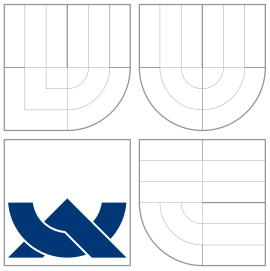
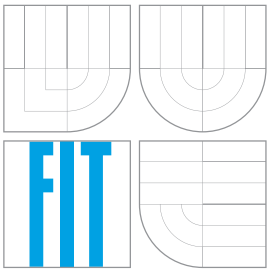


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

VÝUKOVÝ PROGRAM PRO DEMONSTRACI PRINCIPU SPLINE KŘIVEK

TUTORIAL FOR DEMONSTRATING PRINCIPLES OF SPLINE CURVES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — SLAVOMÍR KRBYLA
AUTHOR

VEDOUCÍ PRÁCE — Doc. PŘEMYSL KRŠEK, Ph.D.
SUPERVISOR

BRNO 2009

Abstrakt

Bakalárska práca sa zaoberá návrhom a implementáciou výkového programu pre demonštráciu spline kriviek. Program založený na frameworku Qt v programovacom jazyku C++ dáva užívateľovy možnosť vyskúšať si zadávanie niektorých v počítačovej grafike často používaných kriviek. Pritom sa može bižšie zamerať na ich vybrané vlastnosti.

Klíčová slova

Spline krivka krivky výukový demonštračný program spojitosť stupeň Ferusnova Kochanek-Bartels Catmull-Rom Bezierové NURBS DeCasteljau kubika racionálne neracionálne

Abstract

The bachelor thesis deals with the design and the implementation of tutorial program for demostration of spline curves. The program is based on the Qt framework ported into the C++ programming language. User can interact with feeding in some of the splines mostly used in computer graphic. He can also focus on spline characterisitics.

Keywords

Spline curve curves tutorial demostrate program continuity level Ferguson Kochanek-Bartels Catmull-Rom Bezier NURBS DeCasteljau cubic racional iracional

Citace

Slavomír Krbyla: Výukový program pro demonstraci
principu spline křivek, bakalářská práce, Brno, FIT VUT v Brně, 2009

Výukový program pro demonstraci principu spline křivek

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Přemysla Krška

. . . . . .
Slavomír Krbyla
26. ledna 2009

Obsah

Kapitola 1

Úvod

Rýchli vývoj informačných technológii rozšíril sféru ich použitia už skoro do každej oblasti. Medzi výjimky nepatrí ani použitie pre interaktívnu výuku. Takýchto programov v dnešnej dobe nájdeme mnoho.

Výukovým programom obecne rozumieme konktrétny software, ktorý je určený k výukovým účelom a je schopný plniť aspoň jednu z didaktických funkcií ktorými sú motivácia, expozícia učiva, upevňovanie osvojených vedomostí a dovedností a kontrola získanej úrovne vedomostí a dovedností.[6] Demonštračný program sa zaoberá práve expozíciou učiva a slúži ako pomocka učiteľovy pri vyučovaní, demonštráciou danej látky.

Pri demonštrácii principu spline krviek rozoberáme princip jednotlivých metód pre vykresľovanie krviek, snažíme sa ukázať ich vlastnosti a taktiež vstupné parametre, ktoré určujú tvar krivky. Pre jednoduchšie a prehľadnejšie zobrazenie budeme používat krivky dvojrozmerné, ktoré nám na demonštráciu princípu spline krviek bohate postačujú.

Na začiatoku dokumentu si formulujeme cieľ našej práce a pozrieme sa na dvôvody jej implementácie,potom nasleduje kapitola Teória kriviek v ktorej bližšie pozrieme na teoretickú časť krviek, ktorá nám približi ich matematickú reprezentáciu, spôsoby ako sa môžu medzi sebou jednotlivé krivky spájať, ako aj spôsoby ich modelovania. Ďalšia kapitola s názvom Analýza typu krivek bude venovaná blišie jednotlivým typom krviek použitých v našom programe, ich matematickému zápisu, ako aj ich najvýznamnejším vlastnostiam a parametrom. V kapitole Vlastná implementácia sa zameráme na popis implementovaného programu, rozoberieme použité inplementačné prostredie a framework,popíšeme si jednotlivé triedy a povieme si aj niečo k ovládaniu programu.

Kapitola 2

Formulácia cieľa

Našim cieľom je vytvoriť program spustitelný v operačnom systéme Microsoft Windows v jazyku C++ s frameworkom Qt vo vývojovom porstredí Visual Studio 2005. Implementované typy kriviek budú Fergusnove krivky, Kochanek-Bartels spline a Catmull-Rom Spline s interpolačných kriviek. S pomedzi aproximačných kriviek budeme implementovať Bezierové krivky a to konkrétne Obecné Bezierové krivky, Racionálne Bezierové krivky, Bezierové kubiky a algoritmus DeCasteljau používaný pre vykresľovanie týchto krviek, a krivky NURBS - neuniformný racionálny B-Spline. Princíp kriviek sa demonštruje pomocov vlastností a funkcií jedotliých typov kriviek a to hlavne spojitosť, polynomiálna báza, stupeň krivky ako aj spôsob ich vykreslovania. Aby sme boli schopný pochopiť jedotlivé typy kriviek a ich principy, musíme sa najprv zamerať na teóriu krviek a zistiť čo vlastne jednotlivé vlastnosti znamenajú, čo sú ich výhody a nevýhody a podľa toho ich začleniť do programu a nájsť spôsob ako tieto principy prezentovať užívateľovy čo najjednoduchšie a najintuitívnejšie.

Exisatuje veľa programov podobných tomu našemu. Vätšinov sa jedná o kvalitné výukové programy, ktoré sa vo vačšine prípadov neveľmi odlišujú od našeho programu. Ale tieto malé odlišnosti práve viedli ku nevýhodam pri istých spôsoboch ich použitia, vďaka čomu vznikol náš program. Mezi spomínané nevýhody patrí napríklad práca v online režime pri výukových porgramoch pomocov WWW, alebo prítomnosť teoretickej časti, ktorá je nadbytočná a niekedy ja mätúca keď sa program používa pre demoštráciu učiva učiteľom. Ďalšou nevýhodov je aj že niektoré programy sa zameriavajú iba na nietoré typy krviek (napríklad iba na interpolačné alebo iba aproximačné, alebo len konkrétne na jeden typ krviek).

Kapitola 3

Teória kriviek

Aby sme mohli začat rozoberať jedlotivé typy krviek, musíme najprv pochopiť čo to vlastne krivky sú, ich význam v počítačovej grafike, takisto ako sa zoznámiť s matematickým spôsobom ich zápisu a osvetliť si bližšie niektoré ich základné vlastnosti a parametre.

Krivky sa požívajú v počítačovej grafike a súvisejících aplikáciach na mnoho rôznych miestach. Stretávame sa snimi pri modelovaní v dvoch dimenziach, pri definíci fontov, pri určovaní dráhy pohybujúcich sa objektov v počítačovej animácii, pri definícii objektov pre šablónovanie ai. Rôzne aplikácie majú rôzne požiadavky a preto je táto oblasť pomerne široká.

V počítačovej grafike je pre vätšinu operácií objektov v počítači nutné ich exaktné vymedzenie. Objekty sú obyčajne množiny bodov, ktoré sú v rovine obmedzené krivkamy. Kvôli tomu vznikol požiadavok zjednodušiť fyzické vkladanie pomocou vstupného zariadenia práve pre krivky. Krivky sú najlepšie reprezentované funkciamy a tie je problematické zadávať. Pri ich zadávaní v tvare $y = f(x_1, x_2, ..., x_n)$ si len ťažko dokážeme predstaviť tvar krivky. Hľadajú sa preto metódy, ktoré umožnia užívateľovy čo najednoduchšie zadávať neakú krivku, a to pokým je možné tak aby sa dal dopredu odhadnúť jej tvar. Užívateľ má obyčajne za úlohu zadávať iba niekoľko riadiacich bodov,poprípade parametrov a matematický aparát sa o vytvorenie krivky postará sám.

3.1 Reprezentácia krviek

Krivky sú obyčajne v počítači reprezentované ako sústava parametrov neakej rovnice, ktorá je hneď nato generatívne zobrazovaná. Toto vyjadrenie môže byť trojitého druhu - explicitne, implicitne, parametricky.

