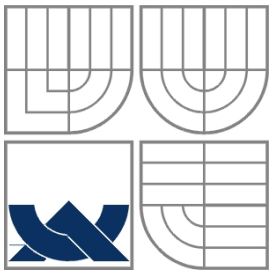
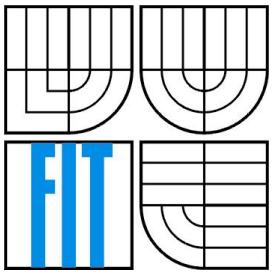


Vysoké učení technické v Brně

Brno university of technology



Fakulta informačních technologií

Ústav počítačové grafiky a multimédií

Faculty of information technology

Department of Computer Graphics and Multimedia

Efektivní uživatelské rozhraní pro editaci parametrických křivek

Effective User Interface for Parametric Curves Editing

Bakalářská práce

Bachelor's thesis

Autor

Author

Milan ONDRUCH

Vedoucí práce

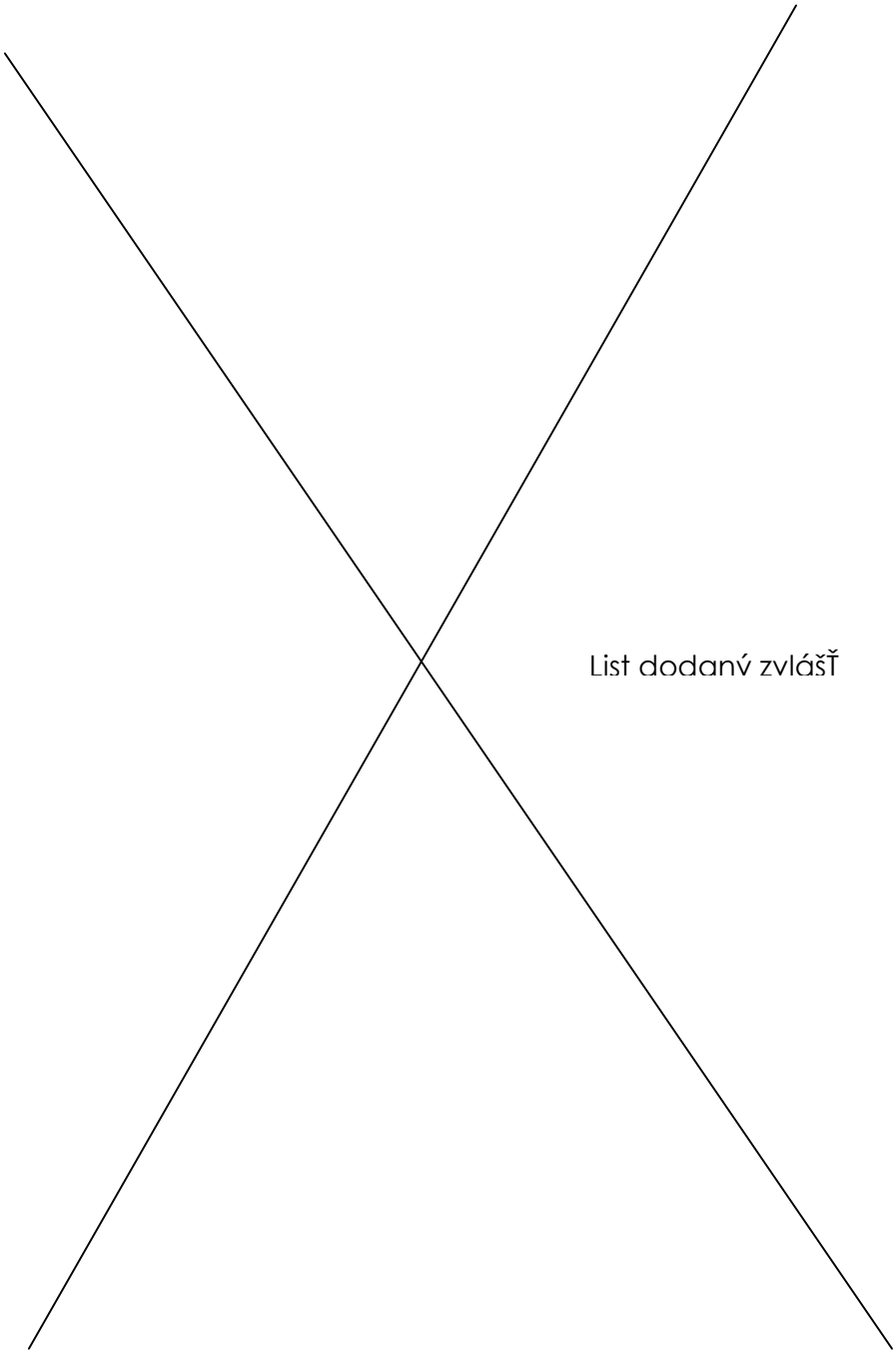
Supervisor

Ing. Adam HEROUT, Ph.D.

Brno 2007

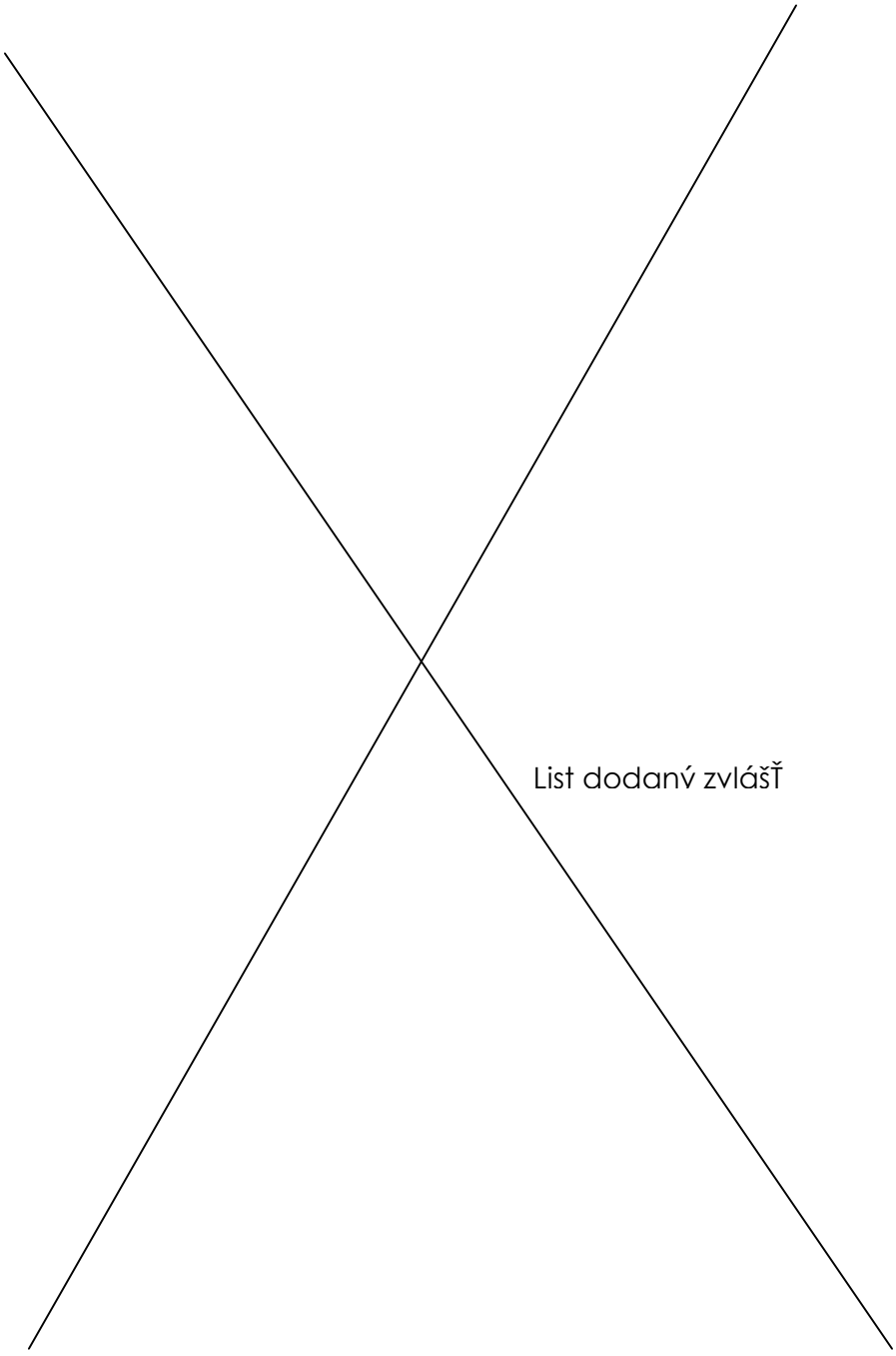
Zadání

Originál zadání bakalářské práce následuje bezprostředně za titulním listem.



Licenční smlouva

Vyplněná licenční smlouva dle přílohy rozhodnutí rektora č.9/2007. Licenční smlouvu obdrží student současně se zadáním bakalářské práce.



Abstrakt

V současnosti je ke křivce přistupováno nejčastěji pomocí polygonu složeného z řídících bodů. Zvláště u složitých křivek se výsledek ztrácí ve změti čar tvořících tento polygon. Tato bakalářská práce navrhuje algoritmus editující křivku bez použití řídících bodů. Výsledkem je nástroj upravující křivku intuitivním způsobem, připomínajícím práci s plastickou hmotou. Ukázkový program je implementován v jazyce C++ s grafickým výstupem přes knihovnu OpenGL a okenní nadstavbou OpenGlut.

Klíčová slova

NURBS, deformace křivky, OpenGL, GLUT, OpenGLUT, C++, uživatelské rozhraní

Abstract

At present, the standard method of accessing a curve is using a polygon composed of control points. When dealing with complex curves, the result becomes lost in a clutter of lines comprising this polygon. This bachelor thesis proposes an algorithm for curve editing without the use of control points. The output is a tool that modifies the curve in an intuitive way reminiscent of work with plastic material. A sample program is implemented in C++ language with a graphic output by means of OpenGL library and OpenGLUT window toolkit.

Keywords

NURBS, curve deformation, OpenGL, GLUT, OpenGLUT, C++, user interface

Bibliografická citace dle ČSN ISO 690

Milan Ondruch: Efektivní uživatelské rozhraní pro editaci parametrických křivek, bakalářská práce, Brno, FIT VUT v Brně, 2007

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Adama Herouta, Ph.D. Uvedl jsem všechny zásadní literární prameny a publikace, ze kterých jsem čerpal.

_____

Milan ONDRUCH

Obsah

1. Úvod

Se zvyšujícím se počtem používání tzv. high polygon modelů definovaných především parametrickými plochami, roste nejen počet uživatelů žádajících editory těchto ploch, ale i počet možností a nástrojů, jak tyto plochy definovat. Velké množství nástrojů je samozřejmě zátěž pro uživatelské rozhraní aplikace. Tato práce mapuje trendy v současném editování parametrických křivek a představuje moderní prvky uživatelských rozhraní.

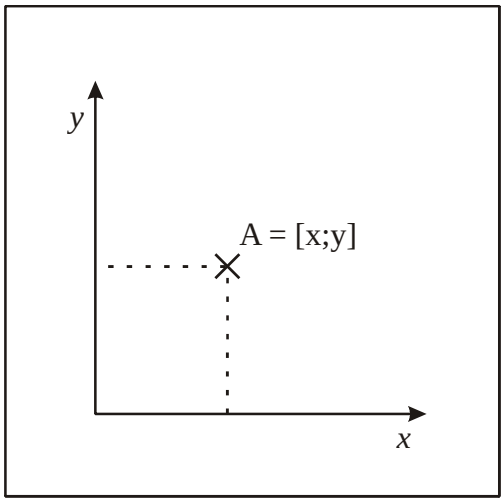
Pro zasvěcení do problematiky křivek se připomínají základy analytické geometrie a následně se prochází používanými typy křivek až ke křivkám NURBS.

Druhou částí práce je návrh nástroje pro intuitivní úpravu tvaru parametrické křivky – konkrétně NURBS. V poslední řadě se zde uvažuje o jeho implementaci ve 3D.

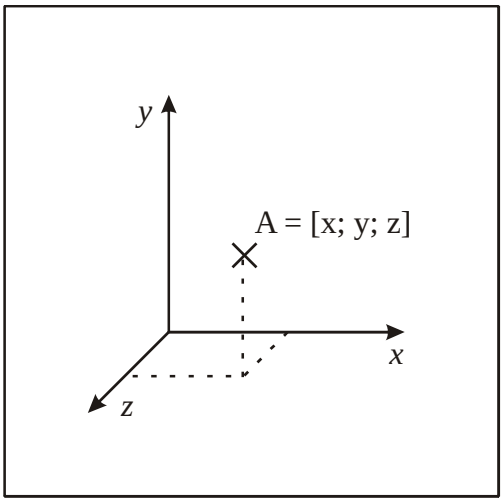
2. Analytická geometrie

Narýsuje-li člověk čáru, nemusí umět ani malou násobilku. Stačí pevná ruka a rýsovací potřeby. Počítačové vidění však nedisponuje takovou vymožeností, jako je představivost, a proto všechno opět řeší matematika. Každý geometrický prvek (bod, přímka, kružnice atd.) má své matematické vyjádření. Každá činnost svou rovnici nebo algoritmus. Jelikož výsledný obraz je složen s pixelů – tedy diskrétních bodů v pravidelné síti – a křivka jakýsi průběh spojité funkce, musí se pro zobrazení výsledku projít několika procesy zjednodušení a převedením do diskrétního prostoru.

K popisu polohy prvku jak v paměti, tak i na obrazovce, je z mnoha důvodů nejvhodnější euklidovský prostor buď druhé nebo třetí dimenze a jeho úprava na kartézskou soustavu souřadnic. Obrazovka je plocha a proto ji popisuje euklidovský prostor druhé dimenze. Pro přesné umístění nejjednoduššího prvku – bodu – na obrazovce stačí dvě hodnoty (pro každou osu či dimenzi jedna). Osy je zvykem nazývat písmeny x a y (resp. z).



Obrázek 1. Bod v dvourozměrné



Obrázek 2. Bod v trojrozměrné

kartézské soustavě (2. dimenze – 2D). *kartézské soustavě (3. dimenze - 3D).*

Jak bude popsáno níže, výsledkem funkce křivky je soubor bodů na křivce. Máme-li body $x_0, x_1, ..., x_n$, pak spojením bodů x_i a x_{i+1}, kde $i \in \langle 0, n-1 \rangle$ dostaneme přibližný obraz křivky složený z úseček. Přesnost zobrazení závisí na parametru t funkce křivky. Rasterizaci úsečky má obsaženu každá základní knihovna pro práci s počítačovou grafikou. Většinou využívají v současné době nejefektivnější Bresenhamův algoritmus pro vykreslení úsečky.

2.1. Bod, úsečka, přímka

Bod, jakožto nejjednodušší prvek nemá žádnou rovnici. Ten v prostoru prostě je a nic víc na něm není. Pozice je určena jeho vzdáleností od počátku vůči každé ose (viz obrázek 1 resp. 2). Například $A = [2;5]$

Přímka lze vyjádřit několika způsoby. Vždy záleží na okolnostech, ale nejpoužívanější jsou parametrické vyjádření přímky a obecná rovnice přímky. K jednoznačnému určení polohy přímky v rovině musím znát dva vzájemně se nepřekrývající se body

Parametrické vyjádření má rovnici pro každou osu zvlášť.

$$p:$$
$$x = A_x + u_x t$$
$$y = A_y + u_y t$$

Kde A je bod na přímce, $\vec{u}$ je směrový vektor a t je parametr rovnice. Výhodou, ale i nevýhodou je, že každý rozměr má svou rovnici.

Další vyjádření přímky – tentokrát pouze jednou rovnicí – obecná rovnice přímky. Rovnice má tvar:

$$p: ax + by + c = 0$$

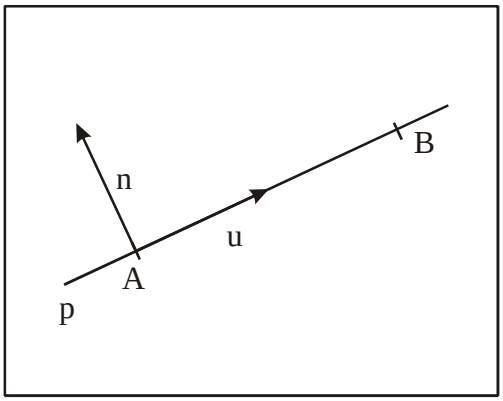
Kde a a b jsou složky normálového vektoru a c určuje polohu na ose y.

Obecnou rovnici ze dvou bodů získáme dosazením libovolného bodu do rovnice za proměnné x a y a dosazením normálového vektoru $\vec{n}$. Ten získáme ze směrového vektoru, určeným dvěma body určujícími přímku.

$$n_x = -u_y$$
$$n_y = u_x$$

Po dosazení nám zbývá jedna neznámá - c, což je poslední chybějící článek obecné rovnice.

$$c = -ax - by$$



Obrázek 3. Přímka v rovině.

Úsečka je jak známo přímka, ohraničená dvěma body. Proto množinu bodů určující přímku omezíme intervalem určeným těmito hraničními body.

2.2. Vzdálenost bodu od bodu a jeho výběr

Chceme-li v uživatelském prostředí programu vybrat bod, je nevhodné aby tomu tak bylo pouze po najetí na přesný pixel reprezentující tento bod. Je potřeba vytvořit kolem bodu aktivní prostředí, které se k danému bodu bude hlásit, jako by to byla samotná pozice onoho bodu. Pozice kurzoru (2. bod) se potom porovnává z aktivní oblastí. Výběr je úspěšný, je-li kurzor uvnitř této oblasti.

Vzdálenost dvou bodů se dá vyjádřit jako velikost přepony pravoúhlého trojúhelníku, jehož odvěsny jsou rovnoběžné s osami. Odečtením poloh těchto bodů získáme jejich vzdálenost na jednotlivých osách. Dále se pokračuje podle Pythagorovy věty.

Protože v Pythagorově větě je odmocnina, což je časově náročná funkce, můžeme se jí vyhnout tím, že tolerovanou vzdálenost umocníme na druhou.

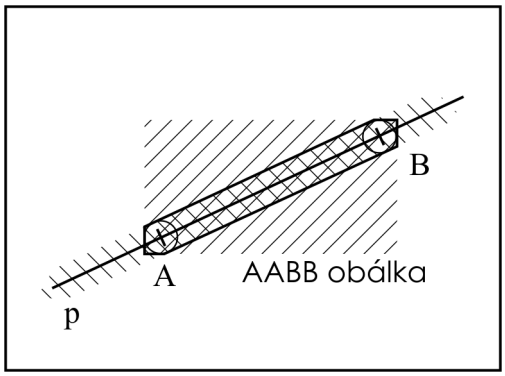
2.3. Vzdálenost bodu od přímky

Vzdálenost bodu od přímky získáme jeho dosazením do obecné rovnice přímky a výslednou hodnotu vydělíme velikostí normálového vektoru přímky.

$$d = \left| \frac{ax + by + c}{\sqrt{a^2 + b^2}} \right|$$

2.4. Vzdálenost bodu od úsečky a její výběr

Chceme-li vytvořit aktivní oblast pro úsečku, zkombinujeme tyto dva postupy dohromady. Body a jejich aktivní okolí vložíme do AABB obálky – to umožní prvotní a efektivní kontrolu. Pokud se kurzor nachází uvnitř obálky zkontrolujeme, zda jsme i uvnitř aktivní oblasti přímky. Pokud je i tato podmínka splněna, je kurzor v její blízkosti a může být vybrána.



Obrázek 4. Aktivní oblast úsečky je průnik aktivní oblasti přímky a AABB obálky aktivních oblastí bodů.

3. Křivky

[1]Křivky a plochy se používají v počítačové grafice a souvisejících aplikacích na mnoha různých místech. Setkáváme se s nimi modelovaní ve třech i ve dvou dimenzích, při definici fontů, při určování dráhy pohybujících se objektů v počítačové animaci, při definici objektů pro šablonování aj. Různé aplikace mají různé požadavky, a proto je tato oblast poměrně široká.

V roce 1959 používal P. de Casteljau u firmy Citroen matematický model křivek a ploch, jenž mu je umožňoval jednoduše zadávat. Podobně v šedesátých letech vedl P. Béziere vývoj programového systému UNISURF pro návrh křivek a ploch u u firmy Renaut. Metody tvarování křivek a ploch se postupem času zdokonalovaly a v současné době je k dispozici velmi silný nástroj který je stále aktivně rozvíjen. Podstatou tohoto rozvoje je kvalitní matematický aparát a v neposlední řade i zřetelný komerční efekt, který se projevil zejména v oblasti průmyslového designu.

Výrazný pokrok do této oblasti a především sjednocení dříve používaných různorodých přístupů přineslo používání racionálních B-spline křivek a ploch s neuniformní parametrizací, NURBS (Non–Uniform Rational B–Spline). Tyto metody umožňují generovat klasické geometrické prvky (úsečky, kružnice, elipsy a v prostoru koule, válce atp.) za pomoci stejných metod, které umožňují vytvořit křivky a plochy se složitými průběhy a tvary.

V počítačové grafice se pro vyjádření křivek nejčastěji používá tvar parametrický. Křivku budeme chápat fyzikálně, jako dráhu pohybujícího se bodu, jehož souřadnice jsou funkcemi parametru t (času).

$$x = x(t), y = y(t), z = z(t)$$

Parametr t je z intervalu $t \in \langle t_{\min}, t_{\max} \rangle$ a nejčastěji je volen v rozsahu $t \in \langle 0,1 \rangle$. Funkcemi je určena bodová rovnice křivky:

$$Q(t) = [x(t), y(t), z(t)]$$

Existují dva základní způsoby interpretace řídících bodů a to interpolace a aproximace. Křivka generovaná při interpolaci probíhá danými body, zatímco při aproximaci je řídícími body tvar křivky určen, ta jimi však procházet nemusí.

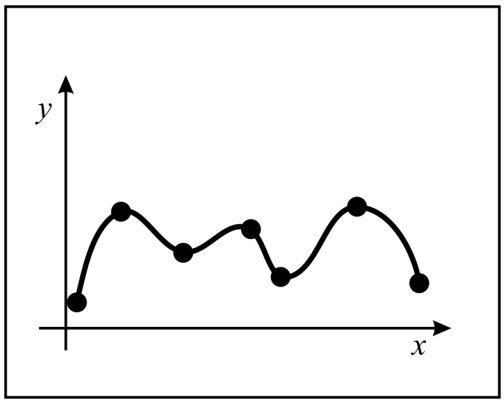
Další důležitou vlastností křivek je jejich racionalita (racionální/neracionální) a uniformnost (uniformní/neuniformní).

Racionální křivky můžou dát řídícímu bodu váhu, která určí velikost jeho vlivu na tvar křivky. Váhá se zadává v intervalu $w \in \langle 0, \infty \rangle$. Je-li váha bodu nulová, bod nemá na tvar křivky žádný vliv. S rostoucí vahou se křivka blíži řídícímu řídícímu bodu. Teoreticky při váze $w = \infty$ křivka bodem vždy prochází.

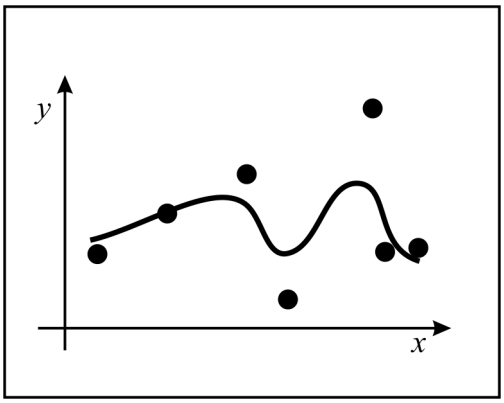
Uniformnost určuje, polohu řídicích bodů ve vztahu k času (prametru t). Řídicí body uniformních křivek mají na časové ose mezi sebou konstantní vzdálenost.

[1] Žára, J., aj.: *Moderní počítačová grafika*. 1. vydání, Brno: Computer Press®, 2004.

