Příprava vstupních dat pro 3D aplikaci

Input data preparation for 3D application

Bc. Tomáš Podoba

Diplomová práce
2007

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2006/2007

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Tomáš PODOBA**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Příprava vstupních dat pro 3D aplikaci**

Zásady pro vypracování:

1.Vytvořte literární rešerši na zadané téma.
2.Obecně popište tvorbu 3D modelů, základní nástroje včetně materiálových vlastností.
3.Seznamte se s plánkem budovy FAI UTB U5 a proveďte návrh vytvoření jejího modelu.
4.Vytvořte model budovy U5 ve dvou verzích. První bude detailní (high-face) verze a druhá bude pouze hrubý návrh (low-poly). Obě verze porovnejte.
5. Low-poly verzi modelu připravte a navrhněte tak, aby se dala prakticky využít v grafickém enginu.

Rozsah práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1.Kříž, J.: 3ds max. Computer press, Praha 2006.
2.http://www.gimp.org/ -domovská stránka programu gimp (zpracování textur)
3.http://www.autodesk.com/ - domovská stránka programu 3DS Max.
4.http://irrlicht.sourceforge.net/ - domovská stránka enginu Irrlicht

Vedoucí diplomové práce: **Ing. Pavel Pokorný, Ph.D.**
Ústav aplikované informatiky

Datum zadání diplomové práce: **13. února 2007**

Termín odevzdání diplomové práce: **28. května 2007**

Ve Zlíně dne 13. února 2007

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.
1

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Hlavním cílem této práce je navrhnout a vytvořit trojrozměrný model budovy fakulty Aplikované informatiky FAI ve Zlíně, který bude optimalizovaný pro implementaci do 3D grafického enginu.

První, teoretická část seznámí čtenáře s barevnými modely a především s reprezentací křivek, které jsou základním stavebním kamenem pro veškeré prostorové struktury. Dále jsou popsány vývojové prostředí a funkce aplikace Blender a 3D grafického enginu Irrlicht. Nemalá část práce je také zaměřena na popis základních modelovacích principů.

Praktická část ve svém úvodu čtenáře názorně seznámí s dostupnými modelovacími technikami. Zbývající prostor je dále plně věnován samotnému vytvoření trojrozměrného modelu fakulty včetně popisu implementace do zvoleného enginu.

Klíčová slova: 3D model, FAI, Křivka, NURBS, Irrlicht engine

ABSTRACT

The main target of this task is to design a three dimensional model of the building of The Faculty of applied informatics in Zlin which will be optimalized for implementation into 3D graphics engine.

The first - the theoretical part - will bring readers to the attention of a color models and especially a state of curves which are the base stones of every dimensional structures. A development environment and the main features of Blender and 3D graphics engine Irrlicht are also described. There is also a part of the task which is aimed at a description of basic modeling priciples.

The firts part of the practical part of the task will present available modelling techniques. The following part is fully dedicated to the creation of the 3D model of the faculty including a decsription of the implementation into the selected engine.

Keywords: 3D model, FAI, Curve, NURBS, Irrlicht engine

V rámci této práce bych chtěl poděkovat panu Ing. Pavlu Pokornému Ph.D., za všechny rady a připomínky, které mi byly velmi nápomocny při tvorbě této práce. Také bych tímto chtěl poděkovat svým rodičům za veškerou doposud projevenou podporu během mého životního studia. A v neposlední řadě všem přátelům, kteří mi s vývojem a kritikou pomáhali.

Prohlašuji, že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně …….

Podpis diplomanta

OBSAH

ÚVOD

Není tomu tolik let, co vznikl osobní počítač. S jeho příchodem se pro lidi naskytla neskutečná sféra možností jak naložit s informacemi nejen v podobě tištěné, nýbrž digitální. Zprvu to bylo těžké, zdlouhavé a drahé, avšak postupem času díky neskutečnému vývoji mikroelektroniky, se stal počítač nezbytnou pomůckou a dnes už standardním vybavením domácnosti. S vývojem počítačů se samozřejmě vyvíjelo i umění, které je s ním úzce spjato, a vznikl termín počítačová grafika.

Počítačová grafika je obor informatiky, který používá počítače na syntetické vytváření umělých snímků (tzv. rendering) a také na úpravu zobrazitelných a prostorových informací, nasnímaných z reálného světa (například digitální fotografie a jejich úprava). Protože je počítačová grafika obor velice dynamicky se rozvíjející, přináší sebou obrovskou expanzi v dalších odvětvích rozmanitých průmyslů a již poměrně přesně nelze určit hranici, vymezující striktně její pojem.

Počítačová grafika je věnována postupům, jak informaci z počítače znázornit v grafické formě, např. výstup jednorozměrných grafů, dvou- i třírozměrných obrazů, na obrazovce nebo zapisovači. Algoritmy počítačové grafiky nabízejí řešení takových problémů jako je vytváření hladkých křivek na rastrových obrazovkách, odstraňování čar zakrytých přední částí obrazu, kreslení třírozměrných útvarů ze zvoleného zorného úhlu, realistické vybarvování ploch a povrchů těles, atd.

Obor počítačové grafiky lze v současné době rozdělit do dvou základních škatulek:

- **Vektorová grafika**

  Základem vektorové grafiky je matematika. Grafické informace se ukládají ve formě matematického zápisu. Ten definuje tvar čáry a křivky, které jsou základními kameny všech zbývajících objektů. Prvním iniciátorem v oblasti vektorové grafiky byl v sedmdesátých letech francouzský matematik a konstruktér Sierr Béziere, který vyvinul matematickou metodu jíž byl schopen popsat libovolný úsek křivky pouze za pomoci čtyř bodů.

- **Rastrová grafika (bitmapová grafika)**
  Celý obrázek je popsán pomocí jednotlivých barevných bodů (pixelů). Body jsou uspořádány do mřížky. Každý bod má určen svou přesnou polohu a barvu pomocí příslušného barevného modelu. Rastrová grafika umožňuje (jako fotografie nebo televize) prostřednictvím tisíců malých bodů vytvořit prakticky libovolný výsledný obraz.

Při realizaci virtuálního modelu budovy fakulty je třeba využít obou grafických škatulek. Jednak model není nic jiného než síťová struktura vektorů, která tvoří konečné uzavřené plochy různých tvarů a jednak jsou tyto plochy „potaženy“ buď enginem vygenerovanou nebo fotorealistickou texturou. Nyní se dostáváme do fáze kdy by bylo vhodné definovat co je engine a jaký byl vybrán.

Funkcionalitu enginu si lze představit jako jakýsi softwarový počítač umožňující chod aplikace na libovolném hardware bez nutnosti jejích úprav. Specifikujme si ho tedy jako technologii, která má za úkol vykreslovat (*renderovat*) vstupní prostředí v reálném čase. Kvalitní grafický engine má vliv na nárůst fps (*frames per second*, počet snímků za sekundu) a tím pádem má program vetší šanci kvalitně uspět v simulačním propočtu autentického matematického modelu budovy. Hodnota *fps* je klíčovým výstupním parametrem a měla by být vyšší jak 25, neboť lidské oko menší počet snímků registruje jako neplynulý, trhaný obraz. Praktické řešení low-polygon verze by se mělo opírat o knihovny OpenGL a Direct3D ve volně dostupném real-timovém 3D enginu Irrlicht.

TEORETICKÁ ČÁST

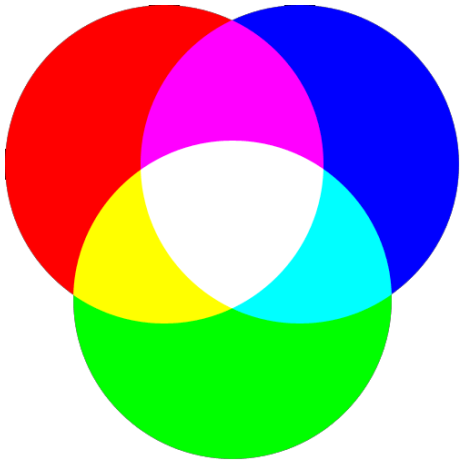
1 ROVINNÁ – 2D GRAFIKA

1.1 Barevné prostory

Popisují základní barvy a modely mísení těchto základních barev do výsledné barvy. Barva je v přírodě dána směsí světla různých vlnových délek a různé barevné modely se snaží napodobit barvu co nejvěrněji. V praxi se používají modely, u kterých je zvolen vhodný kompromis mezi přesností podání barevného dojmu a složitostí konkrétního modelu.

1.1.1 RGB

Různé barvy, které se používají při vytváření obrazu, jsou tvořeny kombinací několika základních barev z barevného spektra. Na barevné obrazovce například vidíme barvu jako výsledek složení tří složek – červené (R, *red*), zelené (G, *green*) a modré (B, *blue*). Barvy lze vyjádřit trojicí (barevným vektorem), jejíž složky nabývají hodnot z intervalu $\langle 0,1 \rangle$. Bývají uváděny i v celočíselném rozsahu 0 – 255, což odpovídá kódování každé ze složek RGB v jednom bytu. Hodnota 0 znamená, že složka není zastoupena, maximální hodnota indikuje, že složka nabývá své největší intenzity. Vyjádření barevných složek pomocí 3 bytů je v současnosti nejběžnější. Používají se ale i jiná kódování (s nižším nebo vyšším počtem bitů), např. 12 nebo 16 bitů na barevný kanál.



Obrázek 1: Aditivní míchání barev

Počet barevných odstínů, který lze reprezentovat trojicí bytů je 256^3 = 16 777 216. Ne všechna výstupní obrazová zařízení jsou schopna takovéto množství barev současně

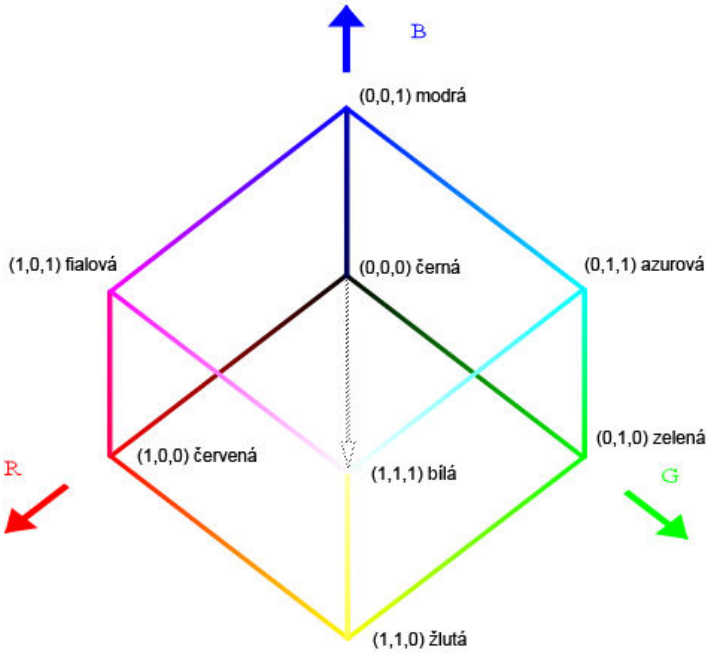
zobrazit. Proto bývá počet barev před vykreslením uměle snižován, ovšem tak, aby lidské oko zaznamenalo co nejmenší ztrátu kvality obrazu.

Složíme-li červenou, zelenou a modrou barvu zobrazovanou displejem v plné intenzitě, získáme barvu bílou. Podobně získáváme stupně šedi skládáním všech tří barev se shodnou, postupně se snižující intenzitou. Kdybychom však chtěli z nějakých důvodů převést různobarevný obraz na šedotónový, nemůžeme jeho barvy nahradit odstíny šedi získanými prostým průměrem ze tří základních barev. Lidské oko vnímá různým způsobem intenzitu jednotlivých barevných složek (nejcitlivější je na zelenožlutou), takže používá pro výpočet *jasu* empirický vztah

$$I = 0.299R + 0.587G + 0.114B \tag{1}$$

Z programátorského hlediska je zřejmé, že konkrétní reprezentace barvy může být značně různorodá. Od jednobitové informace rozlišující pouze mezi stavy černá a bílá, přes osmibitové číslo označující stupeň šedi až po různé zápisy barevných složek. Poznamenejme, že pro reprezentaci barevných složek R,G,B ve dvou bytech (5 – 6 – 5 bitů) se vžilo označení *high color*, pro zápis ve třech bitech označení *true color*. Někdy jsou barvy vyjádřeny nepřímo – pomocí čísla, odkazujícího do tabulky, zvané *paleta*.

Popis barvy třemi složkami R, G a B není jedinou možností. Uvedená volba základních barev je dána technickými vlastnostmi monitorů, resp. Použitými luminiscenčními prvky, které jsou zdrojem vyzařovaného barevného světla. Říká se, že barva je vyjádřena v *barevném prostoru RGB*. Jeho základní vlastností je součtové, ***aditivní skládání*** barev (Obr. 1) – čím více barev složíme (sečteme), tím světlejší je výsledek. Svítivost barevných luminoforů v monitorech roste nelineárně, a proto se při zobrazování barvy v prostoru RGB používá tzv. gama korekce.



Obrázek 2: Geometrická jednotková reprezentace prostoru RGB

Barevný rozsah můžeme v prostoru RGB zobrazit prostorově jako jednotkovou krychli (Obr. 2) umístěnou v osách označených postupně *r*, *g* a *b*. Počátek souřadnic odpovídá černé barvě, zatímco vrchol o souřadnicích [1, 1, 1] odpovídá bílé. Vrcholy krychle, které leží na osách, představují základní barvy a zbývající vrcholy reprezentují doplňkové barvy ke každé ze základních barev.

Každému barevnému vektoru v prostoru RGB odpovídá v této reprezentaci jeden bod krychle. Fialová barva, která je získána součtem červené a modré, je znázorněna bodem o souřadnicích [1, 0, 1]. Bílá, představována vrcholem [1, 1, 1], je složena z barev v červeném, zeleném a modrém vrcholu. Odstíny šedi odpovídají bodům na diagonále krychle spojující černý a bílý vrchol.

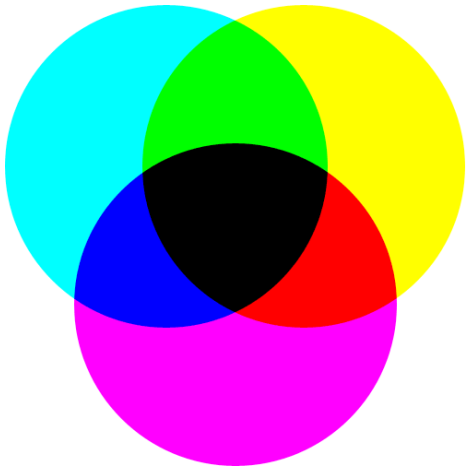
1.1.2 RGBA

V počítačové grafice se setkáváme s další zkratkou, která připomíná prostor RGB. Jedná se o kombinaci RGBA (či výstižněji $RGB\alpha$). Tato zkratka je používána pro vyjádření skutečnosti, že barevný obraz zapsaný v prostoru RGB je doplněn informací o průhlednosti. Každý barevný bod takového obrazu s sebou nese skládání údaj (např.

v rozmezí 0 – 1), který určuje, v jakém rozsahu pokrývá barva plochu obrazového bodu. Hodnota 0.0 znamená neprůhledný barevný bod, maximální hodnota 1.0 zcela průhledný. Při zobrazování samotného obrazu nemá složka α (α*-kanál*, α*-channel*) význam. Pojem RGBα neznamená změnu barevného prostoru, nýbrž přidání další informace. Složka α se ukládá do rozsahu jednoho i více bytů. Používá se zejména při kombinování více obrazů do jednoho celku.

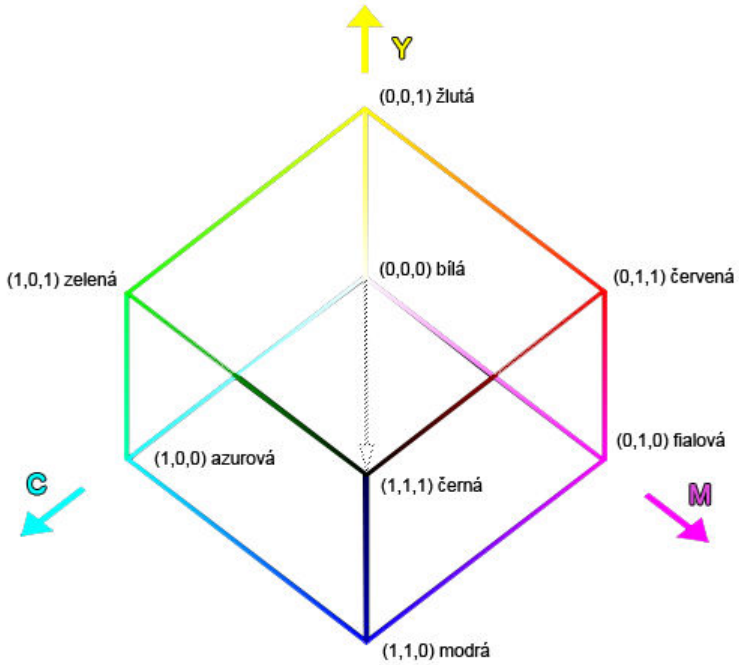
1.1.3 CMY, CMYK

Prostor RGP je technicky orientovaný prostor, vhodný pro displeje. Lidská zkušenost s mícháním bare však vychází ze zcela jiné práce s barvami. Klasickým a všem známým je jistě příklad malíře, jenž nové barvy vytváří míšením jednotlivých barevných pigmentů, přičemž každé přidání pigmentu vytvoří tmavší barvu. Toto skládání barev se nazývá ***subtraktivní***. Složením všech barev vznikne černá, což je právě opačná situace oproti aditivnímu skládání barevného světla.



Obrázek 3: Subtraktivní míchání barev

Míchání barviv je typické nejen pro malíře, ale i pro tiskařské techniky. Těmto účelům vyhovuje prostor CMY, obsahující tři základní barvy: tyrkysovou neboli modrozelenou (C, *Cyan*), fialovou (M, m*agenta*) a žlutou (Y, *yellow*). Také tento prostor lze popsat jednotkovou krychlí. Sčítání hodnot CMY ovšem odpovídá subtraktivnímu skládání barev, takže vrchol [1, 1, 1] reprezentuje černou barvu. Lze tedy říci že CMY je inverzním modelem k RGB.



Obrázek 4: Geometrická jednotková reprezentace prostoru CMY

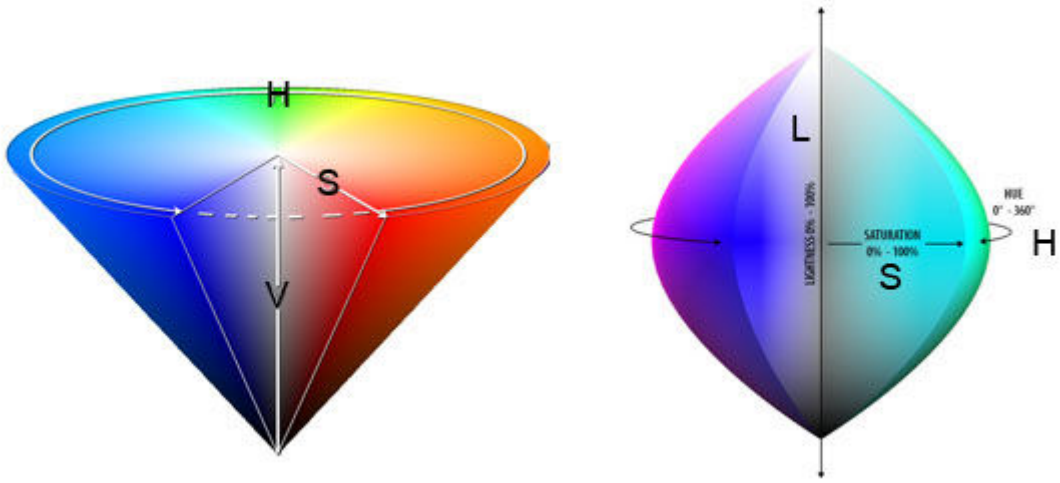
Převod mezi prostory RGB a CMY je jednoduchý. Vyjádříme-li barevný vektor v prostoru RGB tříprvkovou maticí [*r g b*], určíme vektor [*c m y*] v prostoru CMY odčítáním matic:

$$\begin{bmatrix} c \\ m \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} r \\ g \\ b \end{bmatrix} \tag{2}$$

Při tisku jsou barevné obrazy reprodukovány jako soutisk tří obrazů, tvořených základními barvami C, M a Y. Tyto základní barevné pigmenty nesmí být dokonale krycí, neboť nové barvy vznikají vzájemným překrýváním. Složením všech tří barev proto při tisku nevznikne požadovaná černý, ale pouze špinavě hnědá. Mimoto není ekonomické skládat černou barvu ze tří jiných barev. Z těchto důvodů se černá tiskne jako samostatná barva. Lze ji použít i ke ztmavení ostatních barev. Od prostoru CMY se tak v polygrafii přechází k prostoru *CMYK* přidáním černé složky (K, *blacK*) jako čtvrté základní barvy.

1.1.4 HSV a HLS

Oba prostory HSV i HLS definují barvu trojicí složek, které však tentokrát nepředstavují základní barvy. Třemi hlavními parametry prostoru *HSV* jsou *barevný tón* (H, *hue*), *sytost* (S, *saturation*) a *jasová hodnota* (V, *value*). Barevný tón označuje převládající spektrální barvu, sytost určuje příměs jiných barev a jas se dán množstvím bílého (bezbarvého) světla. Pro zobrazení prostoru se nepoužívá krychle, nýbrž šestiboký jehlan či kužele, jehož vrcholy leží v počátku soustavy souřadnic HSV. Souřadnice *s* a *v* se mění od hodnot 0 – 1, souřadnice *h* reprezentuje úhel a nabývá hodnot z intervalu $< 0^\circ, 360^\circ)$. Vrchol představuje černou barvu. Jas roste směrem k podstavě, střed podstavy reprezentuje bílou barvu. Sytost odpovídá relativní vzdálenosti bodu od osy kužele. Dominantní barvy (s hodnotou sytosti jedna) tedy leží na plášti, čisté barvy jsou pak na obvody podstavy. Při pohybu po obvodu ve stejné výši od základny se postupně mění barevný tón, sytost a jas zůstávají nezměněny.



Obrázek 5: Geometrická reprezentace prostorů HSV a HLS

Název prostoru HLS je odvozen z pojmů *barevný tón* (H, *hue*), *světlost* (L, *lightness*) a *sytost* (S, *saturation*). Prostor HLS je obdobou prostoru HSV, v němž byl kužel nahrazen dvojicí kuželů. Barevný tón je opět vyjádřen úhlovou hodnotou, světlost se mění od 0 (černá v dolním vrcholu) do 1 (bílá v horním vrcholu kužele). Sytost nabývá na povrchu kuželů hodnoty 1 a klesá na 0 směrem k ose kuželů. Nejjasnější čisté barvy mají tedy souřadnice

$s = 1$ a $l = 0.5$ a leží na obvodu podstav kuželů.

Tvar prostoru HLS plně odpovídá skutečnosti, že nejvíce různých barev vnímáme při „průměrné“ světlosti což je oblast podstav. Schopnost rozlišit barvy klesá jak při velkém ztmavení, tak při přesvětlení. Lidské oko totiž vnímá spíše jemné odskoky jasných barev, nežli barev přecházejících postupně do černé.

Oba prostory jsou používány i jako vzorníky barev, kde jsou zobrazovány řezy jehlanem, či kuželem pro různého hodnoty barevného tónu.

1.1.5 YUV

Další ze základních barevných prostorů označovaný YUV se používá pro přenos televizních signálu v normě PAL (Austrálie, Evropa mimo Francie, která používá normu SECAM). Setkat se můžeme i s podobným prostorem – YIQ, který je určen pro americkou a japonskou televizní normu NTSC a prostor YC_BC_R pro normu SECAM. Jejich společným rysem je oddělení jasové složky od barevných informací.

Model k popisu barvy používá tříprvkový vektor [Y,U,V], kde Y je jasová složka (pozor Y není označením žluté barvy jak je tomu u barevného prostoru CMY) a U a V jsou barevné složky. U je také někdy označováno jako B-Y a V odpovídá R-Y. Barevné složky se používají v rozsahu od -0.5 do +0.5, jasová složka má rozsah od 0 do 1. Pro převod mezi prostory RGB a YUV se využívá jednoduchého maticového násobení:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.141 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.137 \\ 1 & -0.397 & -0.580 \\ 1 & 2.034 & 0 \end{bmatrix} * \begin{bmatrix} Y \\ U \\ V \end{bmatrix}$$

1.1.6 Ostatní barevné prostory

V předcházejících odstavcích byly popsány globálně nejpoužívanější barevné prostory, které lze díky svému uplatnění v praxi považovat za skupinu dominantní. Přesto existuje i další škála jiných rozmanitých barevných modelů, které mají své uplatnění v jiných oborech. Každopádně princip definování barev bývá ve všech případech obdobný k výše zmíněným modelům ovšem v nestandardním pojetí – rozličné jednotkové modely (třeba i matematické plochy).

- CIE XYZ, xyY, I1I2I3, UVW,
- LSLM, L*a*b*, L*u*v*, LHC, LHS, HSV Polar,
- HSI, HSI Polar, YCbCr
- NCS, Munsell, RAL

2 PROSTOROVÁ – 3D GRAFIKA

2.1 Křivky a plochy

Křivky a plochy se používají v počítačové grafice a souvisejících aplikacích na mnoha různých místech. Setkáváme se s nimi při modelování ve třech i ve dvou dimenzích, při definici fontů, při slučování dráhy pohybujících se objektů v počítačové animaci, při definici objektů pro šablonování aj. Využití prostorové – 3D grafiky s sebou přináší obrovské využití téměř ve všech možných sférách. Různé aplikace mají různé požadavky a proto je tato oblast nesmírně široká.

Vůbec první vizuální grafický obraz generovaný počítačem vytvořil v roce 1950 Ben Laposky. Po zhlédnutí by se vám mohlo zdát, že je obraz směsice rozmazaných bodů, ale není tomu tak. Laposky využil analogového osciloskopu, pomocí něhož vytvořil z částí světélkujících bodů obraz, který zaznamenal na vysokorychlostní film a fotografie, které nazval *oscilony*. Jak je všeobecně známo, osciloskopy jsou obvykle používány pro měření elektrických signálů a dalších fyzikálních stimulů, jako jsou zvuky nebo světelné vlny. Navíc jsou využívány v mnoha variacích, avšak Loposky byl první kdo využil osciloskopu k vytvoření vizuálního obrazu jakožto umělecké dílo.

Vůbec prvního matematického popisu křivek a ploch využíval v roce 1959 P. de Casteljau zaměstnanec firmy Citroen. Pomocí systému DAC-1 (první komerční *CAD* aplikace), který byl vyvinut firmou IBM, tak mohl jednoduše matematicky modelovat první křivky a plochy dle libosti přímo na počítači. Podobně v šedesátých letech vedl P. Béziere vývoj programového systému UNISURF pro návrh křivek a ploch u firmy Renault. Metody tvarování křivek a ploch se postupem času zdokonalovaly a v současné době je k dispozici velmi silný nástroj, která je stále aktivně rozvíjen. Podstatou tohoto rozvoje je kvalitní matematický aparát a v neposlední řadě i zřetelný komerční efekt, který se projevil zejména v oblasti automobilového a průmyslového designu.

Výrazný pokrok do této oblasti a především sjednocení dříve používaných různorodých přístupů přineslo používání racionálních B-spline křivek a ploch s neuniformní parametrizací, NURBS (Non-Uniform Rational B-Spline). Tyto metody umožňují generovat klasické geometrické prvky (úsečky, kružnice, elipsy a v prostoru koule, krychle,

válce atp.) za pomoci stejných metod, které umožňují vytvořit křivky a plochy se složitými průběhy a tvary.

2.2 Vlastnosti křivek

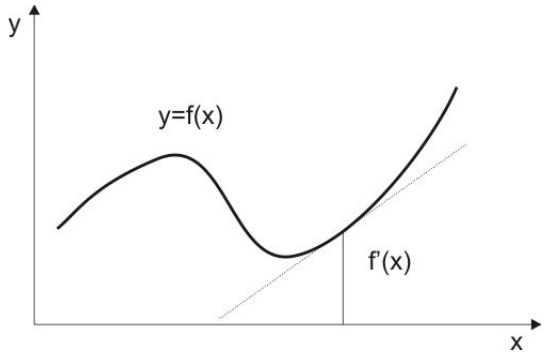
Křivky jsou obyčejně v počítači reprezentovány jako soustava parametrů nějaké rovnice, která je posléze generativně zobrazována. Toto vyjádření může být v podstatě trojího druhu:

- Explicitní
- Implicitní
- Parametrické

Explicitně (obr. 9) vyjádřená křivka může být zadána například jako spojitá funkce ve tvaru

$$y = f(x) \tag{4}$$

A bývá orientována ve směru rostoucího *x*. Jedná se však o zadání, které lze použít pouze pro křivky, které jsou zároveň funkce, tzn. Že hodnotě *x* z definičního oboru dopovídá jediná funkční hodnota *y*.



Obrázek 6: Explicitní zadání křivky

Implicitní zadání křivky má tvar

$$F(x, y) = 0, \tag{5}$$

ideálním příkladem je rovnice kružnice $F(x, y) = (x - xs)^2 + (y - ys)^2 - r^2 = 0$. Toto zadání je poměrně obtížně zobrazitelné v porovnání s ostatními, neboť neumožňuje

v obecnějších případech postupný výpočet křivky. Svůj význam má například při testování oblastí vymezených implicitně zadanou křivkou nebo při výpočtu průsečíku paprsku s křivkou.

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

$$\begin{aligned} x &= x(t), \\ y &= y(t), \\ z &= z(t). \end{aligned} \tag{6}$$

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

$$Q(t) = [x(t), y(t), z(t),] \tag{7}$$

nebo *vektorová rovnice*

$$\vec{q}(t) = (x(t), y(t), z(t)). \tag{8}$$

Vektor $\vec{q}(t) = Q(t) - [0,0,0]$ se nazývá *polohový vektor*, jeho velikost je rovna vzdálenosti bodu $Q(t)$ od počátku. Výhodou parametrického zápisu je závislost souřadnic křivky na jediném parametru *t*, jehož fyzikální interpretací je čas. Díky tomu je možné průběhy, kdy křivka prochází vícekrát (v různých časových okamžicích) stejnými body v prostoru, může se křížit, uzavřít apod.

Tečný vektor v bodě $Q(t_0)$ je určen derivacemi parametricky vyjádřené křivky po složkách ve tvaru

$$\vec{q}'(t_0) = (x'(t_0), y'(t_0), z'(t_0)) = \left(\frac{dx(t_0)}{dt}, \frac{dy(t_0)}{dt}, \frac{dz(t_0)}{dt} \right). \tag{9}$$

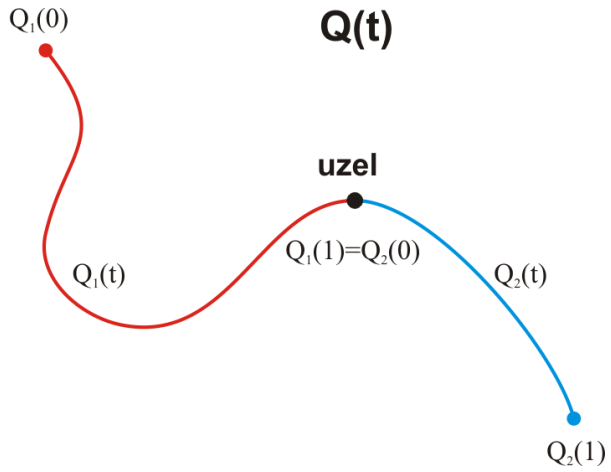
Rovnice *tečny*, tj. přímky která se v tomto bodě dotýká, se vypočítá z tečného vektoru a bodu na křivce jako

$$P(u) = Q(t_0) + u\vec{q}'(t_0) \tag{10}$$

Vektor $\vec{q}'(t_0)$ se také nazývá *směrový vektor přímky*. Ze vztahu (9) je patrné, že parametrická reprezentace křivek umožňuje snadno vyjádřit tečny ke křivce. Toho se s výhodou využívá zejména při navazování křivek a skládání složitých tvarů z jednodušších částí.

Předpokládejme, že $Q_1(t)$ a $Q_2(t)$ jsou dvě části (neboli *segmenty*) jediné křivky $Q(t)$ spojené v bodě $Q_1(1) = Q_2(0)$ (viz obrázek 7). Bod, ve kterém se křivky stýkají, budeme nazývat *uzel* (*knot*).

Segmenty $Q_1(t)$ a $Q_2(t)$ mohou být definovány pomocí různých, nejen polynomiálních reprezentací. Na reprezentaci těchto jednotlivých křivek opravdu nezáleží. Důležitý je však způsob jejich vzájemného napojení (konečného bodu první křivky a počátečního bodu druhé křivky), zejména tzv. *spojitost* (*kontinuity*) v uzlu.



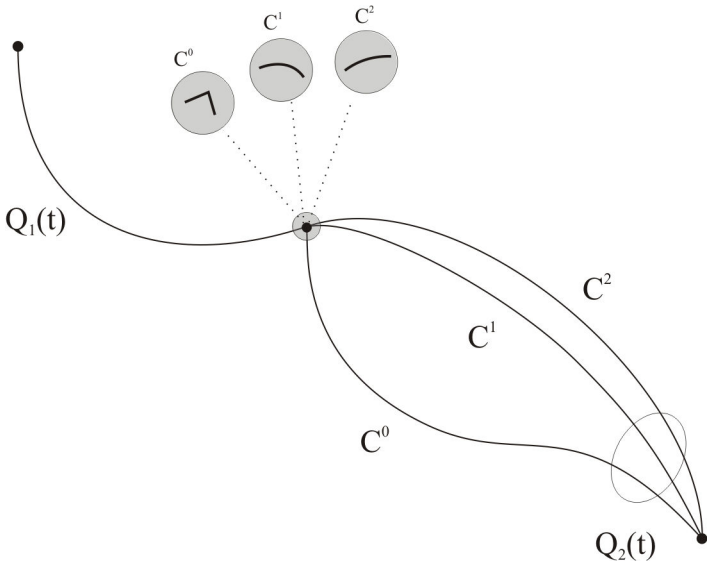
Obrázek 7: Křivka $Q(t)$ vzniklá spojením dvou segmentů $Q_1(t)$ a $Q_2(t)$

Říkáme, že křivka $Q(t)$ je třídy C^n, má-li ve všech bodech spojité derivace podle času t do řádu n. Označení C^n se nazývá *parametrická spojitost stupně n*. Dva segmenty jsou *spojitě* navázány, tj. mají spojení třídy C^0, pokud je koncový bod prvního segmentu počátečním bodem segmentu druhého. Dva segmenty mají spojení C^1, pokud je tečný

vektor v koncovém bodě segmentu Q_1 roven tečnému vektoru segmentu Q_2 v jeho počátečním bodě. Analogicky rovnost vektoru první a druhé derivace je požadována pro C^2, atd. Zkráceně zapisujeme

$$\vec{q}_1^{\,(i)}(1) = \vec{q}_2^{\,(i)}(0); \forall i = 0,1,2,...,n \tag{11}$$

Čím vyšší spojitost je požadována, tím delší „dobu“ (ve smyslu parametru *t*) se oba segmenty přimykají ke stejnému směru. Zjevně $C^{m+1} \Rightarrow C^n$. Ze spojitosti C^0 plyne, že bod se pohybuje po spojité dráze, ale v uzlu může měnit skokem směr pohybu, rychlost i zrychlení. Směr pohybu a velikost složky rychlosti se nemůže měnit skokem při spojitosti C^1 a zrychlení zůstává nezměněné při spojitosti C^2.



Obrázek 8: Spojitost C^0, C^1 a C^2

Hladkost navázání křivek můžeme také posuzovat posle *geometrické spojitosti* označované G^n, nejčastěji se používají geometrické spojitosti G^0 a G^1. Dva segmenty křivky $Q(t)$ jsou G^0 spojité, pokud je koncová bod Q_1 totožný s počátečním bodem Q_2. Dva segmenty jsou G^1 spojité, pokud jsou G^0 spojité a současné tečné vektory $\vec{q}_1'(1)$ segmentu Q_1 a $\vec{q}_2'(0)$ segmentu Q_2 jsou souhlasně kolineární, tj.platí:

$$\vec{q}_1'(1) = k\vec{q}_2'(0); k > 0\,. \tag{12}$$

Tato spojitost zaručuje totožnost tečen (nikoli tečných vektorů). Pohybující se bod v uzlu nemůže změnit skokem směr, ale může změnit skokem rychlost, křivka je vizuálně hladká.

Geometrická spojitost G^n v daném bodě je definována nezávisle na způsobu, jakým byly oběd stýkající se křivky vytvořeny, tj. nezávisle na parametru *t*. Za předpokladu, že obě křivky jsou v místě spojení diferencovatelné, pak Q_1 a Q_2 splňují podmínku geometrické spojitosti G^n, pokud jsou v bodě $[x_0, y_0, z_0] G^{n-1}$ spojité a platí

$$\left[\frac{\partial^n Q_1}{\partial x^n}, \frac{\partial^n Q_1}{\partial y^n}, \frac{\partial^n Q_1}{\partial z^n}\right]_{[x_0,y_0,z_0]} = h \cdot \left[\frac{\partial^n Q_2}{\partial x^n}, \frac{\partial^n Q_2}{\partial y^n}, \frac{\partial^n Q_2}{\partial z^n}\right]_{[x_0,y_0,z_0]}, n > 0, h > 0 . \tag{13}$$

Až na velikosti mají tečné vektory shodný směr.

Ze subjektivního hlediska zaručuje G^1 spojitost „téměř stejnou“ hladkost jako C^1, z hlediska použití bývá daleko snazší zaručit spojitost G^1 nežli C^1. Spojitost C^1 implikuje G^1 s výjimkou jediného případu, kdy vektor rychlosti v místě spojení dvou segmentu je nulový (0,0,0). Obráceně toto neplatí, neboť geometrická spojitost nepostihuje rychlosti a zrychlení pohybu.

2.3 Uživatelský pohled na křivky

Z uživatelského hlediska si na začátku stanovíme určitý cíl – vytvoření křivky, jenž bude splňovat námi požadovaný výstup. Z hlediska matematického se bude jednat o vytvoření požadovaného, většinou složitějšího tvaru, tvaru pomocí jednoduše tvarovatelných částí. Tedy použití segmentů a sestavení *po-částech-polynomiálních křivek*.

Křivkové segmenty definujeme tak, aby splňovaly některé z podmínek:

- Průchod krajními body
- Tečné vektory
- Spojitost mezi navazujícími segmenty

Nejčastěji používané jsou tzv. *kubiky*, jenž jsou omezeny čtyřmi parametry:

- 2 body + 2 tečné vektory — Termite/Fergusson
- 2 koncové body + 2 řídicí body — Bézierovy křivky
- 4 řídicí body — Spline křivky

2.4 Modelování křivek

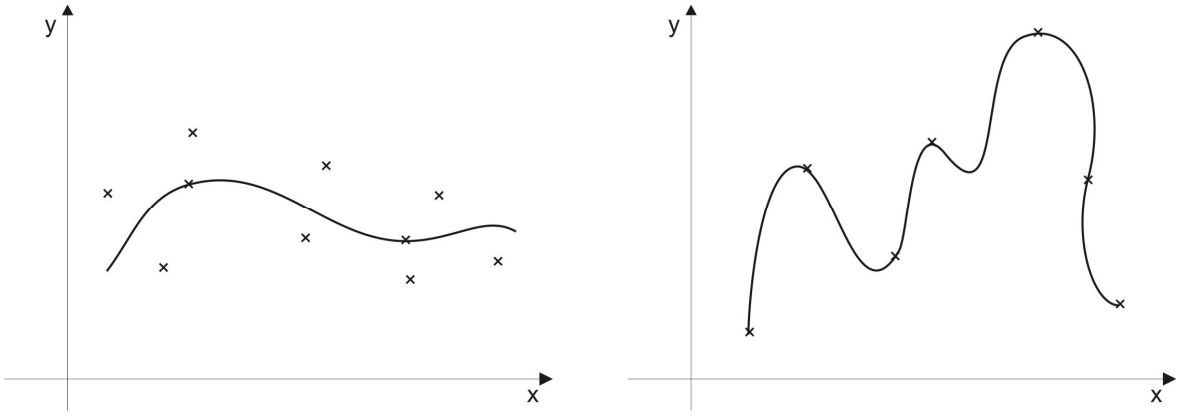
Základním druhem parametrických křivek používaných v počítačové grafice jsou křivky *polynomiální*

$$Q_n(t) = a_0 + a_1 t + \ldots + a_n t^n . \tag{14}$$

.

Polynomiální křivky můžeme snadno vyčíslit a jejich další výhodou je, že jsou jednoduše diferencovatelné. Z polynomiálních křivek lze skládat křivky *po částech polynomiální*, to jsou křivky, jejichž segmenty jsou polynomiálními křivkami. Nejčastěji používané jsou křivky třetího stupně – *kubiky*, které poskytují dostatečně širokou škálu tvarů, jejich výpočet bývá nenáročný, lze s nimi snadno manipulovat a je u nich možné zaručit spojitost C^2, která je často požadována při modelování v CAD systémech.

Modelování probíhá obvykle tak, že je definováno několik *řídicích bodů* (*řídicí polynom*) a matematický aparát z jejich polohy určí průběh křivky. Některé metody umožňují zadávání křivek též pomocí tečných vektorů, je možné zaručit spojitost a hladkost navázání aj.



Obrázek 9: Aproximační (vlevo) a interpolační křivka a jejich řídicí body

Křivku samotnou budu v dalším textu označovat jako $Q(t)$ a její řídicí body P_i. Pokud budu popisovat spojení dvou segmentů křivky, bude první segment značen $Q_1(t)$ a bude zadán řídicími body P_i, zatímco druhý segment bude označovat jako $Q_2(t)$ a bude zadán řídicími body Q_i. Tečný vektor ke křivce bude značen $\vec{q}'(t)$ a druhá derivace potom $\vec{q}''(t)$.

Tečné vektory v řídicích bodech nebo uzlech, kterými křivka prochází, budu označovat $\vec{p}_i'$ resp. $\vec{q}_i'$. Existují dva základní způsoby interpretace řídicích bodů (viz obrázek 12.) a to

- Interpolace
- Aproximace

Křivka generovaná při interpolaci probíhá danými body, zatímco při aproximaci je řídicími body tvar křivky určen, ta jimi však procházet nemusí. Parametricky zadanou kubiku $Q(t)$ ve tvaru:

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y \\ z(t) &= a_z t^3 + b_z t^3 + c_z t + d_z \end{aligned} \tag{15}$$

můžeme zapsat zkráceně v maticovém tvaru

$$Q(t) = TC = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} \tag{16}$$

Derivaci $\vec{q}'(t)$ získáme derivací vektoru **T**

$$\vec{q}'(t) = \frac{d}{dt} Q(t) = \frac{d}{dt} TC = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} C \tag{17}$$

Kubika v prostoru je určena dvanácti parametry, které tvoří prvky matice **C**. Ovládání tvaru křivky pomocí jednotlivých parametrů však není intuitivní, ze změny parametrů nelze jednoduše odhadnout změnu tvaru křivky. Při interaktivním modelování se dá s výhodou spojit tvarování křivek a ploch s viditelnými prvky, jako jsou řídicí body, směry a tečné vektory, zakřivení apod.

Mezi často požadované vlastnosti křivek patří:

1. Invariance k lineární transformaci a projekcím, která zaručuje, že například otočení řídicího polygonu a následné generování křivky dá stejný výsledek, jako otočení každého bodu z vygenerované křivky.
2. Vlastnost konvexní obálky (*convex hull properte*):

(a) silná podmínka – celá křivka leží v konvexní obálce všech svých řídicích bodů,

(b) slabá podmínka – část křivky leží v konvexní obálce některých řídicích bodů (segment v obálce svého generujícího polygonu)

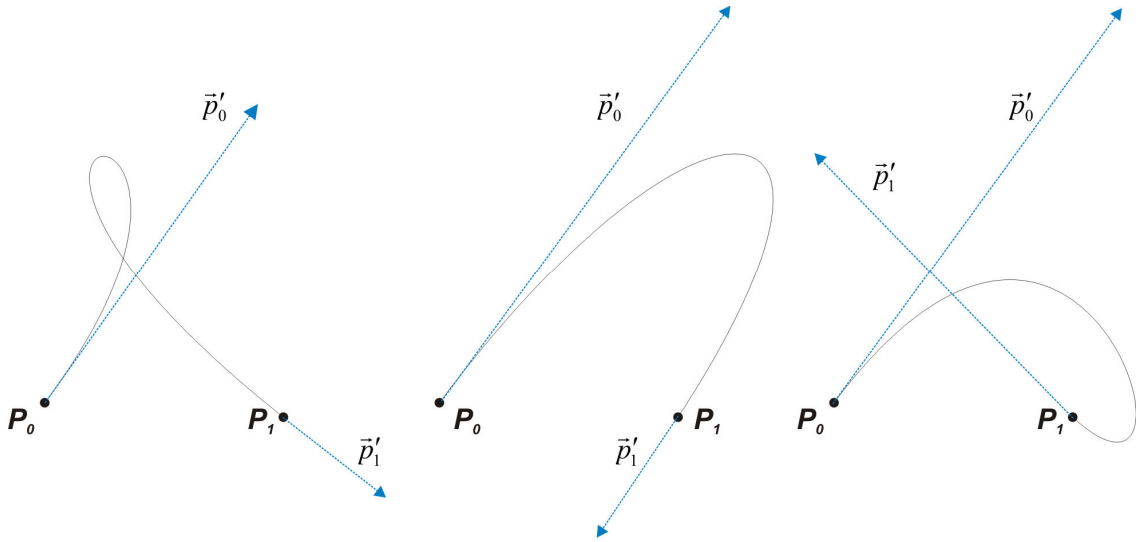
3. Lokalita změn – změnou polohy a/nebo váhy řídicího bodu se mění jen část křivky, nikoli křivka celá.
4. Křivka má procházet krajními body svého řídicího polygonu.

2.5 Interpolační křivky

Existují dva základní druhy interpretace řídicích bodů a to interpolace a aproximace. Při interpolaci (obr. 10) generovaná křivka probíhá danými body. Nejznámější interpolační křivky používané v počítačové grafice jsou tzv. *Hermitovské* křivky či kubiky.

2.5.1 Hermitovské kubiky

Mezi často používané křivky patří Hermitovské kubiky, někdy také označované jako Fergusonovy kubiky. Tyto křivky jsou určeny dvě řídicími body P_0, P_1 a dvěma tečnými vektory $\vec{p}'_0$ a $\vec{p}'_1$ v nich. Body P_0 a P_1 určují polohu křivky, křivka v nich začíná a končí. Poloha této křivky je tedy určena dvěma řídícími body a její tvar závisí na velikosti a směru jejich tečných vektorů. Čím je velikost vektoru větší, tím více se k němu křivka přimyká. Jsou-li oba vektory nulové, pak se křivka stane úsečkou P_0P_1.



Obrázek 10: Změna tvaru Hermitovské kubiky

Předpis pro výpočet Hermitovské kubiky ve stylu (18) má tvar

$$Q(t) = TMG = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ \vec{p}_0' \\ \vec{p}_1' \end{bmatrix}, \tag{18}$$

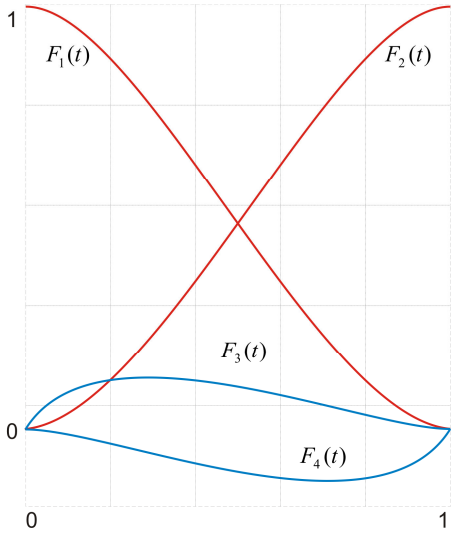
rozepíšeme-li vztah (matice nad tímto textem) do jediné rovnice, dostaneme

$$Q(t) = P_0 F_1(t) + P_1 F_2(t) + \vec{p}_0' F_3(t) + \vec{p}_1' F_4(t), \tag{19}$$

Kde F_1, F_2, F_3, F_4 jsou tzv. *kubické Hermitovské polynomy* ve tvaru (následující zápis) s průběhy znázorněnými na obrázku 14.

$$\begin{aligned} F_1(t) &= 2t^3 - 3t^2 + 1, \\ F_2(t) &= -2t^3 + 3t^2, \\ F_3(t) &= t^3 - 2t^2 + t, \\ F_4(t) &= t^3 - t^2. \end{aligned} \tag{20}$$

Dosazením $t = 0$, resp. $t = 1$ ověříme, že křivka začíná a končí v bodě P_0 resp. P_1. Největší přednost Hermitovských kubik se projeví při jejich navazování. Vzhledem k tomu, že součástí definice křivek jsou tečné vektory v jejich koncových bodech, můžeme jejich hladké navazování realizovat velice snadno.



Obrázek 11: Hermitovské polynomy 3. stupně

Spojitost při navazování dvou kubik docílíme totožností posledního bodu segmentu $Q_1(1)$ a prvního bodu segmentu $Q_2(1)$ (obrázek 10). Spojitost C^1 je zaručena identitou tečných vektorů v uzlu. Spojitost G^1 docílíme jejich lineární závislostí, vyjádřeno pomocí geometrických vektorů **G** dvou po sobě následujících segmentů.

Nevýhodou těchto křivek je poměrně nesnadná editace tečného vektoru ve třech rozměrech.

2.6 Aproximační křivky

Aproximací bodů rozumíme vytvoření takové křivky, která je těmito body vhodně řízena. Není kladen požadavek na procházení opěrnými body (ani prvním a posledním bodem). Způsob řízení odpovídá vytváření této křivky. Existuje v zásadě dvojí přístup k aproximacím.

Přístup první je znám z numerické matematiky a jeho cílem je smysluplná interpretace vstupních dat. Uveďme metodu nejmenších čtverců, její smyslem je nalézt křivku, jež je hladká, a čtverec vzdálenosti řídicích bodů od ní je minimální. Neměnným základem těchto postupů jsou zadané body. Generovaná křivka má jen informativní charakter.

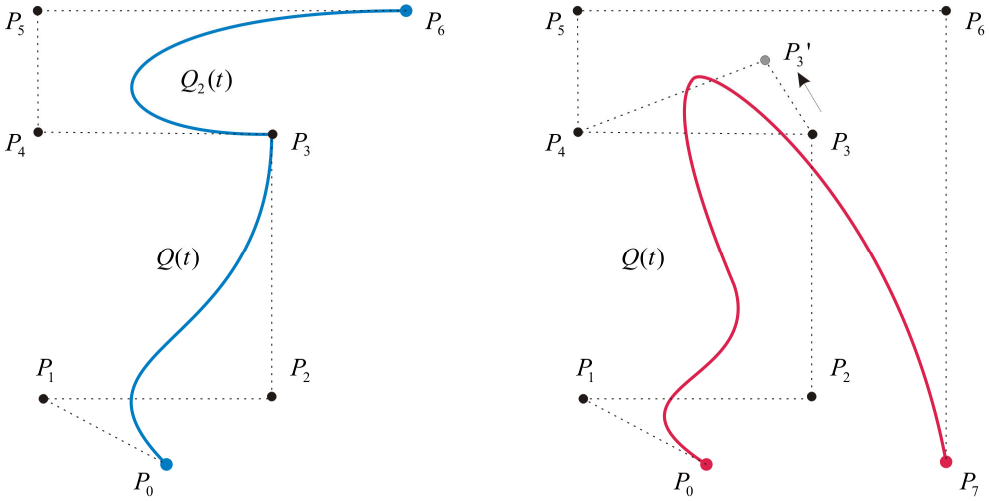
Druhý přístup je používán v počítačové grafice. Smyslem není interpretace bodů, ale generování křivky. Křivka může být řízena body, potom hovoříme o tzv. řídicím polygonu nebo body a vektory. Metoda, která křivku vytváří, zaručuje její vlastnosti. Požadované vlastnosti jsou její hladkost, spojitost, počet inflexí aj. Tyto metody byly navrženy tak, aby vyhovovaly technické praxi, a zaručují tedy například nízké tření navržených objektů aj.

V počítačové grafice se nejčastěji používá aproximace po částech (podobně jako interpolace), a to obyčejně kubikami (tedy křivkami, které jsou generovány polynomy třetího řádu). Důvody pro to jsou dva. Kubiky jsou dostatečně "pružnými" křivkami, aby se jimi dalo vyjádřit téměř vše, co je v praxi potřeba. Dalším důležitým faktorem je, že stupeň polynomu tři umožňuje velmi rychlý výpočet výsledné křivky, což je výhodné pro její interaktivní tvorbu.

2.6.1 Bézierovy křivky

Patrně nejpopulárnější aproximační křivky jsou používané pro modelování ve dvou rozměrech, ale i pro definici trojrozměrných objektů, jsou Bézierovy křivky. Tyto křivky se často používají i při definici písma.

Vlastností Bézerovy křivky je, že při změně polohy jediného řídicího bodu P_i dojde ke změně tvaru celé křivky, jak je patrné na obrázku 15 vpravo při posunutí bodu P_3. Tato vlastnost je jedním z důvodů, proč se tyto křivky dělí na segmenty nižšího stupně (nejčastěji kubiky), které se postupně navazují.



Obrázek 12: Bézierovy křivky 3. stupně (modrá) a 7. stupně (červená)

Bézierovy křivky jsou pro svou snadnou manipulovatelnost často používány v CAD systémech (např. AutoCAD). Jejich jednoduchá tvorba zaručuje jejich snadnou manipulovatelnost (pouhou změnou polohy bodu se mění tvar výsledné křivky). Z hlediska programátora je podstatné i to, že se ve všech uvedených vztazích vyskytuje pouze násobení a sčítání (kombinační čísla v Bernsteinových polygonech jsou spočteny dopředu konstruktérem systému, který se rozhodl, jaký stupeň křivek bude volit). Díky této vlastnosti jsou programy, které používají Bézierovy aproximace, velmi rychlé.

2.6.2 Algoritmus de Casteljau

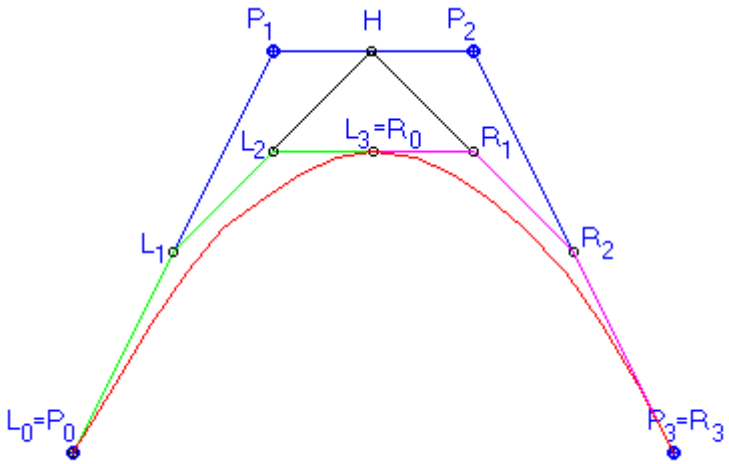
Jinou metodou jak vypočítat Bézierovu křivku stupně *n*, je použít rekurzivní *algoritmus de Casteljau*. Zásadní pro aplikaci tohoto algoritmu je rozdělení Bézierovy křivky na dvě části. Hodnota $P(t)$, která je vstupem algoritmu, určuje bod, ve kterém dojde k jejímu rozdělení a rekurzívní výpočet postupně generuje nové body řídicích polygonů dvou nových křivek. Bod křivky o souřadnicích $Q(t)$ se vypočítá podle rekurentního vztahu

$$P_{j,i}(t) = (1-t)P_{j-1,i-1} + tP_{j,i-1}, \tag{21}$$

kde $i = 1,2, \ldots,n$ a $j = i,i+1,\ldots,n$. Vstupem tohoto algoritmu jsou body řídicího polygonu. Dosadí se za $P_{i,0} = P_i$, a rekurentní vztah (odkaz na rovnici nad textem) postupně vypočítává body $P_{i,j}$. V posledním kroku výpočtu je koeficient $P_{n,n}(t)$ roven hodnotě křivky v bodě $Q(t)$ .Bézierova křivka může být pomocí de Casteljau algoritmu rozdělena

na dvě části v libovolném místě, tzn. Pro zvolenou hodnotu parametru $t \in \langle 0,1 \rangle$. Nejdůležitější volbou je dělení ve středu křivky, tj. $t = 1/2$.

Nechť původní křivka *n*-tého stupně je určena řídicími body $P_0,\ldots,P_n$. Při dělení vytvoříme dvě skupiny řídicích bodů $L_0,\ldots,L_n$ a $R_0,\ldots,R_n$, které určují příslušné části rozpůlené křivky. S využitím výpočetního postupu získáme nové řídicí body, což není nic jiného než vytvoření bodů nových půlením úseček. Pokud tento postup opakujeme, konverzuje polygon určený řídicími body k Bézierově křivek.



Obrázek 13: Algoritmus de Casteljau

2.6.3 Racionální Bézierovy křivky

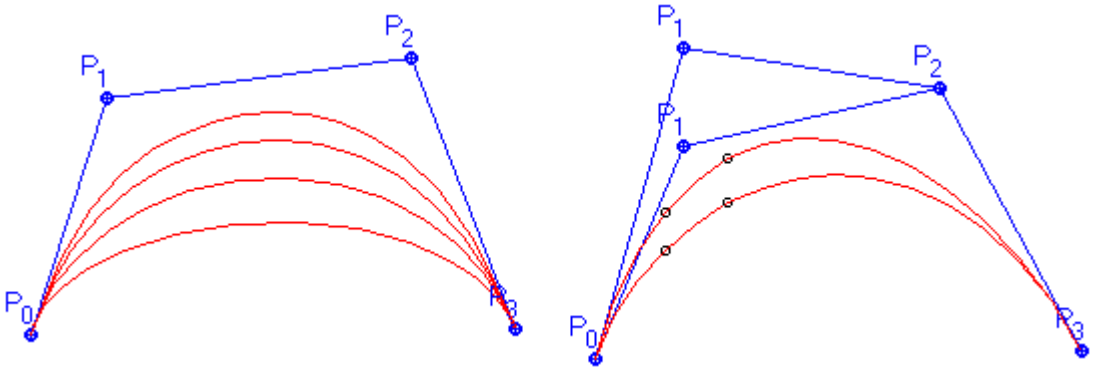
Poměrně novou metodou v CAGD je teorie racionálních Bézierových křivek. Jak napovídá název, jedná se o zobecnění Bézierových křivek. Při klasickém zadávání Bézierových křivek zadává uživatel řídicí polygon a program podle jeho požadavků buď proloží body křivkou patřičného řádu, nebo výslednou křivku složí z křivek řádu nižšího. Pokud chce uživatel změnit tvar křivky, musí identifikovat bod a změnit jeho polohu.

To nemusí být vždy výhodné. Pokud se pracuje v rovině, neprojeví se tento problém tak, jako v trojrozměrném prostoru. Identifikace bodu v prostoru však není vůbec triviální a hlavně nebývá nijak názorná. Přirozenou se potom jeví metoda, která každému bodu řídicího polygonu přiřadí reálné číslo, jehož změnou se mění tvar křivky. Racionální

Bézierova křivka je určena posloupností bodů $P_0, P_1, \ldots, P_n$ a posloupností reálných čísel $w_0, w_1, \ldots, w_n$. Bézierova racionální křivka řádu *n* je potom určena vztahem:

$$P(t) = \frac{\sum_{i=0}^{n} \omega_i P_i B_i^n(t)}{\sum_{i=0}^{n} \omega_i B_i^n(t)} = \sum_{i=0}^{n} R_i^n P_i \tag{22}$$

kde $i = 0,1,\ldots,n$ $\hat{I}N, w \geq 0$ a B_i^n jsou Bernsteinovy polynomy a R_i^n jsou racionální Bernsteinovy polynomy.



Obrázek 14: Změna tvaru křivky s koeficienty v bodech (vlevo) a translace (vpravo)

Největším přínosem racionálních Bézierových křivek je bezesporu možnost manipulace s tvarem křivky bez změny polohy bodů řídicího polygonu.

2.6.4 Coonsovy kubiky

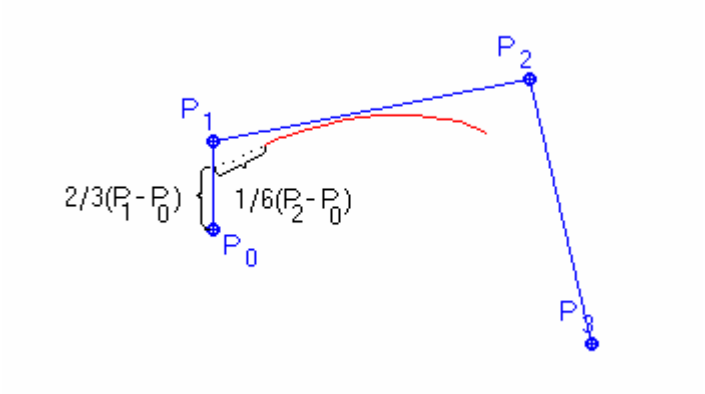
Coonsova kubika zaujímá vedle Bézierových křivek významné postavení v historii parametrických křivek a ploch. Je běžně používána při tvorbě po částech skládaných polynomiálních křivek, úvahy o uniformní a neuniformní parametrizaci výrazně přispěly k zobecněnému pohledu na vlastnosti a tvorbu polynomiálních bází a v nespolední řadě se stala základem obecného aparátu NURBS. S. A. Coons definoval metodu, která má své silné použití díky dobrým geometrickým vlastnostem hlavně v navrhování ploch. Její princip je však vhodné vyložit na modelování křivek.

Coonsova kubika, neboli uniformní neracionální B-spline (zkráceně B-spline), se zadává stejně jako kubika Bézierova čtyřmi řídicími body P_0, P_1, P_2 a P_3 a vztahem

$$P(t) = \frac{1}{6}\left[P_0C_0 + P_1C_1(t) + P_2C_2(t) + P_3C_3(t)\right] = \frac{1}{6}\sum_{i=0}^{3} P_iC_i(t) \quad (23)$$

kde C_0, C_1, C_2, C_3 jsou kubické polynomy tvaru:

$$\begin{aligned} C_0(t) &= (1-t)^3, \\ C_1(t) &= 3t^3 - 6t^2 + 4, \\ C_2(t) &= -3t^3 + 3t^2 + 3t + 1, \\ C_3(t) &= t^3. \end{aligned} \quad (24)$$



Obrázek 15: Coonsova kubika

Největší výhoda Coonsových kubik se stane zřejmou teprve v okamžiku, kdy je použijeme pro skládání aproximačních křivek. Uvažujme řídicí polygon složený z bodů $P_0, P_1, \ldots, P_n$. Budeme-li výslednou křivku skládat z Coonsových oblouků vždy tak, že pro jeden oblouk použijeme vrcholy $P_0P_1P_2P_3$, pro další $P_1P_2P_3P_4$ atd., získáme křivku, která se nazývá B-splajn.

Vlastností B-splajnu je, že má ve všech vnitřních bodech spojitost druhého řádu. Z hlediska konstrukce je pro tímto způsobem vytvářenou křivku výhodné i to, že změnou jednoho bodu dojde pouze k lokální změně čtyř oblouků, jejichž konstrukce se bod účastní.

2.6.5 Spline křivky

Spline křivka stupně *n* je po částech polynomiální křivka, která je třídy C^n. Termín spline pochází od pružného pravítka (křivítka), které tyto křivky modelování.

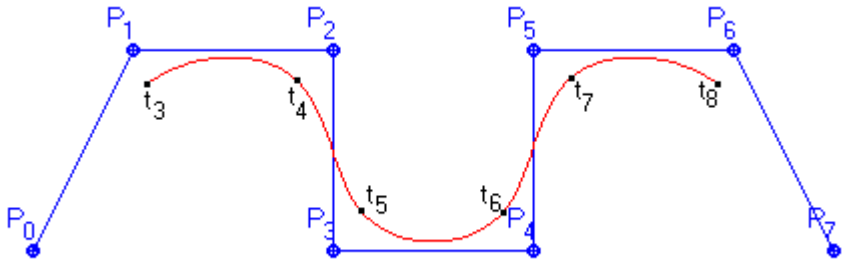
Přirozený spline je spline, který interpoluje své řídicí body. Přirozený kubický spline je tedy interpolační křivka skládající se z polynomiálních křivek stupně tři, která je ve svých

uzlech C^2 spojitá. Výpočet přirozeného spline vede k úloze výpočtu inverzní matice k matici $(m+1)\times(m+1)$, kde *m* je počet bodů řídicího polygonu. Nevýhodou těchto křivek je, že změnou polohy jediného definujícího bodu se změní tvar celé křivky.

