použití bitmapových a vektorových fontů
- Práce s barevnou paletou

7.4.3 The OpenGL utility interface

The OpenGL utility interface (GLUI) představuje uživatelské rozhraní založené na OpenGL GLUT okenním rozhraní. Tato nadstavba umožňuje používat pro práci a uživatelskou interakci klasické ovládací prvky typické pro X-Window aplikace. GLUI umožňuje implementovat do prostředí správu uživatelských panelů, rozbalovací nabídky, spinnery, text boxy, list boxy a další speciální funkční bloky pro ovládání grafických aplikací.



Obr. 31. Ovládací panely pro 3D a 2D mód

ZÁVĚR

Bakalářská práce se zabývá aplikací rasterizačních algoritmů v oblasti počítačové grafiky nazvané počítačová geometrie. V úvodních kapitolách je vypracována teorie 21 algoritmů, které se zabývají rasterizací základních grafických elementů, parametrických polynomiálních křivek a ploch.

V praktické části je naprogramována v *objektovém C++* knihovna obsahující všechny teoreticky popsané modelovací algoritmy. Knihovnu lze provozovat pod rozhraním *OpenGL* nebo *Windows API*. Další grafická rozhraní lze snadno doprogramovat a docílit tak použití na libovolné grafické platformě.

Na základě vytvořené knihovny a teoretické části je srovnána rychlost a vhodnost algoritmů pro použití v počítačové grafice a konstrukci v CAD/CAM.

Dále je v C++ realizována softwarová aplikace *Rasterizer* sloužící pro intuitivní interaktivní modelování 2D a 3D počítačové grafiky. Software je hlavní výstup bakalářské práce a bude sloužit jako výuková pomůcka pro potřeby kabinetu Aplikované informatiky FT-UTB Zlín. Základem softwaru je vyvinutá rasterizační knihovna a grafické rozhraní *OpenGL*. Zdrojový kód aplikace je multipatformní a může být jednoduše portován z domovského *Windows32* prostředí do libovolného *X-Window* systému. Grafika modelovaná tímto programem byla použita pro demonstraci rasterizačních algoritmů v části práce popisující teoretický aparát. Další modelovanou grafiku lze najít v příloze.

SEZNAM POUŽITÉ LITERATURY

[1] DRDLA, J. *Analýza křivek a ploch*. UP, Olomouc, 2001

[2] DRDLA, J. *Geometrické modelování křivek a ploch*. UP, Olomouc, 2001

[3] HUDEC, B. *Základy počítačové grafiky*. ČVUT, Praha,2001, ISBN 80-01-02290-0

[4] LAVOIE, P. *NURBS++ library 3.0.11*. 2002, Dostupné z: <http://yukon.genie.uottawa.ca/~lavoie/software/nurbs>

[5] MARTIŠEK, D. *Matematické principy grafických systémů*. Litera, Brno, 2002, ISBN 80-85763-19-2

[6] MARTIŠEK, D. *Technické aplikace digitální geometrie – Zkrácená verze habilitační práce. VUT*, Brno, 2003, ISBN 80-214-2459-1

[7] PELIKÁN, J. *CGEL LIBRARY*. Univerzita Karlova, Praha, 1995

[8] PELIKÁN, J. *Kreslení čar*. Univerzita Karlova, Praha, 2001, Prezentace

[9] PETZOLD, CH. *Programování ve Windows*. Computer Press, Praha, 1999, ISBN 80-7226-206-8

[10] POKORNÝ, P. *Základy počítačové grafiky*. UTB, Zlín, 2004. ISBN 80-7318-161-4

[11] RADEMACHER, P. *GLUI – A GLUT Based User Interface Library*. Referenční manuál, 1999

[12] LANTINGA, S., *Simple DirectMedia Layer library 1.2.7*. Verze z 18.2.2004, Dostupné z: <http://www.libsdl.org/>

[13] VIRIUS, M. *Programování v C++,* ČVUT, Praha, 2001.ISBN 80-01-01874-1

[14] ŽÁRA, J. a kolektiv. *Počítačová grafika – principy a algoritmy*. Grada, Praha, 1992, ISBN 80-85623-00-5

[15] ŽÁRA, J., BENEŠ, B. A FELKEL, P. *Moderní počítačová grafika*. Computer Press, Praha, 1998, ISBN 80-7226-049-9

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH