Univerzita Komenského
v Bratislave

Matematicko-fyzikálna fakulta Katedra informatiky

# DIPLOMOVÁ PRÁCA

## Objektovo orientovaná knižnica objektov CAGD

Diplomant: Marek Kolinský
Vedúci diplomovej práce: RNDr. Pavel Chalmovianský

Bratislava 2000

## ČESTNÉ VYHLÁSENIE

Čestne vyhlasujem, že diplomovú prácu som vypracoval samostatne a že som uviedol všetku použitú literatúru.

V Bratislave, 16. 4. 2000

.......................
Marek Kolinský

Touto cestou vyslovujem poďakovanie **RNDr. Pavlovi Chalmovianskému** za pomoc, odborné vedenie, cenné rady a pripomienky pri vypracovaní mojej diplomovej práce.

V Bratislave dňa 18.4.2000

# Obsah

# 1 Úvod

Výskum v oblasti počítačovej grafiky a špeciálne v geometrickom modelovaní si vyžaduje jednak teoretický výskum, rovnako ako aj praktickú implementáciu metód a algoritmov. V tejto oblasti cítiť silnú potrebu univerzálnej platformy, nad ktorou môžu byť testované rôzne metódy, nový vývoj a ich výsledky vzájomne porovnávané bez zbytočnej námahy a straty času. Medzi mnohými aspektami (rýchlosť, numerická zložitosť, ...) je veľmi dôležitým aspektom aj kvalita výsledného produktu. Vizualizačné nástroje a metódy môžu byť veľmi nápomocné pri estetickom posudzovaní, prípadne pri testovaní kvality geometrického objektu.

## 1.1 Ciele práce

Cieľom tejto práce je vytvorenie objektovo orientovanej knižnice geometrických objektov, špeciálne kriviek a plôch.

V časti venovanej softvérovým knižniciam (sekcia 2) si preto ozrejmíme pojem knižnica v súvislosti s programovaním, spomenieme si využitie knižníc pri vývoji aplikácií a vlastnosti, ktoré by mala dobrá knižnica spĺňať. Špeciálne sa budeme venovať knižniciam z oblasti počítačovej grafiky a uvedieme knižnicami podporované typy geometrických objektov. V závere tejto časti sa budeme venovať významu našej knižnice pre programátorov grafických aplikácií a popíšeme jej hlavné vlastnosti.

Dôležitým aspektom pri vývoji aplikácií je správny výber programovacieho jazyka, prípadne vývojového prostredia. Dôraz sme kládli na „užívateľský komfort“, rozšíriteľnosť, programátorskú obľubu, prípadne aj nezávislosť od platformy. Dôležitým faktom je samozrejme aj efektivita výsledného produktu. Knižnica by mala slúžiť akademickej obci, študentom hlavne pre testovanie starých, prípadne nových typov kriviek a plôch. Z toho vyplýva aj požiadavka na prehľadnosť zdrojového kódu. My sme sa rozhodli pre programovací jazyk C++. V ďalšej časti (sekcia 3.1) sa teda budeme venovať tomuto jazyku. Vysvetlíme, prečo sme sa rozhodli práve pre tento typ programovacieho jazyka a jeho výhody, prípadne nevýhody pre vývoj grafickej knižnice.

Dôležitou technológiou použitou pri vývoji knižnice je objektovo orientované programovanie. Táto téma je veľmi komplexná, my sa preto budeme venovať iba základnému popisu princípov a vlastností (sekcia 3.2) . Dôraz je kladený hlavne na „otvorenosť“ systému, čiže na možnosť knižnicu jednoducho modifikovať a rozširovať.

V nasledujúcej časti (sekcia 4) si popíšeme základné geometrické pojmy, postupy a definície, ktoré sú nevyhnutné pre pochopenie metód a algoritmov, na ktorých je knižnica postavená.

Pre potreby tejto práce je nevyhnutné zoradiť krivky a plochy do kategórií. Takejto kategorizácii sa venuje ďalšia časť našej práce (sekcia 5). Kategórie potom budú priamym východiskom pre konštrukciu knižnice. Práca samozrejme z rôznych dôvodov neimplementuje všetky kategórie kriviek a plôch. V tejto časti sa teda dozviete, prečo je daná kategória do knižnice zahrnutá, prípadne prečo tomu tak nie je.

Dôležitou vlastnosťou našej knižnice je implementácia prvkov diferenciálnej geometrie, určených na posúdenie kvalitatívnych geometrických vlastností kriviek a plôch. V nasledujúcej časti (sekcia 6) sú preto tieto prvky popísané. Možno tu nájsť aj spôsob, ako konkrétne je daný prvok pužiteľný na spomenuté posudzovanie. Opäť sú popísané a integrované len často používané prvky, so zreteľom na posudzovanie kategórií kriviek a plôch v knižnici implementovaných.

Posledná, kľúčová časť (sekcia 7) sa venuje samotnej hierarchii geometrických objektov. Hierarchia presne popisuje triedy použité v objektovom návrhu našej knižnice. Je nutné si uvedomiť, že implementácia konkrétnych typov kriviek a plôch slúži hlavne ako praktický príklad použitia knižnice a nie je cieľom práce. Napriek tomu sme sa snažili implementovať čo možno najviac typov, minimálne jeden pre každú abstraktnú triedu.

# 2 Softvérové knižnice a ich využitie

Pojem knižnica sa vo výpočtovej technike používa na označenie množiny dát a algoritmov, zväčša vytvorených pre potreby riešenia nejakého konkrétneho (špecializovaného) problému. Väčšina programovacích jazykov a vývojových prostredí priamo podporuje tvorbu a využívanie takýchto knižníc. To vedie k uľahčeniu práce vývojárom, skráteniu času potrebného pre vývoj a k možnosti rozdeliť vývoj do viacerých modulov realizovaných príslušnou knižnicou, prípadne použiteľnou aj v ďalších aplikáciách. V neposlednom rade knižnice obsahujú algoritmy, ktoré už prešli fázou ladenia, testovania a optimalizácie a poskytujú teda algoritmy vhodné k okamžitému použitiu s možnosťou vynechania tejto fázy.

Mnohé z knižníc získajú vďaka svojim vlastnostiam, dostupnosťou a komlexnosťou pohľadu na danú problematiku obľubu u programátorov, čo vedie k ich masovému rozšíreniu. V prípade niektorých z nich sa môžu dokonca dočkať aj hardwarovej podpory (OpenGL), prípadne implementácie priamo do operačného systému (Direct X vo Win32).

Knižnice môžeme podľa použitých technológií rozdeliť do mnohých kategórií. Takýmito kategorizačnými prvkami môže byť programovací jazyk použitý pri jej vývoji, platforma pre ktorú je knižnica určená, druh licencie, cena alebo fakt či je knižnica dodávaná aj so zdrojovými kódmi.

Preferovanou technológiou v oblasti vývoja aplikácií v poslednej dobe je objektovo orientované programovanie (OOP), čím vzniká ďalší segment knižníc využívajúcich túto technológiu. Práve knižnice využívajúce princípy OOP sú pre prirodzený pohľad na objekt veľmi obľúbené. Dôležitým faktorom je, že aplikácie využívajúce OOP sú pomalšie ako aplikácie vytvorené klasickým spôsobom. Keďže primárnym cieľom drvivej väčšiny grafických aplikácií je časová optimálnosť, sú vývojári často nútení vyhnúť sa tejto technológii.

Za veľký problém by sme však mohli považovať cenovú dostupnosť takýchto knižníc. Väčšina dobrých grafických knižníc bola totiž vyvinutá na komerčnej báze, je teda pochopiteľné, že nie sú dostupné zdarma (týka sa to hlavne knižníc pre prostredie Win32). To je ale veľkou prekážkou ich využitia, napríklad v školstve.

Na opačnej strane stoja akademické projekty univerzít, prípadne jednotlivcov, ktoré sa venujú konkrétnejšiemu problému na vysokej kvalitatívnej úrovni.

Nami vytvorená knižnica je charakteristická práve objektovým prístupom, voľnou šíriteľnosťou pre nekomerčné využitie, dostupnosťou zdrojových kódov a možnou nezávislosťou na platforme.

## 2.1 Knižnice v počítačovej grafike

Rovnako ako v ostatných oblastiach informatiky sa knižnice úspešne využívajú aj v počítačovej grafike. Môžeme teda nájsť knižnice špecializované na zobrazovanie grafickej informácie, na riešenie špecializovaných problémov až po knižnice snažiace sa o čo možno najkomplexnejšie poňatie celej problematiky. Väčšina z nich však komplexne nepopisuje problematiku okolo kriviek a plôch.

Knižníc obsahujúcich algoritmy na vyčíslovanie kriviek a plôch je relatívne málo, niektoré dôležité si spomenieme.

- **LEDA**[1],je knižnica geometrických objektov a algoritmov konštruovaná tak, aby bola zrozumiteľná aj pre laika. Podporuje mnoho typov geometrických objektov, ale implementuje iba základné typy kriviek a plôch (Bézierove a B-splinové krivky).

- **OpenGL**[2], je knižnica primárne zameraná na kvalitné zobrazovanie grafickej informácie. Dôležitou charakteristikou OpenGL je, že nevyužíva objektovo orientovaný prístup a podobne ako LEDA implementuje iba základné typy kriviek a plôch (B-splinové krivky a plochy).

## 2.2 Praktické využitie knižnice

Ako sme už spomínali, knižnica je primárne určená pre akademické, vzdelávacie účely na školách venujúcich sa výučbe počítačovej grafiky. Konkrétne je navrhnutá pre vyvíjanie nových typov kriviek, plôch a oboznamovanie sa so známymi algoritmami pre vyčíslovanie a analýzu kriviek a plôch.

V prípade vývoja nových typov je užívateľ nútený implementovať iba samotný algoritmus na vyčíslenie geometrického objektu. Potom sú mu okamžite dostupné najrôznejšie algoritmy na prácu s jeho objektami, algoritmy vstupu a výstupu, ale hlavne algoritmy na zistenie geometrických vlastností tohto objektu. Tie sú veľmi potrebné pri jeho následnom testovaní a posudzovaní. Tento postup oslobodzuje užívateľa od únavnej práce pri implementácii týchto algoritmov a umožňuje mu sústrediť sa iba na skúmaný objekt.

[1]LEDA = Library of Efficient Data types and Algorithms

[2]OpenGL = Open Graphics Library

Inak môže byť naša knižnica využitá v každom type grafickej, štatistickej alebo inej aplikácie, ktorá využíva objekty ako krivka a plocha. Spomenúť môžeme napríklad využitie interpolačných kriviek pri vytváraní spojitého grafu pre namerané diskrétne hodnoty alebo využitie knižnice pri modelovaní.

# 3 Implementácia

V úvode sme spomenuli, že knižnica je vytvorená v prostredí C++ s využitím objektovo orientovaného programovania (OOP[3]). V tejto časti sa budeme venovať popisu základných vlastností C++ a OOP (práca [vvM94]).

Vyčíslené hodnoty je potrebné aj nejakým spôsobom zobraziť, preto uvedieme aj spôsob zobrazovania grafickej informácie na grafický výstup v našej knižnici.

## 3.1 Programovací jazyk C++

Programovací jazyk C++ vznikol ako rozšírenie jazyka C implementáciou postupov a pojmov objektovo orientovaného programovania. Záujem o jazyk C++ vyplýva jednak z jeho kladných vlastností spoločných s jazykom C, t.j. predovšetkým z úspornosti výrazov, bohatstva operátorov, kompaktnosti kódu a zo širokého okruhu aplikovateľnosti, tak z rozšírenia jeho možností implementáciou objektového programovania, ktoré umožňuje jednoduchší vývoj nových a rozvoj už hotových programov alebo programových systémov a zjednodušuje postup modelovania reálneho sveta pri využití prvkov objektového programovania.

Rovnako ako C, nie je ani C++ viazaný k určitému typu počítača alebo operačného systému a je použiteľný pre programy numerického charakteru a programy na spracovanie textu rovnako dobre, ako pre programy hromadných dát, pre písanie operačných systémov, prekladačov alebo pre vývoj náročných grafických aplikácií. Jazyk C++ je jazyk na relatívne nízkej úrovni, avšak vďaka mnohým knižniciam, dovoľuje operácie v širokom rozsahu, od operácií veľmi vysokej úrovne až po operácie blízke prostriedkom assembleru s jednoduchou prenositeľnosťou na iné operačné systémy alebo počítače iných architektúr. To ho predurčuje byť veľmi vhodným jazykom pre potreby vývoja univerzálnych knižníc. Programy v C++ sú zpravidla natoľko efektíne, že nie je treba písať ich ekvivalenty alebo ich časti v assembleri, čo je veľmi dôležité pre vývoj výkonnostne náročných grafických aplikácií.

