Univerzita Komenského
Fakulta Matematiky, Fyziky
a Informatiky
Katedra algebry, geometrie
a didaktiky matematiky
Odbor: informatika

Diplomová práca

PostScriptové nástroje na vykresľovanie geometrických objektov

Autor: Tomáš Macháček
Školiteľ: RNDr. Pavel Chalmovianský, PhD.

Bratislava 2006

Čestne vyhlasujem, že som túto diplomovú prácu napísal sám, len s využitím uvedenej literatúry, pod vedením diplomového vedúceho.

Bratislava, máj 2006 Tomáš Macháček

Abstrakt

Existuje viacero programov, ktoré vykresľujú danú 3D scénu do PostScriptového obrázku. Tieto programy však väčšinou používajú Z-buffer algoritmus na premietanie objektov v scéne. Výsledný obrázok je potom bitmapový a nie je škálovateľný. Iné programy zase neriešia viditeľnosť objektov a tak sa vo výslednom obrázku nachádza zbytočne veľa informácií. My sme sa v našej práci pokúsili zostrojiť návod, ako spraviť program, ktorý by premietal danú 3D scénu pomocou perspektívneho premietania z $\mathbb{R}^3$ do $\mathbb{R}^2$ a riešil viditeľnosť objektov v scéne. Premietnuté objekty reprezentujeme trojuholníkovými mešmi alebo mešmi zloženými z Coonsových záplat. Vďaka tejto reprezentácii potom kreslíme objekty v PostScripte využívajúc tieňovania uvedeného v PostScript Language Level 3.

Kľúčové slová: PostScript, tieňovanie, perspektívna projekcia, viditeľnosť, 3D scéna.

Obsah

Zoznam obrázkov

Úvod

V súčasnosti existuje veľa metód ako zobraziť 3D scénu. Odlišujú sa v tom, čo je pre ne v generovaní obrazu najdôležitejšie. V počítačových hrách je to predovšetkým rýchlosť zobrazovania scény, v technickom kreslení ide o presnosť. Ďalej sú tu rôzne umelecké techniky, ktoré sa zase snažia zobraziť scénu nejakým neštandardným spôsobom.

Naša práca pojednáva o tom, ako z 3D scény vyrobiť vektorový obrázok, ktorý je pekne škálovateľný. Nejde však len o to, aby sa hrany pri zväčšovaní nerozmazávali, ale aj o to, že keď zobrazíme guľu, tak jej okraje budú naozaj okrúhle a nie len do istého zväčšenia.

Boli vyvinuté metódy, ako prerobiť bitmapový obrázok na vektorový. My sme sa však sústredili na priamu transformáciu 3D scény na vektorový obrázok používajúc postupov geometrie a aproximácií.

Okrem toho, výsledok zapisujeme do formátu PostScritp. Táto skutočnosť nás obmedzuje v používaní reprezentácie premietnutých objektov a v možnostiach tieňovania a farbenia.

V prvej kapitole je prehľad možností kreslenia, ktoré nám ponúka PostScript. Ďalej tu môžeme nájsť priblíženie osvetľovacieho modelu v štandarde X3D a predstavenie štandardu X3D. Okrem týchto prehľadových častí nám prvá kapitola poskytuje definície a metódy, ktoré budeme využívať

Druhá kapitola nám predstavuje hrubý náčrt algoritmu, pomocou ktorého objekty 3D scény vykresľujeme.

V tretej kapitole popisujeme metódy, akými sme premietali jednotlivé objekty 3D scény. Popisuje problémy, ktoré nastávajú, keď používame perspektívne premietanie z $\mathbb{R}^3$ do $\mathbb{R}^2$ miesto klasického algoritmu Z-Buffer.

Kapitola 4 je celá určená trojuholníkovým mešom, lebo tie sú pravdepodobne v počítačovej grafike najrozšírenejšie. Zoznámime sa tu s metódami viditeľnosti a hľadania prienikov stien mešov.

V ďalšej časti, kapitole 5, si ukážeme ako riešiť prienik všetkých zobrazovaných objektov, nie len trojuholníkových mešov. Využívame pri tom aproximačných metód na určenie priesečníkov objektov.

Pre správne zobrazovanie objektov v scéne je dôležité určenie ich viditeľnosti, teda ich vzájomného prekrývania sa. O tomto pojednáva šiesta kapitola.

Na záver si ukážeme nejaké výsledky, ktoré sme využitím metód opisovaných v tejto práci dosiahli.

Kapitola 1

Prehľad

1.1 PostScript

1.1.1 Čo je PostScript?

V [Inc99] je PostScript definovaný takto:

PostScript je interpretovaný programovací jazyk s veľkými grafickými možnosťami. Jeho primárna funkcia je opísať vzhľad textu, grafických objektov a obrázkov na tlačených alebo zobrazovaných stranách podľa zobrazovacieho modelu Adobe (Adobe imaging model). Program v tomto jazyku môže prenášať opis dokumentu z kompozičného systému do tlačového systému alebo riadiť vzhľad textu a grafických objektov na obrazovke. Opis je vysoko úrovňový a nezávislý od zariadenia.

PostScriptový dokument môže byť renderovaný na tlačiarni, obrazovke alebo inom výstupnom zariadení pomocou PostScriptového interpretera pre dané zariadenie. Ako interpreter vykonáva príkazy PostScriptu, konvertuje vysoko úrovňový opis do nízko úrovňového rastrového dátového formátu pre dané zariadenie.

Jazyk v sebe zahŕňa bežné dátové typy(čísla, polia, reťazce), riadiace štruktúry (podmienky, cykly, procedúry). Obsahuje aj niektoré neštandardné časti, napr. slovníky.

PostScriptový program môže byť buď v textovom režime alebo ako binárny súbor.

1.1.2 Farbenie v PostScripte

Existuje viacero spôsobov farbenia objektov v PostScripte. Farbenie zvolenou farbou alebo farbenie vzorom (pattern). Pre farbenie farbou sa nastaví nejaká farebná hodnota pre daný priestor farieb (colorSpace napr. DeviceGray, DeviceRGB, DeviceCMYK).

Vzory sú v PostScritpe definované ako slovníky. Delia na dva typy: dlaždicové vzory (tiling patterns), tieňovacie vzory (shading patterns).

Dlaždicové vzory sa skladanú z malých častí nazývaných bunky (pattern cells). Je treba zadefinovať jednu bunku. Tá sa potom opakovane kreslí po celej farbenej ploche.

Zaujímavejšie sú tieňovacie vzory. Pomocou nich, ako aj názov napovedá, je možné vykresľovanie hladkých objektov. Tieto vzory sa delia na sedem typov, podľa toho, aký typ tieňovacej funkcie použijeme:

- Function based shading (typ 1)
- Axial shading (typ 2)
- Radial shading (typ 3)
- Free-form Gouraud shaded triangle meshes (typ 4)
- Lattice-form Gouraud shaded triangle meshes (typ 5)
- Coons patch meshes (typ 6)
- Tensor-product patch meshes (typ 7)

Tieňovacia funkcia je definovaná ako slovník. Slovník je objekt, ktorý v sebe obsahuje kľúče a hodnoty. Každému kľúču prislúcha hodnota, podobne ako u polí.

Farbenie tieňovacími vzormi sa robí v piatich krokoch:

- Zadefinuje sa prototyp vzoru (slovník).
- Vzor sa zainštancuje.
- Vyberie sa priestor farieb vzoru.
- Vzor sa nastaví ako farba na farbenie (current color).
- Zavolaju sa procedúry pre farbenie (napr. fill, stroke,. . .)

1.1.3 Definícia tieňovacieho vzoru v PostScripte

Podobne ako tieňovacia funkcia, aj tieňovací vzor (shading pattern) je v PostScripte reprezentovaný ako slovník. Každý tieňovací vzor musí mať definované nasledovné kľúče:

- *PatternType* – musí byť nastavený na 2 (1 je dlaždicový vzor)
- *Shading* – slovník pre typ tieňovacej funkcie.

Možné je definovať aj ďalšie kľúče. Viac je o tom popísané v [Inc99] a [Inc97].

1.1.4 Slovník pre typ tieňovacej funkcie

Podľa toho, aké tieňovanie chceme použiť sa bude definovať aj slovník. Avšak 2 kľúče musí mať vždy:

- *ShadingType* – typ: 1 – 7, ako sú popísané v kapitole 1.1.2
- *ColorSpace* – Priestor farieb. viď [Inc99]

Ďalšie nepovinné kľúče sú v [Inc99] a [Inc97].

Nebudeme popisovať všetky typy, sústredíme sa len na typ 4 a typ 6. Typ 7 je podobný ako typ 6. Podrobný opis je v [Inc99] a [Inc97].

Typ 4 - Free-form Gouraud shaded triangle meshes

Tento typ tieňovacej funkcie využíva Gouraudovo tieňovanie. Je daný trojuholník. V jeho vrcholoch máme dané farby. Tieňovacia funkcia interpoluje tieto farby.

V slovníku je treba dourčiť nasledujúce kľúče:

- *DataSource* – pole, reťazec alebo súbor. Obsahuje v sebe vrcholy trojuholníka spolu s farbami v týchto vrcholoch
- *BitsPerCoordinate, BitsPerComponent, BitsPerFlag, Decode* – tieto kľúče sa definujú, ak *DataSource* nie je pole.

DataSource je zoznam vrcholov idúcich za sebou. Každý vrchol je definovaný týmito hodnotami:

$$f\ x\ y\ c_1 \ldots c_n$$

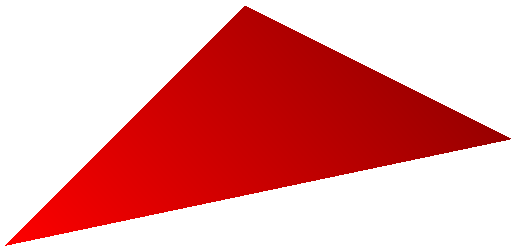
x, y sú súradnice vrcholu, $c_1 \ldots c_n$ sú farebné zložky určené podľa toho, aký priestor farieb sme si zvolili. Napr. pre DeviceRGB je n = 3 a jednotlivé hodnoty zodpovedajú farebným zložkám červená, zelená, modrá. Hodnota f znamená:

- 0 pre nový trojuholník
- 1, 2 pre pripojenie bodu k prvým alebo druhým dvom bodom predchádzajúceho trojuholníka.

Na obrázku 1.1 je príklad použitia tieňovacieho vzoru typu 4. Za ním nasleduje zdrojový jeho zdrojový kód:

```
/DeviceRGB setcolorspace

/shd
<<
```



Obrázok 1.1: Gouraudovo tieňovanie

```
/ShadingType 4
/ColorSpace [/DeviceRGB]
/DataSource [
0 10 10 1 0 0
0 100 100 0.7 0 0
0 200 50 0.6 0 0]
>>
def

/pat
<<
/PatternType 2
/Shading shd
>>
matrix
makepattern
def

newpath
pat setpattern
0 0 200 150 rectfill
stroke
closepath
```

Typ 6 - Coons patch meshes

Typ tieňovacej funkcie č. 6 je zostrojený z jednej alebo viac farebných záplat (Coonsove záplaty). Každá záplata je ohraničená štyrmi Bézierovými krivkami.

Farbenie záplaty je nasledovné. Zoberie sa jednotkový štvorec. V jeho vrcholoch sú definované farby. Pomocou bilineárnej interpolácie sa dopočítajú farby po celom štvorci. Potom sa štvorec zobrazí na záplatu. Vrcholy štvorca sa priradia rohom záplaty, strany štvorca sa premietnu na hranice záplaty (Bézierove krivky).

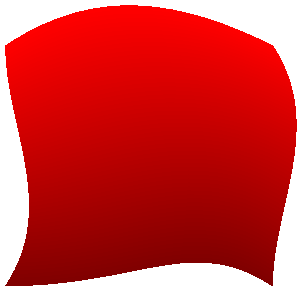
Slovník pre tento typ má povinné kľúče (okrem už spomínaných):

- *DataSource* – pole, reťazec alebo súbor. Obsahuje v sebe 12 riadiacich vrcholov pre Bézierove krivky reprezentujúce hranice záplaty.
- *BitsPerCoordinate, BitsPerComponent, BitsPerFlag, Decode* – tieto kľúče sa definujú, ak *DataSource* nie je pole.

DataSource je zoznam riadiacich vrcholov hraníc + štyroch farieb vo vrcholoch záplaty

$$f\, x_1\, y_1\, x_2\, y_2 \ldots x_{12}\, y_{12}\, c_1\, c_2\, c_3\, c_4 \tag{1.1}$$

f má podobnú funkciu ako pri type 4. Body $x_1\, y_1\, x_2\, y_2 \ldots x_{12}\, y_{12}$ sú riadiace vrcholy hraníc. $c_1\, c_2\, c_3\, c_4$ sú farby vrcholov, pričom každé z $c_1\, c_2\, c_3\, c_4$ obsahuje farebné zložky podľa priestoru farieb napr. $c_1 = 0\,0\,0.1$.



Obrázok 1.2: Tieňovanie typu 6

Na obrázku 1.2 je príklad použitia tieňovacieho vzoru typu 6. Nasleduje jeho zdrojový kód.

```
/DeviceRGB setcolorspace

/shd1
<<
/ShadingType 6
/ColorSpace [/DeviceRGB]
/DataSource [0 10 100 10 70 30 40
10 10 60 10 80 30
110 10 110 40 130 70
```

```
110 100 70 120 40 120
1 0 0 0.5 0 0 0.5 0 0 1 0 0]
>>
def

/pat
<<
/PatternType 2
/Shading shd1
>>
matrix
makepattern
def

newpath
pat setpattern
0 0 150 150 rectfill
stroke
closepath
```

Pozn. Typ 7 je založený na podobnom princípe ako typ 6. Len riadiacich vrcholov je šestnásť. Nemáme teda Coonsovu záplatu ale Bézierovu záplatu.

Bližšie informácie o farbení v PostScripte možno nájsť v [Inc99], [Inc97]. Pekný návod na vykresľovanie geometrických objektov a ich farbenie je ponúka [Cas].

1.2 Osvetľovacie modely

Pri vykresľovaní objektov scény potrebujeme zisťovať farbu bodov na povrchu objektu. Na to slúžia osvetľovacie modely. Osvetľovacie modely sa delia podľa prístupu na empirické (Lambert, Phong) a fyzikálne založené (Cook-Torrance, Oren-Nayar). Empirické modely sa opierajú o namerané a odpozorované skutočnosti. Fyzikálne založené modely pracujú na reálnych fyzikálnych dátach ako je hustota, index lomu a ďalšie. Farebné vlastnosti objektov sú definované ako materiál. Štruktúra materiálu závisí od osvetľovacieho modelu.

Na to, aby sme vedeli určiť farbu v bode na povrchu objektu, potrebujeme vedieť, ako sa v tomto bode odráža dopadajúce svetlo. Na to slúži obojsmerná distribučná funkcia (bidirectional reflectance distribution function BRDF). Počíta pravdepodobnosti odrazu a lomu svetla dopadajúceho pod určitým uhlom.

Niektoré základné princípy práce osvetľovacích modelov sú opísané v [Žár98]. My sme na osvetlenie scény použili osvetľovací model uvedený v špecifikácii X3D [X3D05]. Je veľmi podobný Phongovmu osvetľovaciemu

modelu, ktorý je jedným z najrozšírenejších. Preto si najprv predstavíme Phongov osvetľovací model a potom osvetľovací model uvedený v [X3D05].

1.2.1 Phongov osvetľovací model

Phongov osvetľovací model je empirický model. To znamená, že výpočet farby na povrchu objektu je založený na odpozorovaných a odmeraných hodnotách, nie na presných fyzikálnych hodnotách.

Farebné vlastnosti objektu sa nazývajú materiál. Phongov osvetľovací model vyžaduje v materiáli tri farebné zložky (ambientná, difúzna, spekulárna) a hladkosť (shininess).