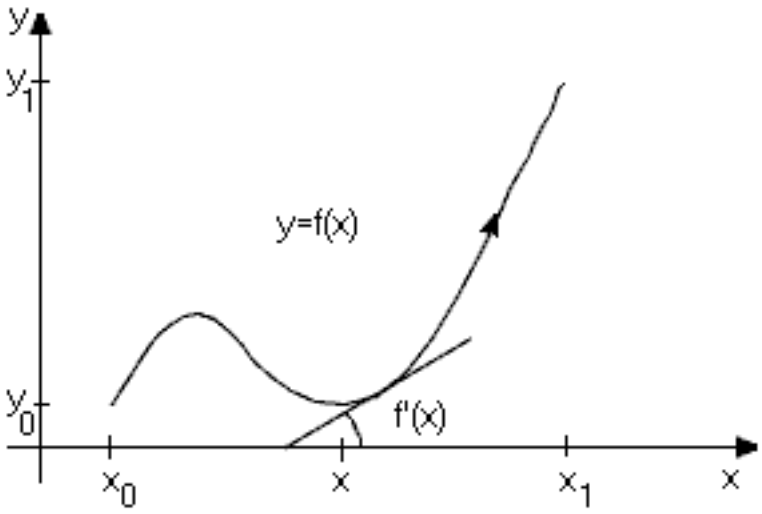
Explicitne zadaná krvika može byť zadaná ako spojitá funkcia v tvare $y = f(x)$ a býva orientovaná v smere rastúceho x (obrázok 3.1). Jedná sa však o zadanie, ktoré je možné použiť pre krivky, ktoré sú zároveň funkciamy, tzn. že hodnote x z definičného oboru odpovedá jediná funkčná hodnota y

Implicitné zadanie krivky má tvar $F(x, y) = 0$. Toto zadanie je pomerne obtiažne zobrazitelné v porovnaní s ostatnými, pretože neumožňuje v obecnejčích prípadoch postupný výpočet krivky. Svoj význam má napríklad pri testovaní oblastí vymedzených implicitne zadnou krivkou.

V počítačovej grafike sa pre vyjadrenie kriviek najčastejšie používa tvar parametrický (obrázok 3.2). Ten krivku chápe fyzikálne, ako dráhu pohybujúceho sa bodu,, jeho súradnice sú funkciamy paramertu t (času).

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

Pramater t je z intervalu $t \in < t_{min}, t_{max} >$ a najčastejšie je volený v rozsahu $t \in < 0, 1 >$.



Obrázek 3.1: Explicitne zadaná krivka

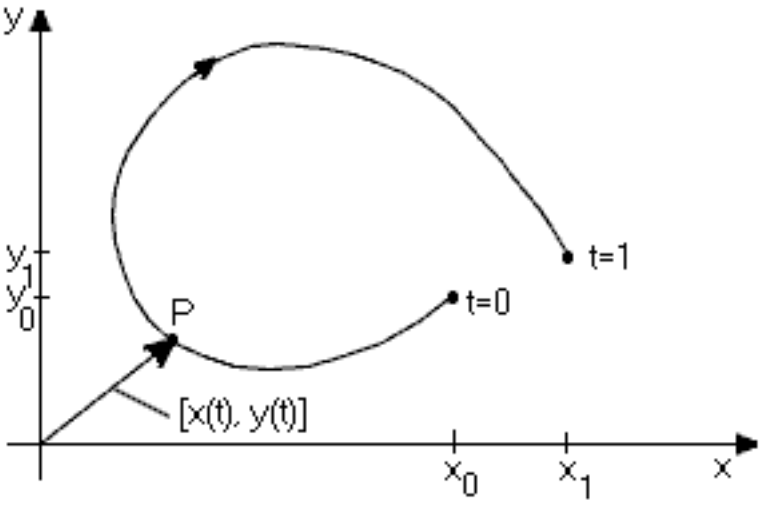
Funkciamy je určená bodová rovnica krivky

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

alebo vektorová rovnica

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

Výhodov parametrického zápisu je závislosť súradníc krivky na jedinom parametry t. Vďaka tomu je možné vyjadriť priebeh, kedy krivka prechádza viackrát rovnakými bodmy v priestore.



Obrázek 3.2: Parametricky zadaná krivka

Tečný vektor v bode $Q(t_0)$ je určený deriváciamy parametricky vyjadrenej krivky po zložkách

v tvare

$$\vec{q'}(t_0) = (x'(t_0), y'(t_0), z'(t_0)) = (\frac{dx(t_0)}{dt}, \frac{dy(t_0)}{dt}, \frac{dz(t_0)}{dt}).$$

Rovnica tečny, tj. priamky ktorá sa krivky dotýka, sa vypočíta z tečného vektoru a bodu na krivke ako

$$P(u) = Q(t_0) + u\vec{q'}(t_0)$$

3.2 Spojitosť

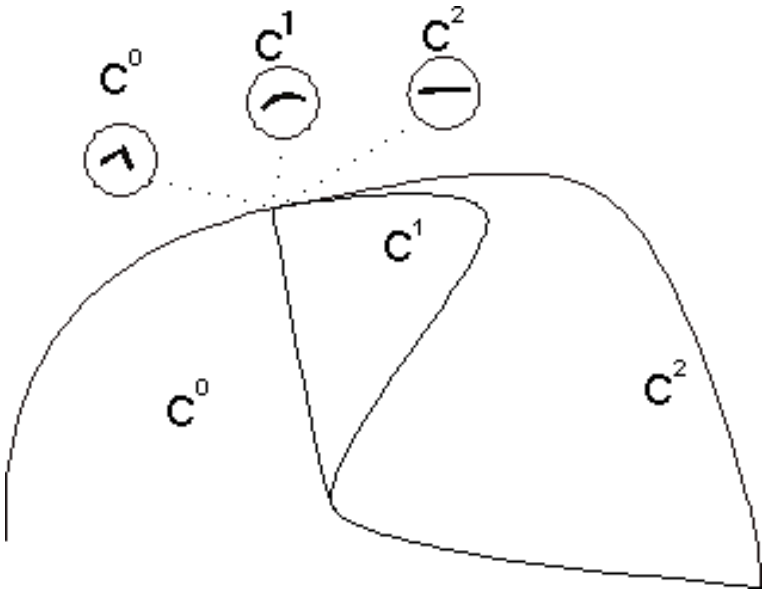
Segmenty $Q_1(t)$ a $Q_2(t)$ sú dve časti jediné krvivky $Q(t)$ spojené v bode $Q_1(1) = Q_2(0)$. Bod v ktorom sa krivky spojujú budeme nazývať uzol (knot). Zaujíma nás hlavne spôsob ich napojenia, hlavne spojitost(continuity) v uzle.

Hovoríme, že krivka $Q(t)$ je spojitosti C^n, ak má vo všetkých bodoch spojité derivácie podľa parametru t do rádu n. Označenie C^n sa nazýva parametrická spojitosť stupňa n (obrázok 3.3).

Dva segmenty sú spojite naviazané, tj. majú spojenie triedy C^0, ak je koncový bod prvého segmentu počiatočným bodom segmentu druhého. Spojitosť C^1 majú, ak tečný vektor v koncovom bode prvého segmentu rovný tečnému vektoru druhého segmentu. Analogicky rovnosť vektoru prvej a druhej derivácie je požadovaná pre C^2 atď. Zkrátene zapisujeme

$$\vec{q_1}^{(i)} = \vec{q_2}^{(i)}(0); \forall i = 0, 1, ..., n.$$

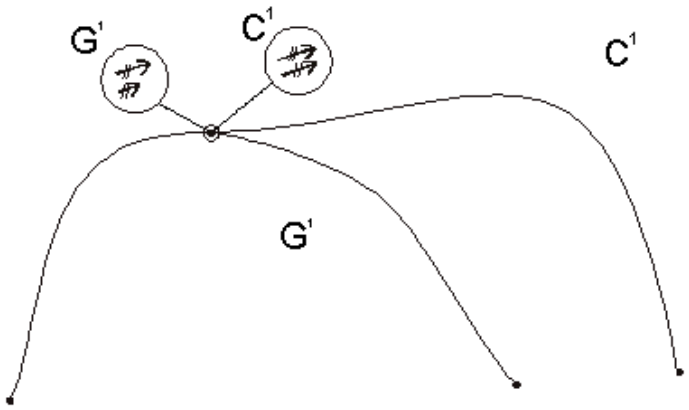
Čím vyššia spojitosť je požadovaná, tým dlhšiu dobu (v zmysle parametra t) sa oba segmenty primykajú k rovnakému smeru. Zjavne $C^{n+1} \Rightarrow C^n$.