Neuniformní křivky mají body rozmístěny v závislosti na tzv. uzlovém vektoru, který je součástí jejich definice. Složky uzlového vektoru mají neklesající posloupnost, hodnota první složky vektoru je rovna $t_{\min}$ a poslední složka $t_{\max}$. Uniformní křivky uzlový vektor nepotřebuji explicitně zadávat, protože jej lze jednoduše generovat.



Obrázek 5. Interpolační křivka prochází všemi řídícími body.



Obrázek 6. Řídicí body aproximační křivky určují její zakřivení.

Jak již bylo řečeno, je mnoho typů křivek a jejich použití se liší na základě řešené problematiky a prostředí. Dále jsou uvedeny nejpoužívanější typy.

3.1. Lagrangeova interpolace

Chceme-li aproximovat funkci, která je dána svými hodnotami v $n+1$ bodech $x_i, i \in \langle 0, n \rangle$ a požadujeme-li, aby aproximace procházela zadanými body, použijeme aproximaci interpolačním polynomem. Aproximace nám potom poslouží k získání přibližné hodnoty zadané funkce v libovolném bodě intervalu $\langle x_0, x_n \rangle$.

Máme-li zadány hodnoty funkce f v $n+1$ různých bodech, tzn. máme zadáno $n+1$ interpolačních podmínek pro polynom, je zřejmé, že stupeň hledaného polynomu bude n. Lze ukázat, že mezi všemi polynomy nejvýše n-tého stupně existuje právě jeden, který je interpolačním polynomem pro zadanou funkci. Pro určení interpolačního polynomu existuje několik postupů, ale je třeba si uvědomit, že pro zadanou funkci všechny postupy určí stejný polynom.

Langrangeův [lagrándžův] interpolační polygon je jedním ze známějších a také snadných způsobů interpolace funkce zadané pouze v diskrétních bodech.

Nechť tedy máme dáno $n+1$ bodů, přes které funkce prochází. Pak můžeme pomocí níže popsané rovnice nalézt interpolační funkci, která se původní rovnici snaží co nejvíce přiblížit.

$$Q(t) = \sum_{i=0}^{n} P_i L_i^n(t)$$

$$L_i^n(t) = \prod_{j=0, j\neq i}^{n} \frac{u - u_j}{u_i - u_j}$$

Nevýhodou interpolačních polynomů je, že neplatí čím více bodů máme zadáno, tím přesnější funkci dostáváme. Pro jakoukoliv volbu uzlů interpolace vždy existuje spojitá funkce, pro kterou interpolační proces nekonverguje stejnoměrně. Navíc se při větším počtu bodů interpolační funkce na koncích více „vlní“, což v řadě případů nemusí být pro interpolaci zrovna nejlepší.

3.2. Hermitovská kubika

Mezi často používané křivky patří Hermitovské kubiky, někdy také označované jako Fergusonovy kubiky. Tyto křivky jsou určeny dvěma řídícími body P_0, P_1 a dvěma tečnými vektory $\vec{p}'_0$ a $\vec{p}'_1$ v nich. Body P_0 a P_1 určují polohu křivky. Křivka v nich začíná a končí. Směr a velikost tečných vektorů pak určuje míru jejího vyklenutí. Čím je velikost vektoru větší, tím více se k němu křivka přimyká. Jsou-li oba vektory nulové, pak se křivka stane úsečkou P_0P_1.

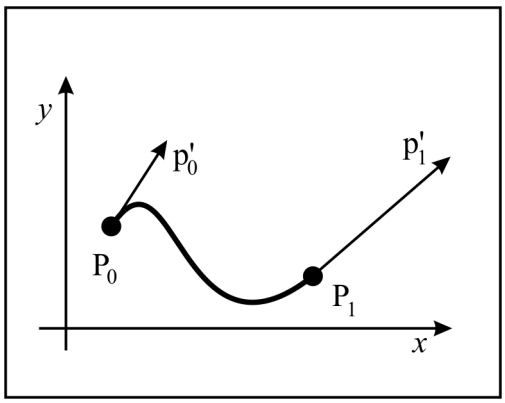
Bodová rovnice Hermitovské kubiky vypadá následovně:

$$Q(t) = P_0F_1(t) + P_1F_2(t) + \vec{p}'_0\,F_3(t) + \vec{p}'_1\,F_4(t)$$

kde F_1, F_2, F_3, F_4 jsou tzv. kubické Hermitovské polynomy ve tvaru:

$$F_1(t) = 2t^3 - 3t^2 + 1$$
$$F_2(t) = -2t^3 + 3t^2$$
$$F_3(t) = t^3 - 2t^2 + t$$
$$F_4(t) = t^3 - t^2$$

Největší přednost Hermitovských kubik se projeví při jejich navazování. Vzhledem k tomu, že součástí definice těchto křivek jsou tečné vektory v jejich koncových bodech, můžeme jejich hladké navazování realizovat velice snadno.



Obrázek 7. Hermitovská kubika.

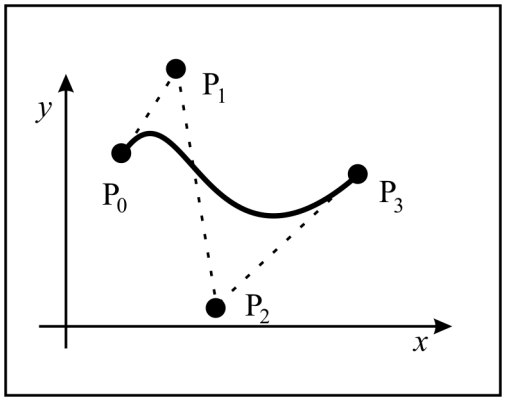
3.3. Bézierova kubika

Patrně nejpopulárnější aproximační křivky používané pro modelování jsou Bézierovy křivky. Tyto křivky se používají i při definici fontů.

Obecná Bézierova křivka je n-tého stupně. Nejpoužívanější je však křivka třetího stupně tedy kubika. Bézierovy kubiky jsou učeny čtyřmi body P_i řídícího polygonu a vztahem:

$$Q(t) = \sum_{i=0}^{3} P_i B_i(t)$$
$$B_0(t) = (1-t)^3$$
$$B_1(t) = 3t(1-t)^2$$
$$B_2(t) = 3t^2(1-t)$$
$$B_3(t) = t^3$$

$B_i(t)$ jsou Bernsteinovy polynomy třetího stupně. Křivka prochází prvním a posledním (čtvrtým) bodem.



Obrázek 8. Bézierova kubika.

Zobrazit Bézierovu křivku můžeme několika způsoby. Za zmínku stojí dva. Jeden způsob spočívá v prostém dosazování parametru t a spojením výsledných bodů.

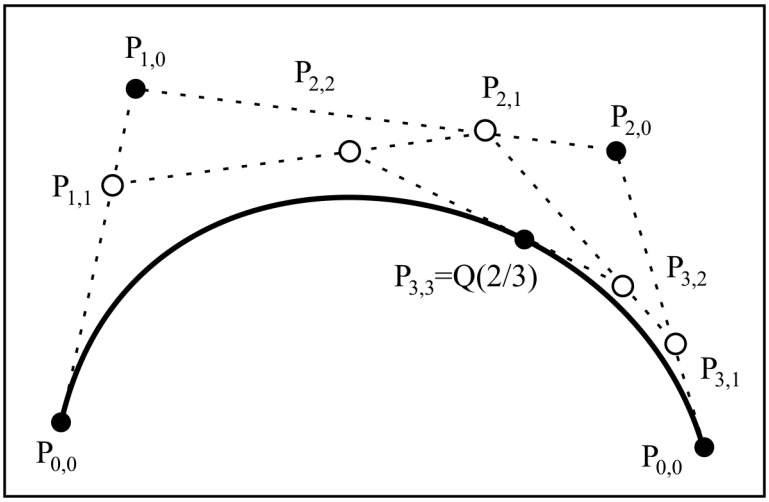
Druhým postupem jak vypočítat Bézierovu křivku je algoritmus de Casteljau. Bod křivky o souřadnicích $Q(t)$ se vypočítá podle rekurentního vztahu

$$P_{j,i}(t) = (1-t)P_{j-1,i-1} + tP_{j,i-1}$$

kde $i = 1,2,...,n$ a $j = i, i+1,...,n$.

Vstupem tohoto algoritmu jsou body řídicího polygonu. Dosadí se za $P_{i,0} = P_i$ a rekurentní vztah postupně vypočítává body $P_{j,i}$.V posledním kroku výpočtu je

koeficient $P_{n,n}(t)$ roven hodnotě křivky v bodě $Q(t)$. Existuje i zobecnění Bézierovy křivky přidávající možnost váhových koeficientů řídicích bodů. Křivka se potom nazývá racionální Beziérova.



Obrázek 9. Postup výpočtu bodu Q(t), pro t=2/3; t∈ <0,1> algoritmem de Casteljau.

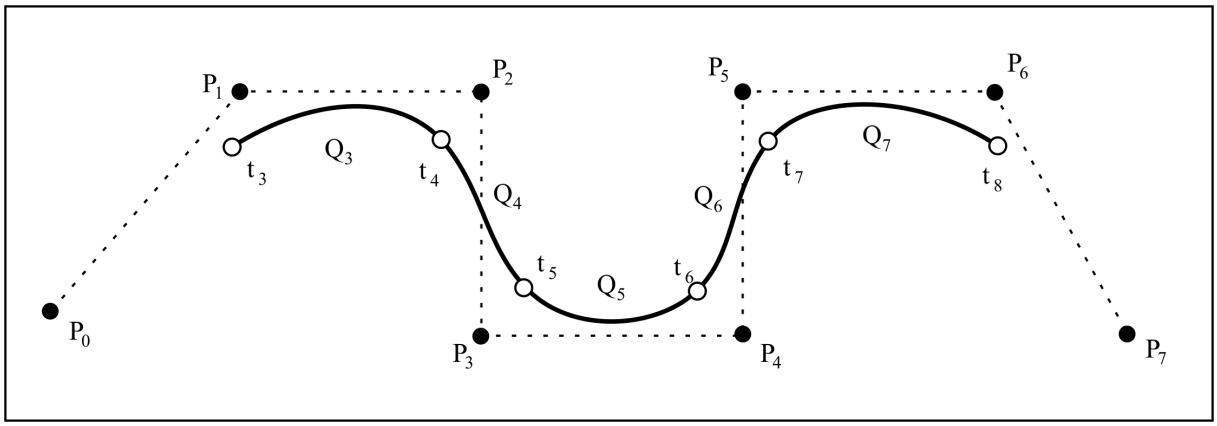
3.4. B-Spline a NURBS

Coonsova kubika zujímá vedle Bézierových křivek významné postavení v historii parametrických křivek a ploch. Je běžně používaná při tvorbě po částech skládaných polynomiálních křivek. Úvahy o uniformní a neuniformní parametrizaci výrazně přispěli k zobecněnému pohledu na vlastnosti a tvorbu polynomiálních bází a v neposlední řadě se stala základem obecného aparátu NURBS.

Coonsova kubika je také nazývána uniformní neracionální B-spline, zkráceně B–spline.

Uniformní kubický B-spline (Coonsův kubický B-spline) vznikne navázáním následujícím navázáním Coonsovych kubik. Segment Q_i je určen body P_{i-3}, P_{i-2}, P_{i-1} a P_i. Následující segment Q_{i+1} je určen body P_{i-2}, P_{i-1}, P_i a P_{i+1}, tedy třemi posledními body segmentu Q_i a jedním bodem segmentu následujícího. Zatímco Coonsova kubika je určena práívě čtyřmi body, Coonsův kubický B-spline je určen minimálně čtyřmi body. Coonsova kubika je sama o sobe B-spline, složený s jednoho segmentu.

B-spline určený $n+1$ body řídicího polygonu $P_0, P_1, ..., P_n$, je složen z $n-2$ segmentů $Q_3, Q_4, ..., Q_n$. Parametr t prochází interval $\langle t_3, t_{n+1} \rangle$ a hodnoty parametru t v uzlech (označujeme je t_i) definují uzlový vektor. Tyto hodnoty mají konstantní vzdálenost a uzlový vektor je tedy uniformní. Interval $\langle t_i, t_{i+1} \rangle$, který určuje bod B-spline v rámci jednoho segmentu, nazýváme lokální parametrizací. Interval $\langle t_3, t_{n+1} \rangle$, který pokrývá celý průběh po částech složeného B-spline, nazýváme globální parametrizací.