Ambientná farba je farba, ktorou sa objekt zafarbí bez ohľadu na to, kde sa nachádza zdroj svetla. Táto zložka sa môže použiť na nahrádzanie nepriameho osvetlenia.

Difúzna farba je farba svetla, ktoré sa po dopade na povrch objektu odráža všetkými smermi. Je závislá od uhlu dopadu.

Spekulárna farba je farba svetla odrazeného v smere hladkého odrazu, t.j. takého, ktorého uhol odrazu je rovný uhol dopadu prichádzajúceho svetla. Táto zložka je závislá od uhlu medzi vektorom pohľadu (vektor z bodu pozorovateľa do bodu dopadu svetla) a vektorom hladkého odrazu svetla.

BRDF pre výpočet farby vyzerá nasledovne:

$$I = L_a k_d + L_d k_d \max\left(0, (\vec{L} \cdot \vec{N})\right) + L_s k_s \max\left(0, (\vec{R} \cdot \vec{V})^n\right) \tag{1.2}$$

I je výsledná farba v bode na povrchu objektu. L_a, L_d a L_s sú ambientná, difúzna a spekulárna zložka svetla, k_a, k_d, k_s sú ambientná, difúzna a spekulárna zložka materiálu objektu. vektor $\vec{L}$ je smer dopadajúceho svetelného lúča, $\vec{N}$ je normálový vektor bodu vzhľadom na povrch objektu, $\vec{V}$ je vektor pohľadu, t.j. vektor z bodu pozorovateľa do bodu, v ktorom počítame farbu. Vektor $\vec{R}$ je vektor hladkého odrazu svetla od povrchu objektu. Platí preň vzťah:

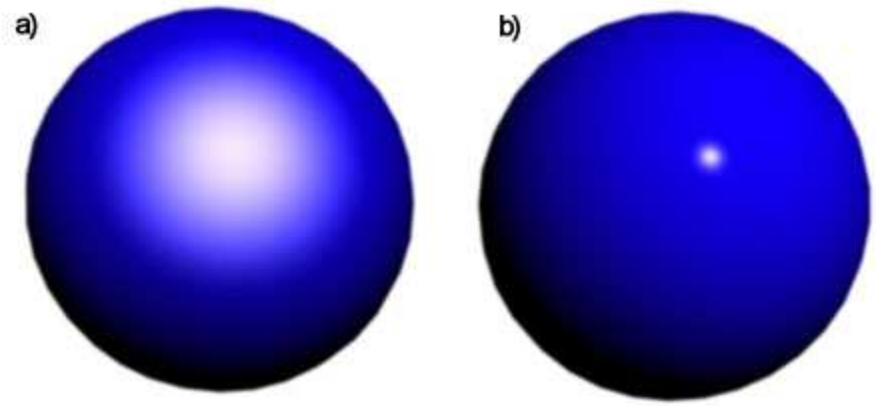
$$\vec{R} = 2(\vec{L} \cdot \vec{N})\vec{N} - \vec{L}$$

Exponent n zo vzťahu (1.2) je koeficient hladkosti. Čím väčšie je n, tým hladší sa zdá byť povrch objektu (obr. 1.3).

Niekedy sa pri výpočte farby na povrchu objektu berie do úvahy aj vzdialenosť svetla od objektu (attenuation). Potom sa BRDF trochu zmení:

$$I = L_a k_d + f_{att} L_d k_d \max\left(0, (\vec{L} \cdot \vec{N})\right) + f_{att} L_s k_s \max\left(0, (\vec{R} \cdot \vec{V})^n\right) \tag{1.3}$$

f_{att} je funkcia počítajúca koeficient öslabenia"svetla závislý od vzdialenosti svetla od bodu na povrchu objektu. Pre každé svetlo v scéne sú dané



Obrázok 1.3: príklad použitia Phongovho osvetľovacieho modelu. a) malý koeficient hladkosti, b) veľký koeficient hladkosti

tri hodnoty súvisiace s oslabením svetla. Označujú sa c_1, c_2, c_3. Ďalej nech d je vzdialenosť svetla od bodu na povrchu objektu. Funkcia f_{att} vyzerá nasledovne:

$$f_{att} = \min\left(1, \frac{1}{c_1 + c_2 d + c_3 d^2}\right)$$

1.2.2 Osvetľovací model v X3D

Než si predstavíme samotný osvetľovací model, povieme si niečo o svetlách a materiáloch v X3D.

Svetlá

V X3D poznáme tri druhy svetla: smerové svetlo (directional light), bodové svetlo (point light) a reflektor (spotlight). Výpočet farby BRDF funkciou je závislí na type svetla v scéne.

Všetky tri typy svetiel majú nasledovné spoločné parametre:

- *ambientIntensity* - určuje, akú intenzitu má ambientná časť svetla.
- *color* - farba svetla.
- *intensity* - intenzita svetla.
- *on* - určuje, či je svetlo zapnuté alebo vypnuté a teda či sa má brať do úvahy pri počítaní osvetlenia.

Smerové svetlo má ďalší parameter *direction*. Tento parameter je vektor, ktorý definuje polpriestor osvetlený týmto svetlom. Smerové svetlo osvetľuje objekty vždy z rovnakého smeru, nezávisle od jeho polohy vzhľadom na objekty. To znamená že vektor dopadajúceho lúča je stále rovnaký[1].

Bodové svetlo má okrem spoločných štyroch parametrov ešte ďalšie tri:

- *attenuation* - pole s tromi hodnotami, určujúce koeficienty c_1, c_2 a c_3 potrebné pre počítanie oslabenia svetla (viď stať 1.2.1).
- *location* - pozícia svetla v scéne.
- *radius* - bodové svetlo vyžaruje lúče na všetky smery. Avšak osvetlí len objekty do vzdialenosti určenej týmto parametrom.

A konečne svetlo typu reflektor. Ako názov napovedá, toto svetlo osvetľuje objekty nachádzajúce sa v kuželi. Parametre pre toto svetlo sú:

- *attenuation* - znamená to isté ako u bodového svetla.
- *direction* - orientácia svetla. Priamka určená týmto vektorom a pozíciou svetla tvorí os kužeľa, v ktorom sú objekty týmto svetlom osvetlené.
- *location* - pozícia svetla.
- *radius* - to isté ako u bodového svetla.
- $beamWidth$ - nech $\vec{v}$ je vektor zo svetla do bodu P. Ak uhol medzi vektorom $\vec{v}$ a vektorom *direction* je menší ako $beamWidth$, bod P je osvetlený plnou intenzitou svetla (keď nerátame oslabenie svetla určené parametrom *attenuation*).
- $cutOffAngle$ - opäť nech $\vec{v}$ je vektor zo svetla do bodu P. Ak uhol medzi vektorom $\vec{v}$ a vektorom *direction* je väčší ako $cutOffAngle$ bod P nie je svetlom osvetlený vôbec.

Ďalšie podrobnosti o svetlách a ich parametroch sú v [X3D05].

Materiál

Podobne ako Phongov osvetľovací model, aj osvetľovací model špecifikovaný v X3D potrebuje mať na výpočet farby bodu na povrchu objektu definovaný materiál objektu.

Materiál v X3D má nasledovné parametre:

- $diffuseColor$, $emissiveColor$, $specularColor$ - difúzna farba, emisná farba a spekulárna farba.

[1] Týmto vektorom je práve vektor *direction*

- *ambientIntensity* - intenzita ambientného osvetlenia. Ambientná farba sa počíta z difúznej (viac v stati 1.2.2).
- *shininess* - koeficient pre hladkosť objektu (vid stať 1.2.1).
- *transparency* - priehľadnosť.

Podrobné informácie možno nájsť v [X3D05].

BRDF

Poďme si teraz priblížiť, ako sa počíta osvetlenie v X3D scéne.

BRDF funkcia:

$$I = (1 - f_0)I_F + f_0\left(O_E + \sum\left(on_i \cdot att_i \cdot spot_i \cdot I_{L_i}(a_i + d_i + s_i)\right)\right) \quad (1.4)$$

- I_F - farba hmly
- f_0 - interpolant hmly. Je závislý od hustoty hmly, typu hmly (lineárna, exponenciálna) a vzdialenosti bodu od pozorovateľa.
- O_E - emisná farba materiálu.
- on_i - parameter *on* i-teho svetla.
- att_i - koeficient oslabenia svetla. Počíta sa rovnako ako u Phongovho osvetľovacieho modelu (stať 1.2.1).
- $spot_i$ - pre bodové a smerové svetlo je $spot_i = 1$. Pre reflektorové svetlo sa tento parameter počíta nasledovne. Nech $\vec{v}$ je vektor zo svetla do bodu P. Nech α je uhol, ktorý zvierajú vektor $\vec{v}$ a vektor *direction*.
  - Ak $\alpha \leq beamWidth$, tak $spot_i = 1$.
  - Ak $\alpha \geq cutOffAngle$, tak $spot_i = 0$.
  - Inak $spot_i = \frac{(cutOffAngle - \alpha)}{(cutOffAngle - beamWidth)}$.
- I_{L_i} - farba i-teho svetla
- a_i - ambientná zložka. Je to súčin ambientnej intenzity svetla, ambientnej intenzity materiálu a difúznej farby materiálu.
- d_i - difúzna zložka.
- s_i - spekulárna zložka.

$$
\begin{aligned}
d_i &= I_i k_d (\vec{N} \cdot \vec{L}) \\
s_i &= I_i k_s \left(\vec{N} \cdot \left(\frac{(\vec{L}+\vec{v})}{|\vec{L}+\vec{v}|} \right) \right)^{128n}
\end{aligned}
$$

I_i je intenzita i-teho svetla, k_d a d_s sú difúzna a spekulárna zložka materiálu, n určuje hladkosť povrchu objektu (shininess). $\vec{N}$ je normálový vektor bodu scény, $\vec{L}$ je smerový vektor dopadajúceho lúča a $\vec{v}$ je vektor z bodu scény do bodu pozorovateľa. Vektory $\vec{N}$, $\vec{L}$, $\vec{v}$ sú normalizované.

Ďalšie informácie o osvetľovacom modeli v X3D sú v [X3D05]

1.3 Bézierove kubiky

1.3.1 Definícia

Definícia 1.3.1 *Polynomická krivka*

$$
B(t) = \sum_{i=0}^{n} B_i^n(t) \cdot V_i, \quad t \in \langle 0, 1 \rangle \tag{1.5}
$$

sa nazýva Bézierova krivka stupňa n. $V_i \in \mathbb{R}^3$ sú riadiace vrcholy krivky a $B_i^n(t)$ je Bernsteinov polynóm.

Bernsteinov polynóm $B_i^k(t)$ je definovaný nasledovne:

$$
B_i^k(t) = \binom{k}{i} t^i (1-t)^{k-i}, \quad i = 0, 1, \ldots, k \tag{1.6}
$$

Existuje dohoda, že $B_i^k(t) = 0$ pre $i > k$ alebo $i < 0$.

Riadiace vrcholy V_i, $i = 0, 1, \ldots, n$ tvoria polygón nazývaný riadiaci polygón.

Definícia 1.3.2 *Bézierova kubika je Bézierova krivka stupňa 3.*

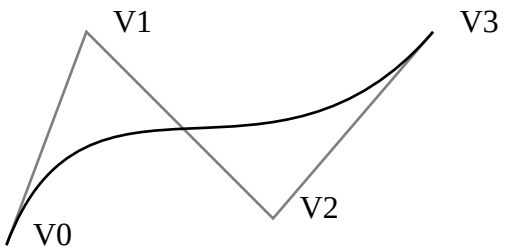
V našej práci sme sa sústredili na Bézierove kubiky, lebo tie nám stačia na popísanie kriviek, s ktorými pracujeme. Okrem toho, Bézierove krivky vyššieho stupňa sú nepraktické, lebo sú časovo náročné na výpočet.

Na obrázku 1.4 je príklad Bézierovej kubiky.

Bézierove krivky a Bernsteinov polynóm sú bližšie opísané v [Sed03a], [Gre02].

1.3.2 Vlastnosti Bézierových kriviek

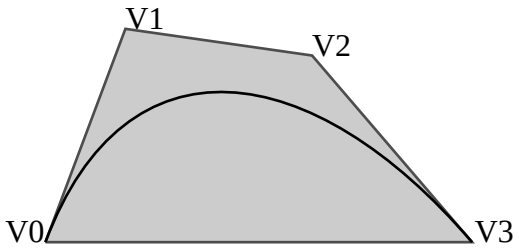
Uvedieme si niekoľko vlastností Bézierových kriviek, ktoré nám v ďalšej práci budú užitočné.



Obrázok 1.4: Príklad Bézierovej kubiky

1. Bézierova krivka interpoluje svoje krajné body (obr. 1.4), t.j. $B(0) = V_0$ a $B(1) = V_n$.

2. Bézierova krivka leží v konvexnom obale svojho riadiaceho polygónu (obr. 1.5).



Obrázok 1.5: Bézierova krivka v konvexnom obale svojho riadiaceho polygónu.

3. Úsečka V_0V_1 tvorí dotyčnicu Bézierovej krivky v bode $B(0)$ a úsečka $V_{n-1}V_n$ tvorí dotyčnicu Bézierovej krivky v bode $B(1)$ (obr. 1.4).

4. Invariantnosť vzhľadom na affíne transformácie. Nech φ je affína transformácia, potom platí:

$$\varphi(B(t)) = \sum_{i=0}^{n} B_i^n(t) \cdot \varphi(V_i)$$

5. Bézierova krivka sa nemení viac ráz ako jej riadiaci polygón. Presnejšie: Pre každú nadrovinu $H \subset \mathbb{R}^d$ platí, že počet prienikov nadroviny H s Bézierovou krivkou $B(t)$ je menší alebo rovný počtu prienikov nadroviny H s riadiacim polygónom Bézierovej krivky $B(t)$.

Vlastnosti 2 a 5 vyžijeme pri hľadaní priesečníkov Bézierových kriviek. Invariantnosť na affíne transformácie je užitočná pri rotácii a posúvaní Bé-

zierovej krivky. Tretia vlastnosť nám pomôže pri konštrukcii Bézierových kriviek v $\mathbb{R}^2$, keď budeme aproximovať premietnuté krivky.

Dôkazy vlastností a niektoré ďalšie vlastnosti sú v [Sed03a], [Gre02].

1.3.3 De Casteljau-ov algoritmus

Poznáme dva spôsoby ako vyrátať bod na Bézierovej krivke. Priamo dosadením parametra t do rovnice (1.5), alebo de Casteljau-ovým algoritmom. De Casteljau-ov algoritmus je založený na lineárnej interpolácii bodov.

Veta 1.3.1 ***(de Casteljau-ov algoritmus)*** *Nech $V_0, V_1, \ldots, V_n$ sú riadiace vrcholy Bézierovej krivky $B(t)$ stupňa n. Označme $V_i^0(t) = V_i$ pre $i = 0, 1, \ldots, n$. Ďalej nech*

$$V_i^j(t) = (1-t) \cdot V_i^{j-1}(t) + t \cdot V_{i+1}^{j-1}(t) \quad i = 0, 1, \ldots, n-j$$

Potom $V_0^n(t) = B(t)$.