Obrázek 3.3: Spojitosť C^0,C^1 a C^2

Hladkosť naväzovania kriviek možme taktiež posudzovať podľa geometrickej postupnosti označovanej G^n, najčastejšie sa používajú geometrické spojitosti G^0 a G^1. Dva segmenty krivky $Q(t)$ sú G^0 spojité, ak je koncový bod Q_1 totožný s počiatočným bodom Q_2. Dva segmenty sú G^1 spojité, ak sú G^0 a súčasne tečné vektory $\vec{q_1'}(1)$ segmentu Q_1 a $\vec{q_2'}(0)$ segmentu Q_2 sú súhlasne kolineárne, tj. platí:

$$\vec{q_1'}(1) = k\vec{q_2'}(0); k > 0.$$



Obrázek 3.4: Geometrická a parametrická spojitosť

Táto spojitosť zaručuje totožnosť tečien (ale nie tečných vektorov).

Geometrická spojitosť G^n v danom bode je definoaná nezávisle na spôsobe akým boli segmenty vytvorené, tj.nezávisle na parametry t. Za predpokladu, že obe krivky sú v mieste spojenia diferencovatelné, potom $Q(1)$ a $Q(2)$ splňujú podmienku gemotricikej spojitosti G^n ,ak sú v bode $[x_0, y_0, z_0]$ G^{n-1} spojité a platí

$$[\frac{\delta^n Q(1)}{\delta x^n}, \frac{\delta^n Q(1)}{\delta y^n}, \frac{\delta^n Q(1)}{\delta z^n}]_{[x_0,y_0,z_0]} = h*[\frac{\delta^n Q(2)}{\delta x^n}, \frac{/delta^n Q(2)}{\delta y^n}, \frac{\delta^n Q(2)}{\delta z^n}]_{[x_0,y_0,z_0]}, n > 0, h > 0.$$

Zo subjektívneho hľadiska zaručuje G^1 spojitosť takmer rovnakú hladkosť ako C^1, z hľadiska použitia je ďaleko jednoduchšie zaručit spojitoť G^1 ako C^1. Spojitosť C^1 implikuje G^1 s výnimkov jediného prípadu a , keď vektor rýchlosti v mieste spojenia dvoch segmentov je (0,0,0). Obrácene toto neplatí, pretože gemoetrická spojitosť nepostihuje rýchlosť a zrýchlenie pohybu. [2]

3.3 Modelovanie krviek

Základným druhom parametrických krivek používaných v počítačovej grafike sú krivky polynomiálne

$$Q_n(t) = a_0 + a_1 t + ... + a_n t^n$$

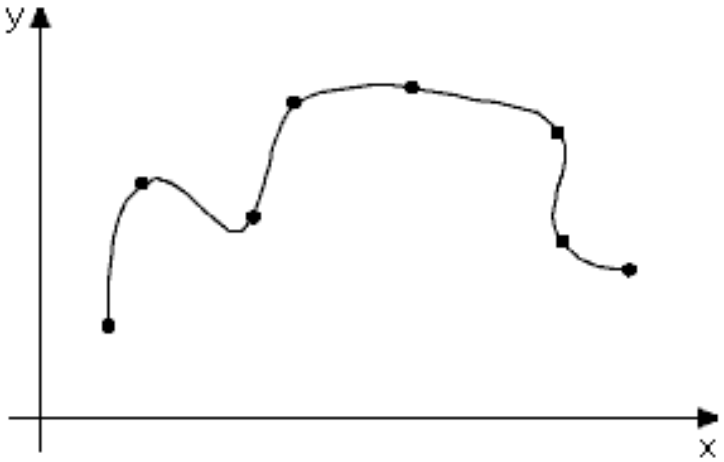
Polynomiálne krivky môžeme jednoducho vyčísliť a ich dalšou výhodov je že sú jednoducho diferencovatelné. Z polynomiálnych krivek je možné skladať krivky po častiach polynomiálne, krivky ich segmentamy sú polynomiálnymi krvikamy. Najčastejšie používané sú krivky tretieho stupňa - kubiky, ktoré poskytujú dostatočne širokú škálu tvarov, ich výpočet býva nenáročný a je možné snimi jednoducho manipulovať.

Existujú dva základne spôsoby interpretácie riadiacich bodov a to interpolácia a aproximácia. Krivka generovaná pri interpoláacii prebieha danými bodmy, zatiaľ čo pri aporximácii je riadiacimy bodmy tvar určený, ale krivka samotná nimi prechádzať nemusí.

Medzi najznámejších predstaviteľov interpolačných krviek patria Hermitovské krivky, Kochanek-Batrels spline a Catmull-Rom spline.

U aproximačných sú to Bezierové krviky a Neuniformný racionálny B-spline(NURBS). Princip a vlastnosti týchto krviek som sa preto rozhodol demonštrovať v našom výukovom programe.

Parametricky zadanú kubiku $Q(t)$ v tvare:



Obrázek 3.5: Interpolačná krivka

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$
$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y$$
$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z$$

môžeme napísať skrátene v matematickom tvare:

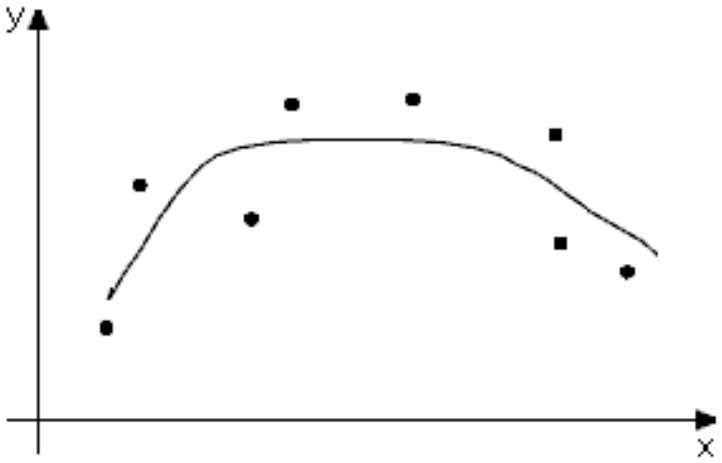
$$Q(t) = \boldsymbol{TC} = [t^3 t^2 t 1] \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}$$

Deriváciou $\vec{q}'(t)$ získame deriváciu vektoru **T**

$$\vec{q}'(t) = \frac{d}{dt} Q(t) = \frac{d}{dt} \boldsymbol{TC} = [3t^2 2t 10] \boldsymbol{C}$$

Kubika v priestore je určená dvanástimy parametramy, ktoré tvoria prvky matice **C**. Ovládanie tvaru krivky pomocou jednotlivých parametrov však nieje dosť intuitívne, zo zmeny parametrov sa nedá jednoducho odhadnúť zmena tvaru krivky. Pri definícii kriviek je s určitými vlastnosťami je teda výhodnejšie oddeliť charakteristiky ktoré sú pre danú krivku individuálne od vlastností ktoré sú rovnaké pre všetky krivky modelované rovnakým spôsobom. V prípade kubik možme maticu **C** rozpísať do súčinu $\boldsymbol{C} = \boldsymbol{MG}$ kde matica konštánt **M** je typu $4x4$ a nazýva sa bázová matica a štovrprkový vektor **G** sa nazýva vektor geomtrických podmienok. Súčin **TM** definuje polynomiálnu bázu, ktorá je spoločná pre všetky krivky určitého typu. Vektor geometrických podmienok obsahuje konkrétne parametre, ktoré ovplyvňujú tvar krivky, napr. riadiacie body alebo riadiacie body a tečné vektory. Kubika je definovaná vzťahom

$$Q(t) = \boldsymbol{TMG} = [t^3 t^2 t 1] \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}$$



Obrázek 3.6: Aproximačná krivka

Túto rovnicu rozpíšeme ako súčet polynomov násobených geometrickými podmienkami

$$Q(t) = (m_{11}t^3+m_{21}t^2+m_{31}t+m_{41})*G_1+(m_{12}t^3+m_{22}t^2+m_{32}t+m_{42})*G_2+(m_{13}t^3+m_{23}t^2+m_{33}t+m_{43})*G_3+(m$$

Polynomy, ktorými sú geometrické podmienky násobené sa uplatnia ako premenlivé váhy riadnené parametrom t. Podľa hodnoty parametru t bázové polynomy určujú ktorá podmienka sa viac uplatní na začiatku krivky a ktorá má vätší vplyv na vnútorný priebeh alebo na koncovú časť.