Jazyk C++ samozrejme neumožnuje naprogramovať efekty, ktoré by nebolo možné vytvorit v C. Jazyk C++ však umožňuje použitie iných myšlienkových postupov poskytujúcich uživateľský komfort z hľadiska analýzy a realizácie projektu a rozširuje a uľahčuje možnosti nového využitia kompilo-

[3]OOP – z anglického Object Oriented Programming

vaného kódu.

## 3.2 Objektovo orientované programovanie v C++

Základnou myšlienkou objektovo orientovaného programovania je spojenie dát s prostriedkami pre manipuláciu s nimi, t.j. určitými algoritmami špecifickými práve pre priradené dáta. V objektovo orientovanom programovaní sa takéto algoritmy nazývajú metódy (methods). V jazyku C++ je toto spojenie dát s funkciami realizované zavedením nového druhu objektu, dátového typu trieda (class). V terminológii C++ sa položky takejto zjednodušenej štruktúry nazývajú členy (members) triedy. Tie členy, ktoré sú dátami sa nazývajú členské dáta (member data) tie, ktoré sú funkciami sa obdobne nazývajú členské funkcie (member function). Kombinácia štruktúry dát s funkciami, ktoré s týmito dátami manipulujú prináša prvú charakteristickú vlastnosť objektového programovania - uzavretosť (encapsulation).

Druhou základnou myšlienkou objektového programovania je zavedenie takej štruktúry dátových typov v programe, ktorá by zodpovedala všeobecnej klasifikácii problematiky reálneho sveta, v programe modelovanej. Klasifikácia pojmov a objektov označenými týmito pojmami vo vedách popisujúcich reálny svet zpravidla vychádza z presne vymedzeného, avšak veľmi všeobecného pojmu. Tento pojem zahŕňa značné množstvo konkrétnych objektov, ktoré majú pojmom definované vlastnosti spoločné, ale každý z nich má mnoho ďalších vlasností, ktoré už nie sú charakteristické pre všetky ostatné objekty označené týmto pojmom. A práve tieto špecifické vlastnosti vedú k ďalším pojmom odvodeným od tohto pojmu, ktoré rozdeľujú pôvodným pojmom vymedzenú skupinu na niekoľko ďalších podskupín. Všetky objekty všetkých podskupín spĺňajú kritériá pôvodnej skupiny určenej pôvodným objektom. Objekty priradené každej skupine majú naviac vlastnosti, ktoré charakterizujú túto podskupinu. Všetky takto vzniknuté podskupiny možno analogicky členiť ďalej na menšie a menšie podskupiny, ktorých objekty majú stále viac spoločných vlastností. Táto druhá vlastnosť objektového programovania sa nazýva dedičnosť (inheritance) - odvodená trieda dedí členy základnej triedy.

Treťou charakteristickou vlastnosťou objektového progarmovania je polymorfizmus (polymorphism), ktorý spočíva v tom, že je možné využit rovnaké meno pre akcie zdieľané nahor a nadol v hierarchii tried, pričom v každej z týchto tried môže byť s týmto menom spojená akcia špecifická práve pre túto triedu a teda pre túto triedu primeraná. Táto vlastnosť je realizovaná prostredníctvom prekrývania (overload) a presmerovania (override) funkcií a symbolov. Rôzne funkcie môžu byť označené rovnakým identifikátorom a rôzne operácie rôznym symbolom. Zo všetkých rovnako označených je vybraný správny podľa toho, s akými argumentami pracuje.

Objektovo orientované programovanie je vďaka týmto vlastnostiam programovacia metóda, ktorá zjednodušuje modelovanie reálnych situácií a približuje metódy a postupy modelovania reálnym situáciám. Objektové programovanie disponuje veľkou možnosťou zovšeobecnenia, triedenia a abstrakcie pri modelovaní zložitých reálnych situácií a umožňuje prirodzenú tendenciu klasifikácie abstraktných pojmov zdieľania určitej spoločnej vlastnosti alebo správania sa prostredníctvom tried. To dáva možnosť rozvíjať a realizovať myšlienkové postupy na všeobecnej úrovni bez toho, aby boli viazané na detaily súvisiace s jedným príliš konkrétnym pojmom.

Jazyk C++ poskytuje všetky prostriedky pre realizáciu objektovo orientovaného programovania, čo umožňuje jednak väčšie možnosti v konštrukcii štruktúry programu, riadení jeho chodu a jeho modularite, jednak možnosť vytvárať nové datové typy s ich vlastnými operátormi a naviac poskytuje nástroje na vytváranie znovupoužiteľného kódu programu. Všetky tieto možnosti prispievajú k vytvoreniu zdrojového kódu, ktorý je vo väčšej miere štrukturovaný, rozšíriteľný a ľahšie udržovateľný.

## 3.3 Grafický systém

Pre potreby zobrazovania grafickej informácie na grafický výstup je nutné implementovať množstvo algoritmov a dátových typov. My sme preto uprednostnili možnosť využiť grafický subsystém OpenGL. OpenGL používa na popis grafických objektov iné dátové typy ako naša knižnica, knižnica preto disponuje metódami na prevod grafických údajov do tohto systému.

Knižnica poskytuje užívateľovi iba základné možnosti grafického výstupu, ktoré sú nevyhnutné pre zobrazovnie. Keďže je však tento subsystém založený na OOP, je možné knižnicu rozšíriť o ľubovolné ďalšie metódy.

# 4 Všeobecné geometrické pojmy a definície

Pre bližšie pochopenie teórie geometrických objektov je nutné objasniť si niektoré základné geometrické pojmy. Geometrické objekty v tejto práci sú totiž postavené na geometrických pojmoch a my s nimi budeme často narábať. Táto práca sa nevenuje detailom, snaží sa popísať iba pojmy a prakticky využívané postupy. Bližšie informácie môžete nájsť v práci [McC94].

## 4.1 Body a vektory

Označíme body, elementy trojrozmerného Euklidovského priestoru $\mathbf{E}^n$ písmenami ako $a$, $b$ atď. Rovnakú notáciu použijeme aj pre vektory, prvky trojrozmerného lineárneho (alebo vektorového) priestoru $\mathbf{R}^n$. Hoci body aj vektory sú reprezentované trojicou reálnych čísel zdôrazňujeme, že je medzi nimi striktný rozdiel. Pre každé dva body $a$, $b$ existuje práve jeden vektor $v$, pre ktorý platí:

$$v = b - a; \qquad a, b \in \mathbf{E}^n, \qquad v \in \mathbf{R}^n.$$

Na druhej strane pre daný vektor $v$ existuje nekonečne veľa párov $a$, $b$ takých, že $v = b - a$.

Priradenie bodu $a+v$ každému bodu $a \in \mathbf{E}^n$ sa nazýva posunutie. Vektory sú invariantné k posunutiu, zatiaľ čo body nie.

## 4.2 Barycentrická kombinácia

Predpokladajme $n$-rozmerný Euklidovský priestor $\mathbf{E}^n$ pre ľubovolné $n$.

**Definícia 4.1** *Simplex v $\mathbf{E}^n$ je $(n+1)$-tica bodov neležiacich v lineárnej variete dimenzie $(n-1)$*

**Definícia 4.2** *Nech $a_0, \ldots, a_n = \Sigma$ je ľubovolný simplex v $\mathbf{E}^n$. Potom každý bod $x \in \mathbf{E}^n$ sa dá jednoznačne vyjadriť v tvare $x = \alpha_0 a_0 + \ldots + \alpha_n a_n$, kde $\alpha_0 + \ldots + \alpha_n = 1$. Potom $(\alpha_0, \ldots, \alpha_n)$ sa nazýva barycentrickými súradnicami bodu $x$ vzľadom na $\Sigma$.*

Na prvý pohľad to vyzerá ako nedefinovaná suma bodov, ale ak ju zapíšeme ako

$$b = b_0 + \sum_{j=0}^{n} \alpha_j (b_j - b_0), \qquad \text{kde } \alpha_0 + \ldots + \alpha_n = 1$$

$$b = \sum_{j=0}^{n} \alpha_j (b_j - b_0), \qquad \text{kde } \alpha_0 + \ldots + \alpha_n = 0$$

dostávame korektný zápis súčtu vektorov a bodu, resp. iba súčtu vektorov. Vektor hodnôt $\alpha_i$ sa nazýva aj váhový vektor. Spomedzi barycentrických kombinácií zaujíma špeciálne postavenie konvexná kombinácia. Je to barycentrická kombinácia, v ktorej je každý koeficient $\alpha_i$ nezáporný. Konvexná kombinácia bodov je bod, ktorý leží v konvexnom obale týchto bodov.

Barycentrická kombinácia je kľúčovým pojmom tejto práce, pretože všetky algoritmy na vyčísľovanie kriviek a plôch ju využívajú.

## 4.3 Afinné zobrazenia

Afinné zobrazenia patria k najčastejšie používaným zobrazeniam v počítačovej grafike. Vyplýva to z mnohých výhodných vlastností tohto typu zobrazení.

**Definícia 4.3** *Zobrazenie $\Phi : \mathbf{E}^n \to \mathbf{E}^n$ sa nazýva afinné, ak zachováva barycentrické kombinácie.*

To znamená, že ak

$$x = \sum \alpha_j a_j; \qquad x, a_j \in \mathbf{E}^n, \qquad \sum \alpha_j = 1$$

a $\Phi$ je afinné zobrazenie, potom platí

$$\Phi(x) = \sum \alpha_j \Phi(a_j); \qquad \Phi(x), \Phi(a_j) \in \mathbf{E}^n$$

## 4.4 Homogénne afinné súradnice

Nech $p = (x_1, \ldots, x_n)$ sú afinné súradnice bodu $p$ v $\mathbf{E}^n$. Homogénne afinné súradnice bodu $p$ dostaneme rozšírením klasických súradníc o prvok $w$, nazývaný váha, nasledovne:

$$(x_1, \ldots, x_n) \to (x_1 : \ldots : x_n : 0), \qquad \text{pre vektory}$$

$$(x_1, \ldots, x_n) \to (x_1 : \ldots : x_n : w); w \neq 0, \qquad \text{pre body}$$

**Definícia 4.4** *$p = (x_1 : \ldots : x_n : w \neq 0)$ sú homogénne afinné súradnice bodu $p \in \mathbf{E}^n$ práve vtedy, ak $p = (\frac{x_1}{w}, \ldots, \frac{x_n}{w})$ sú jeho klasické afinné súradnice.*

V našej knižnici sú body a vektory realizované práve týmto spôsobom[4], s obmedzením na trojrozmerný euklidovský priestor $mathbfE^3(n = 3)$. Váhy sa prakticky využívajú na rozlíšenie bodov od vektorov.

## 4.5 Lineárna interpolácia

**Definícia 4.5** *Lineárna interpolácia bodov* $a, b \neq a$ *je zobrazenie* $L : \langle 0, 1\rangle \rightarrow \langle a, b\rangle$ *také, že* $p = p(t) = (1 - t)a + tb$, *kde* $t \in \mathbf{R}^n$

Pre lineárnu kombináciu platia tieto základné pravidlá:

- $t$ je barycentrickou kombináciou 0 a 1
- $x$ je barycentrickou kombináciou bodov $a$ a $b$
- Lineárna interpolácia zachováva barycentrické kombinácie

Každé tri body, ktoré ležia na jednej priamke sa nazývajú kolineárne. Pre hodnotu parametra $t = 0$ lineárna intepolácia zodpovedá bodu $a$, pre hodnotu $t = 1$ bodu $b$. Ak $0 \leq t \leq 1$, získané body ležia na úsečke $ab$.

S lineárnou interpoláciou je úzko spojený aj pojem barycentrických súradníc. Nech $a$, $x$, $b$ sú tri kolineárne body v $\mathbf{E}^n$:

$$x = \alpha a + \beta b; \qquad \alpha + \beta = 1.$$

Potom $\alpha$ a $\beta$ s nazývajú barycentrické súradnice bodu $x$ vzhľadom na body $a$, $b$. Prepojenie medzi lineárnou interpoláciou a barycentrickými súradnicami je zrejmé: $\alpha = 1-t$ a $\beta = t$. Pre ľubovolné tri kolineárne body $a$, $b$, $c$, môžeme barycentrické súradnice bodu $b$ vzhľadom na body $a$ a $c$ vyjadriť nasledovne:

$$\alpha = \frac{vol_1(b, c)}{vol_1(a, c)}$$

$$\beta = \frac{vol_1(a, b)}{vol_1(a, c)}$$

kde $vol_1$ označuje orientovanú vzdialenosť dvoch bodov[5].