Obrázek 10. B-spline složený z Coonsovych kubik.

Neuniformní racionální B-spline křivky (NURBS) je zobecnění B-spline křivek. Neuniformnost umožňuje nepravidelné rozložení řídících bodů na časové ose určené uzlovým vektorem a racionalita definovatelnou váhu bodu.

Křivky NURBS jsou v dnešní době hojně používaný model jak pro 2D křivky, tak pro 3D plochy. Jejich obrovská výhoda spočívá v mnoha je přednostech. Za prvé umožňují zvyšování počtu řídících bodů, namísto zvyšování stupně křivky. Toto vychází z modelu B-spline. Přidáním nového bodu je do křivky přidán nový segment. Za druhé jsou invariační vůči různým transformacím jako je otáčení, perspektiva atd. a vůči rovnoběžnému a středovému promítání. S jejich pomocí se dají vytvořit základní geometrické obrazce a pomocí vah přesně vyjadřují kuželosečky. Její segmenty leží v konvexní obálce řídicích bodů ovlivňujících tento segment a celá křivka pak leží v konvexní obálce všech řídicích bodů.

NURBS křivka je určena $n+1$ řídicími body $P_i, i = 0,1,...,n$, tvořícími řídící polygon, řádem B-spline k a uzlovým vektorem. Uzlový vektor má tento tvar:

$$U = \{\underbrace{\alpha,...,\alpha}_{k},\underbrace{t_k,...,t_{n-k}}_{n+1-k},\underbrace{\beta,...,\beta}_{k}\}$$

a nazývá se okrajový uzlový vektor. Potom $\alpha = t_{\min}$ a $\beta = t_{\max}$ má za následek, že křivka začíná v prvním řídicím bodě a končí v posledním řídicím bodě, což činí editací křivky B-spline intuitivnější. Křivka je dále určena touto bodovou rovnicí:

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

kde w_i je váha i-tého bodu řídicího polygonu a $N_{i,k}(t)$ je normalizovana B-spline bázové funkce definované rekurentním vztahem

$$N_{i,k}(t) = \begin{cases} 1 \text{ pro } t_i \le t < t_{i+1} \\ 0 \text{ jinde} \end{cases}$$

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t) \text{ pro } t_i < t_{i+1+k}, 0 \le i \le n$$

Bodovou funkci lze také založit na racionální B-spline bázi. Potom lze bodová funkce zapsat jednodušeji:

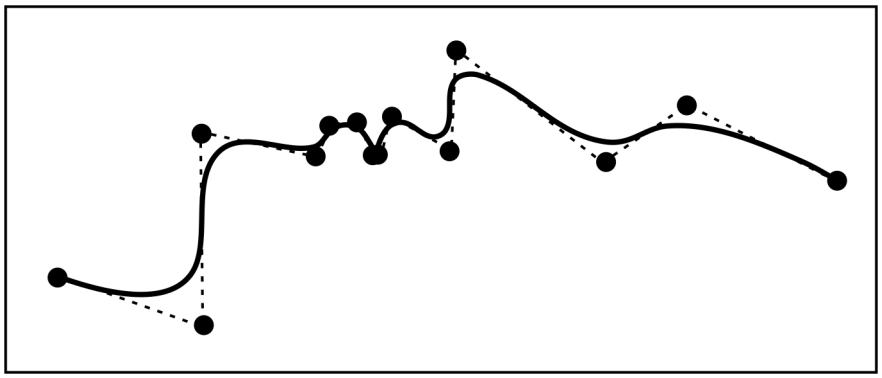
$$R_{i,k}(t) = \frac{w_i N_{i,k}(t)}{\sum_{j=0}^{n} w_i N_{j,k}(t)}$$

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

Další zajímavostí NURBS je,možnost chovat se jako Bézierova křivka. Je-li uzlový vektor

$$U = \{\underbrace{0,...,0}_{k},\underbrace{1,...,1}_{k}\}$$

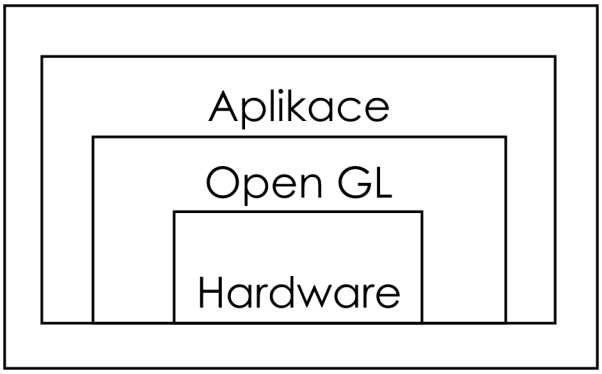
jsou normalizované B-spline polynomy rovny Bernsteinovým polynomům stupně k, a křivka se tedy rovná racionální Bézierově křivce. Pokud jsou všechny váhy $w_i = 1$ a $k = 4$, dostáváme Bézierovu kubiku. Jak již bylo řečeno, Bézierovy kubiky jsou v současnosti nejpoužívanější typ křivek. Tato vlastnost tudíž činí přechod na NURBS velice jednoduchým.



Obrázek 11. NURBS druhého stupně. Jak je vidět, v místě nahuštění řídicích bodů se výsledná křivka ztrácí v řídicím polygonu.

4. OpenGL

[2]Open Graphic Library (OpenGL) je knihovna pro práci s trojrozměrnou grafikou. Jedná se o API obsahující množinu funkcí a struktur. Dnes je brána uživateli za standard obsažený v každém počítači snad i proto, že je platformě nezávislá. Její implementace je na nejnižší, hardwarové úrovni, a to v grafických kartách. Aplikace využívající této knihovny lze však používat i bez této podpory. Tehdy jsou všechny výpočty prováděny softwarově.



Obrázek 12. OpenGL je obsažen přímo na hardwaru a aplikace k němu přistupuje výhradně přes OpenGL.

Knihovna vznikla v roce 1992, když firma Silicon Graphics Inc. (SGI) prováděla výzkumy v oblasti grafického modelování a realistické syntézy obrazu. Nejprve byla vyvinuta pro platformu IRIX, tedy pro počítače firmy SGI, pod názvem IRIX GL. Dnes je OpenGL široce využívaným standardem v oblasti inženýrských systémů, grafických symulací, virtuální reality atd. Dnes je nejrozšířenější verze 1.2, ale do moderních grafických karet je již implementovaná verze 2.0, samozřejmě zpětně kompatibilní s původní verzí. Novinkou je skriptovací jazyk, tzv. Shading language, který popisuje proces zobrazení scény explicitně a dává tak programátorům a grafikům téměř neomezené možnosti.

Open v názvu značí otevřený standard, což znamená, že za novými verzemi nestojí stále jeden autor. Licenční podmínky OpenGL lze shrnout do několika bodů. Nejvýznamnější důsledek je přitom to, že vývojář, který ve svém programu využije služeb knihovny OpenGL, nemusí platit žádné poplatky. Oproti tomu autor určité implementace OpenGL podléhá jedné ze tří úrovní licenčního programu. Level 1 License se týká prodejců implementací OpenGL v binární podobě, a poplatky s ní spojené jsou nízké. Level 2 License je určena pro různé knihovní nadstavby konkrétních implementací (GLU, GLUT). Level 3 License řeší distribuci zdrojových textů implementací.

Koncept zobrazování OpenGL s sebou nese i jistá omezení. Například je obtížné základními prostředky pracovat s objektem, který je oblý. Parametricky popsané plochy (Bézierovy plochy, NURBS) se nejprve musí převést na síť mnohoúhelníků, a

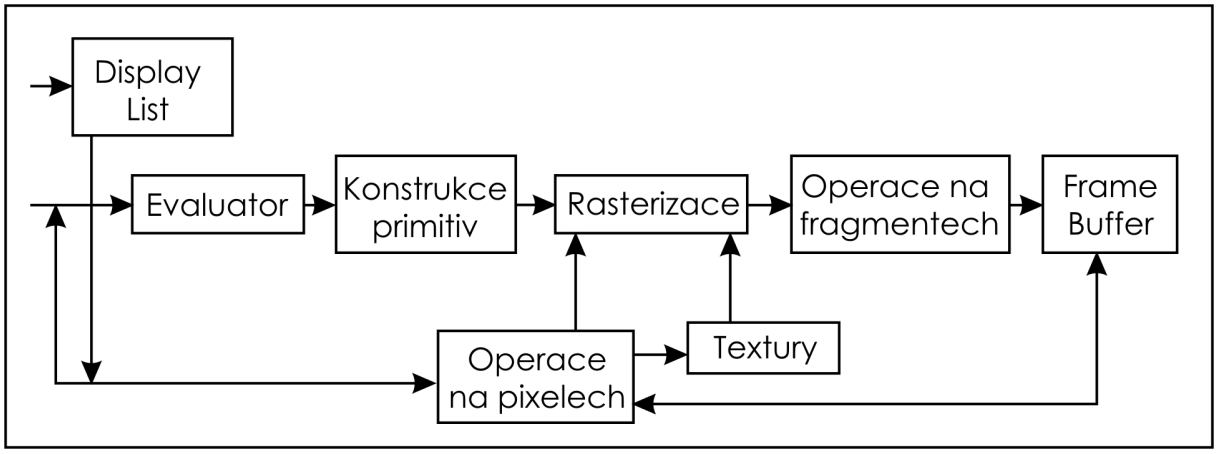
[2] Čech, D.: *OpenGL referát na praktikum z informatiky*.

teprve potom se mohou zobrazovat. To, že se jednotlivé komponenty obrazu zpracovávají odděleně, má vliv na výslednou kvalitu obrazu.

Aby si knihovna zachovala svou rychlost a multiplatformnost, obsahuje pouze ty nejzákladnější funkce pro práci s grafikou. Je však dodávána s přídavnými knihovnami, doplňujícími tento nedostatek. Jsou to GLU s pomocnými funkcemi, např. nastavení perspektivy nebo generování 3D primitivů, GLUT pro práci s okny a GLAUX.

4.1. Jak OpenGL pracuje

Na následujícím obrázku je zběžně popsán postup vykonávání příkazů.