Medzi často požadované vlastnosti kriviek patrí:

1. Invariácia k lineárnym transformáciam a projekciám
2. Vlastnosť konvexnej obálky:
   - silná podmienka - celá krivka leží v konvexnej obálke všetkých svojich riadiacich bodov
   - slabá podmienka - časť krivky leží v konvexnej obálke niektorých riadiacich bodov (segment v obálke svojeho generujúceho polynomu)
3. Lokalita zmien - zmenov polohy a/alebo váhy riadiaceho bodu sa mení iba časť krivky, ale nie krivka celá
4. Krivka má prechádzať krajnými bodmy svojeho riadiaceho polynomu [7]

Kapitola 4

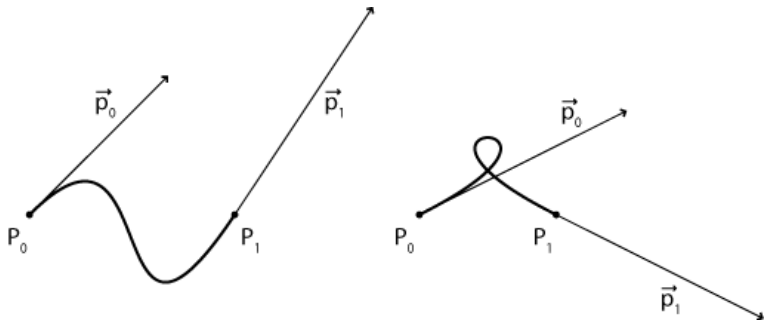
Analýza typov krivek

Pri analýze typov krivek nás zaujímajá hlavne spôsob ich zadávania pomocou periferneho zariadenia a taktiež ich vlastnosti ako sú spojitosť krivky a jej polynomiálna báza ako aj spôsob ako tieto vlastnosti jednoducho a zrozumiteľne demonštrovať pre užívateľov programu. Pri každej zvolenej krivke bude užívateľ vydieť akú má spojitosť ako aj graf polynomov použitých pre jej výpočet (výnimkov je algoritmus DeCasteljau kde sa pri vykreslení polynomy nepoužívjú)

4.1 Interpolačné krivky

4.1.1 Hermitovské kubiky

Jedná sa o krivky, ktoré sú zadávané pomocov dvoch bodov P_0 a P_1 a pomocov vektorov $\vec{p_0'}$ a $\vec{p_1'}$ v týchto bodoch.



Obrázek 4.1: Fergusnova krivka

Predpis pre výpočet Heritovskej kubiky má tvar:

$$Q(t) = [t^3 t^2 t 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ \vec{p_0'} \\ \vec{p_1'} \end{bmatrix}$$

Ak tento vzťah rozpíšeme do jednej rovnice dostávame

$$Q(t) = P_0 F_1(t) + P_1 F_2(t) + \vec{p_0'} F_3(t) + \vec{p_1'} F_4(t)$$

kde F_1,F_2,F_3,F_4 sú takzvané Hermitovské polynomy

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

Po dosadení hodnôt $t = 0$ a $t = 1$ do rovníc Hermitovských polynomov zisťujeme ich vplyv na začiatok a koniec krivky. Pre $t = 0$ sú všetky polynomy okrem F_1 nulové, to znamená, že krivka bude pre $t = 0$ určená funkciou $Q(0) = P_0$. Podobne je tomu aj v prípade $t = 1$ kedy je krivka určená funkciou $Q(1) = P_1$. Stoho vyplýva že krivka začína v prvom a končí v poslednom bode kubiky. Na jej priebeh ale majú hlavný vplyv vektory $\vec{p_0'}$ a $\vec{p_1'}$. Odnich práve závisí tvar krivky, miera jej vyklenutia. Čím je veľkosť vektoru väčšia, tým viac sa knemu krivka primyká. Kvoli tejto vlastnosti som sa rozhodol demonštrovať vplyv tečného vektora na krivku. Najprv demonštrujeme príklad kedy sa bude meniť iba smer vektoru, pričom jeho dĺžka ostane nezmennená, a hňed zaním bude nasledovať demonštrácia vplyvu jeho veľkosti, pričom tentokrát ostane nezmennený smer. [7]

Jeden segment Hermitovej krivky je stupňa 3. Celkový stupeň krivky je teda tiež 3.

Výhodov týchto kriviek je jednoduchá realízacia ich hladkej spojitosti. Pre prípad zadávania bodov a ich vektorov použitý v našom programe, a to že koncový bod predchádzajúceho segmentu a jeho vektor sú vždy počiatočným bodom a vektorom sektoru druhého, čiže tečné vektory oboch sektorov sú totožné a dostávame v každom prípade spojitosť C^1.

4.1.2 Kochaek-Bartels spline

Tieto krivky majú rovnakú interpolačnú schému ako Fergusnove krivky, s tým rozdielom že tečné vektory sa unich musia vypočítať a to tak že máme možnosť riadenia priebehu krivky cez interpolované body, konkrétne umožnňuje definovať näpetie (tension),spojitosť (continuity) a šikomsť (bias). Kvoli tomu sú taktiež nazývané aj TCB krivky.

Krivka je zadaná postupnosťou bodov $P_0, P_1, ..., P_n$ a riadiacmy koeficientamy a_i, b_i, c_i pre každý z vnútorných bodov $P_1, P_2, ..., P_{n-1}$. Krajné body síce definujú tvar krivky ale tá nimi neprechádza. Aby krivka prechádzala aj krajnými bodmy, je potreba tieto body zdvojiť. Pre výpočet krivky sa používa Fergusnova kubika, stým rozdielom že Kochanek-Bartels má pre každý bod definované dva vektory, vstupný $\vec{l_i'}$ - v zmysle parametru t pre ľavý segment - a výstupný $\vec{r_i'}$ - pre pravý segment. Tieto vektory vypočítame podľa vzťahu:

$$\vec{l_i'} = \frac{(1-a)(1+b)(1-c)}{2}(P_i - P_{i-1}) + \frac{(1-a)(1-b)(1+c)}{2}(P_{i+1} - P_i)$$
$$\vec{r_i'} = \frac{(1-a)(1+b)(1+c)}{2}(P_i - P_{i-1}) + \frac{(1-a)(1-b)(1-c)}{2}(P_{i+1} - P_i)$$

Parameter napetia a_i určuje veľkosť tečny v i-tom bode. Parameter šikomosti b_i mení smer a dĺžku tečného vektora. Parameter šikomosti c_i riadi spojitosť v príslušnom bode[7]. Náš program bude naviac obsahovať simuláciu týchto parametrov a ich vplyv na krviku. Simulácia bude prebiehať spôsobom, pri ktorom budú dva parametre rovné nule a ten tretí (ktorý bude simulovaný) bude postupne nadobúdať hodnôt v rámci celej svojej škály.

Zo vzťahu pre výpočet tečných vektorov zisťujeme že krivka je spojitosti C^1 , ak je hodnota parametru c_i pre všetky vnútorné body rovná nule. Potom dostávame vzťah

$$\vec{l_i'} = \frac{(1-a)(1+b)}{2}(P_i - P_{i-1}) + \frac{(1-a)(1-b)}{2}(P_{i+1} - P_i)$$

$$\vec{r_i'} = \frac{(1-a)(1+b)}{2}(P_i - P_{i-1}) + \frac{(1-a)(1-b)}{2}(P_{i+1} - P_i)$$

takže vstupný a výstupný vektor sú totožné.

V opačnom prípade sú vstupný a výstupný vektor odlišné a krvika ma spojitosť C^0.

Segment medzi dvomy vnútornými bodmy je stupňa 3, ciže celkový stupeň je rovnako ako u Hermitovských krviek 3.

4.1.3 Catmull-Rom spline

Jedná sa vlastne o špeciálny prípad krivke Kochanek-Bartels a to ked sú hodnoty parametrov a = b = c = 0. V tomto prípade sú hodnoty vektoru identické a sú dané iba polohov riadiacich bodov. Hodnota vektoru pre vnútorné body je potom rovná polovici vektoru daného bodmy P_{i+1} a P_{i-1}[7].