Dôkaz: Urobíme len dôkaz pre Bézierovu kubiku, lebo len tie využívame v tejto práci. Celý dôkaz je v [Gre02].
$n = 3$, riadiace vrcholy: V_0, V_1, V_2, V_3.

$$\begin{aligned}
V_0^3(t) &= (1-t) \cdot V_0^2(t) + t \cdot V_1^2(t) \\
V_0^3(t) &= (1-t) \cdot ((1-t) \cdot V_0^1(t) + t \cdot V_1^1(t)) + \\
&\quad + t \cdot ((1-t) \cdot V_1^1(t) + t \cdot V_2^1(t)) \\
V_0^3(t) &= (1-t)^2 \cdot V_0^1(t) + 2t(1-t) \cdot V_1^1(t) + t^2 \cdot V_2^1(t) \\
V_0^3(t) &= (1-t)^2 \cdot ((1-t) \cdot V_0 + t \cdot V_1) + \\
&\quad + 2t(1-t) \cdot ((1-t) \cdot V_1 + t \cdot V_2) + \\
&\quad + t^2 \cdot ((1-t) \cdot V_2 + t \cdot V_3) \\
V_0^3(t) &= (1-t)^3 \cdot V_0 + 3t(1-t)^2 \cdot V_1 + 3t^2(1-t) \cdot V_2 + t^3 \cdot V_3 \\
V_0^3(t) &= \tbinom{3}{0} t^0 (1-t)^3 \cdot V_0 + \tbinom{3}{1} t^1 (1-t)^2 \cdot V_1 + \\
&\quad + \tbinom{3}{2} t^2 (1-t)^1 \cdot V_2 + \tbinom{3}{3} t^3 (1-t)^0 \cdot V_3 \\
V_0^3(t) &= B_0^3(t) \cdot V_0 + B_1^3(t) \cdot V_1 + B_2^3(t) \cdot V_2 + B_3^3(t) \cdot V_3 \\
V_0^3(t) &= \textstyle\sum_{i=0}^{3} B_i^3(t) \cdot V_i
\end{aligned}$$

□

De Casteljau-ov algoritmus využívame pri hľadaní priesečníkov Bézierových kriviek, keď prerozdeľujeme krivku na dve.

1.3.4 Prerozdelenie Bézierovej krivky

Riadiaci polygón Bézierovej krivky je istou aproximáciou samotnej Bézierovej krivky. Prerozdelením krivky na dve dostaneme dva nové riadiace polygóny, ktoré spolu tvoria lepšiu aproximáciu pôvodnej krivky a sú riadiacimi polygónmi dvoch častí pôvodnej krivky.

Presnejšie: Predstavme si, že chceme rozdeliť Bézierovu krivku $B(t)$ stupňa n definovanú na intervale $\langle 0,1\rangle$ v bode $B(t_0)$. Dostaneme nové Bézierove krivky $B_1(t)$ a $B_2(t)$ stupňa n, pre ktoré platí:

$$B_1(t) = B(t \cdot t_0) \quad t \in \langle 0,1\rangle$$
$$B_2(t) = B(t \cdot (1-t_0) + t_0) \quad t \in \langle 0,1\rangle$$

Riadiace vrcholy pre Bézierove krivky $B_1(t)$ a $B_2(t)$ nájdeme pomocou de Casteljau-ovho algoritmu.

Veta 1.3.2 *Nech V_{1_i} pre $i = 0,1,\ldots,n$ sú riadiace vrcholy krivky $B_1(t)$ a V_{2_i} pre $i = 0,1,\ldots,n$ sú riadiace vrcholy krivky $B_2(t)$. Potom platí:*

$$V_{1_i} = V_0^i(t_0) \quad i = 0,1,\ldots,n$$
$$V_{2_i} = V_i^{n-i}(t_0) \quad i = 0,1,\ldots,n$$

kde vrcholy $V_0^i(t_0)$ a $V_i^{n-i}(t_0)$ sú vrcholy z de Casteljau-ovho algoritmu (veta 1.3.1).

Dôkaz: Pre Bézierovu kubiku stačí len dosadiť a pomocou de Casteljau-ovho algoritmu dorátať. □

1.4 Coonsove záplaty

Coonsove záplaty sú plochy ohraničené štyrmi krivkami $C_1(u)$, $C_2(u)$, $D_1(v)$, $D_2(v)$, $u,v \in \langle 0,1\rangle$, ktoré majú spoločné krajné body. Coonsova plocha $X(u,v)$ je definovaná nasledovne.

Jej hranice tvoria krivky $C_1(u)$, $C_2(u)$, $D_1(v)$, $D_2(v)$, t.j. $X(u,0) = C_1(u)$, $X(u,1) = C_2(u)$, $X(0,v) = D_1(v)$, $X(1,v) = D_2(v)$. $X(u,v)$ je daná predpisom:

$$\begin{aligned} X(u,v) = \quad & (1-u,u)\begin{pmatrix} X(0,v) \\ X(1,v) \end{pmatrix} + (X(u,0),\ X(u,1))\begin{pmatrix} 1-v \\ v \end{pmatrix} - \\ & (1-u,u)\begin{pmatrix} X(0,0) & X(0,1) \\ X(1,0) & X(1,1) \end{pmatrix}\begin{pmatrix} 1-v \\ v \end{pmatrix} \end{aligned}$$

V našom prípade budú vždy hraničné krivky Bézierove kubiky, lebo tak sú definované Coonsove záplaty v [POSTSCRIPT].

1.5 Perspektívne premietanie

Perspektívne premietanie (ďalej nazývané ako projekcia) je zobrazenie f z $\mathbb{R}^3$ do $\mathbb{R}^2$. Ide vlastne o posunutie zobrazovaného bodu P v smere vektora $\vec{v}$ do roviny π. Rovina π sa nazýva priemetňa. Vektor $\vec{v} = P - C$ sa nazýva smer premietania, bod C je stred premietania.

1.5.1 Rovnica projekcie

Uvažujme trojrozmerný euklidovský priestor E nad poľom reálnych čísel $\mathbb{R}$. Nech stred premietania je v bode $C = (0, 0, 0)$. Priemetňa nech je rovina kolmá na os z. Teda rovnica priemetne je $z = z_1$ pre nejaké $z_1 \in \mathbb{R}, z_1 \neq 0$. Nech $A \in E$. Potom pre obraz $B = (x_B, y_B) \in \mathbb{R}^2$ bodu $A = (x_A, y_A, z_A)$ platí:

$$x_B = \frac{z_1}{z_A} x_A$$
$$y_B = \frac{z_1}{z_A} y_A$$

1.6 Štandard X3D

V našej implementácii sme použili 3D scénu zapísanú v X3D formáte, preto si teraz tento formát trochu predstavíme. Informácia o X3D a jeho presná špecifikácia je v [X3D05].

X3D (Extensible 3D) je štandard pre definovanie 3D obsahu a multimédií (3D scény, animácie, zvuky,...). X3D je nasledovník známeho VRML (Virtual Reality Modeling Language) a dopĺňa ho o nové možnosti ako napríklad nové kódovanie (XML).

V našej implementácii využívame práve XML kódovanie na čítanie 3D scény z x3d súboru.

Štandard X3D v sebe zahŕňa definície nasledovných objektov:

- 3D grafika - Polygonálna geometria, parametrická geometria, hierarchické transformácie, osvetlenie scény, materiály a mapovanie textúr.
- 2D grafika - Text, 2D objekty zobrazované v 3D transformačnej hierarchii.
- animácia - animácie vrátane animácie človeka a morfingu.
- audio a video.
- interakciu s užívateľom.
- navigáciu v scéne - kamery, pohyb užívateľa v scéne, kolízie.
- možnosť definície vlastných objektov.
- skripty
- sprístupnenie X3D cez sieť.
- fyzikálne simulácie - napr. animácia človeka

Pre nás sú zaujímavé definície 3D a 2D grafiky, lebo objekty v týchto definíciách budeme zobrazovať.

X3D je veľmi rozsiahli štandard. Viac o ňom v [X3D05], kde sú aj informácie o XML kódovaní.

Kapitola 2

Náčrt algoritmu

Teraz si predstavíme celý algoritmus, ako budeme riešiť vykresľovanie danej 3D scény. Jednotlivé časti algoritmu riešime v nasledujúcich kapitolách.

1. Najprv objekty scény zobrazíme v perspektívnom premietaní. Niektoré objekty sa dajú premietnuť priamo, ale na niektoré musíme použiť aproximačné metódy. Aby sme s premietnutými objektmi mohli dobre pracovať, reprezentujeme si ich buď ako trojuholníkové meše alebo ako Coonsove meše.

2. Využijúc dáta získané projekciou urobíme lokálnu viditeľnosť objektov. Toto sa týka len trojuholníkových mešov, lebo u ostatných objektoch sme viditeľnosť riešili pri premietaní.

3. Keďže sa objekty v scéne môžu pretínať, nájdeme ich priesečníky a rozdelíme ich na disjunktné časti.

4. Keď máme vyriešenú lokálnu viditeľnosť a priesečníky objektov, vyriešime globálnu viditeľnosť, kde odstraňujeme časti objektov prekryté inými objektmi

5. Zostali nám priemety objektov, ktoré máme reprezentované ako trojuholníkové a Coonsove meše. Pomocou Postskriptových metód tieto meše vykreslíme.

Kapitola 3

Premietanie

3.1 Aproximácie Bézierovými kubikami

V našej práci používame krivky, ktoré máme reprezentované ako postupnosť Bézierových kubík. Preto si teraz uvedieme postupy, ako pomocou Bézierových kubík aproximovať kružnicu a elipsu.

3.1.1 Aproximácia kružnice Bézierovými kubikami

V [Sed03a] je presná reprezentácia časti kružnice pomocou racionálnej Bézierovej kubiky. My si ukážeme spôsob, ako aproximovať časť kružnice pomocou regulárnej Bézierovej kubiky. Celú kružnicu poskladáme z viacerých Bézierových kubík.

Je daná kružnica k so stredom S a polomerom r ležiaca v rovine π, ktorej normálový vektor je $\vec{n}$. Zoberme si dva body P_1 a P_2 ležiace na kružnici k také, že uhol $\alpha = \angle P_1 S P_2 \leq \frac{\pi}{2}$. Časť kružnice medzi bodmi P_1 a P_2 aproximujeme pomocou Bézierovej kubiky $B(t)$.

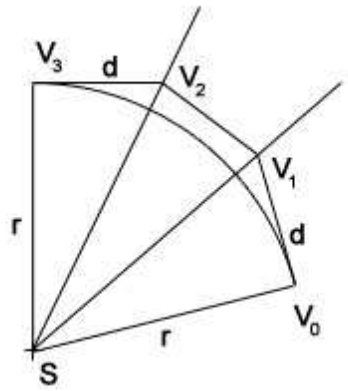
Označme V_0, V_1, V_2 a V_3 riadiace vrcholy Bézierovej kubiky $B(t)$. Vrchol $V_0 = P_1$, vrchol $V_3 = P_2$. Úsečky V_0V_1 a V_2V_3 sú dotyčnice kubiky $B(t)$ (vlastnosť 3 v stati 1.3.2). Na určenie presnej polohy vrcholov V_1 a V_2 potrebujeme zistiť smerové vektory úsečiek V_0V_1 a V_2V_3 a vzdialenosti $|V_0V_1|$ a $|V_2V_3|$.

Keďže úsečka V_0V_1 je dotyčnica kubiky $B(t)$ a kubika $B(t)$ aproximuje kružnicu k, úsečka V_0V_1 je aj dotyčnica kružnice k v bode $V_0 = P_1$. Jej smerový vektor $\vec{v}_1 = \vec{n} \times \vec{u}_1$, kde $\vec{u}_1 = V_0 - S$. Analogicky smerový vektor dotyčnice V_2V_3 je vektor $\vec{v}_2 = \vec{n} \times \vec{u}_2$, kde $\vec{u}_2 = V_3 - S$.

Vrcholy V_1 a V_2 vyrátame zo vzťahu:

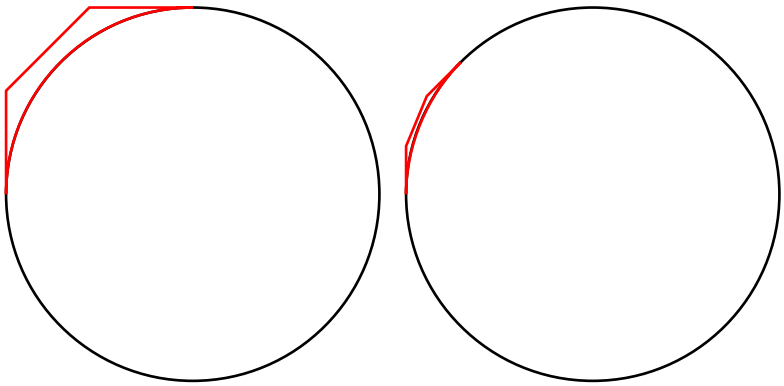
$$\begin{aligned} V_1 &= V_0 + d \cdot \frac{\vec{v}_1}{|\vec{v}_1|} \\ V_2 &= V_3 - d \cdot \frac{\vec{v}_2}{|\vec{v}_2|} \end{aligned}$$

kde $d = r \cdot \tan \frac{\alpha}{3}$ (obr. 3.1).



Obrázok 3.1: Zostrojenie Bézierovej kubiky $B(t)$, aproximujúcej kružnicový oblúk so stredom S a polomerom r. Vrcholy V_0, V_1, V_2, V_3 sú riadiace vrcholy Bézierovej kubiky $B(t)$.

Tento vzťah je založený na tom, kružnicový oblúk je symetrický. Keďže aproximujeme Bézierovou kubikou, rozdelíme oblúka na tri rovnaké časti. Na obrázku 3.2 vidieť príklad použitia uvedenej aproximácie.



Obrázok 3.2: Aproximácia kružnicového oblúka pomocou Bézierovej kubiky.

Teraz, keď vieme aproximovať časť kružnice Bézierovými kubikami, aproximácia celej kružnice k je jednoduchá. Stačí ju rozdeliť na kružnicové oblúky, ktorých uhly sú menšie alebo rovné ako $\frac{\pi}{2}$ a tieto aproximovať Bézierovými kubikami. Kubiky spolu tvoria aproximáciu celej kružnice k.

3.1.2 Aproximácia elipsy Bézierovými kubikami

Je daná elipsa e s polomermi r_1 a r_2 a stredom S. Najprv zostrojíme krivku K, ktorá je zložená s Bézierových kubík a aproximuje kružnicu k so stredom v bode S a polomerom r_1. Elipsa je škáľovaná kružnica. Elipsu e dostaneme z kružnice k škáľovaním faktorom $\frac{r_2}{r_1}$ so stredom škáľovania v bode S v smere vektora osi elipsy, ktorej zodpovedá polomer r_2.

Keďže škáľovanie je affína transformácia a Bézierova kubika je invariantná vzhľadom na affíne transformácie (viď stať 1.3.2), aproximáciu K_e elipsy e dostaneme použitím toho istého škáľovania krivky K aké sme použili na transformáciu kružnice k na elipsu e. T.j. nech $e = s(k)$, potom $K_e = s(K)$.

3.2 Interpolácia bodov Bézierovými kubikami

Pri premietaní kriviek (stať 3.3.2) budeme potrebovať skonštruovať krivku, ktorá interpoluje dané body $P_0, P_1, \ldots, P_n$, pričom poznáme dotyčnice v týchto bodoch. V [Cha99] je popísaných niekoľko takýchto interpolácií. My sme si vybrali nasledovnú.

Sú dané vrcholy $P_0, P_1, \ldots, P_n \in \mathbb{R}^3$ a normalizované smerové vektory v_0, v_1, ..., v_n dotyčníc v týchto bodoch. Vrcholy $P_0, P_1, \ldots, P_n$ budeme interpolovať C^1 spojitou po častiach kubickou krivkou. T.j. medzi vrcholmi $P_0, P_1, \ldots, P_n$ budeme vytvárať Bézierove kubiky $B_0(t)$, $B_1(t)$, ..., $B_{n-1}(t)$, pre ktoré platí:

$$\begin{aligned} &B_0(0) = P_0 \\ &B_{n-1}(1) = P_n \\ &B_{i-1}(1) = B_i(0) = P_i, \quad i = 1, 2, \ldots, n-1 \end{aligned}$$

Krajné riadiace vrcholy kubík $B_0(t), B_1(t), \ldots, B_{n-1}(t)$ teda už máme. Pomocou vektorov $v_0, v_1, \ldots, v_n$ dorátame ostatné riadiace vrcholy. Riadiace vrcholy V_i^0, V_i^1, V_i^2 a V_i^3 Bézierovej kubiky $B_i(t)$ dostaneme takto:

$$\begin{aligned} V_i^0 &= P_i \\ V_i^0 &= P_i + 0.4|P_iP_{i+1}| \cdot v_i \\ V_i^0 &= P_i - 0.4|P_iP_{i+1}| \cdot v_{i+1} \\ V_i^3 &= P_{i+1} \end{aligned}$$

Ako je spomínané v [Cha99], toto riešenie je odôvodnené len peknými výsledkami, ktoré dostávame.