V počítačové grafice jsou nejčastěji reprezentovány spline křivky jako *B-spline kubiky*, které nejsou přirozenými spline křivkami, protože se jedná o křivky aproximační.

2.6.6 Uniformní kubický B-spline

Uniformní kubický B-spline se také nazývá *kubický Coonsův B-spline* (obrázek 16). Vznikne navázáním Coonsových kubik takto: segment Q_i je určen body $P_{i-3}, P_{i-2}, P_{i-1}$ a P_i. Následující segment Q_{i+1} je složen z bodů P_{i-2}, P_{i-1}, P_i a P_{i+1}, tedy ze tří posledních bodů segmentu Q_i a z jednoho bodu nového atd. Coonsův B-spline je určen *n* + 1 body.



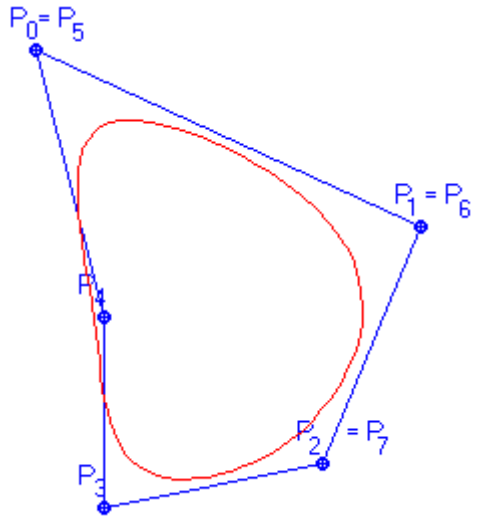
Obrázek 16: Coonsův kubický B-spline

Porovnáme-li vztahy pro tečné vektory a vektory druhých derivací dvou po sobě následujících segmentů Q_i a Q_{i+1}, zjistíme, že segment Q_{i+1} vychází z posledního bodu segmentu Q_i a že jsou identické první a druhé derivace v tomto bodě. Křivka je tedy v uzlech C^2 spojitá.

Využijeme-li chování Coonsových kubik při násobnosti bodů, tak posloupností $P_0, P_0, P_0, \ldots, P_1, \ldots, P_{n-1}, P_n, P_n, P_n$ zajistíme, že výsledný kubický spline bude procházet krajní body svého řídicího polygonu, ovšem za tu cenu, že první segment bude úsečkou spojující bod P_0 s bodem ležícím na jedné šestině vzdálenosti P_0P_1 a analogicky na konci řídicího polygonu.

2.6.7 Uzavřený B-spline

Uzavřený Coonsův B-spline též nazývaný *Periodický spline* (obrázek 17) je vytvořen opakování prvních tří bodů řídicího polygonu na jeho konci. Pro kubický B-spline je tedy řídicím polygonem uzavřené křivky posloupnost bodu: $P_0, P_1, \ldots, P_n, P_0, P_1, P_2$.



Obrázek 17: Uzavřený Coonův B-spline

Mezi důležité vlastnosti B-splajn křivek patří invariance vůči rotaci, translaci a změně měřítka, B-splajn křivka leží celá ve své konvexní obálce a její segmenty leží v konvexních obálkách svých řídicích polygonů. Více [3].

2.6.8 NURBS

Neuniformní racionální B-splajn křivky (NURBS - *non uniform rational B-spline*) jsou dvojím zobecněním B-splajn křivek. Termín *neuniformní* je odvozen od vzdálenosti uzlů ve smyslu parametru *t*, která nemusí být u těchto křivek konstantní. *Racionalita* znamená, že body jsou reprezentovány svými *homogenními souřadnicemi*.

Křivka NURBS je určena $n + 1$ body $P_i, i = 0, \ldots, n$ řídicího polygonu, řádem B-spline k (nejvyšší stupeň je $k - 1$) a uzlovým vektorem **U** délky $n + k + 1$. Uzlový vektor je tvořen posloupností neklesajících reálných čísel – uzlových hodnot $t_0 \leq t_1 \leq \ldots, \leq t_{n+k}$. Uzlové hodnoty v této posloupnosti se mohou opakovat. Příkladem uzlového vektoru

s uniformním rozložením uzlových hodnot je $U_1 = \{-2,-1,0,1,2,3,4,5\}$, tzn. Okrajový uzlový vektor s opakovanými uzlovými hodnotami může mít tvar $U_2 = \{0,0,0,1,1,1\}$.

Obecně je křivka NURBS určena vztahem:

$$Q(t) = \frac{\sum_{i=0}^{n} w_i P_i N_{i,k}(t)}{\sum_{i=0}^{n} w_i N_{i,k}(t)}, \tag{25}$$

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

$$N_{i,1}(t) = \begin{cases} 1 & \text{pro} \quad t_i \le t < t_{i+1} \\ 0 & \text{jinde} \end{cases} \tag{26}$$

$$Ni,k(t) = \frac{t - t_i}{t_{i+k-1} - ti} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t) \text{ pro } t_i < t_{i+1+k}, 0 \le i \le n. \tag{27}$$

Během výpočtu se vyskytují i výrazy, které mají ve jmenovateli nulu. Jejich hodnota je definitoricky položena rovna nule. Jiný zápis využívá *racionální B-spline báze*

$$Ri,k = \frac{w_i N_{i,k}(t)}{\sum_{j=0}^{n} w_j N_{j,k}(t)}. \tag{28}$$

Křivku NURBS podle rovnice (rovnice nurbs Q(t) lze potom jednodušeji zapsat

$$Q(t) = \sum_{i=0}^{n} P_i R_{i,k}(t) \tag{29}$$

Obdobně jako u Bézierových křivek a bázových Bernsteinových polynomů [4] mají polynomy NURBS báze N_i^n následující vlastnosti:

1. $\forall i, n \in N \cup \{0\}$ a $t \in \langle 0,1 \rangle$ je $N_i^n(t) \ge 0$. (30)

2. $\sum_{i=0}^{n} R_n^n(t) = 1$ pro $t \in \langle 0,1 \rangle$.

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

První vztah zaručuje nezápornost polynomů B-spline báze. První a druhý pak zaručují, že výsledný křivka bude vždy ležet v konvexní obálce bodů řídicího polygonu. Třetí vztah je rekurentní definicí bázového polynomu řádu k (stupně $k - 1$) pomocí lineární kombinace dvou po sobě následujících bázových polynomů řádu k (stupně $k - 2$). Třetí vlastnost je základem Cox-deBoorova algoritmu pro výpočet bodu na NURBS křivce.

2.6.8.1 Cox-deBoorův algoritmus

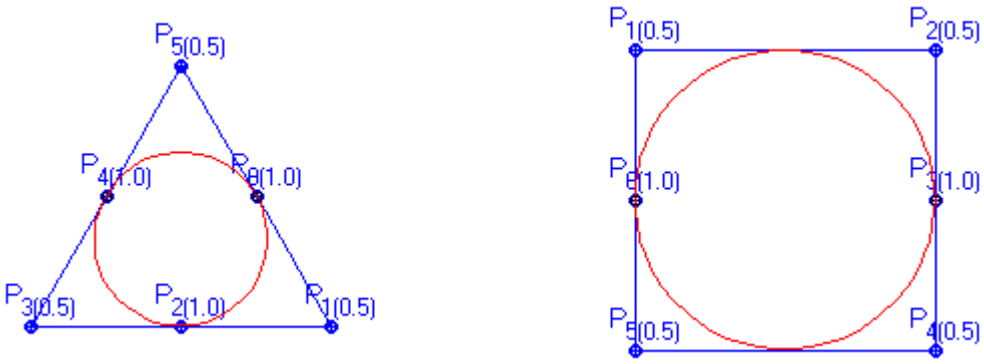
Algoritmus je zobecněním de Casteljau (kap. 2.6.2) algoritmu a je navržen pro spline křivky s neuniformní parametrizací. S využitím rekurentní definice bázových polynomů je bod na křivce NURBS konstruován postupným dělením úseček té části řídicího polygonu, která ovlivňuje segment křivky, k němuž tento bod náleží. Po nalezení příslušného intervalu v uzlovém vektoru, je aplikováno dělení, které respektuje neuniformní časové intervaly určené uzlovým vektorem. Více na [odkaz na odborný text]

Jedním z nejdůležitějších modelovacích prostředků, který poskytují NURBS křivky, je možnost vložení nových uzlových bodů. Namísto zvyšování stupně polynomu umožňují nově vložené uzlové vody společně s příslušně pozměněnými body vymezit ty části křivky, které chceme lokálně ovlivnit.

Křivky NURBS mají dále tyto vlastnosti:

1. Při zadání uzlového vektoru podle **U** (zmíněného na začátku kapitoly) křivky procházejí prvním a posledním bodem řídicího polygonu.
2. Leží v konvexní obálce svého řídicího polygonu, stejně tak jejich jednotlivé segmenty leží v obálkách svých řídicích polygonů. Změna polohy, resp. váhy jednoho bodu má tedy vliv pouze na část křivky.
3. Jsou invariantní vůči transformacím a především vůči rovnoběžnému a středovému promítání.

4. Umožňují přesné vyjádření kuželoseček.
5. Pro uzlový vektor $U = \{0,0,\ldots,0,1,1,\ldots 1\}$ a pro stejný počet nul a jedniček jsou normalizované B-spline báze rovny Bernsteinovým polynomům a NURBS je racionální Bézierovou křivkou. Pro hodnoty $w_i = 1$ je NURBS Bézierovou křivkou.



Obrázek 18: Kružnice jako NURBS definována 6 body v různé topologii

Zejména čtvrtá vlastnost je důležitá pro tvorbu jednotlivých datových struktur, které reprezentují jak "klasické" geometrické tvary, jako jsou např. kružnice (Obr. 18) či parabola, tak i volné (*angl. free-form*). Nevýhodou je, že reprezentace klasických objektů není jednoduchá.

2.6.8.2 Výhody NURBS křivek

Každý, kdo alespoň někdy zabrousil do grafiky, se s NURBS křivkami i plochami setkal. V současné době se NURBS používají při konstrukci obecných tvarů – v automobilovém designu, letectví, filmové animaci. Ale proč? Výhodné na nich jsou:

- neomezené konstrukční možnosti – modifikace polohy bodu, váhy, uzlového vektoru, stupně
- rychlý, stabilní algoritmus, přesný
- lokální kontrolovatelnost – při změně (např. bodu, váhy) dochází pouze k lokální změně křivky, zrychluje výpočet a vykreslování
- zachování spojitosti při změnách

- projektivní invariantnost – při základních transformacích – rotace, posunutí, zkosení – stačí zobrazit pouze řídící body a křivku znovu vypočítat, zrychluje výpočet
- konstrukce kuželosečkových oblouků

2.7 Použití křivek ve 3D

Křivky, popsané v předchozích kapitolách, se díky svým vlastnostem hojně využívají ve 3D a to především při generování povrchů (je to způsobeno tím, že křivky a povrchy jsou si v jistých směrech podobné).

Tato kapitola nepopisuje detailní konstrukce 3D těles a jejich zobrazování, avšak má pouze ukázat jak lze některé z uvedených křivek využít ke konstrukci třírozměrných těles. Pro hlubší seznámení s problematikou zobrazování ve 3D tímto odkazuji čtenáře na literaturu tímto problémem se zabývající.

2.7.1 Konstrukce a zadání plochy

Plocha, ať už plošná nebo prostorová může být zadána:

- Hraničními křivkami
- Síti bodů
- Kinematicky vytvořenými křivkami
    - Rotační plochy (vzniklé rotací křivky okolo přímky)
    - Plochy vzniklé skládáním pohybů – posun, rotace
- Analyticky předepsané plochy – parametrické, explicitní, implicitní vyjádření

2.7.2 Operátory pro generování povrchů ze křivek

Existuje řada známých operací jako např. **tažení** (angl. extrusion), **otáčení** (angl. revolution) či **vedení** (angl. ruled) pro tvorbu povrchů. Pro tvorbu povrchů pomocí těchto operátorů se nejčastěji používá křivka NURBS, kterou se např. předdefinuje profil či obrys daného tělesa (2D) a z něj se pomocí operátorů vygeneruje 3D objekt.

2.7.2.1 Tažení

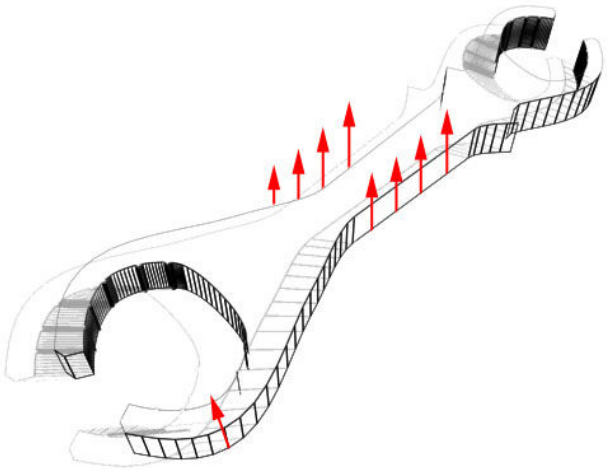
Povrchy těles, které se vytvářejí pomocí operátoru tažení jsou navrhovány tak, že se pomocí křivky nadefinuje profil (tzv. profilová křivka) a pak se pomocí operátorů tažení "táhne" tato křivka v určitém směru. Jestliže má profilová křivka tvar

$$C(u) = \sum_{i=1}^{n} R_i^p(u) Q_i \tag{31}$$

pak vzorec pro tažený povrch je :

$$S(u,v) = \sum_{i=1}^{n} \sum_{j=1}^{2} R_i^p(u) R_j^2(v) P_{i,j} \tag{32}$$

Kde v (konstantní isoparametrická křivka) je původní profil a u (konstantní isoparametrická křivka) je úsečka definující směr tažení, zde je použita lineární NURBS křivka (t.j. $R^2(v)$).

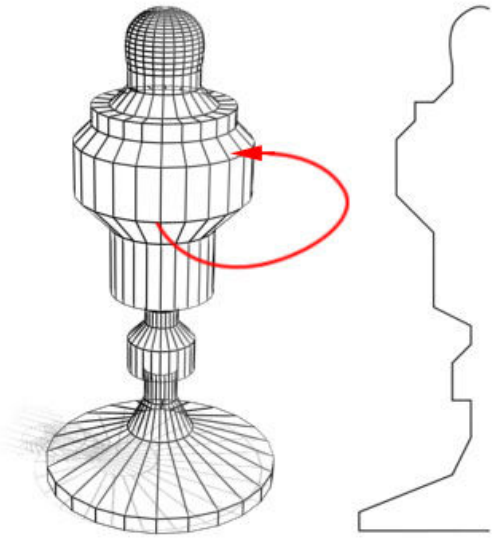


Obrázek 19: Klíč vytvořený pomocí tažení

2.7.2.2 Otáčení

Povrchy vytvářené pomocí otáčení jsou generovány za využití křivky (např. NURBS), která leží v *xz*-rovině. Povrch je vygenerován tak, že se křivka otočí o 360° okolo osy z. Požadovaný povrch (Obr. 19) má tvar:

$$S(u,v) = \sum_{i=1}^{9}\sum_{j=1}^{n} R_i^3(u) R_j^k(v) P_{i,j} \tag{33}$$



Obrázek 20: Povrch vytvořený pomocí otáčení

Obrázek č.20 ukazuje model vytvořený pomocí otáčení profilu. Řídící síť se sestává z 10 rohových řídicích bodů a 9 středových bodů (na hranách). Dále pak každý z řídicích bodů je použit pro generování 9 řídicích bodů čtverce.

2.7.2.3 Vedené (řízené) povrchy (Přímkové plochy)

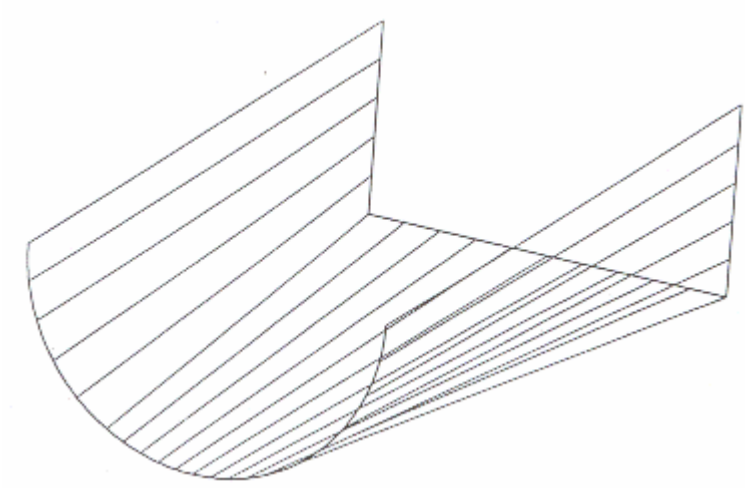
Povrchy vytvářené pomocí operátoru vedení (řízení) využívají toho, že každý bod, který koresponduje s daným parametrem t křivky, je propojen pomocí úsečky s jiným bodem na jiné křivce.

Takto se například vytvářejí povrchy mezi dvěmi křivkami NURBS :

1. Zvýší se stupeň křivky, která je bližší pozorovateli na stupeň křivky vzdálenější.
2. Sjednotí se uzlové vektory obou křivek.
3. Křivky se osadí novým uzlovým vektorem a sjednotí tak, aby měly stejný počet řídicích bodů.

4. Povrch se vytvoří za pomocí řídicích bodů a uzlových vektorů obou křivek. Tento postup je podobný jako postup při tvorbě povrchu operátorem tažení kdy každá křivka se stává řádkem hodnot v matici řídicích bodů.

Obr. 12.3 zobrazuje vedený povrch mezi půlkružnicí a polovinou čtverce. Tento vedený povrch byl vytvořen pomocí u = konstantních isoparametrických čar (přímé čáry) rovnoměrně rozmístěných okolo čtverce a nerovnoměrně okolo kružnice.



Obrázek 21: Vedený povrch

2.7.3 Plochy určené polynomy

V předchozí kapitole byla zmíněna tvorba povrchů 3D těles za využití jedné (maximálně dvou) křivek a případného operátoru. Tato kapitola se zabývá generování ploch definovaných pomocí polynomů, které se používají pro vytváření 2D křivek. Následující text potvrzuje, že pomocí 2D křivek se dají jednoduše generovat 3D plochy.

2.7.3.1 Interpolační plochy

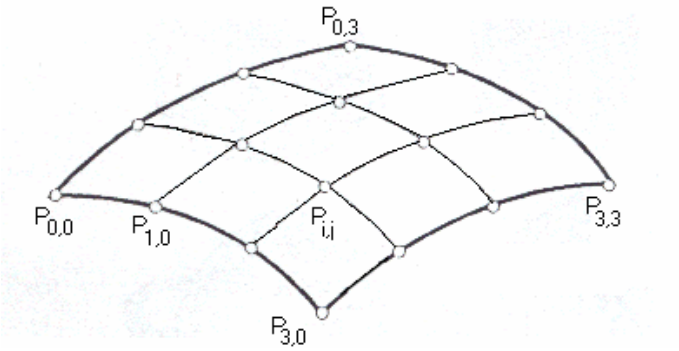
Interpolační plochy procházejí křivkami nebo body, které plochu určují. V technické praxi se používají převážně plochy přímkové, které jsou určeny vektorovým polynomem

$$P(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} a_{ij} u^i v^j \tag{34}$$

Plochy dále dělíme podle toho, kolika křivkami jsou určeny

- Plochy určené dvěma okrajovými křivkami - Bilineární a Bikubické plochy

- Plochy určené čtyřmi okrajovými křivkami - Interpolační plochy



Obrázek 22: Interpolační plocha určená 16-ti body

2.7.3.2 Aproximační plochy

Motivy, které vedly k rozvoji metod pomocí aproximace, jsou charakterizovány především shanou odstínit uživatele od matematiky a zaručit maximální jednoduchost tvorby generovaných ploch. Uživatel tak obyčejně zadává řídicí body (např. rohy), okrajové křivky, tečné vektory, či zkruty a jejich pomocí modeluje tvar výsledné plochy.

Mezi aproximační plochy patří:

- Hermitovské plochy
- Dvanáctivektorové plochy
- Šestnáctivektorové plochy
- Plochy spojené dvěmi (okrajovými) křivkami

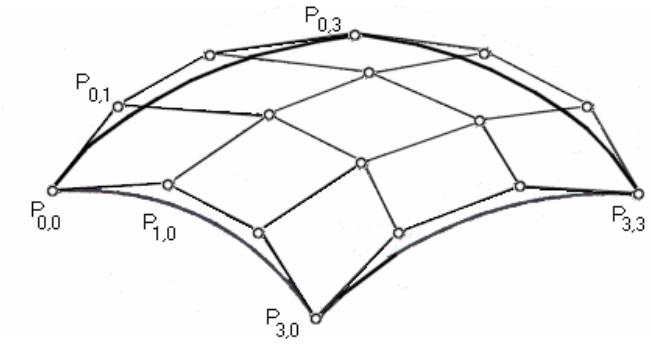
2.7.3.3 Bézierovy plochy

Některé modelovací systém používají k reprezentaci povrchů těles Bézierovy plochy. Tyto plochy jsou snadno diferencovatelné, jednoduše a intuitivně se modelují a relativně snadno se vypočítá průsečík s paprskem. I když se v současných systémech používají pro uchování ploch složitější reprezentace založené na technologii NURBS, tradiční uživatelský pohled a možnost práce s Bézierovými modelovacími nástroji jsou běžnou součástí těchto systémů. Je to umožněno především díky tomu, že Bézierovy plochy jsou speciálním případem ploch NURBS.