Ďalšou vlastnosťou a zároveň nevýhodov týchto krviek je že nie vždy ležia v konvexnej obálke. Túto vlastnosť som sa rozhodol demonštrovať aj v našom demoštračnom programe.

Spojitoť Catmull-Rom kriviek je v každom prípade C^1, kedže vstupný a výstupný vektor sú identické.

4.2 Aproximačné krivky

4.2.1 Bezierové krivky

Obecné Bezierové krivky

Bezierové krivky n-tého stupňa sú určené $n+1$ bodmy P_i riadiaceho polygonu a vzťahom

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

kde B_i^n sú Bernsteinové polynomy n-tého stupňa

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

Vtomto vzťahu je $\binom{n}{0} = 1$ a $0^0 = 1$.

Položíme vo vzťahu $t = 0$, resp. $t = 1$ zistíme že krivka prechádza krajnými bodmy. Po dosadení $t = 0$ a $t = 1$ do derivácie vzťahu získame výrazy pre tečné vektory v krajných bodoch

$$\vec{q'}(0) = n(P_1 - P_0)$$
$$\vec{q'}(1) = n(P_n - P_{n-1}).$$

Medzi nevýhody týchto krviek patrí fakt že pri zmene jedného bodu sa mení tvar celej krivky, s tohto dvôvodu sa tieto krivky robia nižšieho stupňa, ktoré postupne naväzujú [7].

Ich stupeň je rovný počtu riadiacich bodov -1.

Racionálne Bezierové krivky

Jedná sa o zobecnenie Bezierových krviek. Ku každému bodu sa sa pridá reálne číslo, ktorého zmena mení aj tvar krviky.Krvika je potom definovaná riadiacimy bodmy $P_0, P_1, ..., P_n$ a reálnymy

číslamy $w_0, w_1, ..., w_n$ ktoré reprezentujú váhu bodu. Racionálna Berzierová krvika je potom určená vzťahom

$$P(t) = \frac{\sum_{i=0}^{n} w_i P_i B_i^n(t)}{\sum_{i=0}^{n} w_i B_i^n(t)} = \sum_{i=0}^{n} w_i P_i R_i^n; i = 0, 1, ..., n; w >= 0$$

pričom

$$R_i^n = \frac{w_i B_i^n(t)}{\sum_{j=0}^{n} w_j B_j^n(t)}[7]$$

Racionálne Bezierové krivky sú vlastne Obecné berzierové krivky pre ktoré platí že každý bod má váhu práve 1. Výhodu oproti obecným majú hlavne v tom že pre zmenu krivky nemusíme meniť polohu riadiacich bodov. Tú to vlastnosť som rozhodol demonštrovať v našom programe pomocou simulácie, kde sa postupne mení váha bodu ale riadiace body ostávajú nezmenené a užívateľ teda može sledovať vplyv váhy bodu na krivku.

Bezierové kubiky

Sú to Berzierové krivky tretieho stupňa ktoré sú zadané 4 riadiacimy bodmy P_0, P_1, P_2, P_3. Kubika prechádza prvým a posledným bodom a je určená vzťahom

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

pričom Bersteinové polynomy majú tvar

$$B_0(t) = (1-t)^3$$
$$B_1(t) = 3t(1-t)^2$$
$$B_0(t) = 3t^2(1-t)$$
$$B_0(t) = t^3$$

Maticový zápis Bezierovej kubiky je

$$Q(t) = [t^3 t^2 t 1] \begin{bmatrix} 1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

tečné vektory v prvom a v poslednom bode majú tvar

$$\vec{p'}(0) = 3(P_1 - P_0) \vec{p'}(1) = 3(P_3 - P_2)[7]$$

Na tých to krivkach som sa rozhodol demonštrovať náväznosť Bezierových kriviek, ako aj obecne spojitosť $C^0, C^1 a G^1$ kvoli jednoduchému zobrazeniu tečných vektorov v počiatočných a konečných bodoch. Spojitosť C^0 zaistíme totožnosťov posledného bodu prvého segmentu a prvého bodu druhého segmentu. Spojitosť C^1 zaistíme ak je posledný bod prvého segmentu zároveň prvým bodom druhého segmentu a taktiež ak tento bod leží uprostred úsečky danej predposledným bodom prvého segmentu a druhým bodom druhého segmentu. Spojitosť G^1 nastane v prípade že tieto body ležia na priamke, majú rovnaké poradie ako u spojitosti C^1,ale niesú od seba rovnako vzdialené a taktiž niesú totožné.

Kubiky sú vždy stupňa 3.

Algoritmus DeCasteljau

Tento algoritmus je určený pre rasterizáciu Bezierových krviek. Naivným a neadaptívny spôsobom je dosadzovanie parametru t do parametrického vyjadrenia rovnice. Zmena parametru je konšatná kedže Berzierové krivky sú uniformné. Tento spôsob je nefektívny, pretože nerešpektuje zakrivenie úšečky. Tento problém rieši rekurzívny výpočet pomocou algoritmu DeCasteljau. Bod krivky sa vypočíta pomocou rekurentného vztahu

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

kde $i = 1, 2, ..., n$ a $j = i, i+i, ..., n$ Vstupom tohto algoritmu sú body riadiaceho polygonu[7].

Berzierová krivka može byť pomocou tohto algorimu rozdelená na dve ľubovólne časti v ktorom koľvek bode. Najideálnejšie je ju v strede, teda pre t=1/2. Každé rozdelenie generuje dva nové riadiace polygony, ktoré sú presňejšou aproximáciou tejto krivky.

Výhodov tohto algoritmu je možnosť jeho ukončenia napríklad v prípade že sme už dosiahli požadovanú rovnosť medzi dvoma bodmy krivky. Vďaka tomu generuje algoritmus menšie množstvo dat. Rovnejšie krivky sú vlastne reprezentované ako úsečky zatiaľ co u krivších častí použijeme hlbšiu rekurziu čím dosiahneme jemnejšie rozdelenie. Ukážku konverzie tohto algoritmu, konkrétne spôsob delenia pre akúkoľvek hodnotu parametru t som sa rozhodol demonštrovať v našom programe.

4.2.2 NURBS

Neuniformný Racionálny B-Spline (dalej len NURBS) je dvojitým zobecnením B-Spline krviek. Neuniformný znamená, že zmena parametru t nemusí byť vždy konštantná a racionalizmus pridáva ku každému bodu ešte aj jeho váhový koeficient.

Krivka NURBS je určená $n+1$ body $P_0, P_1, ..., P_n$ riadiaceho polygónu, rádom B-Spline k (najvyšší stupeň polygonu je $k-1$) a uzlovým vektorom $\boldsymbol{U}$ dĺžky $n+k+1$. Uzlový vektor je tvorený postupnosťou neklesajúcich reálnych čísel - uzlových hodnôt $t_0 \leq t_1 \leq ... \leq t_{n+k}$ Uzlové hodnoty v tejto postupnsti sa môžu opakovať.

Krvika NURBS je určená vzťahom

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

kde $R_{i,k}$ je racionálna B-Spline báza

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

pričom w_i je váha i-tého bodu riadiaceho polygonu a $N_{i,k}(t)$ sú normalizované B-Spline bázové funkcie definované rekurentným vzťahom

$$N_{i,k}(t) = \frac{t-t_i}{ti+k-1-t_i} N_{i,k-1}(t) + \frac{t_{i+k}-t}{t_{i+k}-t_{i+i}} N_{i+1,k-1}(t) pret_i < t_{i+1+k}, 0 \leq i \leq n.[7]$$

Medzi najdôležitejšie faktory, určujúce tvar NURBS krivky patrí bezpochyby jej stupeň a tvar uzlového vektoru. Vplyv zmeny týchto parametrov na výslednú krviku som sa preto rozhodol demonštrovať. U stupňa krviky to bude prevedením simulácie, ktorá postpne dekrementuje stupeň

krvivky až na minimum, a potom ho inkrementuje na pôvodnú hodnotu. U uzlového vektoru to bude pomocou tlačidiel, ktoré zhustia uzlový vektor buď na začiatku, alebo na konci.

Významnou vlastnosťou týchto krviek je možnosť presne vyjadriť kuželosečky ako podiel polynomov a to pomocou váhových koeficientov. Preto som do programu zahrnul aj dva ukážkové príklady pre vykreslovanie kružníc.