3.3 Projekcia geometrických objektov

Projekcia nie je affina transformácia. Z toho vyplýva, že nemusí platiť nasledujúci vzťah:

$$f(A + t(B - A)) = f(A) + t(f(B) - f(A))$$

$A, B \in \mathbb{R}^3$; $t \in \mathbb{R}$; f je projekcia z $\mathbb{R}^3$do $\mathbb{R}^2$. Tu nastáva problém s premietaním kriviek a plôch, ktoré sú definované parametricky pomocou riadiacich vrcholov. Ak zostrojíme pôvodnú krivku na premietnutých riadiacich vrcholoch, dostaneme spravidla inú krivku ako je priemet pôvodnej krivky.

$$f(K(V_1, V_2, \ldots, V_n)) \neq K(f(V_1), f(V_2), \ldots, f(V_n))$$

kde $V_1, V_2, \ldots, V_n \in \mathbb{R}^3$ sú riadiace vrcholy krivky K, f je projekcia z $\mathbb{R}^3$ do $\mathbb{R}^2$.

My budeme premietať Bézierove kubiky (stať 1.3). Každý objekt popisovaný v ďalšom texte si reprezentujeme zjednotenie plôch, ktoré tvoria Coonsove záplaty. Coonsove záplaty sú plochy, ktorých hranice tvoria nejaké krivky, v našom prípade sú to Bézierove kubiky. Viac o Coonsových plochách môžeme nájsť v stati 1.4.

Coonsove plochy sme si vybrali preto, lebo v PostScripte robíme tieňovanie hladkých objektov pomocou Coonsových plôch (stať 1.1.4).

Obrazom Bézierovej krivky je takzvaná racionálna Bézierova krivka. Od obyčajnej Bézierovej krivky sa odlišuje tým, že pre každý riadiaci vrchol V_i má definovanú ešte váhu $w_i > 0$ vrcholu.

V PostScripte je však Coonsova plocha definovaná pomocou obyčajných Bézierových kubík, nie pomocou racionálnych. Preto budeme musieť použiť pre zobrazenie Bézierových kubík iný spôsob, ktorý bude zobrazenú krivku aproximovať.

3.3.1 Projekcia úsečky

Nech E je trojrozmerný euklidovský priestor nad poľom reálnych čísel $\mathbb{R}$. $A, B \in E$. f je projekcia z E do roviny $\pi \in \mathbb{R}^2$. Potom $f(AB) = f(A)f(B)$.

Toto je dobrá vlastnosť projekcie. Úsečky sa zobrazia na úsečky. Problém však je, že projekcia nezachováva pomery vzdialeností. Nech $A, B, C \in E$. Nech $a = \frac{|AB|}{|BC|}$. Potom nemusí platiť $a = \frac{|f(AB)|}{|f(BC)|}$.

3.3.2 Projekcia krivky

Projekcia krivky je veľmi dôležitá. Využijeme ju pri premietaní kontúr hladkých objektov.

Nech K je ľubovolná krivka definovaná parametricky z $D \subseteq \mathbb{R}$ do $\mathbb{R}^3$. Krivku K budeme premietať tak, že zobrazíme určitý počet jej bodov a tie potom interpolujeme Bézierovými kubikami[1]. Okrem bodov zobrazíme aj dotykové vektory v týchto bodoch. Tie nám pomôžu pri konštrukcii kubiky.

[1]Bézierovou kubikou preto, lebo v PostScripte budeme používať tieňovanie pomocou Coonsových záplat, ktorých hranice sú Bézierove kubiky

Teraz trochu presnejšie. Nech body $V_0, V_1, \ldots, V_n \in \mathbb{R}^3$ sú body patriace krivke K. Body V_0, V_n sú krajné body krivky. Teda $V_i = K(t_i)$, $t_i \in D$ pre $i = 0, 1, \ldots, n$ a nech $t_i < t_{i+1}$, $i = 0, 1, \ldots, n-1$. Ďalej nech $\vec{v}_0, \vec{v}_1, \ldots, \vec{v}_n$ sú vektory dotyčníc v bodoch $V_0, V_1, \ldots, V_n$. Nech f je projekcia. Bézierovu kubiku zostrojíme medzi každými dvoma bodmi $f(V_i), f(V_{i+1})$.

Potrebujeme 4 riadiace vrcholy pre každú kubiku B_i. Nazvime ich $P_{i,0}$, $P_{i,1}$, $P_{i,2}$, $P_{i,3}$. Krajné body už máme: $P_{i,0} = f(V_i)$, $P_{i,3} = f(V_{i+1})$ Body $P_{i,1}$ a $P_{i,2}$ dopočítame pomocou obrazov dotykových vektorov krivky K.

Dotykové vektory $\vec{v}_0, \vec{v}_1, \ldots, \vec{v}_n$ musíme premietať ako priamky prechádzajúce bodmi $V_0, V_1, \ldots, V_n$. Ak totiž premietame dve rovnobežné priamky, ich priemety v perspektívnej projekcii nemusia byť rovnobežné. Nech teda $\vec{u}_i$ je normalizovaný smerový vektor priamky $f(V_i + s\vec{v}_i)$. Riadiace body Bézierovej kubiky B_i sú nasledovné

$$\begin{array}{lcl} P_{i,0} & = & f(V_i) \\ P_{i,1} & = & f(V_i) + d_{i,1}.\vec{u}_i \\ P_{i,2} & = & f(V_{i+1}) - d_{i,2}.\vec{u}_{i+1} \\ P_{i,3} & = & f(V_{i+1}) \end{array} \tag{3.1}$$

$d_{i,1}$ a $d_{i,2}$ sú premenné závislé od toho, aký spôsob interpolácie si vyberieme (stať 3.2).

3.3.3 Projekcia kruhu

V rovnobežnom premietaní sa kružnica premietne na elipsu. Preto stačí premietnuť len stred a dva polomery. Avšak u perspektívneho premietania to nemusí platiť.

Kružnica je krivka. Preto ju premietneme ako krivku, tak ako je to popísané v predchádzajúcej časti.

Vnútro premietnutej kružnice vyplníme trojuholníkmi, ktoré spojíme s krivkou reprezentujúcou hranicu Coonsovými záplatami. Na obrázku 3.3 je vidieť premietnutý kruh.

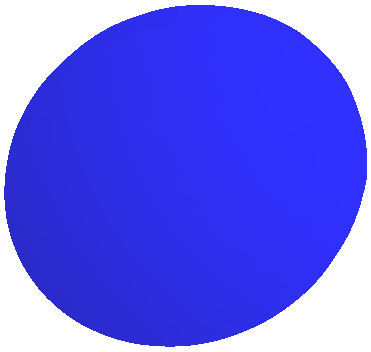
3.3.4 Projekcia gule

Pri premietaní gule budeme podobne ako u iných hladkých telies hľadať kontúry. Ako vyzerá kontúra gule? Ukážeme si, že je to kružnica.

Na obrázku 3.4 vidíme guľu so stredom S a polomerom r. Bod C je stred premietania (pozorovateľ). Body P_1, P_2 ležia na kontúre. Ako sme si už povedali, kontúra je krivka na ploche, ktorej body majú normálu vzhľadom na plochu kolmú na vektor pohľadu. Normálový vektor v bode P ležiacom na povrchu gule je vektor $\vec{n}$, ktorý je smerovým vektorom priamky SP.

Kontúra K gule je krivka ležiaca na povrchu gule.

$$K = \{P; |SP| = r \ \wedge \ \angle CPS = \frac{\pi}{2}\}$$



Obrázok 3.3: Premietnutý kruh

kde S je stred gule, r je polomer gule a C je stred premietania.

Vzdialenosť $|CS|$ je konštantná[2]. Vzdialenosť $|SP|$ je rovná r. To znamená, že všetky trojuholníky CPS, kde P je ľubovolný bod kontúry, sú zhodné. Preto kontúra gule bude kružnica so stredom v bode Y a polomerom $|PY|$, $Y \in CS$, $|\angle CYP| = \frac{\pi}{2}$.

Keďže vzdialenosť všetkých bodov kontúry od stredu premietania je rovnaká, tak normála roviny, v ktorej leží kontúra, je smerový vektor priamky YC.

Na obrázku 7.5 je znázornená premietnutá guľa.

3.3.5 Projekcia valca

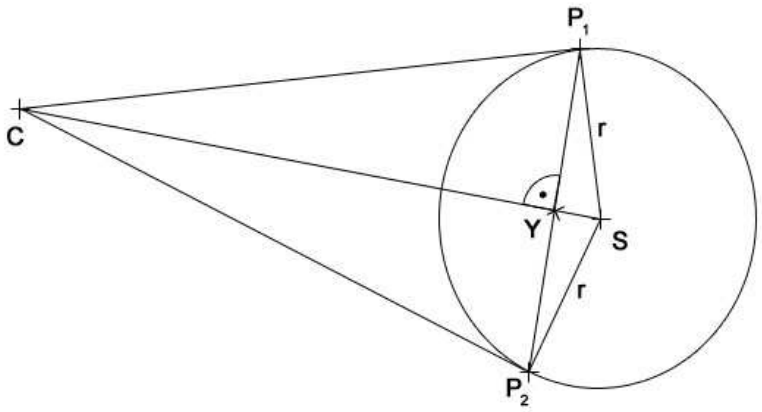
Premietanie valca sa dá rozdeliť na dve časti. Premietanie podstáv a premietanie plášťa. Podstavy sú kruhy a teda ich premietanie už poznáme (viď stať 3.3.3).

Ako premietneme plášť? Potrebujeme nájsť kontúry plášťa. Tie tvoria dve úsečky a dve časti kružnice[3]. Tieto dve úsečky a stred premietania tvoria dve dotykové roviny valca.

Dve časti kružnice sú časťami hraničných kriviek podstáv. Preto tvoria kontúry. Ukážeme si, prečo sú ďalšími kontúrami dve úsečky. Zoberme si ľubovolný bod P na povrchu valca, pre ktorý platí $\vec{v} \perp \vec{n}$, kde $\vec{v} = C - P$, C je stred premietania, $\vec{n}$ je normálový vektor bodu P vzhľadom na plášť valca. Zostrojme úsečku s ležiacu na povrchu plášťa, kolmú na podstavy, pre ktorú platí $P \in s$. Ľahko vidieť, že všetky body tejto úsečky majú rovnakú normálu ($\vec{n}$).

[2]Je konštantná, lebo nehýbeme s objektmi v scéne.

[3]Predpokladáme, že je plášť nie je zakrytý podstavou valca. V opačnom prípade premietame len podstavu a plášť nepremietame vôbec.



Obrázok 3.4: Kontúra gule so stredom S a polomerom r. Y - stred kružnice, ktorá tvorí kontúru. P_1, P_2 - body ležiace na kontúre.

Úsečka s spolu so stredom premietania C tvoria rovinu π. Túto rovinu definujú bod P a vektory $\vec{v}$ ($\vec{PC}$) a $\vec{u}$, ktorý je smerovým vektorom úsečky s. Vieme, že platí $\vec{n} \perp \vec{v}$ a $\vec{n} \perp \vec{u}$[4] a teda $\vec{n}$ je kolmý na celú rovinu π.

Ďalej každá úsečka CQ, kde $Q \in s$ leží v rovine π. Preto platí $\vec{n} \perp CQ$. Z toho vyplýva, že bod Q leží na kontúre plášťa a teda celá úsečka s tvorí kontúru (obr. 3.6).

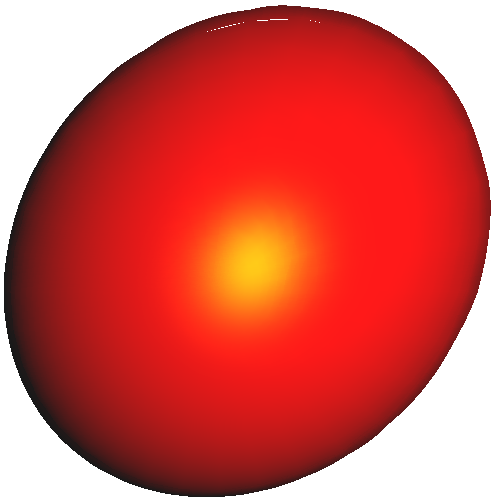
Ukázali sme si, že keď poznáme jeden bod úsečky tvoriacej kontúru, ľahko nájdeme celú kontúru. Teraz si predvedieme postup, ako nájsť dva body, ktoré ležia na rôznych kontúrach. Tak nájdeme aj dve rôzne kontúry.

Posuňme valec v smere normálového vektora podstavy tak, aby stred premietania ležal v rovine podstavy. Dotykové roviny valca sa nezmenia. Podstava valca je kruh so stredom S a polomerom r. Body P_1 a P_2 sú spoločné body dotykových rovín a podstavy. Nájdeme ich analogicky, ako pri hľadaní kontúry gule (obr. 3.4).

Keď máme body P_1 a P_2, posunieme valec aj s týmito bodmi na pôvodné miesto. Ďalej zostrojíme body Q_1 a Q_2 ako priesečník priamok $P_1 + t_1.\vec{n}$ a $P_2 + t_2.\vec{n}$ s druhou podstavou valca, kde vektor $\vec{n}$ je normálový vektor podstáv valca, t_1, $t_2 \in \mathbb{R}$.

Úsečky P_1Q_1 a P_2Q_2 sú kontúry valca. Ešte nám chýbajú dve kontúry, ktoré sú časťou kružníc ohraničujúcich podstavy. Zoberme si prvú podstavu, na nej body P_1, P_2 a stred podstavy S. Uhol $|\angle P_1SP_2| < \pi$ alebo $|\angle P_1SP_2| > \pi$. Nemôže sa stať, že by bol rovný π. To by bolo možné len u rovnobežného

[4] $\vec{u}$ je smerovým vektorom úsečky s ležiacej na povrchu plášťa valca. Vektor $\vec{n}$ je kolmý na povrch valca v bode P. Preto $\vec{n} \perp \vec{u}$.



Obrázok 3.5: Premietnutá guľa

premietania. Platí, že $\lim_{|SC|\to\infty} |\angle P_1SP_2| = \pi$, kde C je stred premietania.

Ktorá časť kružnice teda bude kontúra? Tá, pre ktorú platí

$$|\angle X_1SX_2| < \pi \quad \text{pre } X_1 = P_1 \wedge X_2 = P_2 \vee X_1 = P_2 \wedge X_2 = P_1$$

To isté platí aj pre druhú postavu a body Q_1 a Q_2.

Na obrázku 3.7 je premietnutý valec.