S touto témou súvisí aj pojem bilineárnej interpolácie.

---

[4]Váha bodu je implicitne 1.

[5]závisí od poradia: $vol_1(a, b) = -vol_1(b, a)$

**Definícia 4.6** *Nech $V_{00}$, $V_{01}$, $V_{10}$, $V_{11}$ sú štyri rôzne po troch nekolineárne body priestoru $\mathbf{E}^n$. Bilineárnou interpoláciou týchto bodov rozumieme plochu:*

$$X(u,v) = (1-u,u)\begin{pmatrix} V_{00} & V_{01} \\ V_{10} & V_{11} \end{pmatrix}\begin{pmatrix} 1-v \\ v \end{pmatrix}$$

Táto plocha sa nazýva hyperbolický paraboloid.

## 4.6 Ratio, deliaci pomer

Iným dôležitým pojmom v tejto časti je ratio. Ratio troch kolineárnych bodov $a$, $b$, $c$ je definované ako:

$$ratio(a,b,c) = \frac{vol_1(a,b)}{vol_1(b,c)} = \frac{\beta}{\alpha}$$

Obdobne je deliaci pomer bodu $c$ vzhľadom na body $a$ a $b$ definovaný ako:

$$del(a,b,c) = \frac{vol_1(a,c)}{vol_1(b,c)} = -ratio(a,c,b)$$

## 4.7 Čiastková lineárna interpolácia

Nech $b_0, \ldots, b_n \in \mathbf{E}^n$ tvorí polygón $B$. Polygón pozostáva z postupnosti priamkových segmentov, ktoré interpolujú dvojice bodov $b_i, b_{i+1}$. To sa nazýva lineárny interpolant bodu $b_i$. Ak body $b_i$ ležia na krivke $c$, potom sa $B$ nazýva čiastkový lineárny interpolant ku krivke $c$ a zapisuje sa

$$B = PLc.$$

Jedna z najdôležitejších vlastností čiastkovej lineárnej interpolácie ja afinná invariantnosť: ak je krivka $c$ zobrazená na krivku $\Phi(c)$ afinným zobrazením $\Phi$, potom čiastková lineárna interpolácia krivky $\Phi(c)$ je afinným zobrazením originálneho lineárneho interpolantu:

$$PL\Phi(c) = \Phi(PLc).$$

Čiastková lineárna interpolácia je veľmi dôležitý postup v počítačovej grafike,v našej knižnici je použitá na zobrazovanie kriviek a plôch.

## 4.8 Parametrické krivky

Parametrické krivky patria v počítačovej grafike k veľmi často používaným geometrickým objektom. Vlastnosti a kategórie parametrických kriviek si vysvetlíme v nasledujúcich častiach (sekcia 5). Tu sa budeme venovať iba uvedeniu základných pojmov.

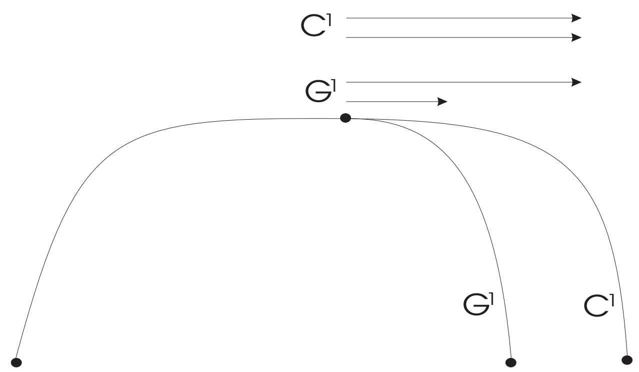
**Definícia 4.7** *Parametrická, diferencovateľná krivka je zobrazenie,* $C : (a, b) \to \mathbf{R}^n$*, pre nejaké* $-\infty \leq a < b \leq \infty$*, také že* $C(t)$ *má derivácie všetkých rádov.*

Krivka sa nazýva regulárna, ak $\frac{\partial C}{\partial t} \neq \vec{0}$ pre všetky $t \in (a, b)$. Taká krivka môže byť zapísaná v súradnicovom tvare ako:

$$C(t) = (x_1(t), \ldots, x_n(t)), \qquad \frac{\partial C}{\partial t} = (x'_1(t), \ldots, x'_n(t))$$

## 4.9 Spojitosť parametrických kriviek

Hovoríme, že parametrická krivka $p(t)$ je z triedy $C^n$, ak má vo všetkých bodoch spojité derivácie až do rádu $n$. Táto spojitosť sa nazýva aj parametrická spojitosť.




Obr. 1: Príklad $C^1$ a $G^1$ spojitej krivky

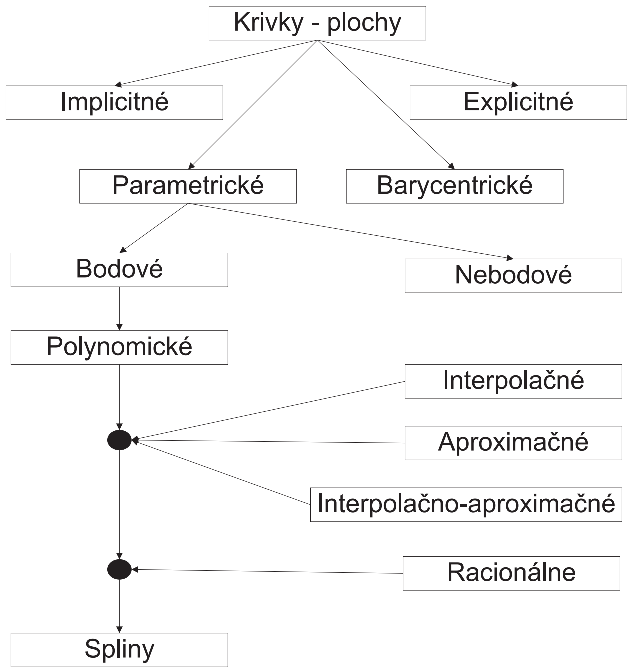
Ak spojíme dve krivky s nadväzujúcimi definičnými obormi, dostaneme krivku zloženú z dvoch častí, nazývaných segmenty. Dva segmenty sú spojito nadviazané ($C^0$ spojitosť), ak sú posledný bod prvého segmentu a prvý bod druhého segmentu totožné. Podobne majú dva segmenty spojenie triedy $C^1$, ak majú v dotykových bodoch totožné vektory prvej derivácie. Analogicky vedie totožnosť druhej, prvej derivácie a dotykového bodu k spojitosti tried $C^2$ atd.

Tieto podmienky sú veľmi silné, a preto sa pre potreby geometrického modelovania zaviedol pojem geometrickej spojitosti $G^n$. Podmienky geometrickej spojitosti sú veľmi podobné spojitosti parametrickej. Podmienky spojitosti $G^0$ sú rovnaké, ako pre $C^0$, pre spojitosť $G^1$ sa namiesto totožnosti vektorov derivácií požaduje iba totožnosť ich smerov a orientácií, nie však veľkostí.

# 5 Kategórie kriviek a plôch

Krivky a plochy zadeľujeme do kategórií podľa mnohých kritérií (práca [FR95]). Každá skupina sa pritom vyznačuje vlastnosťami, čo ju predurčuje k nasadeniu na riešenie špecifického problému. Táto práca sa venuje len kategóriám, ktoré úzko súvisia s tvorbou knižnice a snaží sa o vysvetlenie, prečo a aké z nich sú pre našu potrebu najvhodnejšie.

Obrázok 5 schématicky znázorňuje rozdelenie kriviek a plôch do kategórií, ktoré je použité v tejto práci.




Obr. 2: Kategórie kriviek a plôch

## 5.1 Kategórie kriviek

Krivky patria v počítačovej grafike k veľmi často používaným geometrickým objektom. S krivkami sa môžeme stretnúť napríklad pri definícii fontov, pri popise dráhy pohybujúceho sa objektu v počítačovej animácii, pri definícii – modelovaní objektov, ale napríklad aj v matematickej štatistike. Každá z týchto kategórií má špecifické požiadavky na typy prípadne konkrétne vlastnosti kriviek.

V tejto časti si popíšeme kategórie kriviek a popíšeme, prečo a načo sú jednotlivé kategórie vhodné alebo nevhodné.

### 5.1.1 Analytické rozdelenie

Z analytického hľadiska rovinné krivky rozdeľujeme na krivky zadané:

- explicitne v tvare $y = f(x)$
- implicitne $F(x, y) = 0$
- parametricky $x = x(t), y = y(t)$
- barycentricky $x = x(u, v), y = y(u, v);\ u + v = 1$

Každá z týchto kategórií je vhodná pre iné typy aplikácií a disponuje rozličnými vlastnosťami. Z definícií vyplýva, že explicitné krivky sú špeciálnym prípadom kriviek parametrických (stačí zvoliť $x(t) = t$).

Parametrické krivky (sekcia 4.8), sa chápu ako zobrazenie nejakého parametrického priestoru (najčastejšie intervalu) do priestoru. Je teda jednoduché použiť na ich zobrazenie čiastkovú interpoláciu polynómom (sekcia 4.7). Z parametrického priestoru si podľa nejakého kritéria vyberieme postupnosť bodov a v týchto vyčíslime hodnotu krivky. Výsledná postupnosť vyčíslených bodov tvorí interpolačný polygón danej krivky. V našej knižnici použijeme uniformné rozdelenie parametrického priestoru – vyčíslujeme v cykle so zvoleným krokom. Ďalšou výhodou tohto typu kriviek je, že dáva jednotiaci pohľad na rovinné ako aj priestorové parametrické krivky[6].

Barycentricky definované krivky úzko súvisia s pojmami barycentrická kombinácia a barycentrické súradnice. Definičným oborom je interval, určený dvojicou bodov $a$, $b$. Každý bod na tejto úsečke s barycentrickými súradnicami $u$, $v$, $u + v = 1$ sa zobrazí na bod so súradnicami $x(u, v)$, $y(u, v)$. Každá takto definovaná krivka sa dá jednoducho zapísať v parametrickom tvare, preto sa týmto typom kriviek ďalej nebudeme zaoberať.

Krivky zadané implicitne nie sú veľmi vhodné pre použitie v počítačovej grafike pre ich problémy súvisiace s ich zobrazovaním. Je tu síce možnosť previesť implicitnú krivku v okolí zobrazovaného bodu na krivku parametrickú,

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

tento postup nám však pomôže iba pri vyčíslovaní v konkrétnom bode. Tieto typy kriviek sú však vhodné pri hľadaní prienikov s inými geometrickými objektami.

Implicitne definovaným krivkám sa v tejto práci nebudeme ďalej venovať a všetky kategórie spomenuté nižšie sú kategóriami kriviek parametrických.

### 5.1.2 Bodové a nebodové krivky

Z hľadiska geometrického spôsobu definície sa krivky rozdeľujú na krivky:

- bodové krivky, ktoré na svoje definovanie používajú postupnosť bodov nazývanú tiež riadiaci polygón
- nebodové krivky

Pre potreby modelovania sú samozrejme vhodnejšie krivky bodové, pretože zmena tvaru krivky sa dá docieliť zmenou riadiaceho polygónu bez nutnosti zmeny vyhodnocovacej funkcie. To sa využíva pri modelovaní tak, že okrem samotnej krivky sa zobrazí aj jej riadiaci polygón a umožní sa manipulácia s jeho bodmi.

V našej knižnici a ďalších kategóriach sa budeme venovať iba bodovým krivkám.

### 5.1.3 Polynomické krivky

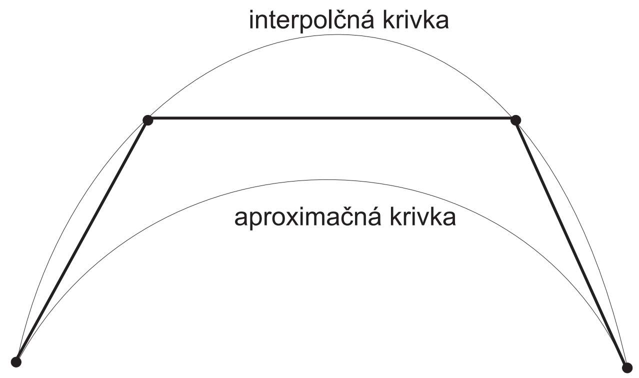
Z hľadiska použitých funkcií pri definícii kriviek zaujímajú dôležitú úlohu polynomické krivky. Výhoda použitia polynómov spočíva v jednoduchosti operácií s nimi (derivovanie, integrovanie, ...). Stupeň polynómu je pritom dôležitým faktorom ovplyvňujúcim tvar a lokálne správanie sa krivky. Bodové krivky druhého stupňa ležia totižto v jednej rovine, čo vedie k problémom pri ich hladkom nadväzovaní. Krivky tretieho stupňa už takéto vlastnosti nemajú a preto sú najrozšírenejším typom polynomických kriviek. Krivky vyšších rádov sa využívajú len sporadicky, pretože sú náchylnejšie k nepriaznivým osciláciám.