Bézierova plocha $n \times m$-tého stupně je určena $(n + 1) \times (m + 1)$ řídicími body $P_{i,j}$ a vztahem:

$$P(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} P_{i,j} B_i^m(u) B_j^n(v) \tag{35}$$

Kde $P_{i,j}$ tvoří řídící síť Bézierovy plochy a $B_i^m(u), B_j^n(v)$ jsou Bernsteinovy polynomy. Výraz $B_i^m(u)B_j^n(v)$ tvoří Bernsteinovy polynomy dvou proměnných u, v stupně (m, n). Platí-li: $n = m = 3$, pak Bézierovu plochu nazýváme Bézierovým bikubickým plátem.



Obrázek 23: Bézierův bikubický plát

2.7.3.4 B-spline plochy

B-spline plochy, které jsou zobecněním B-spline křivek, se velice snadno navazují a jsou proto pro modelování daleko výhodnější, než Hermitovké a Bézierovy pláty. B-spline plochy n-tého stupně zaručují C^{n-1} spojitost ve všech svých bodech a není nutné omezovat některé jejich řídicí body vnějšími podmínkami jako v případě Bézierových ploch. Změnou jediného řídicího bodu měníme tvar vždy pouze části B-spline plochy. B-pline plochy se zadávají sítí řídicích bodů. Na rozdíl od Bézierových plátů se však navazující B-spline plát definuje použitím $m \times (n - 1)$ bodů plátů předchozího a přidáním pouze m dalších bodů.

Tímto způsobem tedy dochází k překrytí reprezentací plátů. Část reprezentace jednoho plátu je součástí plátu následujícího, v případě kubik se jedná o tři sloupce nebo tři řádky v mapě plochy. Změnou polohy jediného řídicího bodů tak manipulujeme s více pláty, počet ovlivněných plátů závisí na zvoleném stupni bázových polynomů obdobně jako při modelování B-spline křivek. To však není nijak na obtíž. Podstatné je, že tato změna je lokální a nezpůsobuje změnu tvaru celé plochy.

Mezi vlastnosti B-spline ploch patří:

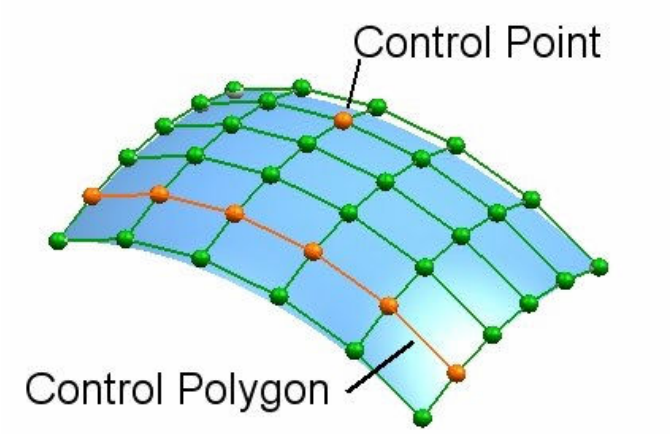
- Plocha leží celá v konvexní obálce svých řídicích bodů.
- Při změně polohy jediného řídicího bodu změní tvar pouze ty pláty, které jsou tímto bodem určeny.
- Plocha obecně neprochází krajními body řídicí sítě, toho lze docílit násobností řídicích bodů a u neuniformních ploch násobností uzlů.
- Jsou invariantní k lineárním transformacím (otáčení posunu, změně měřítka, zkosení).

Bikubické B-spline plochy

Nejjednodušší z B-spline ploch jsou bikubické B-spline plochy (jedná se o neracionální a neuniformní B-spline plochy). Tyto plochy se používají ve složitějších modelovacích programech, avšak jedinou jejich výhodou oproti Bézierovým plochám je, že jsou spojité C^2 bez nutnosti zadávat nějaké vnější omezující podmínky na polohu řídicích bodů. Tyto plochy nejsou invariantní k perspektivnímu promítání.

NURBS plochy

Neuniformní racionální B-spline plochy – NURBS, jsou zobecněním B-spline ploch a představují dnes průmyslový standard v geometrickém modelování. NURBS umožňují definovat širokou třídu ploch, mezi něž patří jak volně tvarovatelné plochy na bázi racionálních polynomů (*free from surfaces*), ale i plochy založené na přímkách, kuželosečkách apod. I tyto tvary se dají vyjádřit v NURBS (obrázek 24) reprezentaci.



Obrázek 24: NURBS plocha

3 BLENDER

Kvalitní vizualizační 3D programy jsou velmi, velmi drahé. Aby také ne, jejich vývoj je docela náročný a klientů, kteří si program koupí je relativně málo (vzhledem například k Adobe Photoshopu, který se používá všude, kde se pracuje s grafikou). Trendem dnešní doby je poskytovat věci zdarma. Platí to i o softwaru pro tvorbu 3D grafiky.

Blender je multiplatformní open source aplikace zaměřená na vytváření 3D modelů, animací, rendering, postprodukční činnost a v neposlední řadě interaktivních aplikací.

Multiplatformní znamená, že Blender lze spustit nejen v systému Windows, ale i pod Linuxem, na Mac OS X a mnoha dalších. Open source znamená, že je program nejen zcela zdarma a to i pro komerční využití, ale také že si můžeme stáhnout kompletní zdrojové kódy, zkompilovat je na vlastní sestavě pro optimalizaci výkonu, libovolně je upravovat a případně se aktivně podílet na dalším vývoji Blenderu.

Kromě nástrojů pro modelovaní, animaci a renderování obsahuje Blender také GameEngine, ve kterém je možné vytvářet interaktivní prezentace, průchozí vizualizace např. interiérů domů a počítačové hry, vše přímo v Blenderu pomocí interního grafického editoru s možností doplnění kódem v objektově orientovaném programovacím jazyce Python.

3.1 Rozhraní

V porovnání s jinými 3D aplikacemi má Blender jednoznačně nestandardní uživatelské prostředí. Flexibilita programu dovoluje potencionálnímu grafikovi nastavit si rozhraní stejně podobně jak byl zvyklý u jiných programů, nebo si může interface nadefinovat dle vlastního uvážení.

Předchozí verze Blenderu byly pro začátečníky kamenem úrazu, jelikož většina nástrojů byla schovány pod klávesovými zkratkami. I když byl tento způsob tvorby rychlejší, mnoho začínajících uživatelů tato skutečnost odradila. Nové verze jsou vybaveny pop-up nabídkami, jež Vám tlačítkově zpřístupní funkce pro rotaci, změn velikostí a pohyby.

Hlavní atributy interface Blenderu jsou:

- Plně nastavitelné uživatelské prostředí, ve kterém si můžeme vytvořit libovolný počet nepřekrývajících se oken.
- rozdělení do oken pro modelování, animační křivky, *outliner*, nelineární videostřih, editování UV map, animování postav (*pose editor*, *NLA editor*), souborový manažer, textový editor, Python editor ...
- funkce „zpět“ je již v několika posledních verzích dostupná ve všech úrovních.
- databázový systém umožňující optimální management scény, instance a dynamické propojování projektů v různých souborech.
- lokalizace do několika jazyků, včetně možnosti zapnout co vše má být lokalizováno a co ponecháno v angličtině (např. tlačítko anglicky, vysvětlující popisek v jiném jazyce) a možnosti vytvářet si vlastní jazykové sady.
- zabudovaný textový editor sloužící k poznámkám a programování Python skriptů.
- interface je stejný na všech operačních systémech, pro které Blender existuje..

3.2 Modelování

- práce s polygony, NURBS plochami, Bézier a B-spline křivkami, *Metabally*, vektorové fonty (TrueType, PostScript, OpenType).
- Tvorba *Catmull-Clark* povrchů (ekvivalent k *meshsmooth*) s editovatelnou ostrostí/oblostí hran.
- editování polygonálního *meshe* s volitelnou selekcí *vertexů*, hran nebo ploch.
- u objektů typu *mesh* podpora Boolean operaci – CGS geometrie.
- mnoho editačních funkcí editovací funkcí jako *extrude*, *bevel*, *cut*, *spin*, *screw*, *warp*, *subdivide*, *noise*, *smooth*…S každou novou verzí přichází alespoň jeden nový modelační nástroj.
- díky Python skriptů do programu implementovat vlastní nástroje pro modelování.

3.3 Animace

- deformační obálky (kosti či skeletony) s dopřednou i inverzní kinematikou (FK, IK), *autoskining* a interaktivní nastavování vah deformačních skupin pomocí nástroje *WeightPaint.*
- několik typů *constraints* určený pro sledování pohybujícího se objektu.
- *pose editor* pro animaci postav.
- editor nelineární animace (NLA) , automatizace posunu postavy ze zacyklenou animací chůze (*walkcycle*) podél definované cesty (*path*).
- animace *vertex keys* a *relative vertex keys* (obdoba *morph targets*) s ovládacími posuvníky.
- částicové generátory - *particle* efekty s deformátory podle větru, gravitace, magnetické přitažlivosti/odpuzování a detekcí kolizí
- nástroj *SoftBodies* (např. simulace látek) s detekcí kolizí.
- animovatelné *lattice* deformace.
- podpora "*motion curve*" (změny animace pomocí křivek) i tradičního *key-frame* (pomocí klíčových snímků) editování
- podpora zvuku a nástrojů pro synchronizaci zvuku a obrazu.
- možnost doprogramovat si pomocí Pythonu animační nástroje případně „řízené animace“ dle potřeby.

3.4 Rendering

- implementovaný velice rychlý *ray tracer*.
- Integrovaná podpora externího *ray traceru Yafray*.
- *oversampling*, *motion blur*, postprodukční efekty (*glow*, *zblur*…) *fields*, *environment* mapy, *halo*, *lens flare*, (efekty) mlha…
- několik materiálových *shaderů* pro difusní a specularitní kanál- *Lambert*, *Phong*, *Orean-nayar*, *Blinn*, *Toon*, *Minnaert*, *Wardlso*
- *Edge rendering* pro efekt vytažených okrajů (*cartoon*)

- procedurální textury.
- *Ambient Occlusion.*
- výpočet Radiosity.
- Množství exportních skriptů do dalších *raytracerů*, např. pro *Povray*, *Renderman*(RIB) *Virtualight.*
- UV editor s několika metodami pro *unwrap*.

3.5 Realtime 3D/tvorba her

- Součástí je grafický editor pro naprogramování logiky aplikace/hry bez nutnosti programovat.
- detekce kolizí a simulace dynamiky.
- přístup do enginu přes Python skripty pro složitější logiku, umělou inteligenci apod.
- podpora všech povrchových módu *OpenGL*, včetně průhlednosti, animovatelných reflexních map apod.
- přehrávání her a interaktivních 3D aplikací bez kompilování a přepočítání.
- audio využívající *SDL toolkit*.
- *multi-layering* scén pro plovoucí interface.

3.6 Soubory a podporované formáty

- všechna data ve scéně se ukládají do jediného souboru s příponou *.blend*
- *.blend* formát podporuje kompresi, digitální podpisy, zakódování, dopřednou i zpětnou kompatibilitu a může být použit jako knihovna, do níž přistupujete z jiného souboru.
- čte/zapisuje formáty *tga, jpg, png, Iris (+ Zbuffer), sgi Movie, iff, avi* a *Quicktime gif, tiff, pds, mov*.

- nativní podpora importu a exportu *dxf* formátu, *Inventor* a VRML souborů, přes Python skripty je umožněn import/export do množství dalších formátů (*obj, lwo, cob, 3ds*…), ty hlavní skripty jsou již součástí staženého Blenderu.
- vytvoření samospustitelných souborů (*.exe*) s interaktivními 3D aplikacemi, hrami apod., nebo je můžete přehrávat ve internetovém prohlížeči s příslušným zásuvným modulem (*pluginem*).

Vzhledem ke skutečnosti, že je Blender multiplatformní aplikace, neomezuje nás její využívání na rozličném operačním systému. Blender podporuje následující platformy:

- Windows 98, ME, 2000, XP, Vista
- Mac OS X
- Linux (i386)
- Linux (PPC)
- FreeBSD 5.3 (i386)
- SGI Irix 6.5
- Sun Solaris 2.8 (sparc)

V současné době je dostupná nová verze s označením Blender 2.44, která odstraňuje několik *bugů* (programové vady) a je plně kompatibilní s 64 bitovými CPU. Samozřejmě, jak je tomu zvykem s každou novou verzí, byly přidány nové modifikátory nejen pro modelování:

- *Composite nodes* – nástroj pro korekci *Bright/Contrast* a *Gamma* světlosti na obrazovém výstupu.
- podpora *Subsurface scattering* (*SSS*) – nový materiál simulující povrchy jako jsou kůže či mléko.
- *Revamp, Smooth, Část*

3.7 Modelování v Blenderu

Kapitola bude stručným a spíše obecným popisem modelovacích technik v prostředí 3D grafického programu Blender. Zmíněny budou základní primitiva a možnosti, jak s nimi manipulovat, deformovat a význam CSG operací. Jak se dají tyto metody využít naleznete dále v praktické části.

3.7.1 Objektový a editační mód

Blender je specifický ve dvou základních módech, jejichž frekventovaným používáním tvoříme 3D scénu. Vložíme-li některý ze základních objektů, obvykle jeho tvar podle potřeby upravujeme. Tyto úpravy můžeme prováděcí ve dvou základních módech – objektovém a editačním (u některých objektů nemá editační mód smysl – např. *Empty*, *Camera* či *Lamp*).

Abychom se mohli přepínat mezi objektovým a editačním módem, je potřeba si daný objekt nejprve označit. Označený objekt je zřetelný světle fialovým okrajovým zabarvením. Pokud objekt označený není, má obrys černé barvy.

Přepínání mezi normálním (objektovým) a editačním módem docílíme opakovaným tisknutím klávesy **Tab**. Pokud nechceme, nebo nemáme k dispozici klávesnici, docílíme stejného výsledku poklepáním na rozbalovací menu v okně *3D View*, ikonka krychle – zvolením příslušného módu – *Object Mode*, *Edit Mode*.

Práce mezi objektovým a editačním módem se liší. V objektovém módu pracujeme s objektem jako celkem, ale v editačním módu můžeme pracovat i s jeho částmi – vertexy, hranami, plochami či objekty.

3.7.2 Základní objekty

Objekty typu *mesh* jsou tvořené základní sítí, která tvoří jejich hranici. Každý mesh objekt je složen z vertexů, hran (hrana spojuje dva vertexy) a ploch (plocha je plná uzavřená oblast, jejíž hranice je určená hranami a může nabývat nejrůznějších tvarů a počtu vrcholů). V programu Blender máme k dispozici hned několik základních objektů, které můžeme dále rozvíjet a deformovat ve složitější.

- Plane – standardní čtverec, tvořený čtyřmi hranami a jedinou plochou. Plane je dvourozměrné těleso, tedy nemá žádnou tloušťku. Využíván bývá např. jako podlaha, stěna, zrcadlo apod.
- Cube – z anglického názvu odvoditelná 3D krychle s 8 vertexy, 12 hranami a 6 plochami. Krychle bývá zpravidla nejpoužívanější objekt pro další *krabicové* modelování.
- Circle – 2D kružnice, opět bez jakékoliv tloušťky. Při vytváření kružnice jsme vždy dotázáni, z kolika vertexů má být složena (standardně je 32). Z kružnice lze jednoduše vytvořit rotační tělesa jako jsou talíře, disky, uzávěry…
- UVsphere – koule složená z *n* segmentů a *m* prstenců. Tyto hodnoty zadáváme při jejím vkládání do scény.
- Iconsphere – koule složená z trojúhelníkových segmentů. Počet těchto segmentů můžeme nastavit parametrem *Subdivision*, který se objeví po zadání tohoto příkazu.
- Cylinder – válec, jehož kruhový průřez je složený z *n* vertexů, které zadáváme při jeho vložení. Válce bývají používány jako zástupné objekty různých topor, klacků, tyčí…
- Tube – dutý válec, jehož kruhový průřez je složený z *n* vertexů. Využívá se pro modelování trubek, sklenic, hrnců.
- Cone – kužel. Jedna z podstav kužele bývá z pravidla větší než druhá, popřípadě ukončena do špičky. Kruhový průřez je složený z *n* vertexů, které opět zadáváme při jeho vložení.
- Grid – mřížka složená z $m \times n$ vertexů. Tyto hodnoty se zadávají, při jejím vkládání do scény. Přestože v normálním módu je podobná objektu typu Plane, ve skutečnosti má více vertexů a lze ji různě tvarovat. Nejčastěji se využívá k vytváření krajin a různých členitých povrchů.
- Monkey – objekt hlavy opice od firmy NaN. Její jméno je Susane a jedná se reprezentačního maskota programu Blender.

3.7.3 Modifikátory

Obecně platí, že spousta modelů lze vytvářen různými způsoby. Záleží čistě na uživateli, která metodu či způsob si osvojí a bude využívat. Program Blender, stejně jako další spousta jiných 3D modelovacích aplikací, je vybaven silnými modifikačními nástroji s jejichž použitím má uživatel prakticky téměř neomezenou příležitost modelovat nejrůznější objekty. Podrobným popisem veškerých metod bychom byli schopni vytvořit samostatnou příručku a proto budou uvedeny metody pouze stručně. Detailnější informace naleznete v literatuře [1].

- **Tažení** (*Extrude*) – patří k nejčastěji prováděným úpravám. Aplikovat lze jak na vertexy, tak i strany. Největší uplatnění nalezneme při tažení celých ploch, kdy se po aplikaci *extrude* vytvoří obrys plochy vytažený v příslušném směru po stranách spojený.
- **Rozdělení** (*Subdivide*). Pokročilejší modelovací metody jsou popsány tzv. subdivizionálním modelováním. Funkce rozdělení (subdivide) přidává k vybraným částem objektu nové vertexy (zhuštění plochy), které nám tyto detaily umožní vytvářet. Blender v módu subdivide umožňuje i několik módů – klasický *subdivide*, *multi subdivide*, *multi fractal subdivide* a *smooth subdivide*, přičemž každý má jinou funkcionální aplikace za vybraný *faceset*.
- **Zaoblení** (Bevel). Zaoblení objektů se používá pro odstranění ostrých hran jejich zjemněním, resp. přidáním nových bodů. Při aplikaci modifikátoru *bevel* se objeví formulářové pole, do kterého zadáváme stupeň rekurze (*Recursion*). Tato hodnota udává kvalitu zaoblení. Z čísla *n*, které zadáme, se jedna hrana změní v 2^n hran, které jsou rozmístěny v oblouku. Poté ještě klepnutím myši definujeme poloměr zaoblení. Zaoblení lze použít pouze na celý objekt, nikoliv na libovolně označené části.
- **Zakřivení** (*Warp tool*). Podle kruhového oblouku určíme míru zakřivení. Středem tohoto oblouku je 3D kurzor a zakřivení probíhá v rovině pohledu okna 3D *view*.
- **Hák** (*Hook*). Je funkce podobná rodičovské vazbě. Tedy objektu vytvořené jako kopie (děti) originálního (rodiče) objektu se mění na základě deformace právě rodiče. Budeme-li tedy měnit kupříkladu polohy některých vertexů, změní se nám taktéž polohy u kopií originálu.

- **Subsurf**. Se nevolá stiskem příslušné klávesy jako v předchozích případech, nikoliv vybrání z *modifiers* panelu. Modifikátor nám vytváří z objektů plochých hladké s tím, že stupeň hladkosti si můžeme volit. Takto vytvořené objekty jsou oproti původnímu menší za to hladší a detailnější. Podobně jako u *bevel* dochází ke zhušťování *mesh* ploch.
- **Decimátor** (*Decimate*). Je nástroj, který nám slouží k redukci počtu vertexů (resp. hran i ploch). Počty vertexů ovlivňují rychlost *renderingu* scény, z tohoto důvodu je nástroj využíván v rámci optimalizace, kdy je vhodné některé vertexy „vypustit“.
- **Rozřezání** (*Knife tool*). Funkce nůž – rozřezání slouží k vytváření nových vertexů v hranách. Pokud je to možné, tak se tyto vertexy spojují (vytváří nové hrany) tak, jak byl veden řez. K dispozici máme celkem 4 možnosti vedení řezu objektem – *Loop Cut*, *Exact knife*, *Midpoints knife* a *Multicut knife*.
- **Šum** (Noise). Funkce umožňuje měnit polohu vertexů na objektu v závislosti na odstínech šedi textury, která je objektu přidělena. Nejčastěji se ji využívá při generování náhodně zvlněných ploch jako jsou krajiny, volní plochy či planety. Čím hustější a kvalitnější textura, tím více bude povrch objektu zvlněný. U modifikátoru můžeme také nastavit hloubku a intenzitu deformující textury.
- **Rotace profilu kolem osy** (*Spin*, *Spin Dup*). Funkce dokáží rotovat libovolný rovinný profil kolem osy. Funkce *Spin Dup* vytváří kopie tohoto profilu po obvodu kruhového oblouku.
- **Šroubovice** (*Screw*). Je podobná příkazů *Spin* s tím rozdílem, že dochází k posunu, takže vznikají různé závitové nebo spirálovité objekty. Šroubovici lze vytvořit pouze v nárysu.
- **Boolean operace**. Nebo-li logické operace s objekty. Blender podporuje celkem tři boolean operace: součet, průnik a rozdíl. Operace se vztahují vždy na dva objekty na které je aplikován příslušný operand. Pokud je potřeba operaci provést s více objekty, musí se provádět postupně. Dále platí, že kombinací dvou vybraných objektů vznikne nové třetí těleso.
- **Metaobjekty** jsou speciální typy objektů, které můžeme v Blenderu používat. Hlavní charakteristikou těchto objektů je, že mají pouze oblé tvary. Jedná se o

implicitní plochy a nejsou tedy definovány ani body ani křivkami. Ve skutečnosti jsou popsány matematickými vzorci, které mohou vykonávat logické operace s jinými objekty. Lze je chápat jako zdroj statického pole. Toto pole může být buď záporné nebo kladné, proto se mohou tyto objekty vzájemně odpuzovat nebo přitahovat. Vhodné jsou především ke tvorbě organických modelů, kdy dochází k sloučení buněk či rozdělení. Síla vazeb a další vlastnosti jsou plně editovatelné.