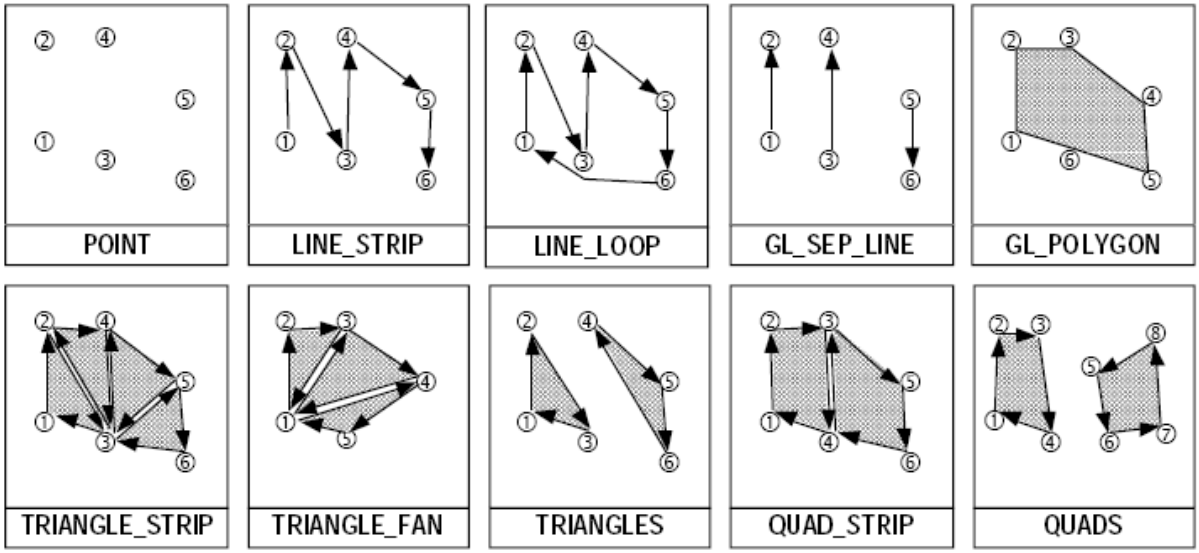
Obrázek 13. Schéma zpracování příkazů

V první, výpočetní fázi, která je zajišťována podpůrnou knihovnou GLU se složitější grafické objekty jako křivky a polynomiální plochy převádí na jednodušší objekty, s kterými už umí OpenGL přímo zacházet. Druhá fáze zahrnuje zpracovávání bodů, úseček a polygonů, výpočet normálových vektorů a osvětlení. Provádí se také ořezávání na rozměry projekčního okna. Třetí fáze - rasterizace znamená vlastní projekci trojrozměrného obrazu na plochu. Rasterizér vytváří sérii adres do framebufferu s hodnotami, při použití dvojrozměrného popisu čar, úseček a polygonů. Nyní se každému bodu přiřazuje konkrétní barva v závislosti na definované barvě objektů, světelných zdrojích, normálovému vektoru plochy a použitém světelném modelu. Pokud má povrch objektu určenou texturu, ve fázi rasterizace se na objekt mapuje. Výsledkem práce rasterizéru není ještě hotový obraz, ale množina tzv. fragmentů. Fragmentu odpovídá jeden pixel výsledného obrázku, který nese navíc informace o barvě, hloubce, popřípadě obsahuje i souřadnice textury. Fragmenty lze dále zpracovávat a docílit tak různým efektů. Můžeme například chtít vykreslovat polygony s vyhlazenými hranami (antialiasing), ovlivňovat průhlednost nebo míchat barvy (blending). Patří sem celá množina efektů (depth cueing), které ovlivňují barvu bodu v závislosti na jeho vzdálenosti od pozorovatele. Tak lze jednoduše docílit mlhy a jiných atmosférických vlivů. Výsledný obraz se uchovává ve framebufferu, kde vlastně proudové zpracování končí. Framebuffer už je přímo pod kontrolou operačního systému. Výsledek se zobrazí v okně.

4.2. Vytváření základních oběktů

Základními stavebními prvky, s nimiž OpenGL pracuje a vytváří scénu jsou body, čáry a polygony. Jednotlivé elementární objekty se v OpenGL definují v sekci uzavřené mezi funkce glBegin a glEnd. Parametrem funkce glBegin je pak jaký primitiv definujeme nebo způsob, jakým jej definujeme.

Mezi těmito funcemi je už definován primitiv funkcemi glVertex, glTexCoord, glNormal, glMaterial, glCallList. Toto jsou nejpoužívanější funkce pro definici tvaru, barvy a jiných vlastností. Tyto další informace se vážou k jednotlivým vrcholům tělesa a ne k tělesu samotnému. Sekce glBegin/glEnd by neměla mimo tyto funkce obsahovat nic jiného.



Obrázek 14. Různé parametry funkce glBegin a jejich vliv na tvorbu primitivu

Pokud zavolám glVertex s dvěma souřadnicemi, definuji přímo akorát x a y, souřadnice z se implicitně nastaví na nulu. Pro tři parametry jsou jasně dány všechny souřadnice. Zvláštní případ nastává tehdy, když zadám souřadnice bodu v 3D čtyřmi složkami. V tom případě znamenají jednotlivé složky x, y, z, w takzvané homogenní souřadnice. Homogenní souřadnice jsou výhodnou reprezentací, protože zjednodušují výpočty spojené s různými lineárními transformacemi scény (otočení, posunutí). Bod v n-rozměrném prostoru je zde popsán $n+1$ souřadnicemi. Souřadný systém se nazývá homogenní (stejnorodý), protože λ-násobek $(\lambda \neq 0)$ vektoru $(x, y.z, w)$ udává stále stejný bod ve 3D. Mezi kartézskými a homogenními souřadnicemi platí jednoduchá převodní pravidla:

$$(x, y, z) \Rightarrow (x, y, z, 1)$$

$$(x, y, z, w) \Rightarrow (\frac{x}{w}, \frac{y}{w}, \frac{z}{w})$$

Základní výhoda, která plyne z použití homogenních souřadnic je, že umožňuje realizovat i posunutí vynásobením vektoru maticí. Třebaže vrchol specifikujeme jiným počtem souřadnic, OpenGL si ho převede právě do homogenních, které používá jako vnitřní reprezentaci.

4.3. Stavy

Řízením stavů se přímo ovlivňuje, které funkce OpenGL mají být v dané chvíli aktivní. Ze začátku je většina vlastností vypnuta a programátor je musí na začátku programu inicializovat, pokud je bude využívat.

Týká se to například používání textur, odstraňování odvrácených stěn objektů nebo nastavení mlhy. Operace se stavy, které mohou nabývat pouze dvou hodnot, se provádí pomocí dvojice funkcí glEnable, která aktivuje vlastnost a glDisable, která deaktivuje vlastnost

OpenGL totiž už od verze 1.1 dává možnost vyhodnocování některých stavů už na straně klienta. Stavy obecně jsou skrytými systémovými proměnnými, které mohou mít různý význam. Stavovými proměnnými jsou i zásobníky framebufferu. Na zjišťování hodnot vnitřních proměnných OpenGL je určena funkce glGet. Těmito funkcemi se můžeme odvolávat na obsah stavových proměnných jednotlivých typů.

Data, jako barva, hloubka a jiná jsou ukládána do zásobníku. Někdy je potřeba tento zásobník vyčistit a připravit tak na nová data. Funkce glClear provádí mazání zásobníků. Parametrem funkce je identifikátor zásobníku Přípustné jsou hodnoty: GL_COLOR_BUFFER_BIT, GL_DEPTH_BUFFER_BIT, GL_ACCUM_BUFFER_BIT, GL_STENCIL_BUFFER_BIT. Mazání zásobníku je poměrně časově náročná operace, ale nové grafické karty umožňují mazání několika zásobníků najednou, proto například volání glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT) může být mnohdy výrazně rychlejší než volání glClear jednotlivě.

4.4. Transformace

Modelovací transformace se používají, když chceme explicitně umístit definovaný objekt v prostoru nebo měnit jeho natočení. Obecněji můžeme souřadnice vrcholů objektu všemožně přetvořit násobením vektorů souřadnic vrcholů maticí. OpenGL pracuje s aktuální maticí, což je stavová proměnná. Na ní se realizují veškeré výpočty tohoto druhu. Transformační příkazy vynásobí aktuální matici danou maticí a výsledek znovu uloží do aktuální matice. Mezi jednotlivými transformacemi je třeba nastavit aktuální matici na jednotkovou funkcí glLoadIdentity, aby nebyly zobrazení vzájemně ovlivněny. Postup aplikací geometrických transformací je opačný vůči klasickému postupu v běžném smyslu. Nejprve se provede transformace a teprve potom se teprve objekt vykreslí.

Zobrazovací transformace jsou analogií k umísťování a rotaci kamery, kterou získáváme obraz scény. Pozici kamery v prostoru určují její souřadnice a navíc pomocný vektor up-vector, který udává lokálně směr nahoru. Implicitně je up-vector nastaven na (0,1,0) tedy ve směru osy y. Zobrazovací transformace jsou v přirozeném duálním vztahu k modelovacím transformacím. Pokud je scénu tvoří jediný objekt, potom posunutí objektu dozadu je ekvivalentní s posunutím kamery v opačném směru. V souvislosti se zobrazovacími a modelovacími transformacemi je výhodné hovořit o lokálním a globálním souřadném systému.

Konečně projekční transformace se již týkají promítání scény do okna. Nastavování vlastností projekční transformace má svou analogii ve vybírání objektivu ke kameře. To jak se bude scéna mapovat do roviny určuje matice projekčního zobrazení. OpenGL nabízí dva vestavěné nejdůležitější druhy projekce – kolmou a perspektivní projekci. Při kolmé projekci nedochází ke zkreslení délek a rovnoběžky zůstávají i po aplikaci transformace rovnoběžné. Toto zobrazení však neodpovídá fyzikálnímu modelu. Jako transformační matice je použita jednotková matice. Kolmá (ortogonální) projekce nalézá uplatnění v různých technických aplikacích a CAD modelerech, kde je důležitější přesné zobrazení proporcí než realistický pohled. Oproti tomu perspektivní projekce dává věrné zobrazení. Nastavení parametrů perspektivy se provádí voláním funkce gluPerspective z GLU. Obecné projekční zobrazení se definuje pomocí funkce glFrustum, závisí ale na mnoha parametrech a její použití není zrovna intuitivní.

4.5. Barevný model

OpenGL používá pro výpočty reprezentaci barev v RGBA. RGB jsou klasické barevné složky červená – zelená – modrá. Písmeno A označuje tzv. alfa kanál, který navíc určuje průhlednost barvy.

Dnes už snad téměř všechny grafické karty pracují v módu TrueColor nebo HighColor, kde není omezení na počet současně zobrazených barev. Protože ne vždy tomu tak bylo a z důvodu zajištění kompatibility umí OpenGL pracovat i v color-index-mode, kde se používají barevné palety. Použití indexovaných barev s OpenGL znamená komplikovanější spolupráci s operačním systémem, který spravuje paletu. Je také pomalejší, protože se musí provádět další přepočty.

OpenGL podporuje dithering, což je metoda, kterou lze dosáhnout uspokojivých výsledků i při použití indexovaných barev. Problém malé barevné hloubky se tak převede na menší prostorové rozlišení. Míchání barev se potom dociluje soutiskem různě barevných pixelů vedle sebe, což dodává na první pohled dojem nových odstínů. Dnes je používání indexovaných barev spíše historickou záležitostí.

Pro definování aktuální barvy je v OpenGL funkce glColor.

4.6. GLUT a OpenGlut

OpenGL Utility Toolkit (GLUT) je programovací rozhraní pro psaní okenních OpenGL aplikací nezávislé na platformě. Zastává následující funkce:

- Více oken pro OpenGL renderování
- Callback správa událostí
- Obsluha vstupních zařízení
- Rutina „idle“ a časovače
- Jednoduché hierarchické pop-up nabídky
- Rutiny pro generování různých povrchových nebo mřížkových těles.
- Podpora bitmapových fontů
- Různé funkce obsluhující okna.

Jedním z prvků specifikace knihovny OpenGL bylo vytěsnění funkcí závislých na platformě mimo OpenGL model.

Před využíváním knihovny je potřeba volat funkci glutInit. První, k čemu GLUT využijeme je určitě vytvoření okna. To je vytvořeno funkcí glutCreateWindow. GLUT okno reaguje na tyto události:

- GlutCloseFunc – zavření okna
- GlutDisplayFunc – povinná callback funkce, vykonávající vykreslování OpenGL
- GlutReshapeFunc – změna velikosti okna
- a jiné

Veškeré vstupy jsou zasílány ke zpracování callback funkcím. Ukazatel na funkci zpracovávající událost se zadá jako parametr příslušné funkce GLUT. Ten potom volá přímo zadanou funkci. GLUT pracuje s těmito vstupními zařízeními: myš, klávesnice, joystick, spaceball a tablet. Knihovna OpenGLUT však mnoho z nich neimplementuje.

Největší pozornost je věnována myši. Pro tu GLUT rozlišuje 5 různých událostí:

- glutEntryFunc - vstup a vystup kurzoru z plochy okna
- glutMotionFunc - tažení myši, tedy pohyb při stisklém tlačítku
- glutMouseFunc - stisknutí nebo uvolnění tlačítka
- glutMouseWheelFunc - pohyb kolečka myši
- glutPassiveMotion - pohyb myši bez stisknutého tlačítka

U klávesnice jsou rozlišovány tři události:

- glutKeyboardFunc – napsaný ASCII znak
- glutKeyboardUpFunc – zachycuje uvolnění klávesy
- glutSpecialFunc – stištěná speciální klávesa (F1 – F12, šiky a klávesy nad nimi)

OpenGLUT je open source projekt pokračující ve vývoji knihovny GLUT pro C/C++. Vychází z podobného projektu freeglut, který používá jako základ, která dále rozvíjí. V současné verzi 0.6.3 implementuje pouze částečně původní GLUT.