V počítačovej grafike sa používajú krivky hlavne na modelovanie, preto sa viac ako iné využívajú polynomické krivky tretieho prípadne druhého stupňa. Naša knižnica a ďalšie kategórie sa preto venujú výhradne týmto krivkám.

### 5.1.4 Interpolačné a aproximačné krivky

Podľa spôsobu, ako sa bodové krivky správajú v závislosti od riadiaceho polygónu, môžeme krivky rozdeliť na krivky:

- interpolačné – krivka prechádza každým bodom riadiaceho polygónu - krivka ich interpoluje
- aproximačné – využívajú riadiaci polygón len na riadenie ich tvaru, samotné body však krivke nemusia patriť – krivka ich aproximuje
- interpolačno - aproximačné – interpolujú časť svojich riadiacich vrcholov, ostatné aproximujú.




Obr. 3: Interpolačná a aproximačná krivka

Interpolačné krivky sa používajú v prípadoch, keď je nevyhnutné, aby krivka prechádzala postupnosťou vopred zadaných bodov (napr. v matematickej štatistike pri výpočte derivácií, určitých integrálov, ...). Na prípadné riadenie tvaru krivky sú však okrem interpolovaných bodov nutné aj ďalšie informácie (napríklad hodnoty vektorov prvých derivácií pre každý bod riadiaceho polygónu).

Na druhej strane sú krivky aproximačné, ktoré sú flexibilnejšie pri potrebe zmeny ich tvaru, ale dopredu nie je známe, kadiaľ krivka povedie. Preto sa pri ich návrhu často postupuje metódou pokusu a omylu (iteratívna metóda), ktorou sa návrhár postupnosťou krokov dopracuje k výslednej krivke.

Tieto kategórie nemajú charakteristiky, ktoré by ich jednoznačne určovali a ktoré sú pre všetky krivky danej kategórie rovnaké. Inak povedané triedy nemajú atribúty, ktoré by nám umožňovali, definovať tieto kategórie kriviek z objektovo orientovaného hľadiska. To, či je krivka aproximačná alebo interpolačná, závisí iba od jej definície.

### 5.1.5 Racionálne krivky

Vynájdením tohto typu kriviek sa modelovanie posunulo výrazne dopredu, pretože je nimi možné modelovať aj kuželosečky ako sú kružnica a elipsa.

Body riadiaceho polygónu sú reprezentované svojimi homogénnymi súradnicami (sekcia 4.4). Tento typ kriviek využíva poslednú súradnicu na vyjadrenie, ako sa krivka v tomto bode správa voči tomuto bodu. Často sa hovorí aj o tom, ako sa krivka k danému bodu „priťahuje“. To dáva návrhárovi ďalšiu možnosť ako meniť tvar výslednej krivky.

Algoritmy na vyčíslenie týchto kriviek využívajú, ako sme už spomínali, barycentrické kombinácie (sekcia 4.2). Výsledný bod teda musí mať poslednú súradnicu rovnú jednej. Preto sú tieto algoritmy charakteristické normovaním tejto súradnice.

Tento typ kriviek opäť nie je charakteristický atribútmi, ktoré by ho odlišovali od klasických typov. Jediný rozdiel je v tom, že algoritmus aktívne pracuje aj z hodnotami váh.

### 5.1.6 Splinové krivky

Často je vyčíslovanie kriviek založené na segmentácii, čiže rozdelení krivky na segmenty. Segmenty sú zväčša krivky tretieho alebo druhého stupňa, krivky sa preto zvyknú nazývať aj po častiach kubické alebo kvadratické krivky. Tieto segmenty sa potom nadväzujú niektorým z typov parametrických, prípadne geometrických spojitostí (sekcia 4.9). Takéto typy kriviek nazývame splinovými krivkami.

Splinové krivky sú krivky, ktorých definičný interval je rozdelený postupnosťou čísel, nazývanou aj uzlový vektor. Dôležitým faktom je, že voľba uzlového vektora ovplyvňuje tvar výslednej krivky, tento postup je preto aj často využívaný. Na základe uzlového vektora, môžeme hovoriť o uniformných splinoch – využívajú uniformnú postupnosť[7] a neuniformných splinoch. Unifromné spliny sú jednoduchšie, pretože uzlový vektor je pevne daný a užívateľ ho nemusí špecifikovať. Potom však nemá možnosť uzlový vektor meniť. My

[7]Uniformná postupnosť je postupnosť $t_i$, kde pre všetky $i$ platí $t_{i+1} = t_i + h$ pre nejaké $h \geq 0$

sme sa zamerali na neuniformné spliny, ale umožnili sme uživateľovi vygenerovať uniformný uzlový vektor.

## 5.2 Kategórie plôch

Plochy sa v poslednej dobe stali v počítačovej grafike veľmi populárne. So zvyšovaním výkonnosti hardwaru sa začali používať aj typy plôch, ktoré boli pre svoje obrovské výpočtové nároky odsunuté do úzadia.

### 5.2.1 Analytické rozdelenie

Podobne ako krivky, zadeľujeme aj plochy do štyroch kategórií:

- explicitné plochy: $z = f(x, y)$
- implicitné plochy: $F(x, y, z) = 0$
- parametrické plochy: $x = x(u, v), y = y(u, v), z = z(u, v)$
- barycentrické plochy: $x = x(u, v, w), y = y(u, v, w), z = z(u, v, w)$

Z rovnakých dôvodov ako u kriviek, sa nebudeme v tejto práci zaoberať explicitnými a implicitnými plochami.

Parametrické plochy sú podobne ako krivky obrazom parametrického priestoru. Parametrický priestor tu však môže byť veľmi komplikovaný. Pre potreby tejto práce sa stačí obmedziť na jeden interval pre parameter $u$ a jeden interval pre parameter $v$. Na zobrazovanie takto definovanej plochy potom môžeme použiť podobný postup ako pri zobrazovaní kriviek. To vedie k aproximácii plochy sieťou štvoruholníkov – bilineárnych interpolantov.

Na rozdiel od barycentrických kriviek, sú barycentrické plochy prakticky využívané a to hlavne pri trojuholníkových záplatách. Definičným oborom je trojuholník určený tromi lineárne nezávislými bodmi $a$, $b$, $c$. Každý bod tohto definičného oboru sa potom dá vyjadriť barycentrickými súradnicami $u$, $v$, $w$, podobne ako u kriviek.

### 5.2.2 Bodové a nebodové plochy

Aj plochy môžeme zadeliť do kategórií podľa toho, či využívajú pre svoju definíciu body alebo nie. Kategórie teda sú:

- bodové plochy, ktoré na svoje definovanie používajú sieť bodov nazývanú tiež riadiaca sieť
- nebodové plochy

Riadiaca sieť je množina $n \times m$ bodov. Podobne ako u kriviek, aj u plôch sa zmenou bodu riadiacej siete mení tvar výslednej plochy.

Nebodové plochy sú na rozdiel od kriviek využívané podstatne častejšie. Je to v dôsledku Coonsových plôch, ktoré sú definované nad dvomi, štyrmi krivkami, prípadne nad sieťou kriviek. Tieto plochy sú v modelovaní často používané a budeme sa nimi zaoberať neskôr.

### 5.2.3 Interpolačné a aproximačné bodové plochy

Rovnako ako krivky, aj bodové plochy rozdeľujeme na

- interpolačné plochy
- aproximačné plochy
- interpolačno-aproximačné plochy

Interpolačné plochy sa objavujú v praxi všade tam, kde máme k dispozícii systém bodov alebo kriviek, ktoré hladaná plocha obsahuje. Okrem toho možno na interpolačné plochy klásť doplňujúce požiadavky, ako je predpísaná hodnota vektorov parciálnych derivácií, apod.

Teória aproximačných a interpolačno-aproximačných plôch v geometrickom modelovaní je založená na podobných základoch ako teória príslušných kriviek (práce [vBF98], [Far90]).

### 5.2.4 Plochy tenzorového súčinu

Plochy tenzorového súčinu si získali popularitu medzi návrhármi z dôvodu jednoduchej konštrukcie a jednoduchého spôsobu modelovania. Východiskom konštrukcií týchto plôch je bilineárna interpolácia.

Plochu tenzorového súčinu chápeme ako množinu všetkých bodov, ktoré patria všetkým možným polohám krivky pohybujúcej sa v priestore, pričom počas tohto pohybu pripúšťame aj zmenu tvaru tejto krivky.

### 5.2.5 Racionálne plochy

Racionálne plochy sú postavené na rovnakých princípoch, ako racionálne krivky (práca [Far90]).

### 5.2.6 Coonsove záplaty

Autormi týchto typov plôch sú američania Coons a Gordon. Namiesto riadiacich bodov vychádzajú zo siete kriviek, ktorú potom vypĺňajú plochou. Táto sieť sa nazýva digitalizáciou kriviek nakreslených na modele plochy. Z tohto popisu vyplýva, že sa jedná o interpolačné plochy. Coonsove záplaty rozdeľujeme na:

- priamkové záplaty
- bilineárne stmeľované záplaty
- čiastkovo kubicky stmeľované záplaty
- bikubicky stmeľované záplaty
- čiastkové Coonsove záplaty

Prvými štyrmi typmi sa budeme podrobnejšie zaoberať neskôr.

Čiastkové Coonsove plochy sú definované nad sieťou pretínajucich sa kriviek a našou úlohou je zostrojiť $C^1$-hladkú plochu. Aby sme to mohli aplikovať potrebujeme zostrojiť twist vektory $X_{uv}(u_i, v_j)$ pre každý bod siete. Potom môžeme zostrojiť vektory derivácií v priečnom smere ku každej hraničnej krivke. Tento typ plochy nebudeme v našej knižnici implementovať.

### 5.2.7 Translačné plochy

Podobným spôsobom ako Coonsove plochy sú definované aj translačné plochy. Sú určené dvomi krivkami, ktoré sa pretínajú v spoločnom bode.

### 5.2.8 Gordonove plochy

Tieto plochy sú zovšeobecnením Coonsovych záplat na celú sieť kompatibilných[8] kriviek. Predpokladajme, že máme danú sieť kriviek. Potrebujeme

[8]krivky s rovnakým parametrom a definičným oborom $\langle 0, 1\rangle$

zostrojiť plochu $G$, ktorá interpoluje všetky tieto krivky v tom zmysle, že sa stanú jej izoparametrickými krivkami. Konštrukcia vychádza z rovnakých princípov ako pri Coonsovych záplatách. Najskôr sa zostrojí plocha $G_1$, ktorá interpoluje jednu sústavu kriviek a podobne plocha $G_2$, ktorá interpoluje druhú sústavu. Ako stmeľovacie funkcie sa použijú Lagrangeove polynómy. Plochu potom možno vyjadriť v tvare:

$$G = G_1 + G_2 - G_{12},$$

kde

$$G_1(u,v) = \sum_{i=0}^{m} G(u_i, v) L_i^m(u) \qquad \& \qquad G_2(u,v) = \sum_{j=0}^{n} G(u, v_j) L_j^m(v)$$

$$G_{12}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} G(u_i, v_j) L_i^m(u) L_j^n(v)$$

# 6 Pojmy diferenciálnej geometrie

Pri konštrukcii hladkých a „dobre vyzerajúcich“ kriviek a plôch je dôležité aplikovať metódy pre posudzovanie ich kvality. Krivky a plochy majú vlastnosti, ktoré je tažké pochopiť z čisto symbolického alebo numerického popisu. Preto je vizualizácia nepostrádateľným nástrojom kontroly kvality a posúdenia estetických vlastností geometrického objektu. Rovnaké techniky môžu byť podobne použité pri vizualizácii vyčíslených hodnôt nových algoritmov a matematických schém.

Jedna z najjednoduchších metód je zobrazenie vytieňovaného objektu z ľubovolného, najlepšie meniteľného uhla pohľadu. Z neho je možné získať prvý dojem z povrchu zobrazeného objektu. Pre detailnejšiu analýzu však sú potrebné oveľa sofistikovanejšie vizualizačné nástroje - metódy. Existuje mnoho spôsobov, ako to urobiť, tu sa budeme venovať najčastejšie používaným z nich.