3.3.6 Projekcia kužeľa

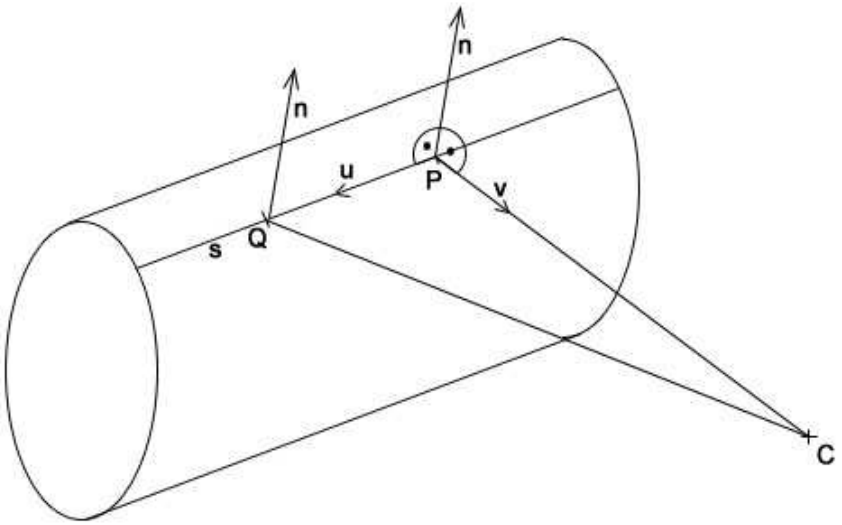
Podobne ako u valca, aj premietanie kužeľa možno rozdeliť na premietanie plášťa a premietanie podstavy. Podstava je kruh, jej premietanie už poznáme (viď stať 3.3.3).

Použijeme dva spôsoby premietania plášťa podľa toho, či sa priemet vrcholu kužeľa nachádza v priemete podstavy (obr. 3.8 (b)) alebo nie[5] (obr. 3.8 (a)).

Ako rozhodneme o tom, kde sa nachádza priemet vrcholu kužeľa? Na obrázku 3.9 je kolmý priemet kužeľa do roviny CVS. V je vrchol kužeľa, S je stred podstavy, C je stred premietania (pozorovateľ). Ak sa pozorovateľ nachádza v šedej oblasti (C_2), priemet vrcholu kužeľa leží v priemete podstavy. Ak sa pozorovateľ nachádza mimo šedej oblasti (C_1) [6], priemet vrcholu kužeľa neleží v priemete podstavy.

[5] V oboch prípadoch predpokladáme, že je plášť aspoň z časti viditeľný, t.j. že nie je zakrytý svojou podstavou.

[6] Ak sa stred premietania nachádza vo vnútri kužeľa, tak kužeľ nezobrazujeme



Obrázok 3.6: Nájdenie kontúry valca. C - stred premietania; P, Q - body kontúry, n - normálový vektor v bodoch P, Q; s -kontúra plášťa; u - smerový vektor úsečky s; v - vektor z bodu P do bodu C. Bod P a vektory u a v určujú rovinu π.

Matematicky to vyjadríme nasledovne. Nech $\vec{v} = S - V$, $\vec{u} = C - V$ a $\vec{w} = P - V$. Ďalej nech $\cos\alpha = \frac{\vec{v}.\vec{u}}{|\vec{v}||\vec{u}|}$ a $\cos\beta = \frac{\vec{v}.\vec{w}}{|\vec{v}||\vec{w}|}$. Stred premietania C sa nachádza v šedej oblasti a teda priemet vrcholu V leží v priemete podstavy kužeľa práve vtedy, keď $|\cos\alpha| > |\cos\beta|$.

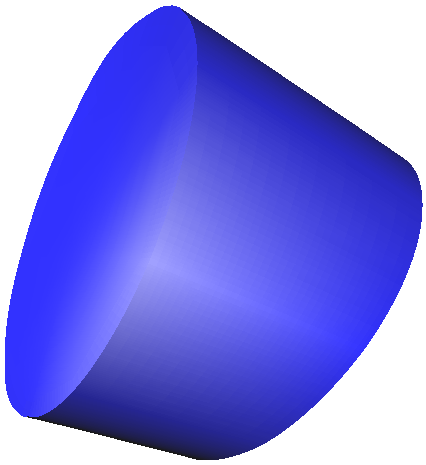
Ak sa priemet vrcholu kužeľa nachádza v priemete jeho podstavy, potom plášť zobrazujeme ako kruh s tým rozdielom, že jeho vnútro vypĺňame celé Coonsovými záplatami a použijeme normály bodov plášťa.

Kontúry plášťa

Teraz predpokladajme, že priemet vrcholu kužeľa leží mimo priemet jeho podstavy. Potom kontúry plášťa tvoria časť kružnice a dve úsečky, ktoré sa pretínajú vo vrchole valca. Úsečky majú s časťou kružnice dva spoločné body. Najprv si dokážeme, že dve úsečky a časť kružnice sú naozaj kontúry.

Kontúra podstavy je kružnica. Tá je zároveň aj kontúrou plášťa. Kružnica môže byť z časti zakrytá plášťom, preto považujeme za kontúru len časť kružnice.

Teraz predpokladajme, že poznáme bod kontúry plášťa (P), ktorý neleží na spomínanej časti kružnice. Cez tento bod vedie z vrcholu kužeľa (V) úsečka do nejakého bodu X ležiaceho na podstave. Všetky body tejto úsečky majú rovnakú normálu $\vec{n}$. Bod P spolu s normálou $\vec{n}$ určujú dotykovú rovinu



Obrázok 3.7: Premietnutý valec

(π) plášťa. Keďže bod P leží na kontúre, tak vektor $\vec{v} = C - P$, kde C je stred premietania, je kolmý na vektor $\vec{n}$. Z toho vyplýva, že $C \in \pi$.

Úsečka XV leží na povrchu plášťa kužeľa, $P \in XV$, preto je vektor $\vec{n}$ kolmý na úsečku XV a teda aj táto úsečka leží v rovine π. Keďže aj stred premietania C leží v rovine π, tak pre všetky body $Y \in XV$ platí, že $CY \perp \vec{n}$. Ako sme si povedali, všetky body ležiace na úsečke XV majú rovnakú normálu ($\vec{n}$). Z toho vyplýva, že všetky body úsečky XV ležia na kontúre plášťa (obr. 3.10).

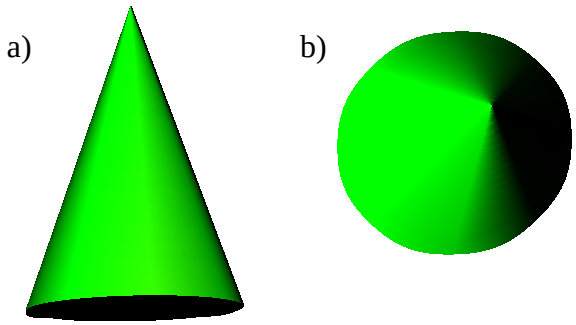
Už vieme, že ak nájdeme jeden bod kontúry (P) ležiaci na plášti, tak všetky body ležiace na prieniku priamky VP a plášťa tvoria kontúru plášťa. Teraz si ukážeme ako nájsť bod P.

Bod kontúry plášťa

Na obrázku 3.11 vidíme kolmý priemet kužeľa do roviny, v ktorej leží podstava kužeľa. Na obrázku 3.12 je kolmý priemet kužeľa do roviny CVS, C je stred premietania, V je vrchol kužeľa, S je stred podstavy.

Bod P, ktorý leží na kontúre plášťa nájdeme tak, že najprv nájdeme bod Y ležiaci na priamke C_1S a ten potom posunieme v smere kolmom na rovinu CVS tak, aby ležal na kružnici ohraničujúcej podstavu.

Keďže body kontúry tvoria úsečky, tak trojuholníky C_1SP a CS_1P_1 sú podobné. Aj trojuholníky YPS a $Y_1P_1S_1$ sú podobné. Preto budeme hľadať bod kontúry v rovine podstavy, lebo tu poznáme vzdialenosť $|SP|$, je to polomer podstavy kužeľa.



Obrázok 3.8: Premietnutý kužeľ.

Poďme si vyjadriť vzdialenosť $|YS|$. Označme $\alpha = \angle YSP$. Potom

$$|YS| = r\cos\alpha, \quad \text{kde } r = |SP|$$

Ďalej $\cos\alpha = \frac{r}{|C_1S|}$, a teda

$$|YS| = \frac{r^2}{|C_1S|} \tag{3.2}$$

Označme $\beta = \angle CVS$ (obr. 3.12). Body C_1, S z obrázku 3.12 sú totožné s bodmi C_1, S z obrázku 3.11. Ideme si vyjadriť vzdialenosť $|C_1S|$ pomocou $h = |SV|$ (výška kužeľa) a uhlu β.

Vieme, že $|C_1S| = h\frac{\sin\beta}{\cos\beta}$. Po doplnení tejto rovnosti do vzťahu (3.2) dostaneme

$$|YS| = \frac{r^2.\cos\beta}{h.\sin\beta} \tag{3.3}$$

$\sin\beta$ nikdy nebude rovné nule, lebo predpokladáme takú polohu stredu premietania, že vrchol V sa nezobrazí do obrazu podstavy kužeľa. Keby $\sin\beta = 0$, tak by polomer r podstavy kužeľa musel byť rovný nule.

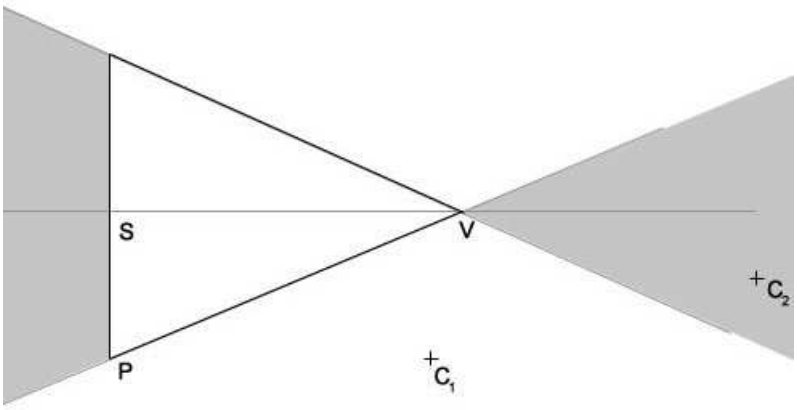
Môže sa stať, že $|YS| < 0$[7]. V tom prípade bod Y bude stále ležať na priamke C_1S, ale už nie na úsečke C_1S.

Zostrojíme priamku p kolmú na rovinu CVS a prechádzajúcu bodom Y. Potom body $P \in p$, $Q \in p$, pre ktoré platí $|SP| = r$, $|SQ| = r$, sú body ležiace na kontúrach plášťa kužeľa a úsečky PV a QV sú tieto kontúry.

3.3.7 Premietanie parametrických plôch

Parametrické plochy sú funkcie dvoch premenných. Ich výstupom je bod. Pre nás sú podstatné parametrické plochy $X(u,v) : \langle 0,1\rangle^2 \to \mathbb{R}^3$, ktoré majú sieť riadiacich vrcholov $V_{i,j}$. Teda $X(u,v)$ je definovaná takto:

[7] Keď $\cos\beta < 0$, t.j. $\frac{\pi}{2} < \beta < \frac{3\pi}{2}$



Obrázok 3.9: Kolmý priemet kužeľa do roviny VSC_1 resp. VSC_2. V - vrchol kužeľa; S - stred podstavy kužeľa; C_1, C_2 - stredy premietania; P - bod ležiaci na prieniku plášťa a podstavy kužeľa, bod P leží v rovine VSC_1 resp. VSC_2.

$$X(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} V_{i,j} \cdot B_i^m(u) \cdot B_j^n(v) \tag{3.4}$$

Príkladom parametrických plôch typu (3.4) sú B-Splajnové plochy a Bézierove plochy ([Gre02]).

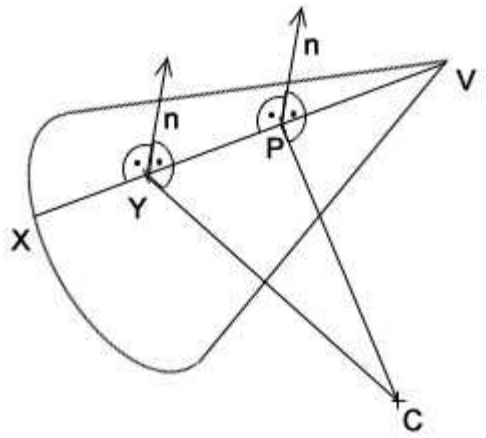
Parametrické plochy budeme zobrazovať tak, že nájdeme kontúry plochy pomocou riadiacich vrcholov a derivácií plochy. Vyriešime viditeľnosť kontúr, zobrazíme kontúry a hraničné krivky a oblasť medzi nimi vyplníme Coonsovým mešom ako u ostatných objektoch.

Hľadanie kontúr parametrických plôch

Hľadanie kontúr je založené na tom, že vyrátame určitý počet bodov plochy rozmiestnených rovnomerne po ploche. Na základe normál v týchto bodoch zistíme, medzi ktorými bodmi vedie kontúra. Bod kontúry potom získame lineárnou interpoláciou.

Koľko bodov budeme zobrazovať? Z vlastností parametrických plôch vieme, že sa plocha nemení viackrát ako jej riadiaca sieť (podobne ako u Bézierovej krivky, stať 1.3.2 vlastnosť 5). Preto nám stačí zobraziť toľko bodov, koľko je riadiacich vrcholov.

Po zobrazení týchto bodov dostávame sieť bodov $P_{i,j} = X(u_i, v_j)$, ktoré nám istým spôsobom aproximujú našu plochu. Teraz vyrátame v týchto bodoch normály $\vec{n}_{i,j}$ vzhľadom na plochu. Pre každý bod $P_{i,j}$ zistíme uhol medzi vektorom $\vec{n}_{i,j}$ a vektorom pohľadu $\vec{w}_{i,j} = C - P_{i,j}$, kde C je stred



Obrázok 3.10: Kontúra kužeľa. C - stred premietania; V - vrchol kužeľa; XV - kontúra plášťa; P, $Y \in XV$; C, P, Y, X, $V \in \pi$; $\vec{n}$ - normála bodov úsečky XV vzhľadom na plášť kužeľa.

premietania. Ak je uhol menší ako $\frac{\pi}{2}$, bod $P_{i,j}$ je privrátený k pozorovateľovi, inak je bod $P_{i,j}$ od pozorovateľa.

Kontúra vedie medzi privrátenými a odvrátenými bodmi $P_{i,j}$.