5. Deformace NURBS křivky

Jednou z výhod NURBS křivek je možnost přidání bodu bez zvyšování řádu křivky. Výše v popisu těchto křivek bylo už zmíněno, že místo zvyšování řádu, přidá se do křivky nový segment. Důležité je však zachovat stejný matematický tvar křivky. Přidaný nový řídicí bod na pozici t ovlivní pozici okolních řídicích bodů.

5.1. Přidání uzlu do NURBS křivky

Algoritmus podle profesora Fugia Yamaguchiho přidává řídicí bod do křivky následovně.[3]

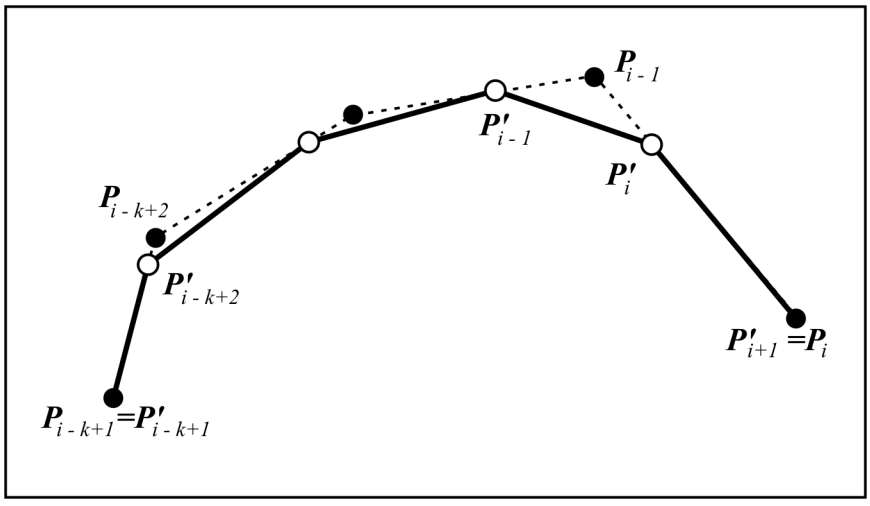
Mějme nový uzel t', který vkládáme mezi uzly t_i a t_{i+1} $(t_i < t' \leq t_{i+1})$. Nový uzel kromě nového řídícího bodu ovlivňuje polohu okolních řídících bodů. Jejich pozice je určena vztahem:

$$P'_j = (1-\alpha_j)P_{j-1} + \alpha_j P_j$$

kde

$$\alpha_j = \begin{cases} 1 & \text{pro } j \leq i\text{-}k+1 \\ \dfrac{t'-t_j}{t_{j+k-1}-t_j} & \text{pro } i-k+2 \leq j \leq i \\ 0 & \text{pro j} \geq i+1 \end{cases}$$

To znamená, že nový vektor řídicích bodů rozdělí segment $Q_{j-1}Q_j$ v poměru $\alpha_j : 1-\alpha_j$ .Následující obrázek ukazuje tuto změnu na řídicím polygonu.



Obrázek 15. Změna řídicího polygonu

[3] Yamaguchi, F.: *Curves and surfaces in computer aided geometric design*. 1st edition, Berlin; Heidelberg: Springer-Verlag, 1988.

5.2. Změna pozice přilehlých řídících bodů

Po přípravě polygonu můžeme přistoupit k vlastní deformaci. Před začátkem posouvání jednotlivých řídicích bodů je jeden z nich určen jako referenční P_i. Od tohoto bodu se bude měřit vzdálenost přilehlých řídicích bodů l_j. Postupuje se po hraně řídicího polygonu na obě strany. Doleva pro $j < i$ a doprava pro $j > i$. Každému bodu se na základě jeho vzdálenosti od reference určí váha w_j jednoduchým vztahem

$$w_j = \begin{cases} 1 - \dfrac{l_j}{l_{\max}} & \text{pro } 0 \le l_j < l_{\max} \\ 0 & \text{pro } l_j \ge l_{\max} \end{cases}$$

kde $l_{\max}$ je maximální vzdálenost mající vliv na přilehlý bod. Výslednou váhu ještě upravíme libovolnou funkcí na $w'_j = f(w_j)$, přičemž implicitně je $f(x) = x$.

Mějme počáteční polohu referenčního řídicího bodu P_i a novou polohu referenčního bodu P'_i. Jejich rozdíl je potom $d = P'_i - P_i$.

Pokud máme k dispozici vzdálenost d a váhy w'_j, vypočítáme novou polohu přilehlého řídicího bodu

$$P'_j = P_j + dw'_j .$$

6. Uživatelské prostředí

S narůstajícím množstvím funkcí a nástrojů v programech je stále větší důraz kladen na vývoj vhodného uživatelského prostředí. Pro intuitivnost a efektivnost uživatelského prostředí je důležité, aby bylo přehledné a nabízelo uživateli to, co zrovna v danou chvíli potřebuje.

Nejčastěji využívanou částí uživatelského rozhraní je pracovní plocha. Uživatel nejvíce ocení, má-li před sebou hodně místa pro svou práci a tím i přehled nad tím, co vytvořil. Není dobré spoléhat na vysoké rozlišení monitoru uživatele. Většina profesionálních aplikací umožňuje skrytí veškerých panelů a nabídek. Uživatel pak ovládá program klávesovými zkratkami o položkami v kontextové nabídce.

Dnes již celkem zažitou vlastností aplikací je nastavitelné prostředí, jak změnou polohy panelů nástrojů, tak i přizpůsobení si obsahu v nich, vytváření si vlastních klávesových zkratek atd. Velice užitečnou vlastností je i automatizace často prováděných činností. Uživatel si tuto činnost „nahraje“, přiřadí jí tlačítko v panelů nástrojů, klávesovou zkratku nebo místo v kontextové nabídce a pak je celá činnost otázkou kliknutí nebo zmáčknutí klávesy. V pozadí je tato automatizace realizována na základě skriptovacího jazyka. Ten se obvykle liší program od programu. Většinou se ale firmy snaží zachovat syntaxi některého z používaných jazyků a přiřadí mu balíčky tříd vycházející z možností programu. Například aplikace firmy Microsoft používají skriptovací jazyk Visual Basic for Applications, vycházející z programovacího jazyku Visual Bacic, 3Dstudio MAX požívá tzv. Maxscript, vycházející z C++.

6.1. Rozložení nástrojů

Pro uživatele, začínajícího v jakémkoliv program a navyklého na klikací prostředí aplikací je nejpříjemnější, když na první pohled vidí všechny nástroje – tedy možnosti zkoušené aplikace – a tyto nástroje navíc sdružené do skupin a podskupin. U větších programů je to však nemožné, protože by celou pracovní plochu zakryli pouze nástroje. Tento problém řeší právě vytváření skupin nástrojů a jejich skrývání či zobrazování buď na podnět uživatele (zobrazení/skrytí panelů nástrojů) nebo na základě právě prováděné činnosti (po výběru štětce se automaticky změní okno vlastností na vlastnosti štětce). Rychlost práce navíc velice zvýší, je-li právě potřebný nástroj co nejblíže kurzoru.

6.2. Klávesové zkratky

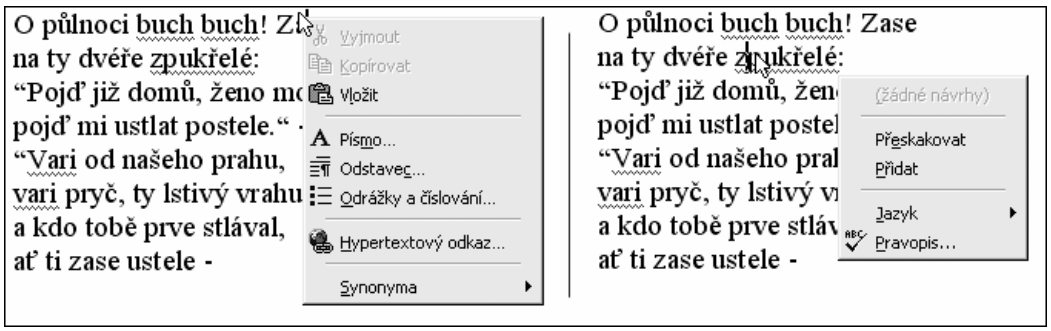
Prvkem, který velice urychluje práci v programu jsou klávesové zkratky. Klávesa navíc nevyužívá ukazovací zařízení (standardně myš) a proto může činnost být vyvolána i při tažení myší. Např. při složité deformaci můžeme snížit složitost modelu, aniž bychom přerušili deformování. Program by měl mít ke všem ovládacím prvkům přidruženou klávesovou zkratku. Jsme sice omezeni počtem kláves na klávesnici, ten se však velice rozrůstá v kombinaci s klávesami Ctrl, Alt a Shift.

Klávesová zkratka navíc nemusí být vždy unikátní. V případě že nemůže nastat kolize, je možné aby jedna klávesa vykonávala rozdílnou činnost. Například při nástroji Štětec může vyvolat jezdce pro nastavení jeho šířky a při nástroji Rydlo jezdce pro nastavení hloubky vrypu. Většinou by ale tyto skupiny měli mít alespoň společný asociační základ.

6.3. Kontextová nabídka

Největší výhoda kontextové nabídky spočívá v její pozici. Je vždy na místě kurzoru a vyvolává se pravým tlačítkem myši (není to pravidlo, ale silně zažitý standard).

Obsah kontextové nabídky by se měl přizpůsobovat nejen na základě vybraného nástroje, ale podle toho co se zrovna nachází pod kurzorem a jakou činnost uživatel zrovna provádí.



Obrázek 16. Změna kontextové nabídky v závislosti umístění kurzoru

6.4. Budoucnost uživatelských prostředí

Moderním způsobem úpravy pracovní plochy a ovládání programu je buď manuální upravení prostředí uživatelem, nebo automatická úprava některým ze samoučících se algoritmů.

Objevují se i myšlenky unifikace uživatelských prostředí, nebo jejich generování na základě vybraného stylu. Uživatel, zvyklý pracovat například v programu Paint Shop Pro si radši pro svůj nový program vybere již zvyklé prostředí a jediné co se učí navíc jsou nástroje programu, aniž by tápal v jeho prostředí. Jako ukázka této myšlenky můžou posloužit moderní skiny programu Winamp. Tyto postupy jsou testovány například v grafickém programu XSI. Ten přecházejícím uživatelům dává možnost výběru chování včetně nastavení klávesových zkratek společných nástrojů stejných jako v 3Dstudiu MAX, Maya nebo jiné.

Další rozšíření pracovní plochy přichází s novými technologiemi pro práci ve 3D. Ve smyslu 3D softwaru dostává pracovní plocha hloubku a tedy obrovské rozšíření prostoru, kdy panely nástrojů jsou sice za sebou, ale pro uživatele stále přehledné. Ve smyslu 3D hardwaru, pomineme-li zobrazovací techniku, to jsou ukazovací vstupní zařízení pracující primárně v prostoru.

Opravdovým výstřelkem jsou potom uživatelská prostředí ve formě virtuální reality, kde uživatel s programem pracuje stejně intuitivně jako kutil v dílně.

7. Implementace

Výše uvedenou matematickou teorii shrnuje ukázkový program napsaný v jazyce C++. Tento objektově orientovaný jazyk umožňuje rozdělení programu do tříd, čímž se zvyšuje přehlednost zdrojového kódu. Je však kompatibilní i se starým jazykem C, což dovoluje použít i dříve navyklé postupy. Knihovna OpenGL poskytuje výkonnou spolehlivou a platformě nezávislou základnu pro nejrozmanitější grafické operace. OpenGLUT jakož náhrada zastaralé nadstavby GLUT pro OpenGL vykonává operace pro multiplatformní práci s okny.

Vývoj probíhal ve vývojovém prostředí Visual Studio 2005 – projekty jsou ve formátu této verze.