## 6.1 Numerické metódy

Pretože všeobecný algoritmus na výpočet derivácií v danom bode neexistuje (máme k dispozícii iba algoritmus na vyčíslenie krivky), ponúka numerická matematika možnosť aproximovať krivku v okolitých bodoch inou krivkou a využiť derivácie tejto krivky na aproximáciu hladanej derivácie. Tento postup sa nazýva lokálna interpolácia. V praxi sa na tento účel veľmi často používa Lagrangeov interpolačný polynóm (sekcia 7.7.2). Keďže budeme potrebovať prvú až tretiu deriváciu, použijeme polynóm tretieho stupňa. Ak za interpolované body zvolíme $x_0 = x - 3h$, $x_1 = x - h$, $x_2 = x + h$, $x_3 = x + 3h$ a interpolovanú funkciu označíme $f$ dostávame nasledujúcu interpoláciu:

$$\begin{aligned}
f'(x) &= \tfrac{1}{48h}(f(x-3h) - 27f(x-h) + 27f(x+h) - f(x+3h)) \\
f''(x) &= \tfrac{1}{8h^2}(f(x-3h) - f(x-h) - f(x+h) + f(x+3h)) \\
f'''(x) &= -\tfrac{1}{8h^3}(f(x-3h) - 3f(x-h) + 3f(x+h) - f(x+3h)).
\end{aligned}$$

Parameter $h$ je voliteľný a jeho voľbou je možné ovplyvniť presnosť výslednej aproximácie. Uspokojivé výsledky sa dajú dosiahnuť voľbou $h = 10^{-4} \times$ (priemer parametrického priestoru).

Samozrejme tento postup nevedie k presným výsledkom, a preto je vhodné v prípade, ak je známy algoritmus na priame vyčíslenie derivácií, použiť tento algoritmus prepísaním príslušnej metódy. Inak nastáva problém, ak chceme použiť derivácie vyšších rádov ako tri, pretože tieto algoritmy nie sú implementované.

Presnejšie informácie možno nájsť v práci [NvV76].

## 6.2 Krivky

### 6.2.1 Dotyčnica, normála a binormála

Základnými prvkami diferenciálnej geometrie používanými na posúdenie kvality geometrických objektov sú:

- normalizovaný smerový vektor dotyčnice $t = \frac{f'(t)}{|f'(t)|}$
- binormála $b = \frac{f'(t) \times f''(t)}{|f'(t) \times f''(t)|}$
- normála $n = b \times t$

Základným prvkom pre posúdenie lokálneho správania sa krivky, je vyčíslenie a spracovanie hodnoty prvej derivácie. Veď dotykový vektor v danom bode je nepostrádateľným atribútom pri hladkom nadväzovaní segmentov krivky. Pre istý stupeň hladkosti je samozrejme nevyhnutné poznať všetky derivácie až do príslušného stupňa. Následným porovnávaním potom získavame podrobnejšie informácie pre posúdenie lokálneho správania sa krivky. Napríklad, veľká zmena smeru a veľkosti dotykového vektora prvej derivácie v bodoch od seba málo vzdialených dáva dôkaz o nižšom stupni hladkosti krivky v daných bodoch (kedže našim cieľom býva dosiahnutie maximálnej hladkosti).

### 6.2.2 Frenetov repér

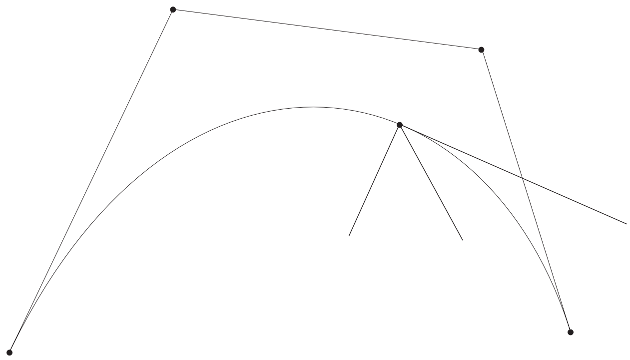
Frenetov repér (práca [WJ98]) je objektom, ktorý spája predchádzajúce tri objekty do jedného pojmu. Jeho označenie je:

$$(t, n, b)$$

Je to vlastne trojica ortonormálnych vektorov[9] (obr. 6.2.2).

Animovaním Frenetovho repéra pozdĺž skúmanej krivky dostávame základnú vizuálnu informáciu o kvalite tejto krivky. Táto metóda je teda jednou z najjednoduchších ale aj najpoužívanejších metód.

[9] ortonormálne vektory = na seba kolmé vektory s dĺžkou 1



Obr. 4: Frenetov repér

### 6.2.3 Krivosť a oskulačná kružnica

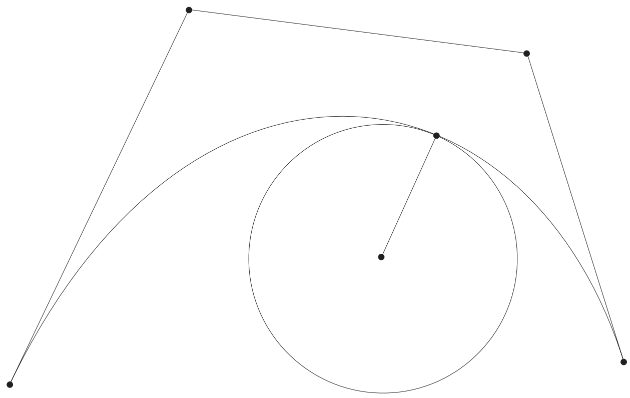
Omnoho zaujímavejším prvkom diferenciálnej geometrie je krivosť:

$$\kappa(t) = \frac{|f'(t) \times f''(t)|}{|f'(s)|^3}$$

Krivosť krivky v danom bode môže byť zobrazená mnohými spôsobmi. Často sa vizualizuje ako kružnica krivosti, nazývaná aj oskulačná kružnica (obr. 6.2.3). Táto kružnica leží v rovine určenej dotyčnicou $t$ a hlavnou normálou $n$, pričom má polomer $\frac{1}{\kappa}$ a dotýka sa krivky v bode, kde sa krivosť vyčíslovala. Táto metóda sa používa v prípade, ak je potrebné posudzovať krivosť pre konkrétny parameter $t$.

V prípade, že chceme vyčísliť krivosť vo viacerých bodoch naraz je použitie oskulačnej kružnice neprehľadné. V takom prípade sa používa animovanie oskulačnej kružnice pozdĺž krivky alebo diagram krivosti (curvature plot). Ide o dvojrozmerný graf, ktorý zobrazuje krivosť nad oblúkom krivky. Vizualizovať ho môžeme ako množinu úsečiek, z ktorých každá je definovaná nad príslušným parametrom a spája bod krivky so stredom oskulačnej kružnice. Samozrejme nemôžeme vyčísliť krivosť v každom bode definičného oboru, a preto krivosť zobrazujeme iba vo vybraných bodoch (napr. s určitým, zvoleným krokom).

Za pozornosť stojí aj spôsob, keď je krivosť zobrazená farebným odtieňom príslušného bodu krivky pri jej zobrazovaní. Táto metóda nedáva presné informácie o krivosti, ale je veľmi vhodná v prípade, ak je aj diagram krivosti neprehľadný (napr. hustá sieť kriviek).



Obr. 5: Oskulačná kružnica

### 6.2.4 Torzia

Často používaným prvkom pri analýze geometrického objektu je torzia:

$$\tau(t) = \frac{f'''(t)(f'(t) \times f''(t))}{|f'(t) \times f''(t)|^2}$$

Je najmenej používaná v počítačovej grafike a uplatnenie nájde v prípadoch, keď je nutné detailne poznať geometrické vlastnosti skúmaného objektu. Podobne ako krivosť, môžeme torziu vizualizovať ako valec, ktorého os je rovnobežná s binormálou $b$ a ktorého polomer je $\frac{1}{\tau}$.

## 6.3 Plochy

### 6.3.1 Dotyčnice, normála

Podobne ako u kriviek, základnými prvkami diferenciálnej geometrie sú

- dotyčnica v smere $u$: $S_u = \frac{\partial S}{\partial u}$
- dotyčnica v smere $v$: $S_v = \frac{\partial S}{\partial v}$
- normála plochy: $n = \frac{S_u \times S_v}{|S_u \times S_v|}$

### 6.3.2 Krivosť

Pre potreby geometrickej analýzy plôch sme implementovali dva typy krivosti plôch

- Gaussova krivosť $K = \frac{LN-M^2}{EG-F^2}$
- priemerná krivosť $H = \frac{GL-2MF+EN}{2(EG-F^2)}$.

Ak $n$ je normála plochy a $S_{uu}$, $S_{uv}$, $S_{vv}$ sú vektory druhých parciálnych derivácií, potom

$$E = S_u.S_u, \qquad F = S_u.S_v, \qquad G = S_v.S_v$$

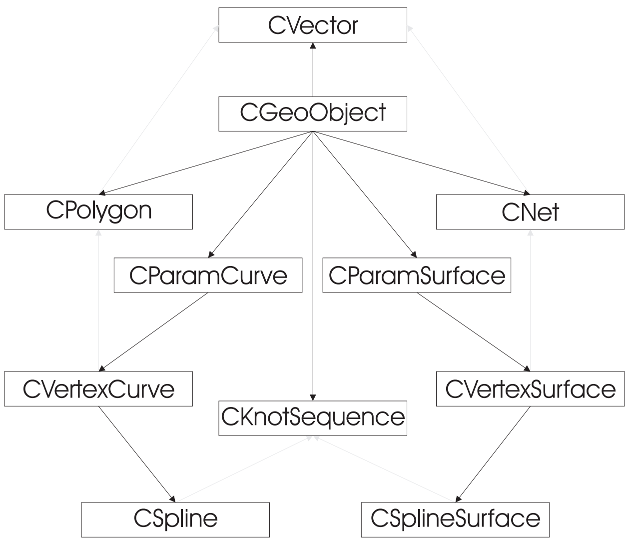
$$L = n.S_{uu}, \qquad M = n.S_{uv}, \qquad N = n.S_{vv}.$$

# 7 Hierarchia tried

Jadrom tejto práce je nesporne hierarchia geometrických objektov. Z nich najpodstatnejšími sú samozrejme triedy kriviek a plôch. Ostatné triedy sú podporné, napomáhajú sprehľadneniu štruktúry a my sa nimi v tejto práci nebudeme zaoberať.

V našej hierarchii sme sa zamerali na rozumnú dekompozíciu tried až na triedy, ktoré zavádzajú nový typ geometrického objektu. Tento postup vplýva sice negatívne na efektivitu výsledného produktu, ale na druhej strane sú tieto triedy prehľadné a algoritmicky jednoduché.

Rozdelenie kriviek a plôch do kategórií v časti (sekcia 5) nebolo samoúčelné. Toto rozdelenie totižto veľmi úzko súvisí práve z hierarchiou kriviek a plôch v tejto práci a mnohé z nich ich presne popisujú. Základné rozdelenie je zobrazené na obrázku (7), kde je tmavými čiarami znázornené dedenie a šedé čiary znamenajú, že trieda obsahuje zástupcu tejto triedy.




Obr. 6: Hierarchia abstraktných tried

Triedy sú rozdelené na triedy abstraktné, ktoré sú podrobnejšie popísané a triedy konkrétnych typov kriviek a plôch. Tieto sú realizované iba predefinovaním metódy pre vyčíslenie svojich bodov, prípadne derivácií a ich popis teda obsahuje iba popis algoritmov týmito metódami realizovaný.

## 7.1 Geometrický objekt – CGeoObject

Trieda *CGeoObject* je primárnou triedou celej hierarchie. Popisuje základné vlastnosti a metódy, ktoré sú spoločné pre všetky geometrické objekty.

Obsahuje dve premenné *ClassName* a *ClassType*. Unikátna hodnota *ClassName* jednoznačne identifikuje triedu, zatiaľ čo *ClassType* identifikuje typ triedy z hľadiska spôsobu vyhodnocovania. Priamymi potomkami sú základné typy geometrických objektov ako vektor, polygón, parametrická krivka a plocha.

Trieda ako taká nedisponuje takmer žiadnou funkcionalitou, umožňuje však združovanie takých rozdielnych tried ako krivky a plochy do jedného objektu. Metódy tejto triedy slúžia hlavne na následné identifikovanie geometrického objektu.

Veľmi dôležitou metódou je metóda *Ready()*, ktorá indikuje či je objekt pripravený na vyčíslenie, zobrazenie a podobne. Každá z nových tried z tejto odvodených samozrejme predefinováva túto metódu, aby do testov zahrnula aj atribúty špecifické pre túto tiedu.

## 7.2 Vektor – CVector

Trieda *CVector* popisuje základný stavebný prvok, ktorý využívajú takmer všetky typy geometrických objektov. Táto trieda slúži na vytváranie bodov ako aj vektorov, kde bod je charakterizovaný svojim polohovým vektorom.