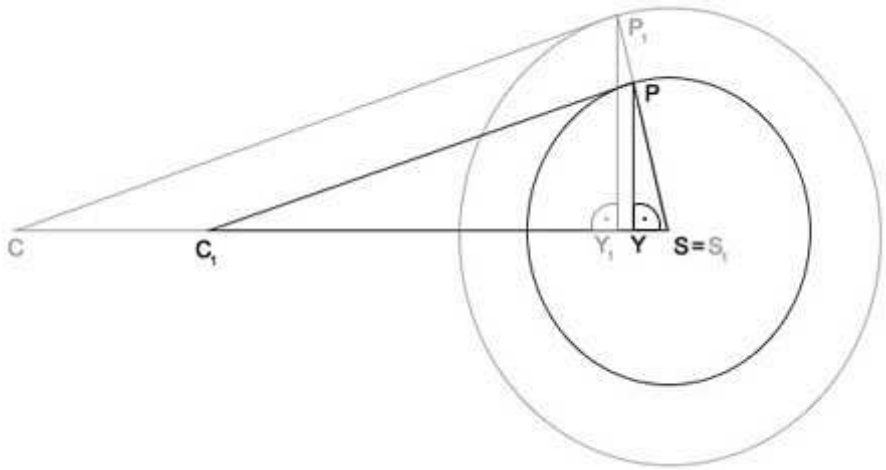
Bod $Q = X(u_Q, v_Q)$ kontúry nájdeme nasledovne. Nech bod $P_{i_1,j_1} = X(u_{i_1}, v_{j_1})$ je odvrátený od pozorovateľa a bod $P_{i_2,j_2} = X(u_{i_2}, v_{j_2})$ je privrátený k pozorovateľovi. Ďalej nech

$$\begin{aligned} d_{i_1,j_1} &= -\frac{\vec{n}_{i_1,j_1} \cdot \vec{w}_{i_1,j_1}}{|\vec{n}_{i_1,j_1}||\vec{w}_{i_1,j_1}|} \\ d_{i_2,j_2} &= \frac{\vec{n}_{i_2,j_2} \cdot \vec{w}_{i_2,j_2}}{|\vec{n}_{i_2,j_2}||\vec{w}_{i_2,j_2}|} \end{aligned}$$

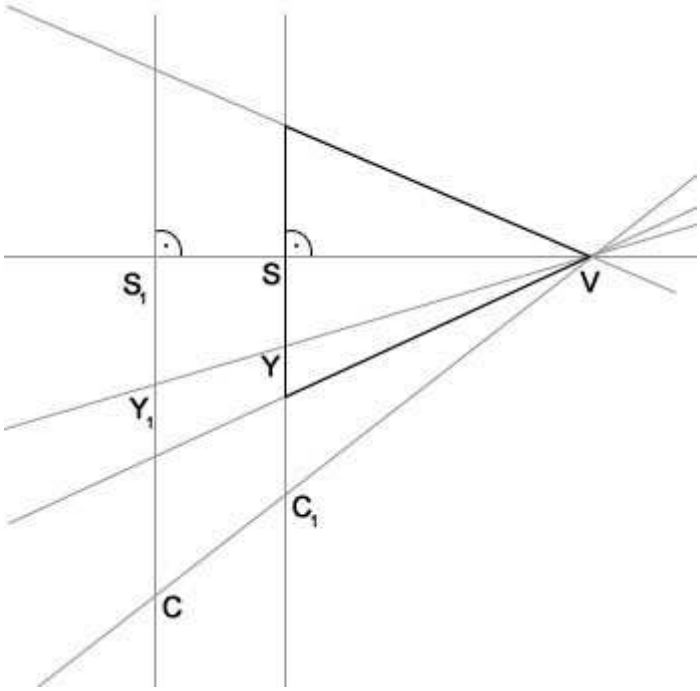
potom

$$\begin{aligned} Q &= X(u_Q, v_Q) \\ u_Q &= \frac{d_{i_1,j_1} \cdot |\vec{n}_{i_1,j_1}| \cdot u_{i_1} + d_{i_2,j_2} \cdot |\vec{n}_{i_2,j_2}| \cdot u_{i_2}}{d_{i_1,j_1} \cdot |\vec{n}_{i_1,j_1}| + d_{i_2,j_2} \cdot |\vec{n}_{i_2,j_2}|} \\ v_Q &= \frac{d_{i_1,j_1} \cdot |\vec{n}_{i_1,j_1}| \cdot v_{i_1} + d_{i_2,j_2} \cdot |\vec{n}_{i_2,j_2}| \cdot v_{i_2}}{d_{i_1,j_1} \cdot |\vec{n}_{i_1,j_1}| + d_{i_2,j_2} \cdot |\vec{n}_{i_2,j_2}|} \end{aligned}$$

Podrobnejší postup ako nájsť kontúry NURBSovej plochy a ako riešiť ich viditeľnosť môžeme nájsť v [Goo98] a v [GC90].



Obrázok 3.11: Kolmý priemet kužeľa do roviny podstavy kužeľa. C - stred premietania; C_1 - stred premietania posunutý v smere vektora CV (V je vrchol kužeľa), C_1 leží v rovine podstavy kužeľa; P, P_1 - body ležiace na priamke obsahujúcej kontúru plášťa; S - stred podstavy kužeľa.



Obrázok 3.12: Kolmý priemet kužeľa do roviny CSV. C - stred premietania; V - vrchol kužeľa; S - stred podstavy kužeľa; C_1 - stred premietania posunutý v smere vektora CV (V je vrchol kužeľa), C_1 leží v rovine podstavy kužeľa;.

Kapitola 4

Viditeľnosť mešov

4.1 Viditeľné a neviditeľné steny

Steny mešu rozdelíme do dvoch skupín. Na viditeľné a neviditeľné. Zobrazovať budeme len viditeľné. Stenu mešu chápeme ako polygón, ktorý leží celý v jednej rovine.

Stenu nazveme neviditeľnou, ak je odvrátená od pozorovateľa[1] alebo ak každú úsečku idúcu z bodu pozorovateľa do ľubovolného bodu steny pretína iný predmet scény. Odvrátená od pozorovateľa znamená, že uhol medzi normálovým vektorom steny a vektorom z bodu steny do bodu pozorovateľa nie je z intervalu $(-\frac{\pi}{2}, \frac{\pi}{2})$. Viditeľná stena je každá stena, ktorá nie je neviditeľná.

Najprv si ukážeme, ako odstránime odvrátené steny a úplne zakryté steny. Steny, ktoré nie sú zakryté úplne a navzájom sa prekrývajú, zoradíme podľa ich vzdialenosti od pozorovateľa.

4.1.1 Odstraňovanie neviditeľných stien

Odvrátené steny

Začneme odstraňovaním odvrátených stien, lebo ide o najjednoduchší prípad. Nech $\vec{n}$ je normálový vektor steny, nech $\vec{v}$ je vektor z niektorého bodu steny[2] do bodu pozorovateľa. Stena je odvrátená práve vtedy, keď platí:

$$\vec{n} \cdot \vec{v} \leq 0$$

Po normalizovaní vektorov $\vec{n}$ a $\vec{v}$ dostaneme kosínus uhla, ktorý zvierajú vektory $\vec{n}$ a $\vec{v}$. Platí

$$\vec{n} \cdot \vec{v} \leq 0 \iff \frac{\vec{n} \cdot \vec{v}}{|\vec{n}||\vec{v}|} \leq 0$$

[1] Niekedy sa môže stať, že chceme zobrazovať aj steny, ktoré sú odvrátené.
[2] Nezáleží na tom, ktorý bod steny si vyberieme

a teda uhol medzi vektormi $\vec{n}$ a $\vec{v}$ nie je z intervalu $(-\frac{\pi}{2}, \frac{\pi}{2})$ práve vtedy, keď $\vec{n} \cdot \vec{v} \leq 0$.

Odstránili sme všetky odvrátené steny. Teraz ideme hľadať zakryté steny.

Odstraňovanie zakrytých stien pomocou vrcholov

Nech p je polygón určujúci stenu a nech je definovaný bodmi $V_1, V_2, \ldots, V_n$. Pre každý s týchto bodov otestujeme, či nie je zakrytý inou stenou. Ak sú zakryté všetky vrcholy, stenu prehlásime za neviditeľnú.

Táto metóda ale v niektorých prípadoch nefunguje. Môže sa totiž stať, že aj keď má polygón zakryté všetky vrcholy $V_1, V_2, \ldots, V_n$, nejaká jeho časť môže byť vidieť. Algoritmus ale označí túto stenu za neviditeľnú.

Triangulácia polygónu

V ďalšom texte budeme pracovať s mešom, ktorého steny sú trojuholníky. Takéto meše sú jednoduchšie na spracovanie. Avšak na vstupe nemusíme mať vždy len trojuholníkový meš. Preto musíme steny mešu striagnulovať ([Eri00]).

Myšlienka algoritmu je založená na tom, že každý polygón obsahuje diagonálu. Diagonála je úsečka medzi dvoma vrcholmi polygónu, ktorá nie sú susedné. Diagonála leží celá vo vnútri polygónu. Algoritmus nájde pre daný polygón diagonálu. Vzniknú dva nové polygóny, pre ktoré opäť hľadáme diagonálu. Pokračujeme induktívne až pokiaľ z pôvodného polygónu nezostanú samé trojuholníky.

Prekrytie vrcholu stenou

Je daný vrchol P a stena mešu reprezentovaná ako trojuholník ABC.

Najprv zistíme pre vrchol P a $\triangle ABC$, či priemet bodu P leží v priemete $\triangle ABC$. Body $\triangle ABC$ sú usporiadané proti smeru chodu hodinových ručičiek. Funkcia $AREA(A, B, C)$ ([Eri00]) vypočíta dvojnásobok orientovaného obsahu $\triangle ABC$.

Funkce `AREA`(A, B, C)

$AREA(A, B, C)$:
return $(C.y - A.y)(B.x - A.x) - (C.x - A.x)(B.y - A.y)$

Ak $AREA(A, B, C) \geq 0$, tak body A, B, C sú usporiadané proti smeru chodu hodinových ručičiek.

Nech P_1 je priemet bodu P a A_1, B_1, C_1 sú priemety bodov A, B, C. Bod P_1 sa nachádza v $\triangle A_1B_1C_1$ ak platí:

$$\begin{gathered} AREA(P_1, A_1, B_1) \geq 0 \wedge \\ AREA(P_1, B_1, C_1) \geq 0 \wedge \\ AREA(P_1, C_1, A_1) \geq 0 \end{gathered} \tag{4.1}$$

Zistili sme, či priemet bodu P leží v priemete $\triangle ABC$. Ak áno, musíme zistiť, či bod P leží pred alebo za $\triangle ABC$.

Body A, B, C ležia v jednej rovine, označme ju π. Dosaďme bod P a stred premietania do rovnice roviny π. Ak majú výsledky opačné znamienka, bod P je zakrytý trojuholníkom ABC a teda ho označíme ako neviditeľný.

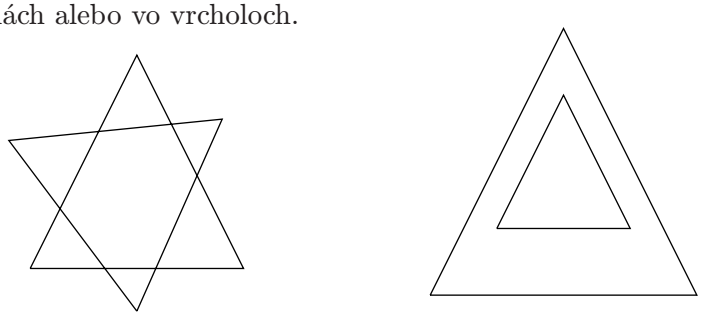
4.1.2 Zoraďovanie stien podľa vzdialenosti od pozorovateľa

Je viacero spôsobov ako riešiť vykresľovanie prekrytých polygónov. Je možné zakrytý polygón rozdeliť na časti, ktoré sú úplne viditeľné alebo vykresliť polygóny postupne od zadného k prednému[3]. My sme si vybrali druhý spôsob. Musíme preto riešiť problém zoradenia prekrytých polygónov podľa vzdialenosti od pozorovateľa. Najprv musíme zistiť, ktoré polygóny sa prekrývajú, potom vypočítať ich vzdialenosť od pozorovateľa a nakoniec ich zoradiť.

Hľadanie prekrytých stien

Pri hľadaní prekrytých stien použijeme dva prístupy. Vrcholový a hranový. Vrcholový prístup spočíva v tom, že zoberiem vrcholy jedného polygónu a testujem, či sa ich priemet nachádza v priemete druhého polygónu. Hranový spôsob testuje, či sa pretínajú premietnuté hrany polygónov.

Naše algoritmy testujú len trojuholníky. To je ale v poriadku, lebo meš si môžeme striangulovať a až potom použiť algoritmy na hľadanie prekrytých polygónov. U obidvoch algoritmov predpokladáme, že polygóny sa pretínajú len v hranách alebo vo vrcholoch.



a. Toto prekrytie nedetekuje vrchol. algoritmus

b. Toto prekrytie nedetekuje hran. algoritmus

Obrázok 4.1: Príklady prekrytia polygónov, ktoré jednotlivé algoritmy nedetekujú

[3] maliarov algoritmus

Hľadanie prekrytých stien - vrcholový prístup

Majme dva polygóny p_1 a p_2. Zoberieme každý vrchol V polygónu p_1 a zistíme, či sa jeho obraz v perspektívnom premietaní nachádza v obraze polygónu p_2. Ak áno, otestujeme, v ktorom polpriestore definovanom rovinou polygónu p_2 sa vrchol V nachádza. Ak v tom istom ako stred premietania, potom je polygón p_2 prekrytý polygónom p_1. Ak je V v opačnom polpriestore ako stred premietania, potom p_1 je prekrytý polygónom p_2 (viď stať 4.1.1).

Keďže sme predpokladali, že sa polygóny pretínajú len v hranách alebo vrcholoch, tak po nájdení prvého prekrytia sa už ostatné vrcholy polygónu netestujú.

Vrcholový algoritmus ale nestačí na nájdenie všetkých prekrytí. Na obrázku 4.1.a. je prípad, ktorý tento algoritmus nedetekuje.

Prienik dvoch úsečiek v rovine

Ešte než si uvedieme hranový algoritmus na detekovanie prekrytých stien, ukážeme si spôsob hľadania prieniku dvoch úsečiek v rovine. Majme úsečky AB a CD s normálovými vektormi $\vec{n}_1$ a $\vec{n}_2$. Všeobecné rovnice týchto úsečiek sú:

$$\begin{aligned} a_1x + b_1y + c_1 = 0 \quad &\text{kde } \vec{n}_1 = (a_1, b_1) \\ a_2x + b_2y + c_2 = 0 \quad &\text{kde } \vec{n}_2 = (a_2, b_2) \end{aligned}$$

Bod P leží na oboch priamkach AB aj CD[4]. Z toho vyplýva, že vyhovuje obom rovniciam. Z nich vyrátame súradnice bodu P. Teraz bod P vyjadríme pomocou bodu A a vektora $\vec{v} = B - A$: $P = A + t\vec{v}$. Ak $t \in \langle 0, 1\rangle$, tak P leží na úsečke AB, inak leží mimo nej. Analogicky otestujeme, či P leží na úsečke CD.

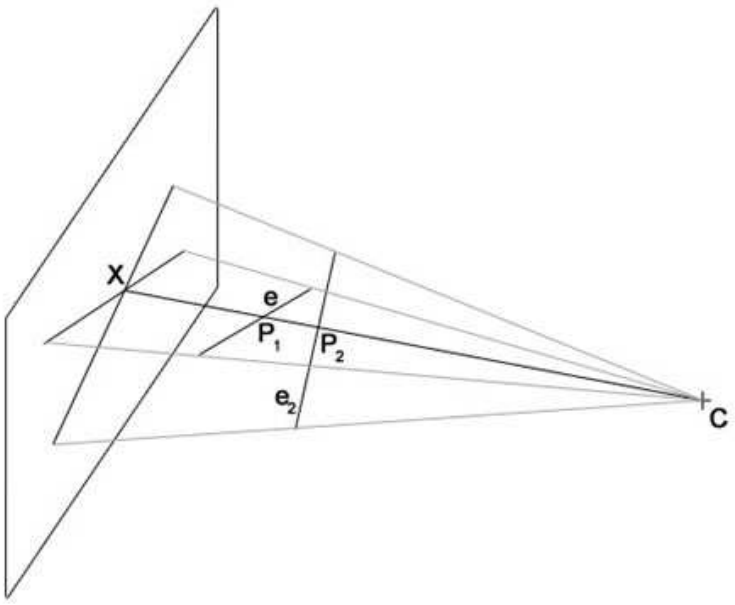
Úsečky AB a CD sa pretínajú, ak P leží na úsečke AB aj na úsečke CD a ich prienikom je bod P.

Hľadanie prekrytých stien - hranový prístup

Opäť majme dva polygóny p_1 a p_2. Tento raz zoberme každú hranu e polygónu p_1 a testujme ju s hranami polygónu p_2. Nech sa priemet hrany e pretne s priemetom niektorej hrany e_2 polygónu p_2 (viď stať 4.1.2). Nazvime tento prienik X. Úsečka XC, kde C je stred premietania, pretína hrany e aj e_2. Nech $P_1 = e \cap XC$ a $P_2 = e_2 \cap XC$. Aby sme zistili, ktorá hrana prekrýva ktorú, musíme porovnať vzdialenosti $|P_1C|$ a $|P_2C|$ (obr. 4.2).