Program demonstruje deformační nástoroj upravující křivku bez nutnosti vidět řídicí body. Nástroj je implementovaný do 2D prostředí a deformuje tedy NURBS křivku. Deformace podle výše uvedeného popisu nijak nevyužívá vah řídicích bodů. U každého bodu je tedy vždy $w_i = 1$ a tudíž v bodové rovnici nemá váha žádný vliv na tvar křivky. Proto byl výpočet optimalizován na rovnici

$$Q(t) = \frac{\sum_{i=0}^{n} P_i N_{i,k}(t)}{\sum_{i=0}^{n} N_{i,k}(t)}.$$

Nástroj deformuje NURBS křivku 3 stupně. Křivky jsou generovány prostým dosazením do rovnice a výpočtem hodnoty $Q(t)$ pro každou osu. Při každém překreslení okna se obnovuje celý jeho obsah a tedy znovu přepočítává křivka.

Detailněji jsou funkce programu popsány v příloženém manuálu.

7.1. Rozdělení programu

Program by měl být demonstrativní a proto je rozdělen jak do několika tříd, tak do tematicky odlišných souborů. Neplatí tedy pravidlo – co třída to soubor. Pro práci s myší je vytvořen speciální soubor, ačkoliv je součástí třídy cGUI.

Třída cGUI se stará o veškerou obsluhu programu. Uchovává, jak nastavení programu, tak metody pro obsluhu vstupů vyvolaných uživatelem a zachycených nadstavbou OpenGLUT

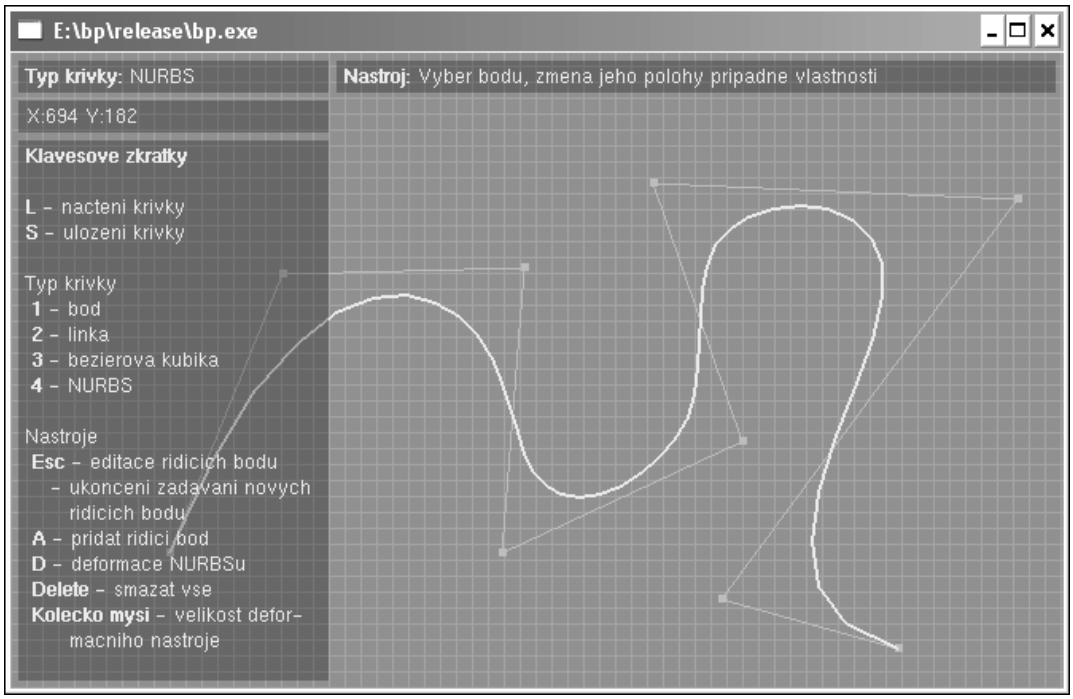
U nastavení programu je potřeba zadat i nějaké textové nebo číselné vstupní hodnoty. K tomu je vytvořena třída cDialog.

Matematický aparát křivek je u každého typu křivky obsažen ve zvláštní třídě. Bézierově kubice patří třída cBezier a NURBSům třída cNURBS. Obě jsou postaveny na stejném základu. Křivka je vypočítána metodou toLines a jako výsledek vrací pole bodů vypočítaných bodovou funkcí $Q(t)$.

7.2. Uživatelské prostředí

Uživatelské prostředí bylo navrženo tak, aby byla pracovní plocha, co největší. Ve výsledné podobě je plocha sloužící k editaci křivky rozprostřena po celém okně. I

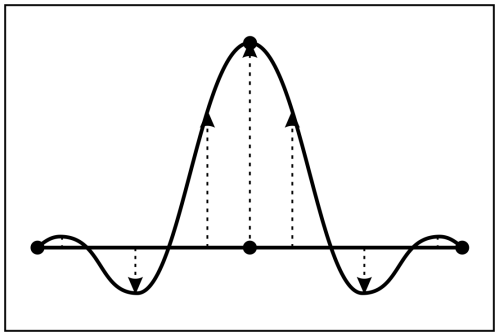
dialogové panely jsou částečně průhledné a blokují přehled pouze minimálně. Každá činnost, mimo nastavení programu, má přiřazenou klávesovou zkratku. Uživatel může věnovat neustále pozornost při práci s myší na úpravu křivky a není nucen nikam kurzorem přejíždět. Kdyby přece jen chtěl aplikaci ovládat pouze jednou rukou, má vše k dispozici i v kontextové nabídce. Vzhledem k jednoduchosti programu není řešena lokální změna obsahu kontextového menu. To má stále stejnou podobu.



Obrázek 17. Rozložení uživatelského prostředí v ukázkové aplikaci.

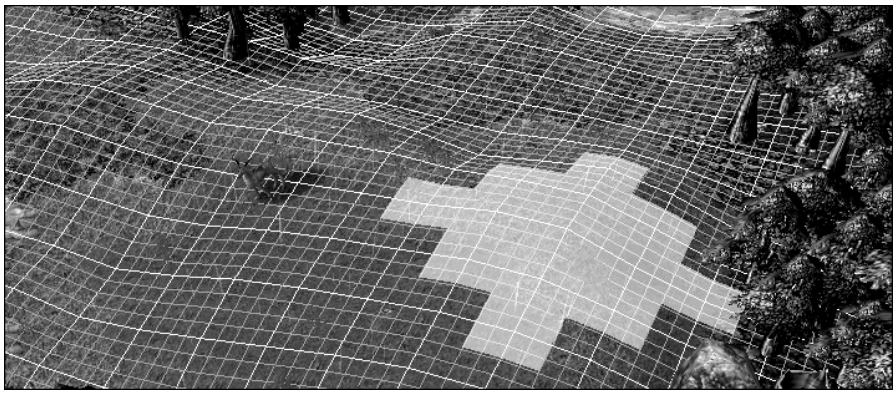
7.3. Jiné možnosti implementace

V průběhu vývoje bylo vyzkoušeno několik implementací nástroje, lišící se jak v počtu přidávaných bodů, tak samotném přístupu ke křivce. I změna polohy přilehlých bodů může být distribuována různými funkcemi, což vytváří vzhledově zajímavé výsledky. Použití dvou základních distribucí je popsáno níže.



Obrázek 18. Nelineární vliv vzdálenosti na okolí referenčního bodu při deformaci

Jiná aplikace této deformace využívá až výsledného pole bodů tvořící křivku. Při deformaci jsou posouvány jednotlivé diskrétní body křivky s případnou zpětnou reparametrizací křivky zpět do spojité formy. Tento postup lze použít například při modelování reliéfu krajiny vycházející z diskrétní mřížky.



Obrázek 19. Nástroj upravující reliéf krajiny deformující pouze diskrétní mřížku. Obrázek je z editoru ke hře Warcraft 3.

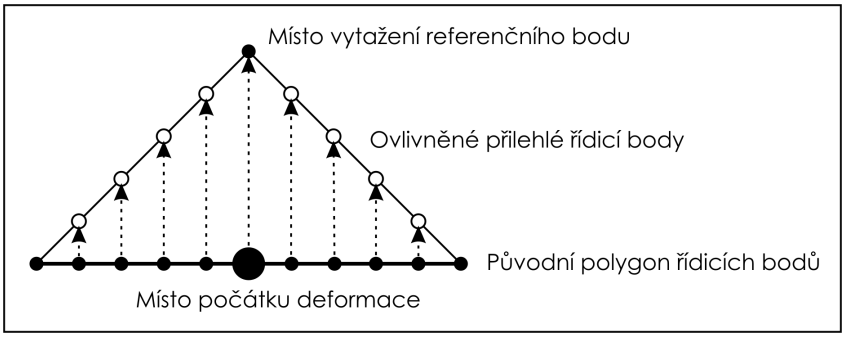
Nejlepší výsledky deformace by nástroj vracel v případě, kdy bude plocha deformace ohraničena okrajovými řídicími body, které by ukotvili kraje deformovaného segmentu. Okrajové body při tom interpolují křivku. Nevýhodou však je rapidní narůstání počtu řídicích bodů, proto je vhodné tento postup upravit na podmíněné vkládání bodu, kdy program nejprve zjistí, zda je potřeba bod vytvářet, případně křivku reparametrizovat a tím vynechat řídicí body s nízkým vlivem na křivost křivky.

7.4. Algoritmus deformačního nástroje

K začátku deformace křivky je potřeba, aby uživatel nastavil velikost deformačního nástoje (viz manuál k ukázkovému programu) a vybral místo na křivce. To je jednoznačně určeno parametrem t. Výkon deformace sestává ze dvou částí.

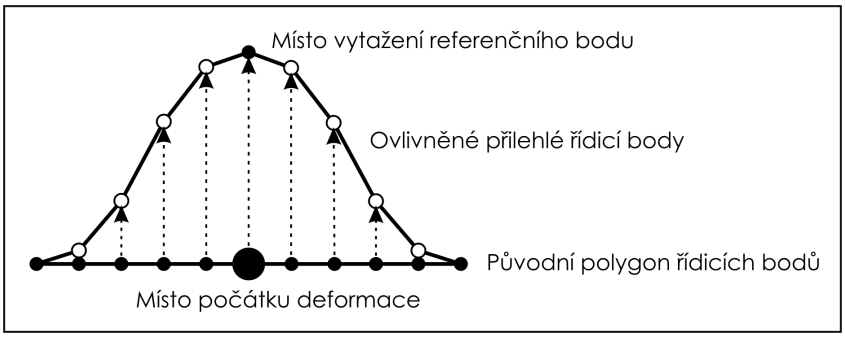
Za prvé je třeba rozhodnout, zda je potřeba přidat nový řídicí bod, nebo je část křivky v čase t dostatečně ovlivněna již existujícím řídícím bodem. Proto se vytvoří nový řídicí bod v určeném bodě křivky, následně se porovná s pozicí předchozích řídicích bodů mezi které byl vložen a pokud je v „dostatečné vzdálenosti“ původní řídicí bod, zruší se přidaný řídicí bod a k nalezený bod se určí jako referenční pro deformaci. V této implementaci byla nastavena „dostatečná vzdálenost“ pevně na polovinu šířky deformačního nástroje. Pokud se v „dostatečné vzdálenosti“ nepodaří nalézt původní řídicí bod, je nový přidán do křivky a vybrán jako referenční pro deformaci.

Druhá část deformace je posouvání referenčního řídícího bodu a poměrné posunutí přilehlých řídicích bodů v závislosti na šířce deformačního nástroje. Vztah přilehlých bodů k velikosti posunutí může být různý. Vždy ale závisí na vzdálenosti od referenčního bodu. Nejjednodušší způsob jak upravit pozici přilehlých bodů je lineární závislost na vzdálenosti od reference.



Obrázek 20. Lineární vztah k vzdálenosti k referenčnímu bodu

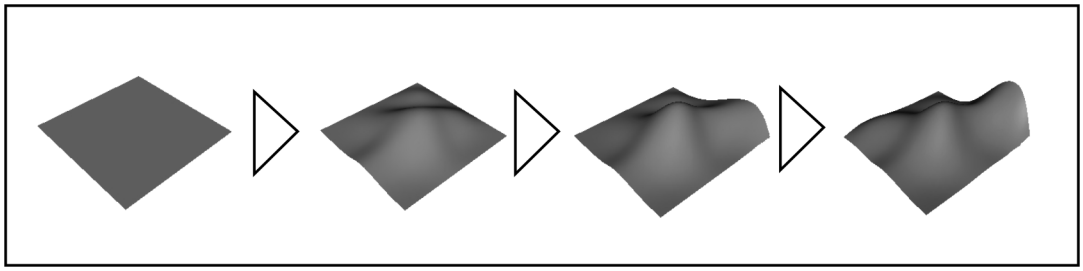
Výsledek však působí příliš hranatě. V mnoha problematikách byla jako funkce vracející vzhledově nepřirozenější výsledky vybrána Gaussova křivka nebo Gaussův oblouk (např. gaussovské rozmazání). I v tomto případě je nejpřirozenější ovlivnění pozic přilehlých bodů převézt z lineárních na Gaussův oblouk. Jelikož vztah generující tuto křivku je výpočetně složitý, byl oblouk nahrazen jednou periodou funkce cosinus. Výsledný tvar je velice podobný, avšak za použití jednoduchého výpočtu.