Každý vektor[10] je reprezentovaný homogénnymi súradnicami (sekcia 4.4), čiže štvoricou reálnych čísel $x$, $y$, $z$, $w$.

Je nutné si uvedomiť, že v dôsledku realizácie matematických operácií na počítači dochádza k numerickým nepresnostiam. V tejto tiede sa to prejavuje konkrétne na hodnote $w$. Je preto nutné zaviesť určitú prahovú hodnotu $\alpha$, a ak platí $-\alpha \leq w \leq \alpha$, tak sa jedná o vektor, inak sa jedná o bod. Hodnota $\alpha$ samozrejme závisí od numerickej chyby algoritmu, pre väčšinu prípadov postačuje hodnota $10^{-4}$. Táto hodnota je v našej knižnici použitá ako implicitná hodnota pre všetky algoritmy pravujúce s vektormi, užívateľ ju teda nemusí špecifikovať.

Implementovaná sada konštruktorov *CVector()*, *CVector(x, y, z)*, *CVector(x, y, z, w)* a *CVector(vector)* umožňuje vytvoriť vektory všetkými spôsobmi potrebnými v algoritmoch tejto knižnice. V prípade, že nie sú špecifikované súradnice $x, y, z$, je pre ich impicitnú hodnotu použitá hodnota 0. Pre

[10] Pojem „vektor“ budeme často používať pre jednotné pomenovanie bodov aj vektorov.

implicitnú hodnotu váhy $w$ je použitá hodnota 1. Posledný zo spomínaných konštruktorov umožňuje vytvoriť presnú kópiu parametra *vector*[11].

Metódy na testovanie typu bod/vektor sú *IsVector(tolerance)* a *IsVertex(tolerance)*. Parameter *tolerance* zodpovedá prahovej hodnote $\alpha$, ktorú sme už spomínali.

Dôležitou funkciou je funkcia na zistenie dĺžky vektora, ktorá je v tejto knižnici implementovaná metódou *Ln()*.

Výhodou C++ je možnosť predefinovať operátory klasických matematických operácií. To sme využili a vytvorili sadu operátorov pre prácu s vektormi. Sú k dispozícii operátory sčítania a odčítania vektorov, vektorového súčinu, skalárneho súčinu[12], násobenie a delenie vektora číslom až po operátor umožňujúci indexový prístup k prvkom.

## 7.3 Polygón – CPolygon

Pod pojmom polygón rozumieme usporiadanú množinu bodov v priestore.

V našej knižnici je teda polygón charakterizovaný poľom vektorov. Trieda bola navrhnutá spôsobom, ktorý umožňuje indexový prístup k jednotlivým prvkom. To samozrejme nepriaznivo vplýva na výslednú zložitosť algoritmov. Na druhej strane je tento prístup prirodzený a uživateľsky komfortný.

Trieda disponuje základnými metódami na pridávanie, editovanie a mazanie bodov, prípadne na spoluprácu s inými polygónmi. Tieto algoritmy sú dostupné vo forme metód, ako aj predefinovaných operátorov, čo opäť sprehľadňuje aplikačný kód.

## 7.4 Sieť bodov – CNet

Sieť bodov je usporiadaná množina $n \times m$ bodov v priestore, ktorá je realizovaná dvojrozmerným poľom.

Podobne, ako u polygónu, umožňuje knižnica indexový prístup k bodom siete a obsahuje všetky potrebné metódy na pridávanie, editovanie a mazanie bodov.

[11] takýto konštruktor sa nazýva kopírovcí konštruktor

[12] predefinovali sme operátor %

## 7.5 Parametrická krivka – CParamCurve

Táto trieda abstraktne popisuje vlastnosti spoločné pre všetky typy parametrických kriviek (sekcia 4.8).

Definičný obor krivky je interval $\langle t_{min}, t_{max}\rangle$, reprezentovaný dvojicou reálnych čísel. Tieto hodnoty nie sú priamo dostupné uživateľovi – sú privátne, preto sú implementované aj metódami pre prácu s nimi. Jedná sa o metódy *SetInterval(min, max)* na zmenu definičného intervalu a metódy *GetIntervalMin()* a *GetIntervalMax()* na zistenie príslušných hodnôt.

Veľký význam má metóda na testovanie, či je krivka definovaná nad zvoleným parametrom - *IsDefined(double t)*, čiže či patrí parameter do definičného intervalu alebo nie. Ďalšou dôležitou metódou je aj metóda *GlobalTransformParam(double t)* na transformáciu parametra, ktorá sa využíva v prípade, ak je krivka definovná na inom intervale, ale algoritmus na vyčíslenie vychádza z definičného intervalu $\langle 0, 1\rangle$.

Najdôležitejšou a najpodstatnejšou metódou je samozrejme metóda na vyčíslenie krivky - *Evalue(double t)* a metóda na vyčíslenie derivácií - *Derive(double t)*. Tieto metódy sú implementované postupmi spomenutými v kapitole o diferenciálnej geometrii (sekcia 6.1). Metóda *Evalue(double t)* je v mnohých prípadoch jediná metóda, ktorú je treba predefinovať v novovytvorených triedach parametrických kriviek. Ako sme už spomínali, v tejto práci sa stretneme výhradne s polynomickými krivkami a preto trieda obsahuje aj metódu na zistenie jej stupňa - *Degree()*.

Trieda ďalej obsahuje spomínané metódy na výpočet základných prvkov diferenciálnej geometrie. Metóda *Tangent(double t)* slúži na výpočet dotykového vektora v danom bode (aj normalizovanej), metóda *Binormal(double t)* slúži na vyčíslenie vektora binormály a metóda *Normal(double t)* slúži podobne na vyčíslenie normálového vektora. Implementované sú aj metódy na vyčíslenie krivosti - *Curvature(double t)* a torzie *Torsion(double t)*. Tieto objekty sú popísané v kapitole (sekcia 6.2).

Metóda *Ready()* je rozšírená o test, či je definičný interval korektne definovaný, teda či platí $t_{min} < t_{max}$.

## 7.6 Bodová krivka – CVertexCurve

Najpoužívanejšími typmi parametrických kriviek sú nesporne bodové krivky (sekcia 5.1.2). Táto trieda abstraktne popisuje práve tento typ kriviek.

Bodové krivky sú charakteristické tým, že na svoje vyčíslovanie používajú riadiaci polygón, čiže polygón riadiacich bodov. Trieda samotná pri svojom vzniku nevytvára žiadu inštanciu typu *CPolygon* ako riadiaci polygón, ani svoj riadiaci polygón pri zániku neuvoľňuje. Užívateľ je teda nútený tento objekt vytvoriť a krivke ho jednoducho priradiť. Tento spôsob má výhodu v tom, že jeden polygón môžeme priradiť viacerým krivkám (napríklad pri ich porovnávaní). Zmeny riadiaceho polygónu oboch kriviek sú potom realizované zmenou iba tohto polygónu.

Trieda teda disponuje metódou *SetControllPolygon(CPolygon)* na spomínané priradenie polygónu triede. Riadiaci polygón je privátnym atribútom a preto je pre mnohé algoritmy nevyhnutné získať jeho ukazovateľ. Na to slúži metóda *GetControllPolygon()*.

Stupeň bodovej krivky často závisí od počtu bodov riadiaceho polygónu, konkrétne stupeň = počet bodov - 1. Metóda *Degree()* preto implicitne vracia práve túto hodnotu.

Klasická metóda *Ready()* je rozšírená o test na existenciu radiaceho polygónu, keďže tento nemusí existovať.

### 7.6.1 Parametrická čiara – CParamLine

Parametrická čiara je jedným z najjednoduchších typov parametrických kriviek. Určená je dvojicou bodov $a$, $b$ a predpisom

$$p(t) = (1-t)a + tb.$$

Body $a$, $b$ sú privátne atribúty, trieda preto obsahuje metódy na získanie a nastavenie ich hodnôt *SetBegin()*, *SetEnd()* a *GetBegin()*, *GetEnd()*.

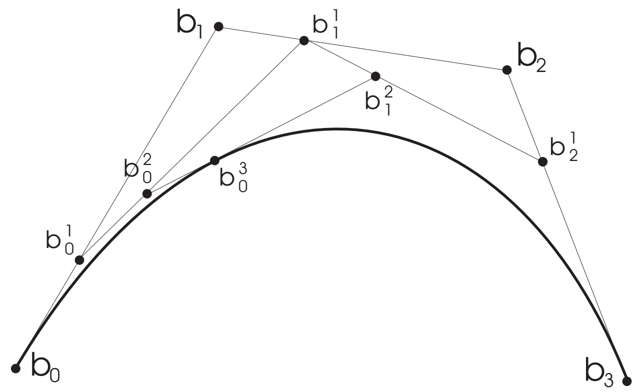
### 7.6.2 Bézierova krivka (de Casteljau) – CBezierCasteljau

Táto trieda bodových kriviek popisuje Bézierove krivky $n$-tého stupňa, ktoré používajú na vyčíslenie de Casteljauov algoritmus. Tento algoritmus patrí vôbec k najzákladnejším algoritmom v krivkovom a plošnom modelovaní.

Máme dané body $b_0, \ldots, b_n \in \mathbf{E}^3$ a $t \in \mathbf{R}$. Nech

$$b_i^r(t) = (1-t)b_i^{r-1}(t) + tb_{i+1}^{r-1}(t)$$

pre $r = 1, \ldots, n$ , $i = 0, \ldots, n-r$ a $b_i^0 = b_i$. Potom $b_0^n(t)$ je bod Bézierovej krivky zodpovedajúci parametru $t$. Uvedený postup sa nazýva de Casteljauov algoritmus, podľa jeho autora. Riadiaci polygón Bézierovej krivky sa




Obr. 7: De Casteljauov algoritmus pre $t = \frac{1}{3}$

nazýva aj Bézierovým polygónom, podobne jeho body sa nazývajú Bézierovými bodmi.

Pomocné koeficienty $b_i^r(t)$ sa zvyknú zapisovať do trojuholníkovej siete, nazývanej de Casteljuova schéma (obr. 7.6.2). Tu je príklad kubického prípadu:

$$\begin{array}{llll} b_0 & & & \\ b_1 & b_0^1 & & \\ b_2 & b_1^1 & b_0^2 & \\ b_3 & b_2^1 & b_1^2 & b_0^3. \end{array}$$

Pre potreby vyčíslenia však nie je potreba vytvárať takúto štruktúru, pretože algoritmus dokáže pracovať aj z jednorozmerným poľom dĺžky $n$, v ktorom postupne prepisuje príslušné hodnoty. Výsledok je nakoniec umiestnený v prvom prvku poľa.

Základné vlastnosti Bézierovych kriviek sú:

- Afinná invariantnosť – barycentrické kombinácie sú invariantné od afinných transformácií
- Krivka leží v konvexnom obale svojich riadiacich bodov
- Krivka prechádza svojimi koncovými bodmi
- Krivka je symetrická

- Zmena polohy bodu riadiaceho polygónu vedie k zmene tvaru celej krivky

### 7.6.3 Bézierova krivka (Bernstein) – CBezierBernstein

Iným, často používaným spôsobom na vyčíslenie Bézierovych kriviek je algoritmus využívajúci Bézierove krivky vyjadrené v Bernsteinovom tvare:

$$b^n(t) = \sum_{j=0}^{n} b_j B_j^n(t),$$

kde $B_j^n(t)$ sa nazývajú Bernsteinove polynómy n-tého stupňa a $b_j$ sú body riadiaceho polygónu.

Bernsteinove polynómy sú často používanými polynómami v počítačovej grafike a stretnúť sa s nimi môžeme aj pri iných typov kriviek ako sú Bezierove krivky. Bernsteinov polynóm stupňa $n$, s indexom $i$ je definovaný nasledovne:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

kde

$$\binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!} & \text{ak } 0 \le i \le n \\ 0 & \text{inak.} \end{cases}$$

Bernsteinove polynómy spĺňajú aj nasledovnú rekurzívnu vlastnosť, ktorá je často využívaná na ich vyčíslenie. Tento postup je použitý aj v tejto knižnici.