Nech d_1 je vzdialenosť stredu premietania C od roviny π_1 polygónu p_1 a d_2 je vzdialenosť stredu premietania C od roviny π_2 polygónu p_2. Tieto vzdialenosti vyrátame z rovníc spomínaných rovín. Nech $\vec{n}_1$ je normálový

[4] okrem prípadu, kedy sú priamky AB, CD rovnobežné. A to je vtedy, ak ich normálové vektory sú lineárne závislé.



Obrázok 4.2: Prienik X priemetu hrán e a e_2. C - stred premietania; Body P_1 a P_2 sú prieniky hrán e a e_2 s úsečkou XC.

vektor roviny π_1, $\vec{n}_2$ normálový vektor roviny π_2 a nech $\vec{v}$ je smerový vektor úsečky XC (taktiež úsečiek P_1C a P_2C) (obr. 4.3). Potom

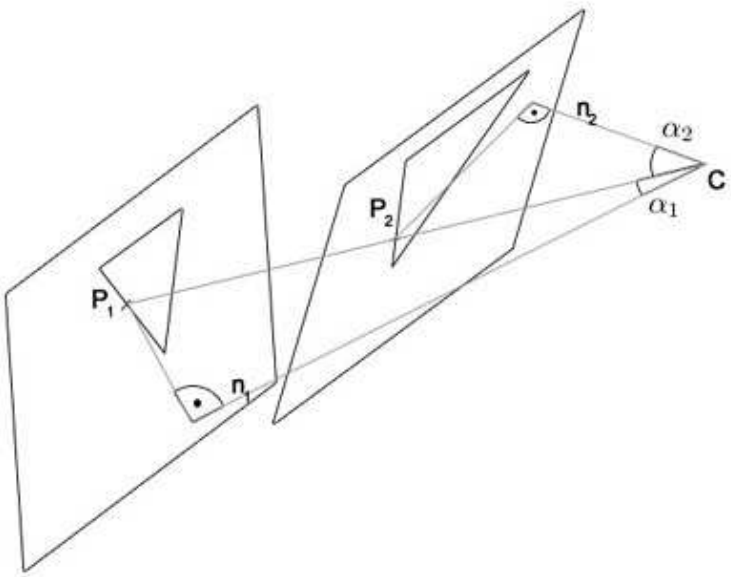
$$\begin{aligned} |P_1C| &= \frac{d_1}{\cos\alpha_1} \\ |P_2C| &= \frac{d_2}{\cos\alpha_2} \end{aligned}$$

kde α_1 je uhol medzi vektormi $\vec{n}_1$ a $\vec{v}$ a α_2 je uhol medzi vektormi $\vec{n}_2$ a $\vec{v}$, pričom $\cos\alpha_1 = \vec{n}_1.\vec{v}$ a $\cos\alpha_2 = \vec{n}_2.\vec{v}$

Hranový algoritmus taktiež nestačí na nájdenie všetkých prekrytí. Na obrázku 4.1.b. je prípad, ktorý tento algoritmus nedetekuje. Ale hranový a vrcholový algoritmus nájdu všetky prípady prekrytia za podmienok uvedených na začiatku.

Zoraďovanie prekrytých stien

Pomocou hore uvedených algoritmov (state 4.1.2, 4.1.2) vieme pre každé dva polygóny rozhodnúť, ktorý z nich je vpredu a ktorý vzadu, prípadne, že sa neprekrývajú. Máme teda množinu polygónov a na nej definovanú reláciu poradie ($<$), pričom $p_1 < p_2$ znamená, že polygón p_1 je prekrytý polygónom p_2. Táto relácia nám na množine polygónov vytvára orientovaný graf. Na tomto grafe si ukážeme algoritmus, ktorý rozhodne o poradí vykresľovania polygónov tak, aby sa polygóny umiestnené vzadu vykreslili ako prvé a



Obrázok 4.3: Porovnanie vzdialeností bodov P_1 a P_2 od stredu premietania C.

polygóny vpredu ako posledné[5].

Ideme zostrojiť graf G(V,E). V je množina uzlov, E $\subseteq$ V $\times$ V je množina hrán. Pre každý polygón p vytvoríme uzol $v \in$ V grafu G.

Ďalej E = $\{(v_1, v_2)$; $v_1, v_2 \in$ V, pre polygóny p_1 a p_2 prislúchajúce uzlom v_1 a v_2 platí: $p_1 < p_2$ $\}$

Uzol v_2 nazveme potomkom uzla v_1 a uzol v_1 nazveme predchodcom uzla v_2, ak $(v_1, v_2) \in$ E. Uzol, ktorý nemá žiadneho predchodcu nazveme koreň.

Každý uzol bude mať tieto premenné:

- *pos* - počet polygónov, ktoré sú z pohľadu pozorovateľa za polygónom reprezentovaným daným uzlom. T.j. ak $pos = 0$, tak za polygónom už nie je žiadny iný a bude sa vykresľovať medzi prvými.
- *snd* - táto premenná sa využíva na zabránenie zacyklenia.

Algoritmus bude pre každý uzol hľadať najdlhšiu cestu z koreňa do tohto uzla. Cesta medzi uzlami v_1 a v_2 je postupnosť hrán $e_1, e_2, \ldots, e_n \in$ E, pre ktoré platí: $e_1 = (v_1, u)$, $e_n = (v, v_2)$ a nech $e_i = (u_{i_1}, u_{i_2})$, $e_{i+1} = (u_{i+1_1}, u_{i+1_2})$, potom $u_{i_2} = u_{i+1_1}$.

Dĺžka cesty je číslo n, t.j. počet hrán v ceste.

Na začiatku nastavíme pre každý uzol u hodnota $u.pos = 0$. Každý uzol u pošle svojim potomkom číslo $u.pos + 1$. Ak uzol v dostane od svojho

[5]Toto je princíp maliarovho algoritmu. Keďže sme ale najprv odstránili úplne neviditeľné steny, náš algoritmus bude rýchlejší

predchodcu číslo n väčšie ako $v.pos$, nastaví si $v.pos = n$. Ako prvé posielajú správy korene. Ich hodnota *pos* sa nezmení, ostane rovná nule.

Po skončení algoritmu bude mať každý uzol v premennej *pos* dĺžku najdlhšej cesty z najvzdialenejšieho koreňa. Polygóny sa budú vykresľovať v nasledovnom poradí. Najprv sa nakreslia tie, pre ktoré má prislúchajúci uzol $pos = 0$, potom tie s $pos = 1$ atď. Z definície relácie $<$ vyplýva, že sa prekrývajúce polygóny vykreslia v poradí od najvzdialenejších po najbližšie.

Algoritmus 2 popisuje postup podrobnejšie.

Algoritmus 2: Zoraďovanie polygónov

Data: graf $G = (V, E)$, každý uzol $v \in V$ má premenné *pos*, *snd*, *par*.
Result: Pre $\forall v \in V$ $v.pos$ je číslo určujúce poradie vykreslenia polygónu

inicializácia:
forall $v \in V$ **do**
    $v.pos \leftarrow 0$
    $v.snd \leftarrow false$
    $v.par \leftarrow NULL$
end
forall $v \in V \wedge v$ *je koreň* **do**
    v pošle správu $(1, snd)$ všetkým svojím potomkom
end

uzol v prijme správu (n, snd) od uzla u:
if $v.snd$ **then**
    zacyklenie
else
    if $n > v.pos$ **then**
        $v.pos \leftarrow n$
        if v *má potomkov* **then**
            $v.par \leftarrow u$
            $v.snd \leftarrow true$
            v pošle svojim potomkom správu $(v.pos + 1, snd)$
        else
            v pošle u správu $(done)$
        end
    else
        v pošle u správu $(done)$
    end
end

uzol v prijme správu $(done)$ od uzla u:
$v.snd \leftarrow false$
v pošle uzlu $v.par$ správu $(done)$.

4.2 Efektívny algoritmus

Ako zefektívniť algoritmus viditeľnosti? Jeden zo spôsobov je ten, že nebudeme porovnávať každý polygón s každým, ale len niektoré vybrané dvojice. Nemá zmysel testovať také polygóny, ktoré sú príliš ďaleko od seba, lebo takéto sa nikdy neprekryjú.

Predstavme si meš premietnutý v rovine. Túto rovinu si rozdelíme na niekoľko obdĺžnikových oblastí. Testovať sa medzi sebou budú len polygóny spadajúce do rovnakej oblasti.

4.2.1 Rozdelenie na oblasti

Nech V je množina vrcholov premietnutého mešu. Definujme

$$\begin{aligned} minx &= \min\{x\,; P = (x,y),\ P \in \mathrm{V}\} \\ miny &= \min\{y\,; P = (x,y),\ P \in \mathrm{V}\} \\ maxx &= \max\{x\,; P = (x,y),\ P \in \mathrm{V}\} \\ maxy &= \max\{y\,; P = (x,y),\ P \in \mathrm{V}\} \end{aligned}$$

Vytvoríme v rovine obdĺžnik so súradnicami $(minx, miny)$ vľavo dole a $(maxx, maxy)$ vpravo hore. Obdĺžnik rozdelíme na a stĺpcov a b riadkov. Parametre a a b sa vhodne zvolia podľa priemernej veľkosti polygónov. Čím väčšie a a b zvolíme, tým rýchlejšie algoritmus pobeží. Avšak pre príliš veľké hodnoty réžia štruktúry opísanej ďalej môže naopak algoritmus spomaľovať.

Pre každý bod $P = (x, y) \in V$ zistíme, do ktorej oblasti patrí a naopak pre každú oblasť si budeme pamätať, ktoré body do nej patria. Oblasti si reprezentujeme ako dvojrozmerné pole Obl. Musíme nájsť indexy i, j určujúce oblasť pre bod P.

$$\begin{aligned} i &= \left\lfloor a.\frac{x-minx}{maxx-minx} \right\rfloor \\ j &= \left\lfloor b.\frac{y-miny}{maxy-miny} \right\rfloor \end{aligned}$$

Bod P patrí do oblasti $Obl[i][j]$.

Ďalej zistíme, do ktorých oblastí patria polygóny. Na to využijeme znalosť o tom, kam patria jednotlivé jeho vrcholy.

Pre každý polygón p nájdeme $mini = \min\{i \mid P \in Obl[i][j],\ P$ je vrchol polygónu $p\}$, $minj = \min\{j \mid P \in Obl[i][j],\ P$ je vrchol polygónu $p\}$, $maxi = \max\{i \mid P \in Obl[i][j],\ P$ je vrchol polygónu $p\}$, $maxj = \max\{j \mid P \in Obl[i][j],\ P$ je vrchol polygónu $p\}$. Polygón p patrí do týchto oblastí:

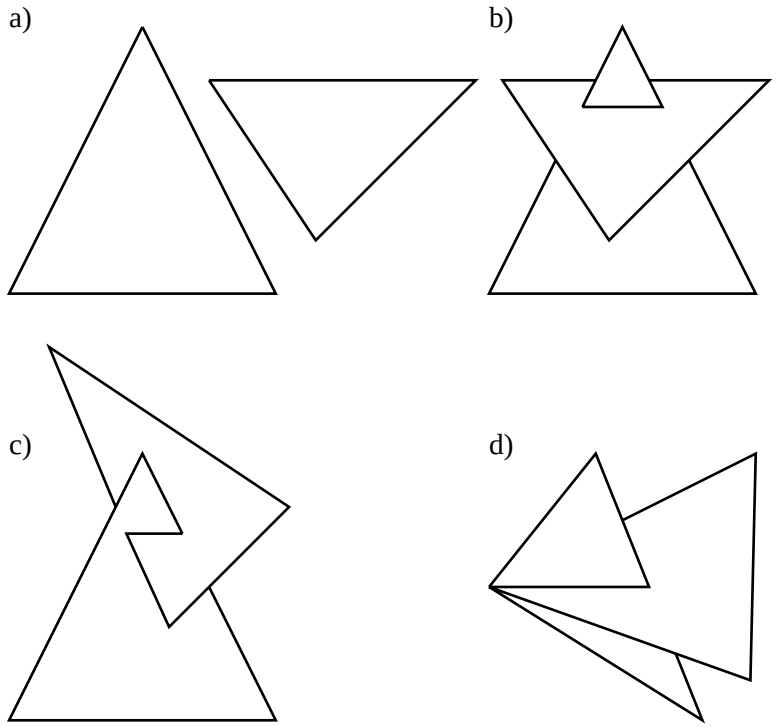
$$\begin{array}{cccc} Obl[mini][minj] & Obl[mini+1][minj] & \dots & Obl[maxi][minj] \\ Obl[mini][minj+1] & Obl[mini+1][minj+1] & \dots & Obl[maxi][minj+1] \\ \vdots & \vdots & \ddots & \vdots \\ Obl[mini][maxj] & Obl[mini+1][maxj] & \dots & Obl[maxi][maxj] \end{array}$$

Teraz pri riešení viditeľnosti, keď je treba zistiť, či je bod P zakrytý polygónom p, budeme testovať prekrytie len vtedy, ak $P \in Obl[i][j] \wedge p \in Obl[i][j]$, pre nejaké i, j.

4.3 Pretínajúce sa steny

Ďalší problém, ktorý sa objavuje pri riešení viditeľnosti mešov sú pretínajúce sa steny. Steny budeme mať reprezentované ako trojuholníky, preto si uvedieme definíciu pretínajúcich sa trojuholníkov.

Definícia 4.3.1 *Dva trojuholníky $\triangle A_1B_1C_1$ a $\triangle A_2B_2C_2$ sa pretínajú, ak ich prienikom je úsečka, ktorej aspoň jeden bod neleží na žiadnej hrane ani jedného z trojuholníkov $\triangle A_1B_1C_1$ a $\triangle A_2B_2C_2$.*



Obrázok 4.4: Štyri prípady vzájomnej polohy dvoch trojuholníkov v zmysle preťatia sa.

Budeme rozoznávať tri prípady vzájomnej polohy dvoch trojuholníkov v priestore pre potreby riešenia ich preťatia sa:

- Prípad 1: trojuholníky sa nepretínajú (obr. 4.4a.). Sem patrí aj prípad, keď sa trojuholníky dotýkajú, napr. hrana jedného trojuholníka leží druhom trojuholníku.

- Prípad 2: trojuholníky sa pretínajú tak, že jeden z nich je rozdelený na dva nové polygóny, pričom nemajú spoločný vrchol (obr. 4.4b.) alebo trojuholníky sa pretínajú tak, že ani jeden z nich nie je rozdelený na dva polygóny (obr. 4.4c.)

- Prípad 3: trojuholníky sa pretínajú a jeden z vrcholov niektorého trojuholníka leží v rovine druhého trojuholníka (obr. 4.4d.)

Problém budeme riešiť tak, že keď nájdeme dva pretínajúce sa trojuholníky, tak jeden z nich ponecháme a druhý rozdelíme na niekoľko ďalších. Preťatie trojuholníkov je nezávislé od polohy pozorovateľa a tak stačí, keď sa rieši len raz.

My ho budeme riešiť po odstránení odvrátených stien, aby sme netestovali zbytočne veľa polygónov.

Ideme nájsť pretínajúce sa trojuholníky a zaradiť ich do spomínaných skupín. Majme dva trojuholníky, $\triangle A_1B_1C_1$ a $\triangle A_2B_2C_2$. Ak majú spoločnú hranu, tak sa môžu pretínať len tak, že ležia v jednej rovine. Tento prípad nepotrebujeme riešiť.