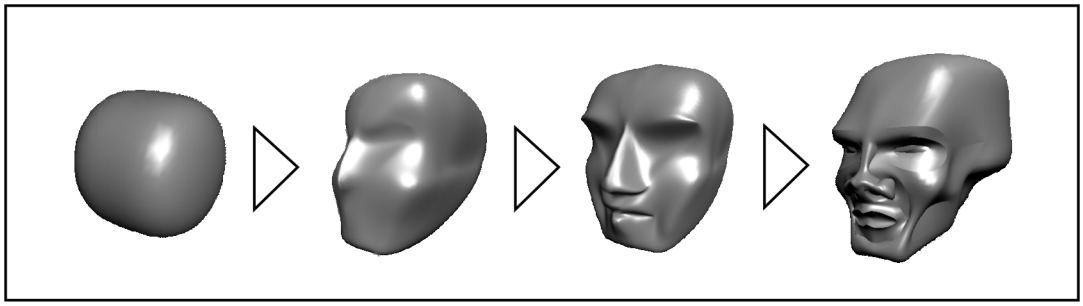
Obrázek 21. Posunutí přilehlých bodů podle průběhu funkce cosinus

8. Možnosti využití

Tento deformační nástroj je primárně určen k nepreciznímu modelování. Je vhodný pro tvorbu grafických vektorových obrazů, nepoužitelní však k vytváření přesných logotypů. V trojrozměrném světě pak k vytvoření modelu z NURBS ploch a následné využití pro animaci. Nikoliv však pro průmyslový design. Několik programů už s podobným nástrojem experimentovalo. Nejznámějším průkopníkem v oblasti NURBS ploch je program Rhinoceros (Rhino). Podobný nástroj se objevil i v programu XSI. Pro vytvoření si názoru na budoucnost tohoto nástroje určitě dostatečně poslouží následující obrázky.



Obrázek 22. Postupná deformace plochy na reliéf krajiny.



Obrázek 23. Deformace primitivu na model hlavy.(model poskytl Igor Bureš)

Závěr

Parametrické křivky a především parametrické plochy generované v reálném čase budou s narůstajícím výkonem počítačů stoupat v oblibě. To zapříčiní nárůst počtu uživatelů ze strany laiků, kteří budou chtít tyto technologie využívat.

Nejen profesionálové ocení výkonné a hlavně efektivní uživatelské prostředí. Zkombinováním jednoduchosti a výkonnosti vzniknou aplikace, ve kterých se bude uživatel soustředit hlavně na svou práci, aniž by jej rozptylovaly zbytečné panely, či dialogy a obtěžovalo neintuitivní ovládání programu.

Navržený nástroj, jakožto součást této práce, má velmi široké využití i pro svou nepřesnost. V mnoha aplikacích byl již podobný nástroj implementován a ocení jej zejména začátečníci nebo uživatelé nerozumějící problematice křivek.

9. Použitá literatura

[1] Žára, J., aj.: *Moderní počítačová grafika*. 1. vydání, Brno: Computer Press®, 2004.

[2] Piegl, L., Tiller, W.: *The NURBS Book*. 2nd edition, Berlin; Heidelberg: Springer, 1997.

[3] Yamaguchi, F.: *Curves and surfaces in computer aided geometric design*. 1st edition, Berlin; Heidelberg: Springer-Verlag, 1988.

[4] *Wikipedie*. [online] URL: <http://cs.wikipedia.org/> [cit. 2007-05-10]

[5] *Wikibooks*. [online] URL: <http://cs.wikibooks.org> [cit. 2007-05-11]

[6] *CZ NeHe OpenGL*. [online] URL: <http://nehe.ceskehry.cz/ [cit. 2007-05-10]

[7] Čech, D.: *OpenGL referát na praktikum z informatiky*. [online] URL: <http://hippo.nipax.cz/docs/opengl_cz.pdf> [cit 2007-05-13]

[8] *OpenGLUT Project*. [online] URL: <http://openglut.sourceforge.net/ > [cit. 2007-05-14]

10. Přílohy

1. Manuál k ukázkovému programu
2. Plakát projektu

Manuál k ukázkovému programu

Jako demonstrace deformačního nástroje byl vytvořen program pro jednoduchou tvorbu a editaci parametrických křivek. Kromě křivek NURBS na kterých je prováděna vlastní deformace aplikace zběžně ukazuje i práci s nejčastěji používanou křivkou – Bézierovou kubikou

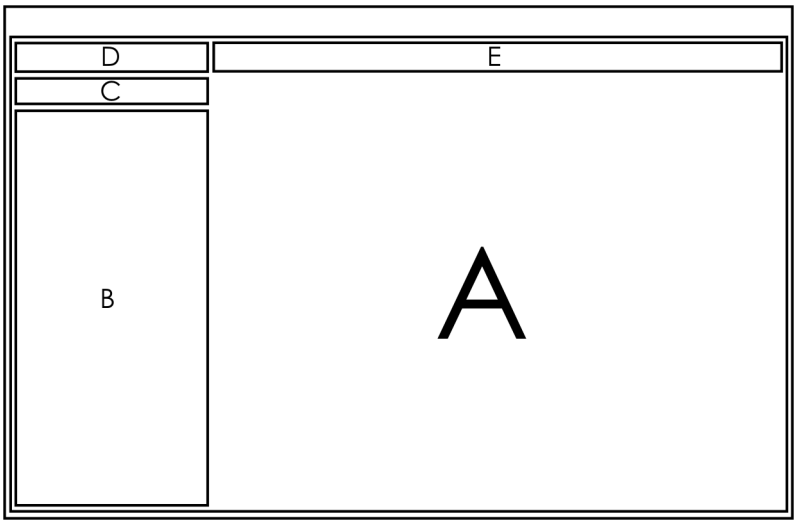
Spuštění

Aplikace využívá multiplatformní knihovny, a proto by nemel být problém při jejím spouštění na jakékoliv platformě podporující OpenGL a OpenGLUT.

Pro platformu Win32 je kromě samotné aplikace reprezentované souborem „bp.exe“ je nezbytná i dynamická knihovna nadstavby OpenGLUT „openglut.dll“. Knihovna nemusí být registrovaná v systému, stačí její umístění do stejné složky jako spoustitelný soubor.

Po spuštění se otevřou dvě okna. Konzolové a okno editoru.

Okno editoru



A – Pracovní plocha

B – Klávesové zkratky

C – Pozice kurzoru na ploše

D – Typ křivky

E – Popis právě používaného nástroje

Pracovní plocha je celá plocha okna, včetně částí B, C, D a E. Znamená to, že je možno zadávat body i „pod“ tyto části.

V části C, je-li typ křivky NURBS, se při pohybu nad křivkou ukazuje hodnota aktuálního parametru t a při deformaci je potom na toto místo vložený nový uzel.

Ovládání programu

Uživatel si může zvolit způsob jakým bude přepínat nástroje. Má k dispozici buď kontextové menu (pravé tlačítko myši) nebo klávesové zkratky.

Zadávání bodů (**L**): Na špičce kurzoru se objeví křížek, indikující právě tento nástroj. Levým tlačítkem myši jsou postupně přidávány body. Při tažení myši je možné umístit nový bod na přesnou pozici. Zadávání je ukončeno výběrem jiného nástroje nebo klávesou Escape.

Smazání všech bodů (**Delete**): Smaže všechny body nezávisle na aktuální činnost.

Deformace NURBSu (**D**): Deformační nástroj indikuje desetiúhelník (kružnice) vyznačující velikost deformačního nástroje. Ta se dá měnit buď zadáním přesné hodnoty, nebo kolečkem myši. Více o tomto nástroji níže.

Výběr typu křivky (**1, 2, 3, 4**): Přepíná mezi typy grafických elementů tvořených řídicími body. 1 pro body, 2 pro linky, 3 pro Bézierovy kubiky a konečně 4 pro NURBS křivky.

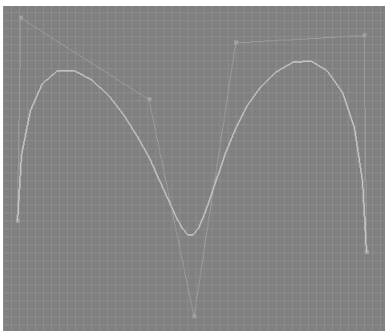
Načtení křivky (**L**): Smaže aktuální křivku a načte nový NURBS ze souboru "nurbs.txt"

Uložení křivky (**S**): Uloží aktuální křivku do souboru "nurbs.txt"

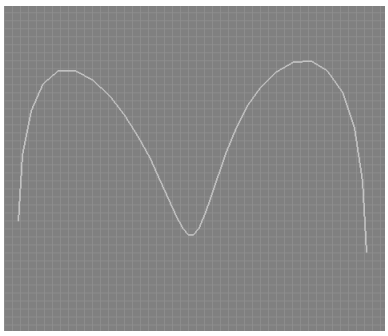
Nastavení zobrazení

Uživatel má možnost zobrazit si jednotlivé body křivky nebo skrýt řídicí polygon. Je potřeba připomenout, že deformační nástroj nepotřebuje řídicí body, a proto pro správnou představu použití tohoto nástroje v praxi je vhodné skrýt řídicí polygon.

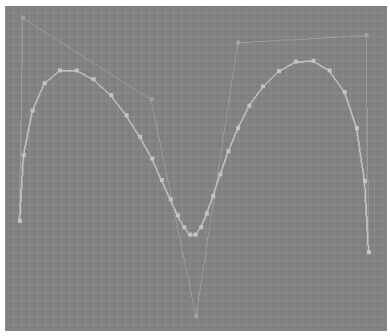
Obě možnosti jsou v položce "Nastaveni…" v kontextovém menu.



Standardní nastavení



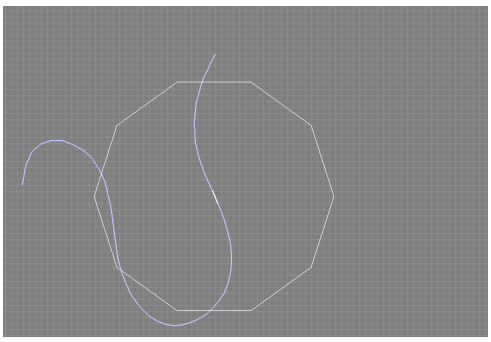
Skryté řídicí body



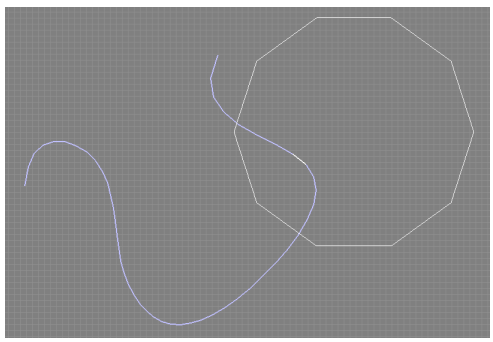
Zobrazené body křivky

Deformace

K deformaci křivky je potřeba mít již nějakou jednoduchou křivku vytvořenou. Po výběru deformačního nástroje si uživatel kolečkem myši nebo v kontextovém menu „Nastavení…" -> „Velikost deformace" určí velikost nástroje a tažením myši z místa na křivce zahájí deformaci. Deformací v aktuálním bodě ukončí uvolněním tlačítka myši.



Zahájení deformace (kliknutí na křivce)…



…ukončení deformace (uvolnění tlačítka)

Plakát projektu