- **Křivky a povrchy**. Prostorové objekty lze modelovat i pomocí křivek (*curves*) a ploch (*surfaces*). Veškeré animační programy stejně tak i Blender jsou vybaveny těmito nástroji pro modelování. Nejsilnější metodou vytváření ploch právě v Blenderu jsou NURBS plochy jejichž význam a aplikace byly teoreticky popsány v předchozí kapitole. Šikovný animátor je schopen pomocí těchto plátů vymodelovat cokoliv.

3.7.4 Materiálové nastavení

Materiál udává každé objektu vzhled. Pro dosažení realistického vzhledu je proto nutná důkladná znalost materiálové problematiky. Ve většině případech zabere autorům několikanásobně více času než tvorba objektu.

Při renderování obrázku je potřeba vždy určit barvu každého bodu (pixelu). To se provádí nejlépe zpětným sledováním světelného paprsku. Bodem, jehož barvu chceme určit zpětně sledujeme paprsek světla a hledá s jeho průsečík s libovolným objektem ve scéně. V tomto bodě se poté určuje, který světelný zdroj na něj působí. Vliv na výslednou barvu mají vlastnosti povrchu tohoto objektu (například hrubost, průhlednost, reflexe…) a dále dopadající paprsky od světelných zdrojů. Každý individuální paprsek, dopadá na povrch nějakého objektu se může zachovat dvěma způsoby – může se okamžitě odrazit (tento jev se nazývá specularita – *specular*) nebo dochází k vícenásobnému odrazu a lomu (tento jen se nazývá difúze – *diffuse*).

Intenzita odraženého světla je dána součtem speculárních a difúzních složek, nazývaných též shadery. Charakteristikou pro speculární složku je její směrovost, která narůstá s hladkostí povrchu. Tato složka je příčinou odlesků na zobrazovaném objektu. Kombinováním různých nastavení podle barevných modelů (kap. 1) vytvoříme v materiálovém editoru příslušný shader, který aplikujeme na objekt.

Další, složitější, možností jak v přidělit objektům materiál je definovat jeho Texturu. Textura je obrázek, který slouží později k texturování. Nejpoužívanější formáty pro textury jsou *jpg*, *bmp*, *gif*, *tga* .... Nejčastěji se textury získávají úpravou digitálních fotografií, ale někdy se textura tvoří přímo v grafickém prostředí programu (například *noise*, *celluar*, *checker*, *gradient*, *fractal* a jiné). Je několik druhů textur a jejich využití. Při použití textur se často využívají *shadery*.

Blender [1] nabízí poměrně silný nástroj pro definování textur a není třeba externího programu. Například hrubě vypadající povrchy vytvoříme aplikováním *Bump* mapy, reflexi či refrakci definováním barevné složky či textury, difúzní barva může být buď číselně určena z barevného spektra nebo externě z obrázku. Materiálové nastavení je opravdu složitou a samostatnou disciplínou, jejíž důkladným pochopením a ovládnutím můžeme elegantně zamaskovat i špatně vymodelovaný objekt.

4 IRRLICHT

Irrlicht Engine je open-source realtimový 3D engine napsaný v C++ , dostupný také pro .NET platformu. Jedná se o vysoce výkonné API rozhraní pro vytváření 3D a 2D aplikací jako jsou hry, vědecké vizualizace, spořiče obrazovky a další multimediální software.

API je zkratka z anglických slov Application Programming Interface, což znamená rozhraní pro programování aplikací. Jedná se tedy o specifickou sbírku procedur, funkcní či jiných tříd určité knihovny (ale třeba i jiného programu nebo jádra operačního systému), které může využívat programátor, který knihovnu využívá. API určuje, jakým způsobem se funkce knihovny mají volat ze zdrojového kódu programu. V Irrlichtu jsou důležitá především grafická API jako OpenGL a DirectX, která jsou standardizována a programátor jich pomocí tříd využívá ve své aplikaci.

Aktuální verze Irrlicht enginu – 1.3, s sebou přináší vynikající nástroj v němž jsou integrovány nejmodernější doplňky pro vizuální prezentace prostorů jako jsou dynamické stíny, generátory částic, nástroje pro animaci postav, generátory uzavřených a otevřených prostředí či kolizní systémy. Vše s dostupnou dokumentací pro C++ a .NET.

Irrlicht je kompletně multiplatformní, lze jej bez problémů využívat na operačních systémech jako jsou:

- Windows 98, ME, NT 4, 2000, XP, XP64, Vista
- Linux
- MacOS
- Sun Solaris/SPARC

4.1 Specifikace 3D enginu

- Výkonný *real-time* 3D *renderer* využívající Direct3D a OpenGL.
- Platformově nezávislý. Spustitelný na mnoha operačních systémech.
- Obsahuje rozsáhlou vestavěnou a libovolně rozšířitelnou knihovnu s materiály včetně podpory *vertex* a *pixel shaderů*.
- Hladké a přehledné mixování uzavřených a otevřených scén.

- Systém pro animaci postav včetně aplikace kostí či *morph* pro animace (morfování se využívá pro simulaci různých výrazů – smutek, smích, zloba …).
- Částicové systémy, *billboards*, *light* mapy, *stencil buffer shadows*, a spousta dalších speciálních efektů.
- Implementace .NET programovacího jazyka, který zpřístupňuje své funkce pro veškeré .NET programovací jazyky C#, VisualBasic a Delphi.NET.
- Dvouplatformní rychlý a nezávislý *softwarový render* obsahující *perspective correct texture mapping, bilinear filtering, sub pixel correctness, z-buffer, gouraud shading, alpha-blending and transparency* a rychlé 2D vykreslování.
- Přehledné a libovolně nastavitelné 2D grafické prostředí s tlačítky, lištami, editačními boxy a další.
- 2D vykreslovací funkce jako jsou a*lpha blending, color key based blitting*, vykreslování fontů a mixování 3D grafiky s 2D.
- Jednoduché, lehce pochopitelné a plně zdokumentované API se spoustou příkladů a návodů.
- Napsáno v čistém C++ kódu vše objektově orientováno.
- Přímý import rozličných *mesh* objektů/formátů přímo do scény: Maya (*.obj*), 3DStudio (*.3ds*), COLLADA (*.dae*), DeleD (*.dmf*), Milkshape (*.ms3d*), Quake 3 mapy (*.bsp*), Quake2 modely (*.md2*), Microsoft DirectX (.X) a další.
- Přímý import textur ve formátech - *.bmp*, *png*, *psd*, *jpg*, *tga*, *pcx* a další.
- Rychlý a jednoduchý systém pro detekci kolizí a reakcí.
- Optimalizované 3D matematické knihovny.
- Možnost čtení z komprimovaných formátů *zip*.
- Integrovaný XML *parser*.
- Podpora standardu Unicode.
- Pracuje pod vývojovými prostředími Microsoft VisualStudio6.0™, VisualStudio.NET 7.0-8.0™, Metroworks Codewarrior, a Bloodshed Dev-C++ s g++3.2-4.0.

- Celý *engine* je zadarmo a volně stažitelný (*open source*) včetně zdrojových kódů. Potenciální uživatel může v kódu programu měnit cokoliv. *Engine* je vedený pod zlib licencí.

4.1.1 Speciální efekty

Irrlicht engine obsahuje poměrně velké množství nejpoužívanějších speciálních efektů, na které jsme u obdobných či komerčních aplikací zvyklí. Jejich použití není vůbec složité a v mnohých případech stačí, aby je programátor jednoduše aktivoval pouhým zavolám či zapnutím. Engine s novými verzemi postupně vybavován o další nové efekty, nicméně současná 1.3 verze má v sobě implementovány následující speciální efekty:

- Simulace realistických povrchů vody
- Dynamické světla
- Dynamické stínování (stíny mění svůj rozměr i pozice na základě modifikace světelných zdrojů a to vše v reálném čase) s využitím *stencil bufferu*.
- Billboardy
- Bump mapping – simulace zdrsněných, členitých povrchů pouhou texturou.
- Paralax mapování
- Transparentní objekty
- Light mapy
- Libovolně nastavitelné částicové systémy pro simulaci sněhu, kouře, ohně, explozí aj.
- Sférické mapování
- Animované textury
- Skyboxy
- Mlha

4.1.2 Ovladače

Engine v současné době podporuje celkem šest různých renderovacích API rozhraní, přičemž pět z nich jsou typické pro 3D využití:

- Direct3D 8.1
- Direct3D 9.0c
- OpenGL 2.0
- Vlastní Irrlicht Engine softwarový *renderer*.
- Apfelbaum softwarový *renderer.*
- Nulový

Pokud se uživatel rozhodne právě pro Irrlicht engine, čistě teoreticky a částečně i prakticky nepotřebuje programátor znát, které API zrovna engine používá, neboť je zcela abstraktní. Jedinou nutností je enginu sdělit, které API rozhraní bude preferováno. Především tu máme tři důvody proč není výběr engine zaměřen pouze na API:

- **Výkon**. Některé grafické adaptéry jsou optimalizovány pro OpenGL a některé naopak běží rychleji s Direct3D.
- **Závislost platformy**. Direct3D nebude zřejmě nikdy uveden pro Mac nebo pro pracovní stanice běžících na operačním systému Linux, kde OpenGL ano. A pokud ani OpenGL nebude plně podporováno na žádné z platforem, stále máme k dispozici softwarový render, který nám na zvoleném operačním systému poběží. Přestože nebudou využity veškeré funkce, stále docílíme toho, že uživatel na své obrazovce něco uvidí.
- **Problémy s ovladači** – nejčastější problém s kterým uživatelé nepočítají. V současné době máme na trhu nesmírné množství grafických karet a každým měsícem přichází nové a nové. Pokud je na počítači nainstalován zastaralý nebo výrobcem necertifikovaný ovladač, často dochází k pádům aplikací a kosa narazí na kámen. Možnost přepnout mezi příslušnými ovladači by mohlo problém vyřešit.

4.1.3 Materiály a shadery

Abychom byli schopni rychlého vytvoření realistických prostředí, nalezneme jako součást enginu i běžné přednastavené materiály. Některé materiály jsou založeny na bázi fixní funkcionální *pipeline* (například light mapped geometrie) a některé se odkazují na programovatelné *pipeline* (normálově mapované/parallaxané na pixel osvětlené materiály) jenž dnešní 3D hardware poskytuje. Samozřejmě je možné oba tyto typy mezi sebou

mixovat. V případě potřeby doplnit knihovnu o další materiály se nejedná o žádný problém a ani nebudete muset měnit či znovu kompilovat jádro enginu.

V současném vydání jsou podporovány následující shadery:

- Pixel a Vertex Shaders 1.1 do 3.0 verze
- ARB Fragment a Vertex Programs
- HLSL
- GLSL

4.1.4 Animace postav

Irrlicht mimo výše uvedených vlastností také obsahuje dva specifické nástroje pro animaci postav. Díky tomu není třeba pořizovat si drahé komerční nástroje, ale elegantně využít implementovaných nástrojů, které jsou:

- **Morph target** animace. Meshe jsou lineárně interpolovány z jednoho snímku do druhého. Tohohle způsobu bylo využíváno i sériích hry Quake, kde nejsou postavy statické, ale pohybují se. Příkladem budiž stojící *mesh* postavičky, který se nadechuje a vydechuje – zvedají se mu ramena, zvětšuje objem hrudníku při nádechu. Díky možnosti importovat *.md2* soubory (což jsou animované modely z Quake) dostaneme do naší aplikace jakékoliv animované postavy na bázi *Morph target animation*.
- **Skeletal animation**. Povrch, skin, je manipulován animovanými spoji. Tyto spoje můžeme do Irrlicht enginu nahrát ze souborů s koncovkou *ms3d* nebo *x* a různé části objektů na tyto spoje posléze přiřadit. Názorným příkladem může být napojení objektu zbraně do rukou animovaného skeletonu člověka, přičemž poloha zbraně se přizpůsobí pohybům rukou a celého těla.

4.1.5 Podporované formáty

Součástí enginu je i rozsáhlá podpora všech možných formátů, které můžeme do programu nahrát přímo použitím jednoduchých kódů. Právě rozmanitost a podpora rozsáhlé škály formátů nás oprošťuje od nutnosti konvertování z jednoho standardu do druhého, čímž

ušetříme spoustu času. Seznam podporovaných formátů se konstantně rozšiřuje s každou novou verzí enginu. Vývojáři jsou navíc tak perspektivní, že v případě potřeby některé z formátů, které doposud Irrlich nepodporuje, jsou schopni import doprogramovat.

Aktuální podporované formáty textur:

- Adobe Photoshop (*.psd*)
- JPEG File Interchange Format (*.jpg*)
- Portable Network Graphics (*.png*)
- Truevision Targa (*.tga*)
- Windows Bitmap (*.bmp*)
- Zsoft Paintbrush (*.pcx*)

Aktulní podporované *mesh* formáty:

- 3D Studio meshes (*.3ds*)
- B3D files (*.b3d*)
- Alias Wavefront Maya (*.obj*)
- Cartography shop 4 (*.csm*)
- COLLADA (*.xml*, *.dae*)
- DeleD (*.dmf*)
- FSRad oct (*.oct*)
- Irrlicht scenes (*.irr*)
- Microsoft DirectX (.x) (*binary* & text)
- Milkshape (*.ms3d*)
- My3DTools 3 (*.my3D*)
- OGRE meshes (*.mesh*)
- Pulsar LMTools (*.lmts*)

- Quake 3 levels (*.bsp*)
- Quake 2 models (*.md2*)

Dodatkem jsou navíc k dispozici nástroje pro export do populárních modelovacích softwarů jako jsou **Blender**, 3D Studio Max či Gile a další…

4.2 Přídavné nástroje

Irrlich samotný je sofistikovaný 3D real-time engine určený pro tvorbu rozmanitých aplikací, her, multimediálních prezentací apod. Vývojáři se snaží cestu k používání Irrlichtu co nejvíce zjednodušit a tak pro potenciální vývojáře, ve spolupráci s nezávislými lidmi s projektem spojenými, připravují další volně stažitelné nástroje, jenž jim ušetří a zjednoduší spoustu práce.

Jedná se volně stažitelné externí balíky či knihovny, které jsou přímo propojeny s Irrlicht enginem a většinou jsou vyvíjeny třetími stranami.

4.2.1 irrKlang

irrKlang je freeware 2D a 3D *sound engine* a zároveň knihovna, která slouží k přehrávání hudebních záznamů ve formátech *wav*, *mp3*, *ogg*, *mod*, *xm*, *it*, *s3m*.

4.2.2 irrEdit

Asi nejoblíbenějším nástrojem, který je s Irrlicht enginem spojený je bezesporu irrEdit nástroj. Jedná se opět o volně stažitelný real-time 3D editor a *radiosity lightmap* generátor. Prostředí irrEditu je velice jednoduché se specifickým GUI. Součástí aplikace jsou přeprogramované primitiva (objekty), částicové systémy, světla, jednoduché modifikátory a nástroje pro transformaci. Díky těmto funkcím jsme schopni vytvářet jednoduché, ale i složitější scény.

4.2.3 irrXML

Jedná se o jednoduchý a rychlý Open Source *xml parser* pro programovací jazyk C++. irrXML se s novými verzemi Irrlichtu stal neoddělitelnou součástí a je přímo integrován.

Používán je například pro čtení COLLADA a *.irr* souborů. Je velice rychlý a nevyžaduje velké množství operační paměti.

irrXML není určen jenom pro Irrlicht. Jeho využitý má i globální význam a pokud bychom jej chtěli použít pro projekt, který neběží na Irrlicht enginu, lze irrXML nástroj implementovat jako knihovnu.

PRAKTICKÁ ČÁST

5 DEMOSTRACE MODELOVACÍCH TECHNIK

V teoretické části bylo zmíněno několik dostupných modelovacích technik obsažených v Open Source 3D modelovacím programu Blender. V současné době se na trhu pohybuje poměrně slušné množství aplikací, jenž jsou určeny pro modelování prostorových objektů a scén.

Z pochopitelného hlediska najdeme v různých animačních a modelačních programech modifikátory a modelovací techniky skryty pod jinými názvy a třeba i v jiné hierarchii či aplikaci. Principielně se ovšem nic nemění a přestože se nám může zdát používání těchto technik v různých programech zcela odlišné, není tomu tak, neboť matematické aparáty se nemění.

Jak již bylo řečeno, osvojení si příslušné metody může znamenat rychlý a přesný modelovací počin, nicméně není vyloučeno, že by to šlo jinak. Většinou ano, ale kupřípadu Boolean operace zůstanou nadobro Boolean operacemi a organika bude i nadále přednostně modelována pomocí *metaballů* nebo *surface* povrchů.

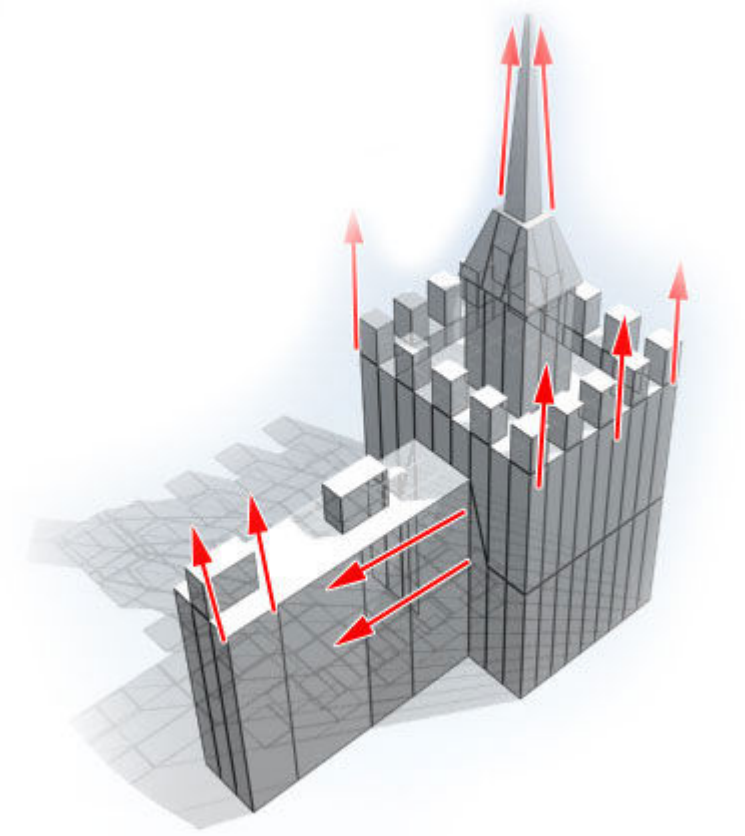
Modelovací funkce lze rozdělit do několika kategorií. K základním patří matematická 2D a 3D primitiva, a to nejenom kvádry, koule, válce, ale i různé hvězdy, kapsle, barely či uzly, všechny definované parametricky a zpětně editovatelné. Pomocí nich se dají vytvářet další tělesa a to mnoha způsoby, využitím *booleanovských* operací, tažením pomocí trajektorie, prokládáním plochy uzavřenými křivkami, atd. Specializovanými nástroji tzv. pláty lze modelovat tvarově složitější tělesa. Lze též použít NURBS modelování. Všechna tělesa *mesh*, *patch* i NURBS, lze upravovat na úrovni svých vrcholů, hran, segmentů, křivek a ploch. Dostupné modelovací techniky jsou:

- Konstruktivní prostorová geometrie
- NURBS
- Polygonární
- Subdivizionální
- Impicitní

Na celá tělesa nebo jejich části mohou být aplikovány modifikátory - ohnutí, zkroucení, zúžení, šum, vytažení, rotace kolem křivky, a desítky dalších. Pomocí těchto modifikátorů lze upravit těleso podle autorových představ. Obrovskou výhodou je funkce, která si pamatuje všechny modifikátory použité na tělese a umožňuje se kdykoli vrátit a změnit konkrétní modifikaci. Mnoho těles lze také modelovat pomocí ovlivňování jinými tělesy.

5.1 Modelování tažením

Modelování rozmanitých objektů tažením je, nepoužívanější technikou v prostorovém modelovaní. Tažení můžeme provádět se všemi objekty typu *mesh* po zvolení příslušného koncového prvku – vertex, hrana, polygon.



Obrázek 25: Modelování hradby tažením

Na obrázku 24 je metoda výstižně znázorněna. V první fázi byl vytvořen objekt krychle tvořený 9 stranami na výšku, 9 na šířku a 2 na tloušťku. Dále postupným označováním ploch, jejich tažením (jak znázorňují černé šipky) a případnou změnou měřítka koncových stěn,

(v případě špičaté věže) lze docílit vytvoření takovéhoto jednoduchého objektu během několika vteřin.

Pokud bychom se rozhodli modelovat takovýto objekt jinou metodou, například *surface* či NURBS, určitě bychom nedosáhli takového výsledku za více jak desetinásobek času. Případným zdetailováním objektu (přidávání nových vertexu, resp. hran) dokážeme i z primitivního objektu vymodelovat velice složité struktury. Metoda, která využívá tohoto postupu se nazývá metodou *Subdivizionálního* modelování.

Během tvorby hlavního objektu univerzity jsem nejčastěji využíval právě funkce pro tažení. Stěny nejsou nic jiného než prosté kvádry o různých rozměrech specificky rozmístěné po scéně. V některých situacích, kdy bylo třeba zjednodušit hierarchii modelu, se funkce tažení – *extrude* příkladně hodila. Především při modelování různých zákoutí, rohů či překližek.