Kapitola 5

Vlastná implementácia

5.1 Programovací jazyk, framework a vývojové prostredie

Pre implemenáciu programu sa ponúka hňed niekoľko programovacích jazykov. Mňa snich najviac zaujal jazyk *C++*, ktorý podporuje vytváranie GUI aplikácií pomocov prídavných knižníc, pričom je ho možné napojiť na hardware. Pre jazyk C++ existujú rôzne nadstavby nástrojov, tzv. framevorky, ktoré značne uľachčujú písanie programov. Medzi jeho nevýhody patrí hlavne závisloť na platforme, pretože zdrojový kód je nutné pred prelopžiť pred spustením do strojového kódu, ktorý je závislý na platfomre. Na druhej strane ale vďaka tomu výsledný spustiteľný program získava na svojej rýchlosti[5].

Tým sa nám ale naskytá dalšia voľba a to vybrať pre našu implementáciu vhodný framework. Spomedzi všetkých vyšiel najvhodnejšie framework *Qt* od firmy TrollTech. Prostredie umožňuje vývoj aj pre viac platforiem, takisto podporuje i iné programovacie jazyky. Framework obsahuje moduly s rôznymi funkciamy pre uľahčenie práce programátora. Mimo iné obsahuje vlastné nástroje pre tvorbu GUI, preklad programu do viac jazykov, ako aj nástroje pre vykreslovanie a základnú prácu s grafikov. K dispozícii je Qt Desinger, prehľadný grafický nástroj pre tvorbu GUI programov. Sada nástrojov QT neobsahuje iba grafické triedy ale taktiež kompletnú prestavbu štandardnej C++ knižnice. Tieto triedy majú predponu Q. Patrí medzi ne napríklad `QArray`, `QVector`, `QFile`, `QDir`, `QString`, `QTime` či `QDate`. Ďalej podporuje nadstavbu fontu triedamy `QFont`, `QFontInfo`. Pre kreslenie je možné použiť triedy `QPainter`,`QColor`,`QPixmap`,`QGraphicView`. Vytváranie viac vlákonvých aplikácií Qt zjednodušuje triedamy `QThread`,`QProcess`. Taktiež podporuje grafickú knižnicu OpenGL, prácu s XML, znakovými sadamy a databázamy[4].

Pre vývojové prostredie som sa rozhodol použiť komerčný produkt firmy Microsoft a to Visual Studio 2005, ktoré je v rámci školskej licencie poskytované pre štúdijné účely zdarma. Na internete sa mi podarilo nájsť jednoduchý tutoriál[1], ktorý ukazuje spôsob akým sa dá framework Qt integrovať do tohoto vývojového prostredia. Visual Studio v sebe obsahuje Microsoft Visual C++, integrované vývojárske prostredie pre programovacie jazky C,C++ a C++/CLI [3]. Taktiež sa Visual Studio stará o prehľadné odsadzovanie zdrojového kódu a zvýraznoanie kľúčových slov, čo nám výrazne zjednoušuje prácu.

5.2 Popis tried

Medzi najzákladnejšiu triedu programu patrí trieda *Application*, ktorá je potomkom triedy *QApplication*. Konštruktor tejto triedy je volaný s funkcie *main*, ktorá je vlastnme vstupným bodom programu. Táto trieda si ukladá instanciu aplikácie, teda ukazovateľ na samu seba z dvovodu vi-

acnásobného spustenia aplikácie. Ďalej ukladá operačný typ, ktorý určuje s akým typom kriviek zrovna pracujeme a je dvoležitov premmenou vetvenia programu v mnohých funkciách, hlavne tých súvsejúcich s ovládaním programu a vykreslovaním. Taktiež ukladá vstupné parametre programu pre prípad že by si ich použitie vyžadovalo neskoršie vylepšenie programu. Vo svojom konštruktore taktiež volá konštruktor triedy *MainWindow*, ktorá je srdcom celého našeho programu, a ukladá si ukazovateľ na túto triedu.

Trieda *MainWindow* sa stará hlavne o grafické užívatelské rozhranie, jedná sa o hlavné okno aplikáacie. Táto trieda je potomkom triedy *QMainWindow*, určuje rozmiestnenie jedolivých elementov(tlačidiel, prepínačov ...) v hlavnom okne a ako aj obsluhu signálov nimi generovaných. Podľa ich stavu nastavuje operačný typ aplikáacie, vykoná príslušnú reinicializáciu alebo vykoná potrebnú postupnosť úkonov. Takisto vypočíta nové veľkosti a rozmiestnenie elementov v prípade že bola zmenená veľkosť hlavného okna. Ďalej obsahuje vektory riadiacich bodov, ich váhového parametru ako aj TCB parametrov v prípade krivky Kochanek-Bartels a inicializáciu týchto vektorov pre špecifický operačný typ. Obsahuje aj implementáciu všetkých funkcií potrebných pre vypočítanie bodov priamky z riadiacich bodov podľa vyššie uvedeného matematického zápisu pre každú implementovanú krivku. Pri krvike Kochanek-Bartels som sa rozhodol najprv použiť funkciu vypočíta vektory v riadiacich bodoch a vráti riadiace body a vektory Fergusnovej kubiky a s týchto riadiacich bodov a vektorov potom vypočita pomocou funckie pre výpočet Fergusnovej krviky body krivky. Pri funkcii pre výpočet bodov krivky z riadiacich bodov pre Fergusna som sa zase rozhodol riadiace vektory v jednolivých bodoch 3x vynásobiť, krivka tak nebudem mať presne tvar určený matematickým modelom, ale na druhej strane sa tak jednoduchšie ukazuje vplyv tochto vektoru na tvar výslednje krivky. Taktiež sú v tejto triede implementované jednotlivé simulácie. Pre riadenie simulácii som použil triedu *QTimeLine*, ktorá zabezpečuje časovanie v jednotlivých simuláciach. Počas priebehu simulácii sa deaktivuje celé hlavné okno, aby nemohlo dojsť k zmeneniu inicializácie simulácie a tým k jej nečakanému správaniu ktoré vo vatšine prípadov končilo pádom programu. Trieda si taktiež ukladá ukazovatele na triedy *PaintArea* a *FunctionW*, ktoré sú určené pre obsluhu chovania tzv. Widgetov, čo sú komponenty grafického užívatelského rozhrania na ktoré posobý užívateľ. Konštruktory týchto tried sú volané v konštrutkore triedy *MainWindow*, kde sa zároveň nastaví aj geometriu týchto Widgetov.