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t),$$

Na výpočet $n$-tej derivácie môžeme použiť algoritmus, ktorým nahradíme abstraktný algoritmus založený na lineárnej interpolácii (sekcia 4.5). Algoritmus realizuje nasledovný výpočet:

$$\frac{d^r}{dt^r} b^n(t) = \frac{n!}{(n-r)!} \sum_{j=0}^{n-r} \Delta^r b_j B_j^{n-r}(t),$$

kde

$$\Delta^r b_i = \sum_{j=0}^{r} \binom{r}{j} (-1)^{r-j} b_{i+j}.$$

### 7.6.4 Racionálna Bézierova krivka – CRationalBezier

Racionálna Bézierova krivka môže byť opäť vyčíslovaná aplikovaním modifikovaného de Casteljauovho algoritmu:

$$b_i^r(t) = (1-t)\frac{w_i^{r-1}}{w_i^r}b_i^{r-1} + t\frac{w_{i+1}^{r-1}}{w_i^r}b_{i+1}^{r-1},$$

kde

$$w_i^r(t) = (1-t)w_i^{r-1}(t) + tw_{i+1}^{r-1}t.$$

Inou často používanou metódou na vyčíslenie bodov racionálnej Bezierovej krivky, ktorá sa využíva aj v tejto knižnici je explicitná forma využívajúca Bernsteinove polynómy:

$$b^n(t) = \sum_{i=0}^{n} \frac{b_i w_i B_i^n(t)}{w_i B_i^n(t)}.$$

Vzťah je veľmi podobný vzťahu na vyčíslenie klasickej Bezierovej krivky – stačí, ak za každú z hodnôt $w_i$ dosadíme hodnotu 1.

Hodnoty $w_i$ sa často používajú ako tvarovacie parametre. Zvyšovaním hodnoty $w_i$ je krivka totižto viac priťahovaná k bodu $b_i$.

## 7.7 Splinové krivky – CSpline

Ako sme už spomínali, táto trieda obsahuje nový dátový typ – uzlový vektor. Uzlový vektor je postupnosť (vždy usporiadaná) reálnych čísel. Podobne, ako riadiaci polygón, ani uzlový vektor nie je dostupný po vytvorení tejto triedy – trieda ho nevytvára. Musí byť teda vytvorený mimo a triede priradený. Dôvody sú rovnaké ako pri riadiacom polygóne.

Keďže je počet bodov uzlovej postupnosti závislý na počte bodov riadiaceho polygónu, je impementovaná aj metóda na zistenie potrebného počtu uzlov v uzlovej postupnosti – *ExceptedNumberOfKnots()*. Na uľahčenie práce užívateľovi knižnica obsahuje aj metódu na vygenerovanie uniformného uzlového vektora *CreateUniformKnots()*. Užívateľ je teda oslobodený od jeho vytvárania a uzlový vektor môže modifikovať až v prípade potreby zmeniť ním výsledný tvar krivky.

Príslušnými zmenami prešla aj metóda *Ready()*, ktorá je rozšírená o test na existenciu uzlovej postupnosti, ako aj o test, či je počet jej členov rovný očakávanému počtu.

### 7.7.1 Aitkenov spline – CAitken

Častým problémom v modelovaní je bodová interpolácia bodov $p_i$ so zodpovedajúcimi hodnotami parametra $t_i$ krivkou, ktorá prechádza danými bodmi $p_i$. Nech

$$p_i^r(t) = \frac{t_{i+r} - t}{t_{i+r} - t_i} p_i^{r-1}(t) + \frac{t - t_i}{t_{i+r} - t_i} p_{i+1}^{r-1}(t),$$

kde $r = 1, \ldots, n$, $i = 0, \ldots, n-r$ a $p(t_i) = p_i$. Potom $p_0^n(t)$ zodpovedá bodu interpolačného polynómu prislúchajúceho parametru $t$. Tento algoritmus popisuje postup pri vyčíslovaní Aitkenovho splinu.

### 7.7.2 Lagrangeov spline – CLagrangeSpline

Aitkenov algoritmus nám umožňuje vyčísliť bod $p^n(t)$ na interpolovanej krivke cez $n+1$ bodov riadiaceho polynómu. Nedáva však odpoveď na otázku, či je interpolačná krivka jedinečná. Na túto otázku dávajú odpoveď Lagrangeove polynómy $L_i^n$.

Explicitná forma interpolačnej krivky je daná

$$p(t) = \sum_{i=0}^{n} p_i L_i^n(t),$$

kde $L_i^n(t)$ sú Lagrangeove bázové polynómy

$$L_i^n(t) = \frac{\prod_{j=0, j\neq i}^{n}(t - t_j)}{\prod_{j=0, j\neq i}^{n}(t_i - t_j)}.$$

### 7.7.3 Hermittov kubický spline – CHermitte3

Polynomická interpolácia nezávisí vždy iba od bodových dát, niekedy môže nastať potreba interpolovať aj iné informácie, ako napríklad derivácie. To vedie k interpolácii nazvanej Hermitova interpolácia. V tejto práci sa budeme zaoberať iba kubickým prípadom, ktorý interpoluje dva body $p_0$, $p_1$ a dva dotykové vektory $m_0$, $m_1$ v týchto bodoch.

Postup na vyčíslenie interpolačnej krivky je nasledovný:

$$p(t) = p_0 H_0^3(t) + m_0 H_1^3(t) + m_1 H_2^3(t) + p_1 H_3^3(t),$$

kde $H_i^3(t)$ sú Hermitove polynómy tretieho stupňa. Vyjadriť ich môžeme pomocou Bernsteinovych polynómov nasledovne

$$\begin{aligned}
H_0^3(t) &= B_0^3(t) + B_1^3(t) \\
H_1^3(t) &= \tfrac{1}{3}B_1^3(t) \\
H_2^3(t) &= -\tfrac{1}{3}B_2^3(t) \\
H_3^3(t) &= B_2^3(t) + B_3^3(t).
\end{aligned}$$

Hermitov interpolačný spline vytvoríme z dvoch segmentov tak, že stotožníme koncový bod $p_1$ a vektor derivácie $m_1$ prvého segmentu z prvým bodom $p_0$ a vektorom $m_0$ druhého segmentu. Z tejto konštrukcie vyplýva, že segmenty majú v spoločnom bode spoločný vektor prvej derivácie, spline je tam teda $C^1$ spojitý.

Pre jednoduchosť zadávania a spájania segmentov sa tento spline stal jedným z najpoužívanejších v počítačovej grafike.

### 7.7.4 Coonsov kubický spline – CCoons3

Jedná sa o aproximačný spline, ktorý využíva na vyčíslenie Coonsovu kubiku. Je určená štvoricou riadiacich bodov $p_0$, $p_1$, $p_2$, $p_3$ a vzťahom

$$p(t) = \frac{1}{6}(t^3, t^2, t, 1)\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix}\begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix} \tag{1}$$

Coonsova kubika na rozdiel od od ostatných kriviek spomínaných v tejto práci neprechádza krajnými bodmi svojho riadiaceho polygónu. Dosadením $t = 0$ a $t = 1$ do (1) zistíme, že krivka začína a končí v bodoch

$$p(0) = \frac{p_0 + 4p_1 + p_2}{6}$$

$$p(1) = \frac{p_1 + 4p_2 + p_3}{6}.$$

Tento nedostatok sa dá odstrániť vložením prvého a posledného bodu do riadiaceho polygónu tri krát. Dôležitou vlastnosťou vyplývajúcou z definície je totižto fakt, že krivka prechádza trojnásobnými bodmi.

Coonsov spline vznikne nadviazaním Coonsovych kubík (1) tak, že nasledujúci segment bude určený poslednými tromi bodmi predchádzajúceho segmentu a jedným novým. Coonsov spline, ktorý je určený $n \geq 4$ bodmi pozostáva teda z $n - 3$ segmentov. Podrobnejšie informácie o tomto type kriviek možno nájsť v práci [vBF98].

### 7.7.5 De Boorov B-Spline – CBSplineDeBoor

B-spliny pre štatistické vyhladzovanie dát odštartovali modernú teóriu splinovej aproximácie.

Opis krivky zložitého tvaru si vyžaduje krivku vysokého stupňa, čo nie je vhodné z numerických dôvodov. Preto vznikli splinové krivky, ktoré zobrazujú každý z intervalov $\langle t_i, t_{i+1} \rangle$ na polynomický krivkový segment.

Pre B-splinovú krivku je možné zmodifikovať de Casteljauov algoritmus, ktorý je známy ako de Boorov algoritmus. Je to algoritmus manipulujúci s riadiacim polygónom, ktorý využíva na vyčíslovanie bodov B-Splinovej krivky metódu nazvanú „algoritmus vkladania uzla“ (práca [Far90]).

Nech $u \in \langle u_I, u_{I+1}) \subset \langle u_{n-1}, u_{L+n-1})$, kde $I$ je maximálny index taký, že $u_I \le u \le u_{I+1}$ a $L$ je počet segmentov. Nech $d_i = d_i^0$ sú body riadiaceho polygónu. Definujeme

$$d_i^k = \frac{u_{i+n-k} - u}{u_{i+n-k} - u_{i-1}} d_{i-1}^{k-1}(u) + \frac{u - u_{i-1}}{u_{i+n-k} - u_{i-1}} d_i^{k-1}(u)$$

pre $k = 1, \dots n - r$ a $i = I - n + k + 1, \dots, I + 1$. Potom

$$b(u) = d_{i+1}^{n-r}.$$

### 7.7.6 Bázový B-Spline – CBSplineBase

B-splinová krivka využívajúca bázové funkcie je krivka určená stupňom $n$, riadiacimi vrcholmi $p_0, \dots, p_m$ a uzlovou postupnosťou $u_0, \dots, u_{L+2n-2}$, kde $L$ je potenciálny počet segmentov. Samotná krivka je definovaná na intervale $\langle u_{n-1}, u_{L+n-1} \rangle$. B-splinovú krivku stupňa $n$ potom definujeme:

$$p(t) = \sum_{i=0}^{m} p_i N_{i,n}(u).$$

Bázové funkcie $N_{i,n}(u)$ sú definované rekurzívne:

$$N_{i,n}(u) = \frac{u - u_i}{u_{i+n-1} - u_i} N_{i,n-1}(u) + \frac{u_{i+n} - u}{u_{i+n} - u_{i+1}} N_{i+1,n-1}(u)$$

$$N_{i,0}(t) = \begin{cases} 1 & u \in \langle u_i, u_{i+1}) \\ 0 & inak \end{cases}$$

### 7.7.7 Neuniformné racionálne B-spliny – CNURBS

Neuniformný racionálny B-spline (NURBS) je zovšeobecnením B-spline kriviek. Krivka je určená $n+1$ bodmi riadiaceho polygónu, uzlovým vektorom dĺžky $m+1$ a stupňom krivky $p$. Hodnoty uzlov musia byť samozrejme neklesajúce. Prvý a posledný bod riadiaceho polygónu má násobnosť $p+1$. Medzi počtom uzlov, stupňom krivky a počtom bodov riadiaceho polygónu platí vzťah

$$m = n + p + 1.$$

NURBS krivka je potom určená vzťahom

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

kde $w_i$ je váha $i$-teho bodu riadiaceho polygónu a $N_{i,p}(t)$ sú bázové B-splinové funkcie. Vo výpočte sa vyskytujú aj výrazy, ktoré majú v menovateli nulu. Ich hodnota je definitoricky rovná nule.

NURBS krivky majú tieto dôležité vlastnosti:

- Ak $w_i \geq 0; \forall i$, krivka leží v konvexnom obale svojho riadiaceho polygónu.
- Zmena polohy, resp. váhy má vplyv iba na časť krivky.
- Sú invariantné voči afinným transformáciám.
- Umožňujú presné vyjadrenie kužeľosečiek (kružnica, elipsa, parabola, hyperbola)

## 7.8 Parametrická plocha – CParamSurface

Trieda *CParamCurve* popisuje triedu všetkých typov parametrických plôch (sekcia 5.2.1).

Ako sme spomínali, parametrický priestor v tejto knižnici obmedzíme na obdĺžnik realizovaný dvojicou intervalov reálnych čísel $\langle u_{min}, u_{max}\rangle \times \langle v_{min}, v_{max}\rangle$. Podobne ako u kriviek, sú tieto atribúty privátne. Prístup k nim, nastavovanie a zisťovanie ich hodnôt, je teda realizovaný sadou metód *SetUInterval(umin, umax)*, *SetVInterval(vmin, vmax)*, *SetIntervals(umin, umax, vmin, vmax)* a metódami *GetUIntervalMin()*, *GetUIntervalMax()*, *GetVIntervalMin()*, *GetVIntervalMax()*.

Na zistenie, či je plocha definovaná nad dvojicou parametrov $u$, $v$ je vhodné použiť metódu *IsDefined(u, v)*. Funkcia na transformovanie parametrov do parametrického priestoru $\langle 0,1\rangle \times \langle 0,1\rangle$ je realizovaná dvojicou metód *GlobalTransformUParam(u)* a *GlobalTransformVParam(v)*.

Základnými metódami sú samozrejme metódy *Evalue(u, v)* na vyčíslenie krivky a *Derive(u, v, degu, degv)* na vyčíslenie parciálnych derivácií príslušných stupňov.