5.2 Modelování rotací

Princip modelování pomocí rotace byl matematicky i teoreticky popsán v kapitole o křivkách. Z praktického hlediska bylo vytvořeno několik ilustrativních objektů, které názorně ukazují využití modelovací techniky rotací.

V prvním kroku je vždy vytvořit obrys objektu, který budeme rotovat. Pomocí křivek si buď obtáhneme obrázek na pozadí nebo vytvoříme vlastní dle fantazie. Jakmile jsme dosáhli požadovaného obrysu, aplikujeme na křivku příslušný modifikátor, který se samozřejmě v různých programech liší. V našem případě se bude jednat o použití funkce *Spin* (Blender) nebo *Lathe* (3D Studio Max).

Obrázek 26: Objekty vzniklé rotací kolem osy – pohárky, vinná sklenice, kanystr

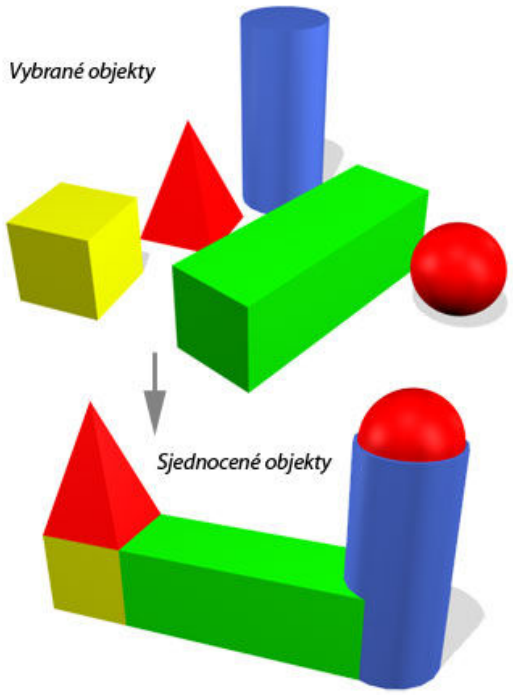
5.3 Konstruktivní prostorová geometrie

Než-li se pustíme do práce s booleanovskými operacemi – konstruktivní prostorovou geometrií, je třeba si představit nebo nejlépe na papír nakreslit, jakého výsledku chceme dosáhnout.

Na začátku bylo zvoleno pět základních primitivních objektů, s kterými budou názorně předvedeny všechny čtyři uvedené logické *boolean* operace. Vybranými objekty jsou koule, kvádry, krychle, válce a čtyřboké hranoly.

5.3.1 Sjednocení objektů

V prvním kroku jsem si na papír nakreslil jaký výsledný objekt chci vymodelovat. V 3ds max jsem si nakreslil základní objekty a přesně jim určil rozměry - krychle o straně a = 20cm; kvádr o stranách a = 60cm, b = 20cm, c = 20cm; koule o r = 10cm; válec o poloměru r = 11cm, výšce h = 50cm; hranol o straně a = 20cm a výšce v = 30cm (Obrázek 27).



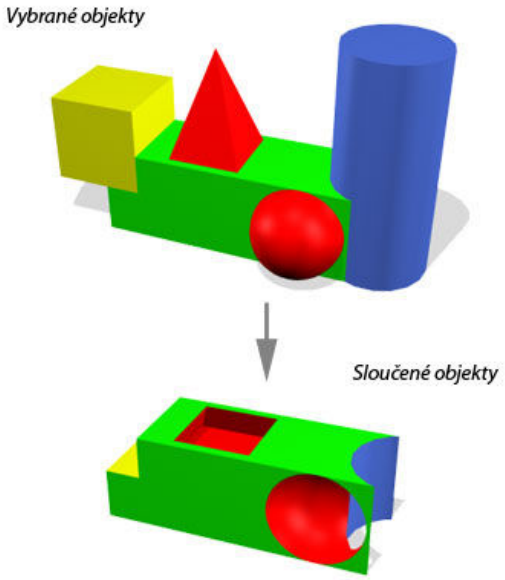
Obrázek 27: Sjednocení

5.3.2 Sloučení objektů

Vybrané objekty a jejich rozměry budu i nadále používat. V následujících logických operacích je budu vzájemně slučovat v tomto pořadí:

1. Zelený + žlutý
2. Zelený + červený hranol
3. Zelený + modrý
4. Zelený + červená koule

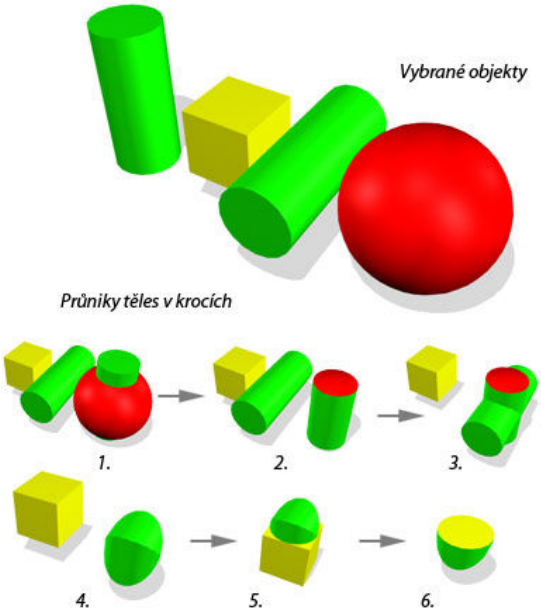
Výsledek operací je znázorněn na následujícím obrázku (Obrázek 28).



Obrázek 28: Sloučení

5.3.3 Průnik objektů (těles)

Při operaci průniků objektů resp. těles jsem vybral jako výchozí „průnikové“ těleso kouli o poloměru r = 20cm. Tu jsem postupně „ořezával“ dvěmi objekty válce o poloměru r = 10 cm a délce h = 50cm a jednou krychlí o straně a = 20cm. Grafické znázornění operací v krocích je na následujícím obrázku (Obrázek 29).



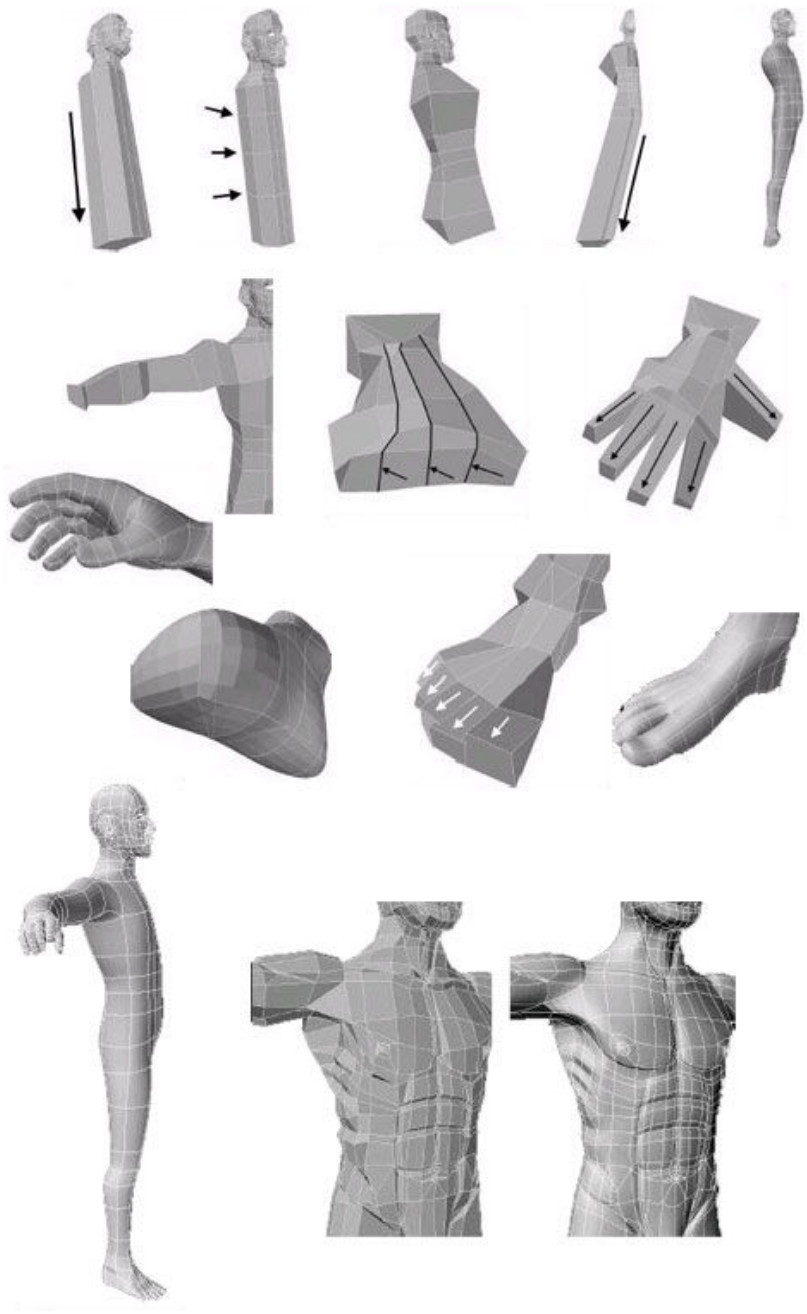
Obrázek 29: Průnik

5.3.4 Rozdíl těles

Poslední metodou konstruktivní prostorové geometrie je rozdíl dvou objektů resp.těles. Metoda je stejná jako sloučení s tím rozdílem, že v objektu A nezůstává objekt B, jednoduše se jeho obsah odečte.

5.4 Subdivizionální modelování

V teoretické části jsem se zmínil o faktu, že modelovací techniky Subdiv surfaces se využívá především při tvorbě organických objektů. Ano je to opravdu tak, největší předností Subdiv techniky je téměř nekonečné přidávaní detailů až po žíly, jizvy, puchýře, znaménka apod. Většinou se tak detailní modely netvoří – detaily jsou nahrazeny kvalitními texturami, přesto se najdou lidé, kteří si na detail modelu potrpí. Na následujícím obrázku je zobrazen postup tvorby mužského těla (Obrázek 30.).



Obrázek 30: Subdivizionální modelování mužského těla v krocích

Na začátku si umělec z obyčejného *boxu* vymodeloval hlavu (není na obrázku vidět), dále pokračoval vytažením horní části těla směrem dolů k pánvi. Postupně model rozřezal tak, aby mohl vytáhnout ruce a dolní končetiny. V další části si rozřezal konce polygonů končetin na přesné počty tak, aby získal prsty jak u nohou, tak u rukou. Tím získal hrubý, ale postačující model pro aplikování *Subdiv*. Na posledních dvou obrázcích je již ukázán nezaoblený věrohodný detailní *mesh* mužského těla.

6 MODEL BUDOVY U5 – FAI

Než-li se pustíme do jakékoliv realizace trojrozměrného komplexu, je třeba se opřít o základní nástroje pro obecnou tvorbu takovýchto 3D modelů. 3D modelování umožňuje zpracovat jakýkoliv předmět do prostoru a také tento předmět použít v celkové scéně nebo ho nechat osamocený. V neposlední a důležité fázi před samotným započetím sofistikovaného a přesného modelování je třeba zajistit nezbytné informace, které nám o koncovém požadovaném výstupu prozradí více:

- Technický, architektonický plánek objektu, budovy
- Návrh vytvoření
- Zvolené měřítko
- Modelovací technika a postupy
- Materiálové vlastnosti
- Fotografické materiály
- Vizualizační 3D software

Všechny tyto údaje byly čerpány z technického nákresu budovy U5 obdrženého na rektorátě FAI, dále technických nákresů rozmístěných jako únikové schémata z budovy a množstvím fotek pořízených autorem modelu.

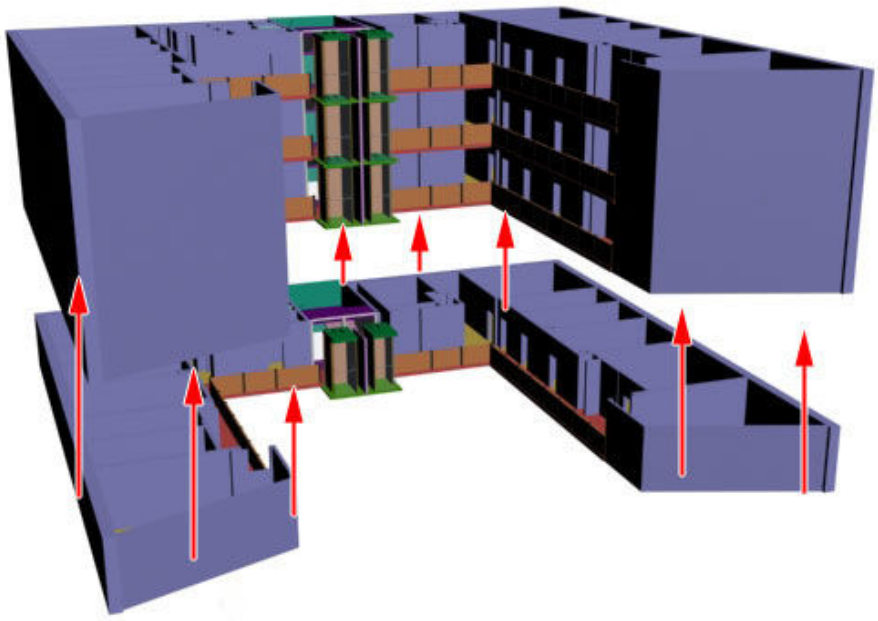
Cílem nebylo vytvoření přesného modelu, který by byl rozměrově totožný jako budova reálná, nýbrž navržení dvou verzí *low-poly* a *high-face* verze. Obě verze byly následně porovnány a *low-poly* verze mimo jiné optimalizována pro implementaci do vybraného 3D *enginu*.

Modelování samotné probíhalo v několika fázích v první fázi byl pořízen a milimetrovým pravítkem přeměřen a rozkreslen model na papír. Tohoto kroku muselo být podstoupeno, neboť technický plán nezahrnoval tak důležité měřítko, které by usnadnili spoustu práce. Jako první byly položeny základní kameny, chcete-li pilíře modelu. Obyčejné primitiva typu *Box*. Ostatně boxy tvoří téměř 90% celého modelu, číselně více jak 2000 objektů poskládaných v různých patrech, rozměrech a transformacích.

Vzhledem k tomu, že modelování v 3D grafickém prostředí, není nikterak limitovány svým prostorem, bylo zvoleno měřítko milimetrového pravítka, které ve skutečnosti představuje odhadem 1 mm = 7 cm skutečné délky. Model lze samozřejmě jakkoliv měnit spolu z jednotkami, které byly zvoleny.

6.1 Základna budovy – kanceláře

Nejobtížnější část modelu jsem si vybral hned na začátek. Na papíře jsem měl rozkreslené rozměry všech místností a postupným přidáváním objektů *box*, definováním rozměrů a pozic jsem vytvořil jedno podlaží hlavního čela fakulty. Při detailnější studii budovy zjistíte, že patra 3 – 8 se opakují, tedy v rámci flexibilního modelování bylo zvoleno jedno patro jako rodičovské a další byly pouze instancemi původního. Tedy jakákoliv změna na rodičovském modelu se projevila i dceřiném (instančním modelu). Podlaží 3 – 5 bez použitých textur včetně výtahu vypadá stejně jak na následujícím obrázku.



Obrázek 31: Modelování jednotlivých podlaží

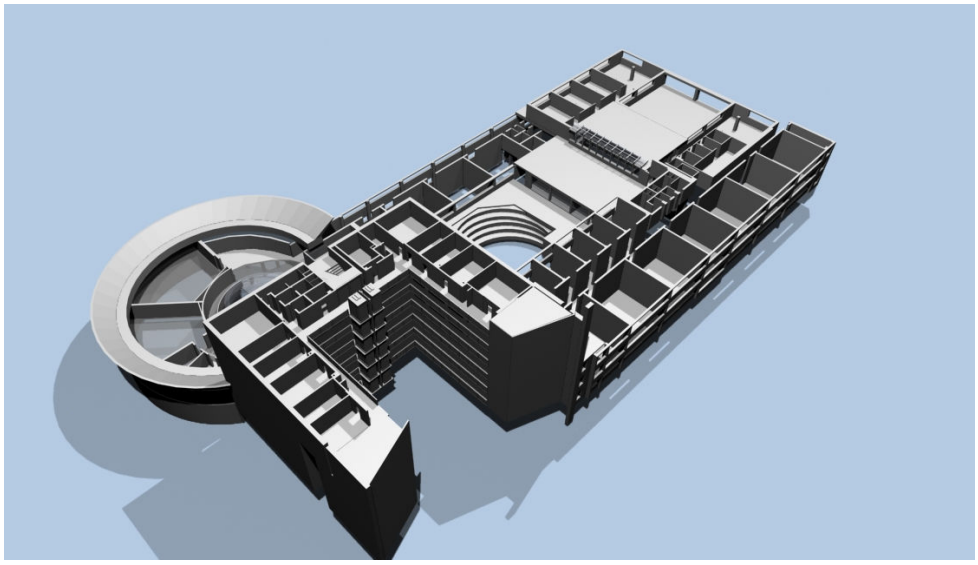
Jedno patro s optimalizovanými detaily včetně schodišť a výtahu bylo přibližně složeno s 220 objektů a pouze 6000 faců. Při rozkopírování pěti pater vzhůru a jednoho lehce rozdílného patra dolů dojdeme k číslu kolem 36.000 faců. Při představě, že postavičky v dnešních akčních hrách mají síťovinu přibližně deset a dvacet tisíc polygonů, můžeme v této fázi říci že model je z hlediska hierarchie topologie *meshů* výborně optimalizován.

6.2 Cvičebny, přednáškové a ostatní místnosti

Naprosto stejným postupem byla modelována i zbývající část budovy. Nejprve byly na papír zakresleny a rozměřeny odstupy zdí, jejich tloušťky, pozice a poté parametricky vkládány přímo do programu. V grafickém programu byly dále jednotlivé objekty skládány tak, aby na sebe navazovaly a utvořily souvislou a uzavřenou část příslušného sektoru budovy. V některých případech bylo využito *Boolean* operací pro výřezy oken a dveřních prostor, či změn měřítka pro roztažení koncových objektů do požadovaného rozměru.

V rámci modelování zbývajících prostor, místností, byly tvořeny následující sektory:

- Cvičebny
- Laboratoře
- Přednáškové místnosti
- Strojovny
- Elektrické rozvodové skříně
- Menza



Obrázek 32: Rozpitvaný model budovy U5

V modelu jsou záměrně vynechány dveře a panely oken, které byly nahrazeny velkými pláty zasahující skrze zdi s příslušným materiálovým nastavením, kvůli optimalizaci modelu do low-poly. Ukázky výsledného modelu naleznete v příloze na CD jak v různých formátech pro import tak i vykreslení z několika pohledů.

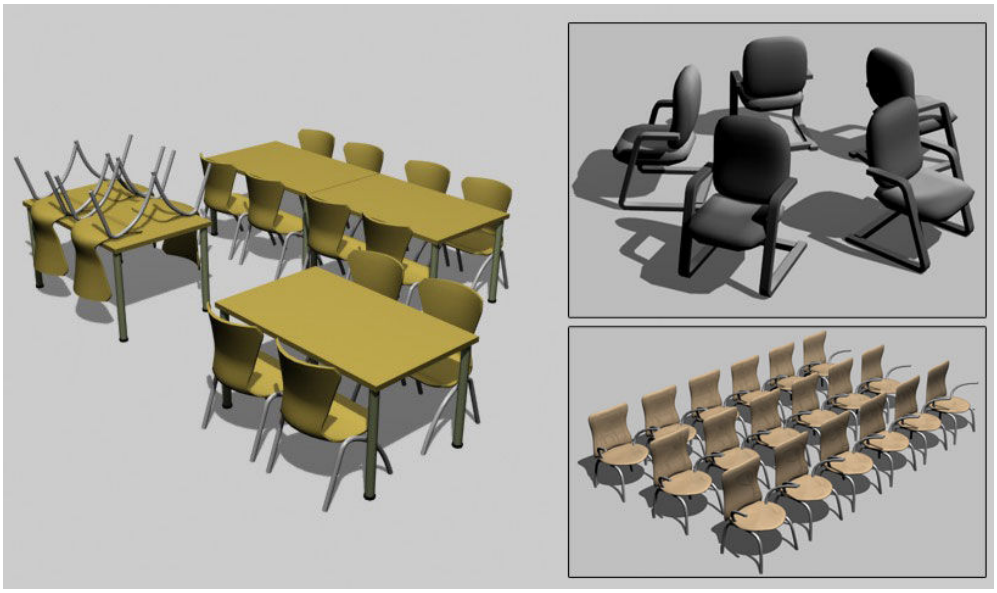
6.3 Doplňkové objekty, interiér

V rámci práce byly vytvořeni i různé doplňkové objekty, které budou použity především pro vizualizaci v 3D enginu nebo pro kompoziční rendering. Objekty jsou z části optimalizovány svým počtem polygonů na přijatelnou hierarchii. Nicméně při představě, že počet židliček v menze svým počtem polygonů jednoduše převýší celkový počet polygonů budovy, si jistě uvědomíte, že do low-poly verze nebyly tyto předměty umístěny.

Mezi vymodelovanými doplňkovými objekty, které byly tvořeny odhadem nikoliv dle předloh patří:

- Plastové nádrže na vodu (nápojový koutek)
- Plastové kelímky
- Židličky v jídelně
- Polička u kopírky
- Poličky podél zdí u studovny
- Dřevěný totem v příchodové místnosti u výtahů
- Nástěnky
- Čtvercový stoleček
- Židle na chodbách
- Luxusní měkká židle v zasedacích místnostech
- Stoly
- Hasicí přístroj
- Umyvadlo
- Topení
- Toaleta

Samozřejmě že seznam objektů umístěných uvnitř fakulty je delší než celá diplomová práce a proto byly vybrány a vymodelovány ty nejdůležitější a na oko pozorovatele nejčastěji se opakující předměty/věci.