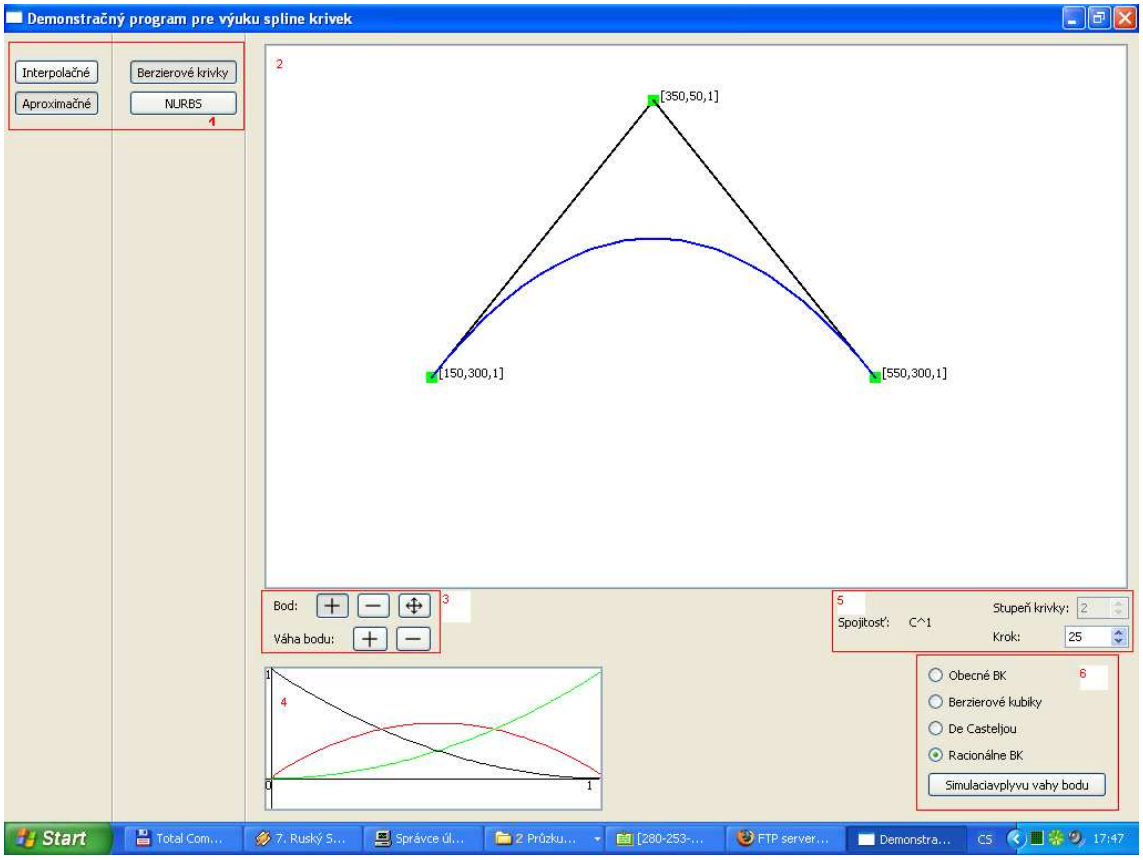
PaintArea je trieda slúžiaca pre vykreslovanie krviek a obsluhu udalostí myši. Je dedená s triedy *QWidget*a vždy keď obdrží požiadavok o prekreslenie, buď implicitne zmenov jej veľkosti vykreslovacieho okna alebo explicitne od triedy *MainWindow* alebo sama od seba pomocov funkcie *Update()* prekreslí obsah vykresľovacieho okna podľa aktuálneho operačného typu. Okrem samotnej krivky a riadiacich bodov vykresľuje aj všetky ostatné požadované veci, ako napríklad konvexnú obálku u Catmull-Rom krivky alebo ukážku rozdelenia krviky v strede pomocou algoritmu De-Casteljau. Na vykreslovanie používame triedu *QPainter*, ktorá v sebe obsahuje základné funkcie pre vykresľovanie geometrických tvaroch, ako aj zmenu šírky čiary alebo farby vykresľovaného objektu. Trieda sa stará aj o obsluhu myši súvisejúcej s vykresľovacím oknom. Pri kliknutí pridá bod, alebo zisťí či bolo kliknuté na bod a potom ho poprípade vymaže, zvätší/zmenší váhu bodu alebo označí bod pre posúvanie ak je pohnuté myšou pred pustením tlačidla, záleží na momentálonom stave aplikácie. Kolečko myši slúži pri označovaní bodu pri Kochanek-Bartels krivke pre ktorý chceme upravovať parametre TCB. Pri pohybe myši sa okrem pohybu bodu taktiež mení aj kurzor myši. Súčasťou tejto triedy mala byť aj funkcia pre vykresľovanie vektorov *drawArrow*, túto funkciu som však už nestihol doimplementovať.

Pre vyykreslovanie polynomov slúži trieda *FunctionW*. Rovnako ako trieda *PaintArea* je dedená sa triedy *QWidget*. Jej úlohov je vykresliť jednotlivé polygony podľa operačného typu aplikácie. Prekreslenie tejto funkcie je explicitne vyžiadané od triedy *PaintArea* pomocou metody *Update()* v momente keď trieda *PaintArea* dokončí vykresľovanie.

5.3 Vzhľad fomrmuláru

Na vzhľad a rozmiestnenie prvkov GUI bol kladený veľký dôraz, aby bolo docielené jednoduché a intuitívne ovládanie. Pre grafické prvky GUI má framework Qt špecíalny modul nazvaný výstižňe QtGui. Pre vytvorenie formuláru hlavného okna a rozmiestnenie jednolivých elementov v ňom vynikajúco poslúžil nástroj pre tvorbu GUI programov QDesigner.

5.3.1 Hlavné okno



Obrázek 5.1: Hlavné okno programu

Hlavné okno programu je potomkom triedy `QMainWindow`, čo je základný stavebný prvok v Qt pre vätšie GUI aplikácie. Ako vydíme s obrázku (5.1) prvky hlavného okna sa delia do 6 základných častí. Majú nasledujúce použitie

1. Výber typu krivky
2. Vykrelsenie krivky
3. Správa riadiacich bodov
4. Vykreslenie polynomiálnej bázy
5. Vlastnosti krivky

6. Parametre krivky

Výber typu krviky je realizovaný pomocov tlačidiel triedy `QButton`, ktoré sú nastavené tak, aby bolo možné tlačidlo zatlačiť, nielen naň kliknúť. Dvojica tlačidiel umiestnená na ľavo slúži pre vybranie skupiny krviek podľa spôsobu ich modelovania, a to aproximačné alebo interpolačné a určí počet tlačidiel na pravej strane a ich popisky, kedže tieto tlačidlá už reprezentujú konkrétne typy kriviek.

Pre vykresľovanie krivky používame triedu `QWidget`. Samotné kreslenie je realizované pomocov triedy `QPainter`. Pre každý vykreslený riadiaci bod je uvedenená X-ová a Y-ová súradnica ako aj váha tohto bodu. Ak nám to matematický aparát dovoľuje (napríklad niesu zadané iba dva riadiace body pre vykresľovanie kubiky) vykresľujeme aj krviku samotnú. Niektoré krivky si vyžadujú ešte vykreslenie ich špecíaônych parametrov alebo vlastností, napr. tečny ku krvike Catmull-Rom, alebo ukážka algoritmu DeCasteljau.

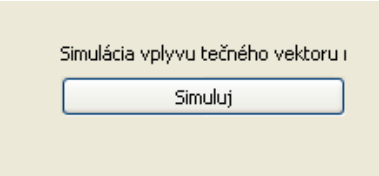
Správa riadiacich bodov je rovnako ako výber typu krivky realizovaný pomocov tlačidiel triedy `QButton`. Pomocov stavu týchto tlačidiel nastavujeme operáciu, ktorá sa prevedie pri udalosti myši vyvolanej v oblasti pre vykresľovanie krivky. Vrchné tri tlačidlá slúžia pre vloženie nového bodu, vymazanie existujúceho bodu a premiestnenie exisstujúceho bodu. Zaťiaľ čo dva spodné tlačidlá sú vyditeľne iba pri racionálnych krivkách a slúžia pre zvýšenie a znýženie váhy bodu. bodu.

Vykresľovanie polynomiálnej bázy je zase rovnako ako pre vykresľovanie krivky použitá trieda `QWidget` a pre samotné kreslenie trieda `QPainter`. Slúží nám pre vykresľovanie polynomiálnej bázy danej krivky. Každý polynom vykresľujeme inou farbou.

Pre zobrazenie vlastností každej krivky sme použili triedy `QLabel` a `QSpinBox`. Výsledná spojitosť krviky je zobrazená pomocov triedy `QLabel`, ktorá slúži pre zobrazovanie reťazca a nieje možné ju uživateľsky modifikovať zatiaľ čo stupeň krivky a krok krivky su dané triedov `QSpinBox`, ktorá služi pre zobrazovanie a zadávanie čísel užívateľom s programátorsky zvolenej množiny. Stupeň krivky je ale okrem jedného prípadu (krvika NURBS) celý beh programu deaktivovaný a slúži iba pre zobrazenie stupňa krivky, pretože v týchto prípadoch je stupeň konštantný. Krokom krivky rozumieme počet zmien hodnoty t v intervale $< 0, 1 >$.

Parametre krivky záležia od typu krivky, preto sú vo formuláry použité rozne tirdy pre rôzne typy krviek.

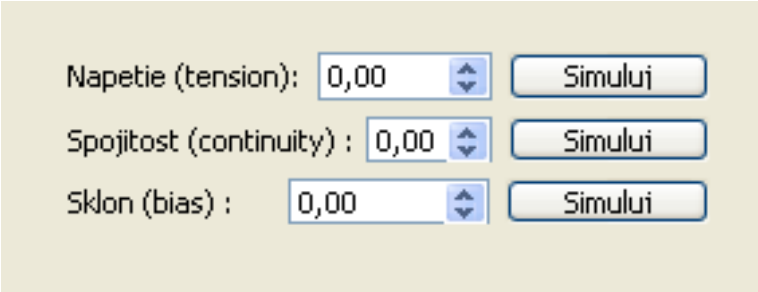
5.3.2 Parametre kriviek



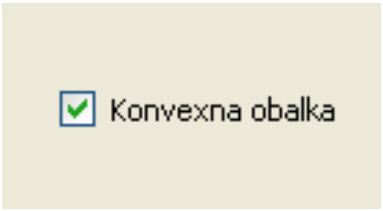
Obrázek 5.2: Parametre Fergusnovej krivky

Pre vyjadrenie parametrov Fergusnovej krivky je použité jediné tlačidlo triedy `QButton`, ktoré slúži pre spustenie simulácie vplyvu tečného vektora na výslednú krivku.