Pre potreby diferenciálnej geometrie sú implementované metódy na výpočet základných parciálnych derivácií *PartialU(u)*, *PartialV(v)* a metódu na výpočet normály *Normal(u, v)*. Z prvkov vyšších rádov sú implementované algoritmy na výpočet Gaussovej krivosti *GaussianCurvature(u, v)* a priemernej krivosti *MeanCurvature(u, v)*.

Aj táto trieda má implementovanú metódu *Ready()*, ktorá na tejto úrovni zisťuje, či je parametrický priestor korektne zadefinovaný podobne ako u kriviek (s príslušným rozšírením o ďalší parameter).

### 7.8.1 Coonsova priamková plocha – CLinearCoons

Priamkové plochy sú základom plošného designu. Riešia problém: sú dané dve kompatibilné priestorové krivky $C_1$ a $C_2$. Treba nájsť plochu $X$, ktorá má dané krivky za protiľahlé hraničné krivky. Hľadáme teda plochu $X(u,v)$ tak, aby

$$X(u,0) = C_1(u) \qquad \& \qquad X(u,1) = C_2(u), \qquad \text{pre } \forall u \in \langle 0,1\rangle.$$

Riešení je samozrejme nekonečne veľa, tu je jedno z najjednoduchších:

$$X(u,v) = (1-v)C_1(u) + vC_2(u).$$

Priamkové plochy pripomínajú lineárnu interpoláciu, každá izoparametrická čiara $u = const$ je úsečka. Teraz však neinterpolujeme body, ale celé krivky.

### 7.8.2 Coonsova bilineárne stmeľovaná plocha – CBilinearCoons

Bilineárne stmeľované plochy sú určené ľubovolnými štyrmi krivkami $C_1$, $C_2$ definované na intervale $\langle 0,1\rangle$ a $D_1$, $D_2$ definované na intervale $\langle 0,1\rangle$ so spoločnými rohovými bodmi. Pri riešení využijeme nasledujúce priamkové plochy:

$$R_c(u,v) = (1-v)C_1(u) + vC_2(u) \qquad R_d(u,v) = (1-u)D_1(v) + uD_2(v).$$

Výslednú plochu potom dostaneme ako:

$$X = R_c + R_d - R_{cd}$$

kde $R_{cd}$ je hyperboloický paraboloid určený rohovými bodmi.

### 7.8.3 Coonsova čiastkovo bikubicky stmeľovaná plocha – CPart-BicubicCoons

Čiastkovo bikubicky stmeľované plochy sa využívajú v prípade, ak chceme $C^1$ hladko spájať viacero záplat. Hladkosť je pritom dosiahnutá tým, že namiesto stmeľovacích funkcií $(1-u)$, $(u)$, resp. $(1-v)$, $(v)$ sa použijú kubické Hermittove polynómy $H_0^3$ a $H_3^3$. Tie totižto majú nulové derivácie, dôsledkom čoho je, že aj záplaty majú na hraničných krivkách nulové derivácie v smere priečnom ku krivke a môžeme ich teda spájať. Tieto plochy však nadobúdajú v dôsledku použitia len dvoch Hermittovych polynómov v rohoch nežiaduce „rovinné“ plôšky.

### 7.8.4 Coonsova bikubicky stmeľovaná plocha – CBicubicCoons

Predchádzajúci nedostatok odstraňujú bikubicky stmeľované plochy. Pri vyčíslovaní používajú všetky Hermittove polynómy, ich konštrukcia si však vyžaduje zadanie aj vektorových funkcií reprezentujúcich derivácie v smere priečnom ku krivke. Data takýchto plôch teda sú:

$$X(u,0), X_v(u,0), X(u,1), X_v(u,1), X(0,v), X_u(0,v), X(1,v), X_u(1,v).$$

Plocha je teda definovaná vzťahom:

$$X(u,v) = H_c(u,v) + H_d(u,v) - H_{cd}(u,v)$$

kde

$$H_c(u,v) = H_0^3(u)X(0,v) + H_1^3(u)X_u(0,v) + H_2^3(u)X_u(1,v) + H_3^3(u)X(1,v)$$

$$H_d(u,v) = H_0^3(v)X(u,0) + H_1^3(v)X_u(u,0) + H_2^3(v)X_u(u,1) + H_3^3(v)X(u,1)$$

$$H_{cd}(u,v) = (H_0^3(u), H_1^3(u), H_2^3(u), H_3^3(u))\mathbf{C}(H_0^3(u), H_1^3(u), H_2^3(u), H_3^3(u))^T$$

$$\mathbf{C} = \begin{pmatrix} X(0,0) & X_v(0,0) & X_v(0,1) & X(0,1) \\ X_u(0,0) & X_{uv}(0,0) & X_{uv}(0,1) & X_u(0,1) \\ X_u(1,0) & X_{uv}(1,0) & X_{uv}(1,1) & X_u(1,1) \\ X(1,0) & X_v(1,0) & X_v(1,1) & X(1,1) \end{pmatrix}$$

### 7.8.5 Translačná plocha – CTransSurface

Máme dané dve krivky $C_1(u)$ a $C_2(2)$ so spoločným bodom $A = C_1(0) = C_2(0)$. Potom definujeme translačnú plochu nasledovne:

$$T(u,v) = C_1(u) + C_2(v) - A.$$

Ak $\hat{u}$, $\hat{v}$ sú pevne zvolené hodnoty parametra, tak platí:

$$T(\hat{u},v) = C_2(v) + (C_1(\hat{u}) - A) \qquad T(u,\hat{v}) = C_1(u) + (C_2(\hat{v}) - A).$$

To znamená, že izoparametrické krivky tejto plochy sa získavajú transláciami. Odtiaľ pochádza aj ich názov. Derivovaním zistíme, že $T'_{uv}(u,v) = 0$ pre všetky $u$, $v$. Teda tieto plochy majú všade nulové twisty, nemusia mať však všade nulovú krivosť

## 7.9 Bodová plocha – CVertexSurface

Podobne, ako bodové krivky, používajú aj bodové plochy na svoje definovanie body. Tieto body sú zoradené v riadiacej sieti, realizovanej dvojrozmerným poľom bodov.

Ako je zvykom v našej knižnici, riadiaca sieť je externý objekt, ktorý je potrebné vytvoriť mimo triedy a príslušnou metódou ho triede priradiť. Trieda teda disponuje metódou *SetNet(CNet)* na spomínané priradenie a metódou *GetNet(CNet)* na získanie ukazovateľa na riadiacu sieť triedy.

Potrebnými úpravami prešla aj metóda *Ready()*, ktorá obsahuje test, či je riadiaca sieť zadaná korektne.

### 7.9.1 Bézierova plocha tenzorového súčinu – CTensorBezier

Túto plochu si môžeme predstaviť ako množinu bodov Bézierovej krivky, ktorá sa pohybuje v priestore tak, že každý bod jej riadiaceho polygónu sa pohybuje po Bézierovej krivke. Všetky tieto krivky sú pritom rovnakého stupňa. Počas svojho pohybu zostáva krivka Bézierovou krivkou.

Algoritmus na vyjadrenie tejto plochy je:

$$b^{m,n}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} b_{ij} B_i^m(u) B_j^n(v),$$

kde $m$ je stupeň prislúchajúci parametru $u$ a $n$ je stupeň prislúchajúci parametru $v$. Body $b_{ij}$ sú body riadiacej siete a $B_i^m$, $B_j^n$ sú Bernsteinove polynómy stupňa $m$, $n$.

## 7.10 Splinová plocha – CSplineSurface

Splinová plocha je bodová plocha, ktorá je rozšírená o dvojicu uzlových postupností, jednu pre $u$-krivky a jednu pre $v$-krivky. Tieto atribúty sú opäť privátne a trieda teda disponuje sadou metód na prácu s nimi.

Metóda *Ready()* je rozšírená o test na existenciu uzlových postupností.

### 7.10.1 Bikubická B-splinová plocha – CBicubBSpline

Bikubická B-splinová plocha je určená dvoma uzlovými vektormi $u_0, \ldots, u_{m+3}$, $v_0, \ldots, v_{n+3}$ a sieťou riadiacich vrcholov $D_{ij}$ pre $i = 0, \ldots, m$ a $v_0, \ldots, v_{n+3}$. Plocha je potom definovaná ako

$$p(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} D_{ij} N_i^3(u) N_j^3(v).$$

Z definície vyplýva, že sa jedná o bikubickú plochu – plochu tretieho stupňa v každej premennej.

# 8 Záver

Vytvoriť kvalitnú objektovo orientovanú knižnicu kriviek a plôch je pomerne náročný a rozsiahly projekt.

Našou snahou bolo vytvoriť flexibilný abstraktný objektový model na vyčíslovanie a prípadnú vizualizáciu geometrických objektov, nad ktorým sú implementované základné typy kriviek a plôch. Na integrovanie nového typu krivky alebo plochy je nutné vytvoriť iba najzákladnejšiu metódu, metódu na jej vyčíslenie, všetky ostatné metódy sú už implementované v abstraktných triedach. Programátor sa môže venovať experimentovaniu bez nutnosti poznania a implementovania algoritmov na vyčíslovanie derivácií, krivosti alebo iných typov objektov, ktoré sú potrebné pre zistenie geometrickej kvality krivky alebo plochy.

## 8.1 Práca do budúcna

Knižnica zatiaľ nedisponuje kompletnou sadou metód na zobrazovanie geometrických objektov a na interakciu s užívateľom. Implementované sú len základné algoritmy na zobrazovanie, čo je dostatočné v prípade, že chceme vizuálne testovať vlastnosti gemetrických objektov. Pre potreby modelovania sú však tieto metódy nedostačujúce. Preto by sme sa chceli v budúcnosti venovať aj rozšíreniu funkčnosti grafického výstupu.

Prvky diferenciálnej geometrie pre krivky sú implementované v dostatočnom počte, čo umožňuje ich pomerne kvalitnú analýzu. Pre detailnú analýzu plôch je však nevyhnutné implementovať aj ďalšie prvky, ktoré pre svoju zložitosť do knižnice zatiaľ neboli začlenené. Abstraktné triedy síce poskytujú algoritmy na vyčíslovanie derivácií, tieto algoritmy však v niektorých prípadoch môžu viesť k nepresným výsledkom. Preto by sme chceli pre každú triedu implementovať vlastný algoritmus (ak je známy) a spresniť tak výpočty.

Už sme spomínali, že knižnica obsahuje iba základné typy kriviek a plôch. Do budúcnosti by sme chceli knižnicu rozširovať aj o ďalšie, známe aj menej známe typy, ktoré by však pre niektorých užívateľov mohli znamenať výrazný prínos. Algoritmy na vyčíslovanie sú implementované s ohľadom na prehľadnosť kódu, to však neznamená, že nie sú optimalizovateľné. Preto je optimalizácia ďalšou dôležitou úlohou nasledujúceho vývoja.

Nezávislosť na platforme a operačnom systéme je významnou vlastnosťou každého projektu, pretože sa stáva dostupný väčšiemu počtu užívateľov.

Využitie C++ na tvorbu našej knižnice bolo prvým krokom k dosiahnutiu tejto nezávislosti. Knižnica zatiaľ využíva dáta viazané k operačnému systému Win32, čo však chceme odstrániť v nasledujúcich verziách knižnice. Knižnica je dostupná zdarma pre nekomerčné použitie a je dodávaná aj so zdrojovými kódmi. To umožňuje komukoľvek zapojiť sa do vývoja a odhaliť tak jej prípadné nedostatky.

# Zoznam obrázkov

# Literatúra


[Far90] G. E. Farin, *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, San Diego, 1990.

[FR95] A. Ferko and E. Ružický, *Počítačová grafika*, Sapientia, Bratislava, 1995.

[McC94] J. McCleary, *Geometry from a Differentiable Viewpoint*, Cambridge University Press, Cambridge, 1994.

[NvV76] M. Nekvida, J. Šrubař, and J. Vild, *Úvod do numerické matematiky*, Nakladatelství technické literatury, Praha, 1976.

[vBF98] J. Žára, B. Beneš, and P. Felkel, *Moderní počítačová grafika*, Computer Press, Praha, 1998.

[vvM94] Ľubor Šešera and Andrej Mičovský, *Objektovo - orientovaná tvorba systémov a jazyk C++*, PERFEKT a.s., Bratislava, 1994.

[WJ98] G. Westgaard and J.Heinmann, *Analysis and Visualization of Geometric Curve Properties*, Mathematical Methods for Curves and Surfaces II (M. Deahlen, T. Lyche, and L. L. Schumaker, eds.), Vanderbilt University Press, Nasvhille, 1998.