Obrázek 33: Doplňkové objekty – židličky

Obrázek 34: Doplňkové objekty – totem, hasicí přístroj, umyvadlo, topení

6.4 Materiálové nastavení

Aby model nepůsobil fádně a smutně, pouze v odstínech šedi nebo náhodně přednastavených barev, byly vytvořeny základní textury s příslušnými *shadery*. Vzhledem ke skutečnosti, že má být model připraven pro 3D aplikaci, musely být pořízeny fotografie, které budou v dalším kroku přetransformovány pro použití v grafickém vizualizačním softwaru nebo grafickém 3D enginu.

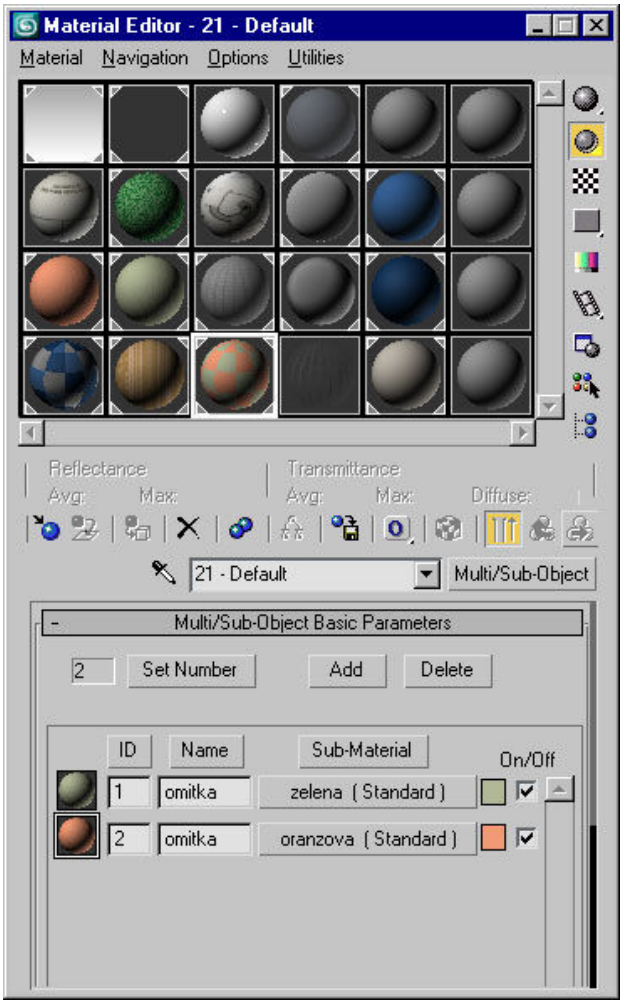
V rámci práce bylo pořízeno celkem 368 fotografií ve vysokém rozlišení. Byly vybrány tak, aby posloužili vývojáři jako pomůcka nejen pro přípravu fotorealistických textur, ale také pro lepší prostorovou orientaci.

Většina vytvořených materiálů se stala klasickými shadery bez komplexnějšího nastavení pomocí rozmanitých textury. Využity byly pouze složky pro Bump mapping, reflexi a *self-illumination*. Materiály byly vždy definovány nabráním barvy z fotografie (kapátkem) a poté v RGB složkách přepsány do materiálového editoru příslušného programu. Nejdůležitějšími parametry se staly složky:

- Prostředí (*Ambient*)
- Difúzní barva (*Diffuse*)
- Spekulární barva (*Specular*)

Materiál	Shader	Ambient	Diffuse	Specular	Diffuse map/Bump
Sklo	Blinn	[0,0,0]	[218,230,255]	[230,230,230]	
Tráva	Blinn	[0,0,0]	[0,94,22]	[255,255,255]	Bump: Celluar
Kov	Metal	[0,0,0]	[211,211,211]	Level 10	
Sklo-panel	Blinn	[160,160,160]	[55,100,155]	[230,230,230]	Diffuse: textura nebe
Omítka oranž	Strauss		[242,154,118]		Bump: Celluar
Omítka zelená	Oren-Nayran-Blinn	[129,140,86]	[177,184,150]	[255,255,255]	Bump: Celluar
Balkon	Blinn	[0,0,0]	[238,244,255]	[230,230,230]	Diffuse: Dlaždice
Dřevo	Blinn	[0,0,0]	[190,155,126]	[255,255,255]	Diffuse: textura dřeva
Hasicí přístroj	Metal	[0,0,0]	[255,20,40]	[255,255,255]	

Tabulka 1: Přehled použitých materiálů



Obrázek 35: Okno material editoru

Na obrázku 35 je výřez z materiálového editoru programu 3D Studio Max. Při exportu do jiných formátů jsou ve většině případech zachovány difúzní a spekulární složky objektu a proto při importu do grafického prostředí Blender zůstaly materiálové nastavení stejné.

V případě použití procedurální textur jako jsou *Celluar* nebo *Noise*, se textura neexportuje a třeba ji v každém programu definovat znovu. Engine samotný pak tyto procedurální textury nepodporuje a jsou nahrazovány vlastními nebo texturami v podobě rastrové grafiky. Pro simulaci traviny byly právě procedurální textury využity, nicméně v Enginu mohou být nahrazeny texturou a afa kanálem nebo v lepším případě částicovým systémem. Textury dřeva a oblohy naleznete v příloze na CD.

7 3D ENGINE – VSTUPNÍ DATA

Nyní se dostáváme do fáze kdy by bylo vhodné definovat co je engine a jaký byl vybrán. Engine funguje jako jakýsi softwarový počítač umožňující chod aplikace na libovolném hardware bez nutnosti jejích úprav. Představme si ho tedy jako technologii, která má za úkol vykreslovat (renderovat) vstupní prostředí v reálném čase. Kvalitní grafický engine má vliv na nárůst fps (*frames per second*, počet snímků za sekundu) a tím pádem má program vetší šanci kvalitně uspět v simulačním propočtu autentického matematického modelu budovy.

Hodnota *fps* je klíčovým výstupním parametrem a měla by být vyšší jak 25, neboť lidské oko menší počet snímků registruje jako neplynulý, trhaný obraz. Praktické řešení low-polygon verze se opírá o knihovnu OpenGL a Direct3D ve volně dostupném real-timovém 3D enginu **Irrlicht**.

Finální low-poly model je tvořen 62 083 polygony, zatímco high-face model, který je libovolně rozšiřitelný na základě výstupních *fps*, čítá více jak půl milionů polygonů. Low-poly verze byla nejdříve exportována do formátu *.3ds*, který je standardizovaným formátem pro vstupní data.

Dále s použitím Open Source 3D *world builderu* irrEdit načtena. IrrEdit umožňuje importovaná data přímo texturovat, definovat světelné zdroje a generovat částicové systémy. Nicméně při importu došlu k sjednocení všech objektů do jednoho *meshe*, který lze samozřejmě dále rozvíjet, ale texturování samotné by se stalo předmětem několika měsíční práce.

Enginu Irrlicht lze využít poměrně snadno s irrEditorem. V tom se načtené objekty otexturují, definují se světelné zdroje, jejich dosah a intenzita, popřípadě provedeme detekci kolizí. Jakmile přípravu dat v editoru ukončíme, exportujeme výsledek do *.irr* souboru, který je vlastně .xml datovým souborem obsahující informace o veškerých objektech ve scéně, jejich pozici, materiálové nastavení, lokalizace importovaných souborů, textury, nastavení světelných zdrojů, *skyboxy*, částicové systémy – emitace, frekvence a další spoustu informací, které umí Irrlicht engine přečíst a zpracovat.

Jakmile docílíme požadovaného výstupního souboru, stane se záhy souborem vstupním přímo pro engine. Tato metoda je obecně nejjednodušší a v enginu stačí pouhé zavolání funkce, která přečte vstupní *.xml* soubor a s ním data spojená.

Následuje zdrojový kód, z kterého je jasně viditelné, jakým způsobem se napojují data z externího souboru.

```
#include <irrlicht.h>
#include <iostream>
using namespace irr;
#pragma comment(lib, "Irrlicht.lib")

int main()
{
	// definice driveru
	video::E_DRIVER_TYPE driverType;
	printf("Vyberte driver pro vykreslovani:\n"\
		" (a) Direct3D 9.0c\n (b) Direct3D 8.1\n (c) OpenGL 2.0\n"\
		" (d) Software Renderer\n (e) Burning's Software Renderer\n"\
		" (f) NullDevice\n (otherKey) exit\n\n");

	char i;
	std::cin >> i;

	switch(i)
	{
		case 'a': driverType = video::EDT_DIRECT3D9;break;
		case 'b': driverType = video::EDT_DIRECT3D8;break;
		case 'c': driverType = video::EDT_OPENGL;   break;
		case 'd': driverType = video::EDT_SOFTWARE; break;
		case 'e': driverType = video::EDT_BURNINGSVIDEO;break;
		case 'f': driverType = video::EDT_NULL;     break;
		default: return 1;
	}
	// vytvoření device, pokud nenastane ukonči

	IrrlichtDevice* device =
		createDevice(driverType, core::dimension2d<s32>(640, 480));

	if (device == 0)
		return 1; // nepovedlo ze použít zvolený driver.

	device->setWindowCaption(L"Titulek okna aplikace");

	video::IVideoDriver* driver = device->getVideoDriver();
	scene::ISceneManager* smgr = device->getSceneManager();
```

```
	smgr->loadScene("test/mistnost.irr");  //  načtení  vstupního  xml
souboru

	// přidání kamery pro volný pohyb
	smgr->addCameraSceneNodeFPS();

	// vykreslení všech objektů
	int lastFPS = -1;

	while(device->run())
	if (device->isWindowActive())
	{
		driver->beginScene(true, true, video::SColor(0,200,200,200));
		smgr->drawAll();
		driver->endScene();
		int fps = driver->getFPS();
		if (lastFPS != fps)
		{
		  core::stringw str = L"Irrlicht – ukazkova scena";
		  str += driver->getName();
		  str += "] FPS:";
		  str += fps;
		  device->setWindowCaption(str.c_str());
		  lastFPS = fps;
		}
	}
	device->drop();
	return 0;
}
```

Po úspěšném zkompilování se vytvoří nový spustitelný *.exe* soubor, který bude reprezentovat aplikaci v níž se budeme moci volně pohybovat pomocí šipek na klávesnici a změnou polohy myši měnit úhel pohledu po trojrozměrném modelu. Pokud bychom chtěli posílat tuto aplikaci dalším zájemcům, je nezbytné společně se spustitelným souborem přiložit i grafické soubory s texturami a datové soubory s objekty včetně *.irr* souboru.

ZÁVĚR

Hlavním cílem práce bylo vytvoření virtuálního modelu budovy Fakulty aplikované informatiky ve Zlíně s kódovým označením U5. Dále tento vstupní model připravit pro některý z dostupných 3D enginů tak, aby bylo možné se po celém virtuálním modelu volně pohybovat.

Výstupním kritériem pro zvolený engine bude především plynulý chod na průměrně výkonné pracovní stanici, tedy aplikace vykreslující real-time model s více jak dvaceti pěti snímky za vteřinu. Docílení požadovaného *fps* bude záležet na detailech a množství vložených 3D objektů a rozmanitosti celé budovy respektive scény. Pro aplikaci bude použit tzv. low-polygon model. Přesto není vyloučeno, že 3D engine díky dobré algoritmizaci a matematických výpočtů bude schopen v reálném čase zpracovat i objemnější data.

Implementovaný low-poly model nebude zřejmě obsahovat výbavu jednotlivých kanceláří (které jsou také přesně rozměřeny a vymodelovány), tím je myšlen nábytek, šanony, složky či další příslušenství. Právě optimalizace GPU a lepší algoritmizace výpočtů scény by mohla přispět k zakomponování i složitějších objektů. Proto bych chtěl i po magisterském studiu pokračovat v doktorandském studiu, kde jsem podal přihlášku na téma: Návrh metodiky pro hodnocení výkonnosti GPU a modifikací pro evoluční princip činnosti.

V teoretické části bylo záměrně věnováno poměrně hodně prostoru pro popis křivek, jejich matematických definicí a také aplikací v praxi. Při hlubším zamyšlení totiž nakonec dojdeme k závěru, že 3D grafika není nic jiného než aplikace matematiky popisující geometrické tvary a jejich vzájemnou kombinací vytváříme rozsáhlé prostorové scény či rozmanité modely.

Samotný 3D model je tvořen ve vývojovém prostředí aplikace 3D Studio Max a Blender, který je součástí výuky studijního předmětu na univerzitě a prakticky tak ukazuje studentům jaké využití může mít v praxi.

Hlavní vizí celého projektu je do budoucna vytvoření aplikace, která bude zveřejněna v multiplatformním provedení a studenti budou mít možnost projekt rozšiřovat či optimalizovat k vyšší výkonnosti úpravou programových algoritmů.

Mimo to bude určena nejen studentům, ale i široké veřejnosti jako pomůcka v orientaci po budově FAI včetně volného pohybu po něm.

Význam a obsah práce byl také prezentován na mezinárodní soutěži studentské tvůrčí a odborné činnosti STOČ‘2007 v Ostravě v kategorii S2 – Aplikovaná informatika.

ZÁVĚR V ANGLIČTINĚ

The main target of the task has been to create of a virtual model of the Faculty of Applicated informatics building in Zlin with the code sign U5. Also that input model should be prepared for any available 3D engine in that way so that it should be possible to freely move around the whole virtual model.

The output criteron for an selected engine will be particularly the smooth run on an average workstation, so the application depicting a real-time model with more than 25 frames per second. The final FPS will depend on a details and an amount of rendered 3D objects and a variety of the whole building or more precisely on the scene. A Low-polygon model will be used for the application. But still, it is not excluded that the 3D engine, according to a good algorithm development and math calculation, will be able to process also the huge data in real time.

The implemented low-polygon model probably won't contain the equipment of offices (which are also exactly measured and modelled) by which I mean eg. forniture, shannons, folders and other accessories. Just the optimalization of GPU and better algorithm calculations might help with implementation of more complex obejcts. That is why, after mine ingeneering studies, I would like to continue with graduant studies where I entered on the topic: Design of methodics for evaluation of GPU performance and modifications for evolutional principle aktivity

In theoretical part were a lot purposely topic focused on curves description, their mathematical definitons and also aplication in working experience. Throught the mention we finally achieve a fact, that 3D graphics is nothing else then mathematic formula which describes geometric shapes and their relative combination we do huge scenes or various models.

Single 3D model is done in 3D Studio Max and mainly in Blender graphic development software, that is a part of study program on university and practically shows to students how they can take advantage of Blender in profession.

The general vision for future is done an application, which would be published in multiplaform versions and students could

extend or optimize that project to better performace by programming effective algorithm. Besides will be indicated not only for students but for world wide community as a help in orientation over the FAI building with the inclusion of free move and look on virtual model.

Sense and project content was either presented on international contest of students creative and professional activity STOČ'2007 in Ostrava in S2 - Applied informatics category..

SEZNAM POUŽITÉ LITERATURY

[1] POKORNÝ, Pavel. *Blender - naučte se 3D grafiku*, BEN 2006, ISBN 80-7300-203-5

[2] KŘÍŽ, J.: *3ds max – hotová řešení*. Computer press, Praha 2006, ISBN 80-2510-885-6

[3] ŽÁRA Jiří, BENEŠ Bedřich, SOCHOL Jiří, FELKEL Petr, *Moderní počítačová grafika (2. vydání)*, Computer Press, 2005, ISBN 80-251-0454-0

[4] ALEXAND, Lubomír. Diplomová práce - *Výuka počítačové grafiky cestou WWW*, VUT Brno. Dostupný z: <http://lubovo.misto.cz/_MAIL_/curves/obsah.html>

[5] Internetový portál Science World. Dostupný z <http://www.scienceworld.cz/>

[6] Autodesk, Dostupný z: <http://www.autodesk.com>

[7] WxWidgets Library. Dostupný z: <http://www.wxwodgets.org>

[8] GIMP Dokumentace. Dostupný z: <http://www.gimp.org>

[9] Blender 3D Dokumentace. Dostupný z: <http://www.blender.org>

[10] Irrlich Engine. Dostupný z: <http://irrlicht.sourceforge.net>

[11] Irrlicht Nico blog. Dostupný z: <http://www.irrlicht3d.org/>

[12] Wikipedia. Dostupný z: http://www.wikipedia.org

[13] Zpravodajský server o počítačové grafice. Dostupný z <http://www.grafika.cz/>.

[14] NVIDIA SDK. Dostupný z: <http://developer.nvidia.com>

[15] NURBS křivky. Dostupný z: <http://www.root.cz/clanky/krivky-nurbs-1/>

[16] BOARDMAN, Ted. 3DS MAX 5 *Podrobná příručka*, Computer Press Praha, 2004. 424 s. ISBN: 80-722-6798-1.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

Aditivní	Součtový
CAD	Computer Aires Design – Volný počítačový design
Edge	Strana polygonu
Extrude	vytažení
Gamma	Míra kontrastu
GUI	Graphic User Interface – Grafické prostředí
Hue	Barevný tón
Channel	Kanál
Kubika	Geometrický utvar složený z křivek
Lightness	Světlost
Mesh	Sítový povrch objektu
NTSC	National Television Standards Committee - televizní standard pro USA a Japonsko
NURBS	Non-Uniform Rational B-Splines
Particle	Částice
Plugin	Zásuvný modul
Polygon	n-úhelník
Real-time	V reálném čase
Rendering	Vykreslování
Saturation	Sytost
SECAM	Séquentiel couleur à mémoire, televizní norma
Spline	Křivka
Surface	Síťový povrch
UNISURF	Programovací systém pro návrh křivek

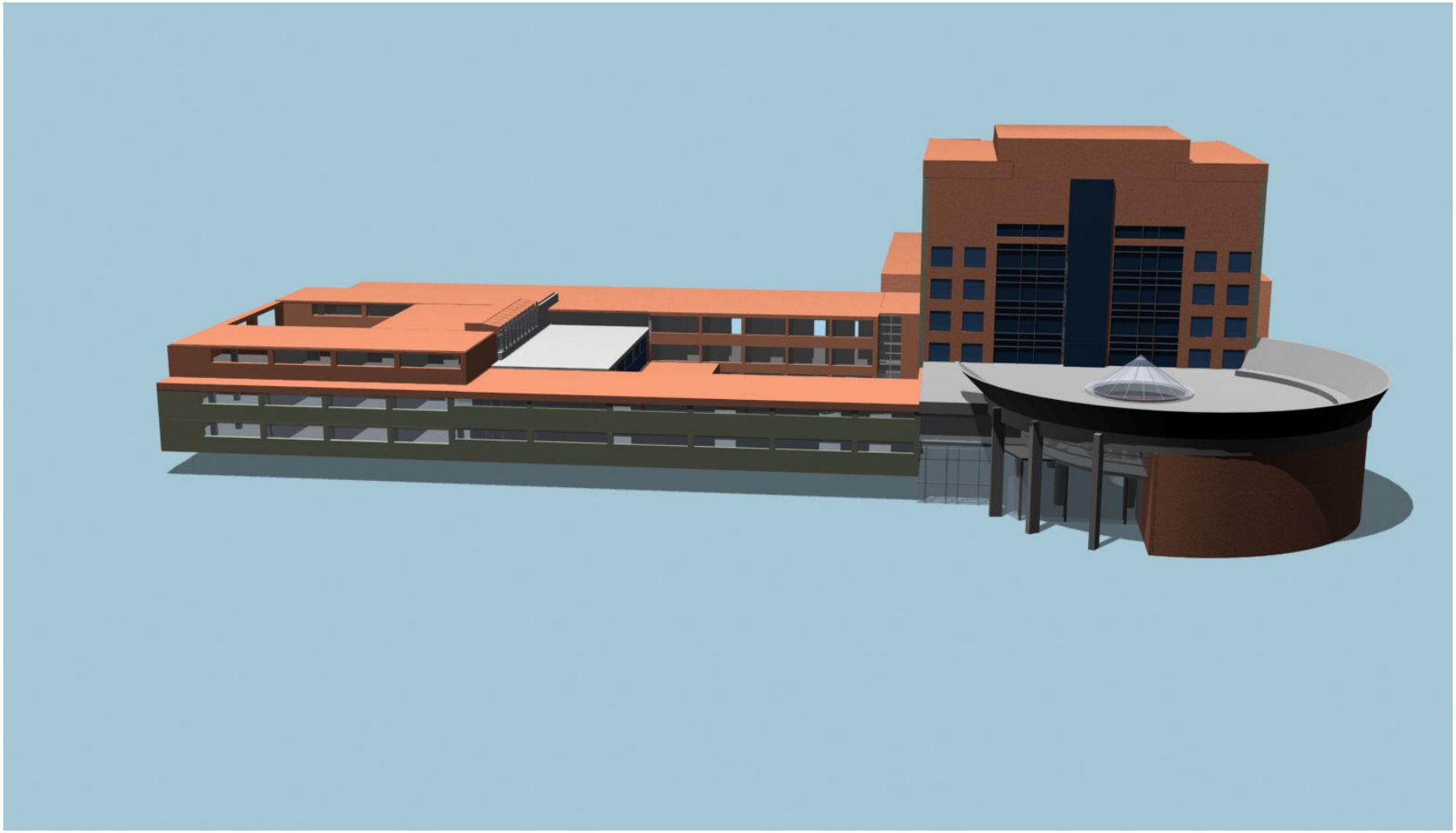
Value	Hodnota
Vertex	Bod, vrchol
VRML	Virtual Reality Modeling Language, programovací jazyk pro vytváření virtuálních světů

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: 3D MODEL FAI UTB – U5



PŘÍLOHA P II: POROVNÁNÍ 3D MODELU S REALITOU