U kriviek Kochanek-Bartels je pre každý z parametrov TCB použitý jeden prvok triedy `QSpinBox` pre zobrazenie a zmenu hodnoty a takisto aj tlačidlo triedy `QButton` pre štart simuláacie vplyvu tohto parametru.

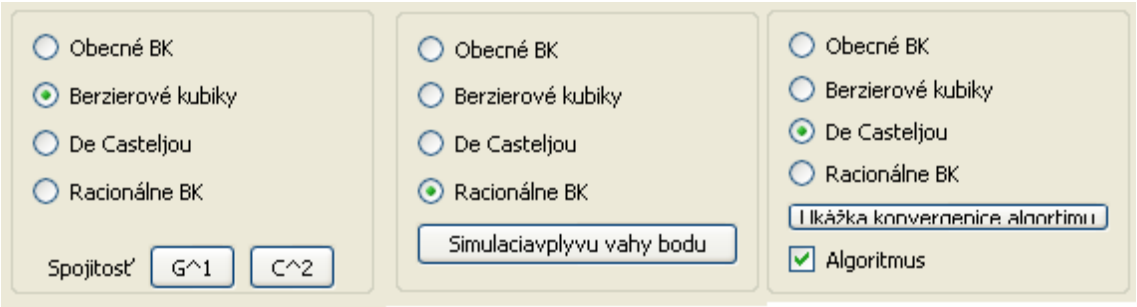


Obrázek 5.3: Parametre krivky Kochanek-Bartles



Obrázek 5.4: Parametre krivky Catmull-Rom

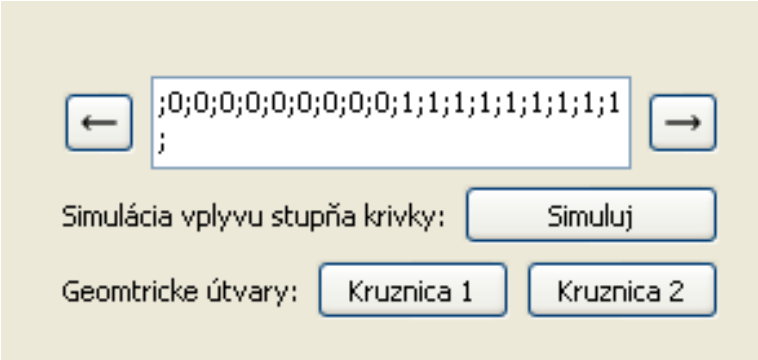
Pre krivku Catmull-Rom je použitý jeden prvok triedy `QCheckBox`, ktorý udáva či sa má konvexná obálka vykresliť alebo nie. Akýmkoľvek zmenením riadiacich bodov krivky sa vykresľovanie konvexnej obálky vypne.



Obrázek 5.5: Parametre Bezierových krviek

Pri Bezierových krivkách sú použité 4 prvky triedy `QRadioButton`, ktoré slúžia pre výber typu Bezierovej krivky. Ďalej obsahuje dve tlačidlá triedy `QButton`, ktoré sú viditelné iba pre Bezierové Kubické krviky a slúžia pre demoštráciu spojitosti týchto krviek. Dalšie prvky a to konkrétne tlačidlo triedy `QButton` pre simuláciu konvergencie algorimu Decasteljau a prvok triedy `QCheckBox` určujúci zobrazenie algoritmu DeCateljau. Ako je už zrejmé tieto dva prvky sú viditelné iba pre algoritmus DeCasteljau. Posledným prvokom je tlačidlo triedy `QButton`, ktoré spúšta simuláaciu vplyvu váhy bodu na výsledný tvar krivky a je viditelné iba pre Racionálne Bezierové krivky.

Ako posledné si popíšeme parametre krviky NURBS. Tie obsahujú prvok triedy `QTextBrowser`, ktorý je iba na čítanie a zobrazuje uzlový vektor danej krivky. Z oboch strán tochto prvku je umi-



Obrázek 5.6: Parametre krivky NURBS

estnené jedno tlačidlo triedy `QButton`. Tieto tlačidlá menia vstupný vektor tak že ho zhusťujú na stranu kde su umiestnené aby tak ukázali vplyv tohto vektoru na výslednu krviku. Ďalej obsahuje ešte tri prvky typu `QButton`, jeden pre simuláaciu vplyvu stupňa krivky a dva pre demonštráciu spôsobu vykreslovania geometrických tvarov, konkrétne kružníc.

5.4 Testovanie

Program bol testovaný na operačných systémoch Microsoft Windows XP a Microsoft Windows Vista.

Pri testovaní boli najdené 2 chyby, ktoré sa mi nepodarilo odstrániť. Chybné dokročenie simulácie pri simulácii vplyvu tečného vektoru na Fergusnove krviky. Krátku dobu po ukončení simulácie a aktivovaní hlavného okna vedie akákolvek akcia spôsobená užívateľom ku neočakávanému správaniu a následnému pádu aplikácie. Pri obecných Bezierových krivkách bolo zistené nesprávne modelovanie krviek pri vätšom počte riadiacich bodov. Kedže táto chyba nastáva približne pri 14 a viac riadiacich bodoch nemusíme túto chybu brať v potaz pretože pri používaní programu vätšinou pracujeme s menším počtom riadiacich bodov.

Kapitola 6

Záver

Úkolom mojej bakalárskej práce bolo vytvoriť čo najjednoduchší a čo najprehľadnejší výkový program. Myslím že vďaka minimalizácii ovládacich prvkov a intuitívnemu ovládaniu sa mi podarilo vytvoriť program, s ktorým sa užívateľ ľahko stotožní a može si tak vyskúšať ako implementované krivky pracujú a zároveň pochopiť niektoré ich principy na použitých príkladoch, ako aj použiť tento program ako pomôcku pri vysvetľovaní princípu krviek niekomu inému. Či sa mi ale naozaj podarilo vytvoriť požadovaný program a splniť krteriá preňho dané ukáže až jeho používanie v praxi.

Pri ďalšom vývoji programu by určite bolo vhodné doimplementovať niektoré vňom neimplementované typy kriviek a algoritmov pre ich vykreslenie, ako aj sa zamerať na niektoré tu nedemonštrované vlastnosti jednotlivých krivke a algoritmov, poprípade nájsť sposob pre rošírenie programu do tretej dimenzie a tým ukázať aj princípy vytvárania ploch pomocov trojrozmerných spline krviek. Ďalším smerom ktorým by sa mohol program uberať je pridanie rozhranie pre zobrazovanie teorie o danej krivke, čiže by sa užívateľ mohol hneď popri demonštrácii vzdelávať aj v teórii kriviek.

Náš výukový program, spoločne s ostatnými výukovými programmy, ktoré sú v tomto školskom roku vyvýjané v rámci predmetu Bakalárska práca ostatnými študentamy by mohli slúžiť ako ineraktívna učiteľská pomocka a tým významnov mierov prispieť k skvalitneniu výuky na našej fakulte ako aj pomocť ostatným študentom pri samovýuke.

Literatura

[1] Using Qt 4.0 with Visual Studio 2005. [online].
URL http://www.telldus.se/qt/tutorial.pdf

[2] Geometrie/Úvod do křivek. [online], septeber 2008.
URL http://cs.wikibooks.org/wiki/Geometrie/vod_do_kivek

[3] Microsoft Visual Studio. [online], december 2008.
URL http://en.wikipedia.org/wiki/Microsoft_Visual_Studio

[4] Qt - Wikipedia, the free encyclopedia. [online], deceber 2008.
URL http://en.wikipedia.org/wiki/Qt_(toolkit)

[5] C++. [online], január 2009.
URL http://en.wikipedia.org/wiki/C++

[6] Výukový program. [online], november 2009.
URL http://cs.wikipedia.org/wiki/Vukov_program

[7] Žara J., F. P., Beneš B.: *Moderná počítačová grafika. 1. vyd.* Praha, Computer press, 1998, ISBN 80-7726-049-9.