Predpokladajme teda, že $\triangle A_1B_1C_1$ a $\triangle A_2B_2C_2$ nemajú spoločnú hranu. Uveďme si teraz nutnú podmienku pre preťatie sa dvoch trojuholníkov. Nech sa trojuholníky $\triangle A_1B_1C_1$ a $\triangle A_2B_2C_2$ pretínajú a nech $\triangle A_1B_1C_1$ leží v rovine π_1 a $\triangle A_2B_2C_2$ leží v rovine π_2. Potom jeden z vrcholov $\triangle A_1B_1C_1$ leží v opačnom polpriestore určenom rovinou π_2 ako ostatné dva vrcholy $\triangle A_1B_1C_1$ a súčasne jeden z vrcholov $\triangle A_2B_2C_2$ leží v opačnom polpriestore určenom rovinou π_1 ako ostatné dva vrcholy $\triangle A_2B_2C_2$.

Dosaďme teda postupne vrcholy A_1, B_1, C_1 do rovnice roviny π_2 (a_1, b_1, c_1) a vrcholy A_2, B_2, C_2 do rovnice roviny π_1 (a_2, b_2, c_2). Zadefinujme si funkciu $SIGN(t)$ (algoritmus 3).

Funkce `SIGN(`t`)`

```
SIGN(t):
if t < 0 then
    return 0
else
    return 1
end
```

Ak $SIGN(a_1) = SIGN(b_1) = SIGN(c_1)$ alebo $SIGN(a_2) = SIGN(b_2) = SIGN(c_2)$, potom sa trojuholníky nepretínajú (prípad 1).

Nech teda nutná podmienka platí. Ďalej nech žiaden vrchol ani jedného z trojuholníkov neleží v rovine druhého trojuholníka. Bez ujmy na všeobec-

nosti predpokladajme, že $SIGN(a_1) \neq SIGN(b_1)$ a súčasne $SIGN(a_1) \neq SIGN(c_1)$. Potrebujeme nájsť priesečník X úsečky A_1B_1 a roviny π_2 a priesečník Y úsečky A_1C_1 a roviny π_2. Tie nájdeme jednoducho pomocou a_1, b_1 a c_1:

$$\begin{aligned} X = &\quad A_1 + \frac{|a_1|}{|a_1|+|b_1|}\vec{v} \\ Y = &\quad A_1 + \frac{|a_1|}{|a_1|+|c_1|}\vec{u} \end{aligned}$$

kde $\vec{v} = B_1 - A_1$ a $\vec{u} = C_1 - A_1$.

Teraz pomocou nerovností (4.1) zistíme, či bod X alebo bod Y ležia v trojuholníku $A_2B_2C_2$[6]. Ak aspoň jeden z nich leží v $\triangle A_2B_2C_2$, tak dostávame prípad 2.

Predstavme si situáciu, keď $X, Y \notin \triangle A_2B_2C_2$. Tak zoberme dve vhodné hrany[7] z trojuholníka $A_2B_2C_2$ a zistime, či pretínajú $\triangle A_1B_1C_1$. Ak obe hrany pretínajú $\triangle A_1B_1C_1$, dostávame prípad 2, inak sa trojuholníky nepretínajú (prípad 1). Ak by len jedna hrana trojuholníka $A_2B_2C_2$ preťala trojuholník $A_1B_1C_1$, musel by jeden z bodov X, Y ležať v trojuholníku $A_2B_2C_2$. To sme ale vylúčili.

Ešte nám chýba prípad 3. Bez ujmy na všeobecnosti predpokladajme, že vrchol A_1 trojuholníka $A_1B_1C_1$ leží v rovine π_2 trojuholníka $A_2B_2C_2$. Ak hrana B_1C_1 preťala trojuholník $A_2B_2C_2$ alebo bod A_1 leží v trojuholníku $A_2B_2C_2$, nastal prípad 3.

Zostal nám ešte jeden prípad, ktorý sme neriešili. A to keď jeden vrchol trojuholníka leží v rovine druhého trojuholníka, ale neleží v druhom trojuholníku a ani žiadna hrana prvého trojuholníka nepretína druhý trojuholník. Trojuholníky sa však preťali tak, že druhý trojuholník je rozdelený na dva polygóny (podobne ako obr. 4.4 (b)). Túto situáciu riešime ako prípad 2.

4.3.1 Rozdeľovanie trojuholníkov

Uviedli sme si, ako detekujeme jednotlivé prípady preťatia sa dvoch trojuholníkov. Teraz si ukážeme, čo urobíme, ak niektorý z prípadov nastal.

Prípad 1

Keďže sa trojuholníky nepreťali, nie je čo riešiť.

Prípad 2

Nech $\triangle A_2B_2C_2$ pretína hranu A_1B_1 trojuholníka $A_1B_1C_1$. Nech $X = A_1B_1 \cap A_2B_2C_2$ a $Y = A_1C_1 \cap \pi_2$, $\triangle A_2B_2C_2 \in \pi_2$. Trojuholník $A_1B_1C_1$ rozdelíme

[6]Spomínaná nerovnosť určuje, či bod leží v trojuholníku pre dvojrozmerný priestor. To nám ale neprekáža, lebo body X, Y, A_2, B_2, C_2 si môžeme premietnuť.

[7]Ich konce ležia vždy v opačných polpriestoroch definovaných rovinou π_1

na tri nové: $\triangle A_1XY$, $\triangle XB_1C_1$ a $\triangle XC_1Y$. $\triangle A_2B_2C_2$ ponecháme nezmenený.

Prípad 3

Nastal prípad 3. Bez ujmy na všeobecnosti predpokladajme, že vrchol A_1 trojuholníka $A_1B_1C_1$ leží v rovine $\pi_2 \supset \triangle A_2B_2C_2$. Nech $X = B_1C_1 \cap \pi_2$. Trojuholník $A_1B_1C_1$ rozdelíme na dva nové: $\triangle A_1B_1X$ a $\triangle A_1XC_1$. Trojuholník $A_2B_2C_2$ ponecháme nezmenený.

Rozdelenie susedných trojuholníkov

Keď pri rozdeľovaní trojuholníka rozdelíme hranu tohto trojuholníka na dve, tak ostatné trojuholníky obsahujúce túto hranu rozdelíme na dva nové trojuholníky.

4.3.2 Algoritmus na riešenie problému pretínajúcich sa stien

Teraz si ukážeme kompletný algoritmus na riešenie problému pretínajúcich sa stien. Na vstupe máme trojuholníkový meš. Na výstupe dostávame modifikovaný trojuholníkový meš, ktorého steny sa nepretínajú. Pre lepšiu prehľadnosť uvádzame algoritmus riešiaci problém pre všetky steny, nie len pre viditeľné (algoritmus 4). Algoritmus porovnáva steny $F_1[i]$ a $F_1[j]$, pričom pre jednoduchosť vždy rozdelí $\triangle F_1[i]$. Doplnenie o test, ktorý trojuholník sa má deliť, je triviálne.

Algoritmus netestuje dvakrát tú istú dvojicu trojuholníkov. $parent(F_1[i]) \neq parent(F_1[j])$ znamená, že trojuholníky $F_1[i]$ a $F_1[j]$ nevznikli rozdelením toho istého trojuholníka. Ak áno, nie je potrebné ich testovať.

Algoritmus 4: Riešenie problému pretínajúcich sa stien

Data: meš M, množina vrcholov V, množina stien F, počet stien $fcount$

Result: meš M_1, nová množina vrcholov V_1, nová množina stien F_1, nový počet stien $fcount_1$

inicializácia:
$M_1 \leftarrow M$, $V_1 \leftarrow V$, $F_1 \leftarrow F$, $fcount_1 \leftarrow fcount$, $i \leftarrow 0$, $j \leftarrow 1$,
$oldfcount \leftarrow fcount$,

for $i < fcount_1$ **do**
  for $j < oldfcount$ **do**
    testujeme $F_1[i]$ a $F_1[j]$, $parent(F_1[i]) \neq parent(F_1[j])$
    switch *poloha trojuholníkov* $F_1[i]$ *a* $F_1[j]$ **do**
      case *prípad 2*
        Vznikli dva nové vrcholy X, Y: $V_1 = V_1 \cup \{X, Y\}$
        Z $F_1[i]$ vznikli tri nové trojuholníky t_1, t_2, t_3:
        $fcount_1 = fcount_1 + 2$
        $F_1[i] \leftarrow t_1$, $F_1[fcount_1 - 2] \leftarrow t_2$, $F_1[fcount_1 - 1] \leftarrow t_3$
        Rozdeľ susedné trojuholníky
      end
      case *prípad 3*
        Vznikli dva nové vrcholy X, Y: $V_1 = V_1 \cup \{X, Y\}$
        Z $F_1[i]$ vznikli dva nové trojuholníky t_1, t_2:
        $fcount_1 = fcount_1 + 1$
        $F_1[i] \leftarrow t_1$, $F_1[fcount_1 - 1] \leftarrow t_2$
        Rozdeľ susedné trojuholníky
      end
      otherwise
        nie je čo riešiť
      end
    end
    $j \leftarrow j + 1$
  end
  $oldfcount \leftarrow fcount_1$, $i \leftarrow i + 1$, $j \leftarrow i + 1$
end

Kapitola 5

Priesečníky objektov

V tejto časti si ukážeme, ako nájsť priesečníky telies v priestore ($\mathbb{R}^3$). Priesečníky si reprezentujeme ako krivky zložené z Bézierových kubík. Premietnuté objekty určené na vykreslenie sú uložené ako plochy pospájané z Coonsových záplat, ktorých hranice tvoria Bézierove kubiky. Na určenie viditeľnosti budeme potrebovať rozdeliť pretínajúce sa objekty podľa ich priesečníku, teda budeme počítať prieniky Coonsových záplat a Bézierových kubík v rovine.

5.1 Prienik Bézierových kriviek

Ukážeme si, ako nájsť prieniky Bézierových kubík v $\mathbb{R}^2$ s využitím vlastnosti, že krivka leží v konvexnom obale svojho riadiaceho polygónu. Postup pre nájdenie prieniku dvoch Bézierových kriviek ľubovoľného stupňa je analogický. Nám stačí prienik Bézierových kubík.

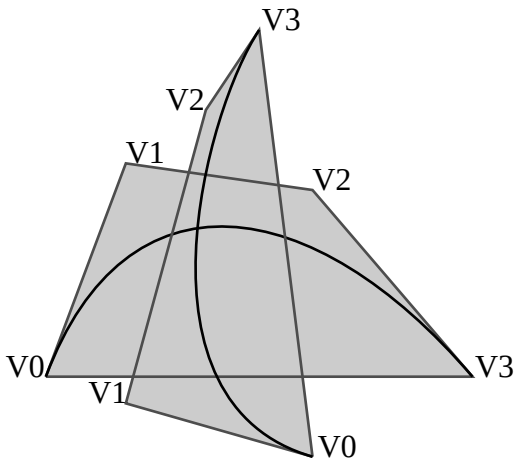
Sú dané dve Bézierove kubiky B_1 a B_2. Keďže krivky ležia v konvexnom obale svojich riadiacich polygónov, tak nutnou podmienkou preťatia sa kriviek je preťatie sa konvexných obalov riadiacich polygónov (obr. 5.1)(viď [Sed03b]).

Tento postup sme trochu zjednodušili tak, že pre obe krivky nájdeme najmenší obdĺžnik obsahujúci konvexný obal ich riadiacich polygónov a strany týchto obdĺžnikov sú rovnobežné s osami x a y. Pre takéto obdĺžniky sa ľahšie a rýchlejšie zisťuje, či sa preťali. Je zrejmé, že ak sa pretnú konvexné obaly riadiacich polygónov kriviek, tak sa pretnú aj spomínané obdĺžniky.

Než si uvedieme algoritmus na nájdenie prienikov, treba poznamenať, že maximálny počet prienikov dvoch Bézierových kubík je deväť (viď vlastnosť 5 v stati 1.3.2).

5.1.1 Algoritmus hľadajúci prieniky Bézierových kubík

Algoritmus bude pracovať nasledovne:



Obrázok 5.1: Ak sa pretínajú dve Bézierove krivky, musia sa pretnúť aj konvexné obaly ich riadiacich polygónov.

1. Pre obe Bézierove kubiky $B_1(t)$ a $B_2(t)$ nájdeme najmenšie obdĺžniky o_1 a o_2 obsahujúce riadiaci polygón kubiky. Strany obdĺžnikov o_1 a o_2 sú rovnobežné s osami x a y.

2. Ak sa obdĺžniky o_1 a o_2 pretínajú, kubiky $B_1(t)$ a $B_2(t)$ prerozdelíme v bode $t = \frac{1}{2}$. Dostaneme tak štyri Bézierove kubiky. Zoberieme všetky dvojice týchto kriviek tak, že v jednej dvojici je jedna kubika vzniknutá z kubiky $B_1(t)$ a druhá kubika vzniknutá z kubiky $B_2(t)$. Pre každú takúto dvojicu testujeme ich prienik induktívne od bodu 1.

3. Ak sa obdĺžniky o_1 a o_2 nepretínajú, nerobíme nič.

Aby algoritmus nebežal donekonečna, určíme si $\varepsilon \in \mathbb{R}$, $\varepsilon > 0$. Ak strany obdĺžnikov o_1 a o_2 budú menšie ako ε, zoberieme bod, ktorý spája dve kubiky prislúchajúce obdĺžnikom o_1 a o_2 a ten prehlásime za prienik kriviek $B_1(t)$ a $B_2(t)$. Potom pokračujeme v hľadaní ďalších prienikov.

Než si napíšeme algoritmus formálne, ešte si ukážeme, ako zistiť, či sa pretínajú dva obdĺžniky v rovine.

Sú dané dva obdĺžniky o_1 a o_2. Zadefinujeme si štyri funkcie:

$$\begin{aligned} minx(o) &= \min\{x; A = [x, y] \wedge A \in o\} \\ miny(o) &= \min\{y; A = [x, y] \wedge A \in o\} \\ maxx(o) &= \max\{x; A = [x, y] \wedge A \in o\} \\ maxy(o) &= \max\{x; A = [x, y] \wedge A \in o\} \end{aligned}$$

Obdĺžniky o_1 a o_2 sa pretínajú práve vtedy, keď platí:

$$\begin{aligned} not \quad & (minx(o_1) > maxx(o_2) \ \vee \ maxx(o_1) < minx(o_2) \\ & \vee miny(o_1) > maxy(o_2) \ \vee \ maxy(o_1) < miny(o_2)) \end{aligned}$$

Ďalej si urobíme dve polia K_1 a K_2, ktoré budú predstavovať pôvodné Bézierove kubiky. Keď niektorú kubiku prerozdelíme, odstránime ju z poľa K_i a namiesto nej vložíme dve vzniknuté kubiky. Výstupom algoritmu bude pole res obsahujúce záznamy o prienikoch.

Funkcia $Subdivide(B, t_0)$ rozdelí Bézierovu krivku B v bode t_0 a vráti dvojicu kriviek vzniknutých po prerozdelení. Pre každú krivku si budeme v premennej $B.sub$ pamätať, koľkým je prerozdelením pôvodnej krivky. Teda ak $(B_1, B_2) = Subdivide(B, t_0)$, tak $B_1.sub = B.sub{+}1$ a $B_2.sub = B.sub{+}1$.

Funkcia $Rect(B)$ vráti najmenší obdĺžnik obsahujúci riadiaci polygón Bézierovej krivky B, ktorého hrany sú rovnobežné s osami x a y.

Funkcia $Insert(K, B, i)$ vloží do poľa K na pozíciu i Bézierovu krivku B, pričom pôvodné krivky od pozície i do konca posunie o jednu pozíciu ďalej:

$$\begin{aligned} K[j+1] \ &\leftarrow \ K[j] \quad j = Count(K), Count(K) - 1, \ldots, i \\ K[i] \ &\leftarrow \ B \end{aligned}$$

Nastavíme si ε na požadovanú hodnotu, napr. $\varepsilon = 0,1$.

Funkcia $Compute(K, i)$ vyráta pre pole Bézierových kriviek K a index i hodnotu parametra $t = t_0$, pre ktorý platí: $B(t_0) = K[i](1)$, kde $B(t)$ je pôvodná Bézierova krivka.

Funkce `Compute(`K, i`)`

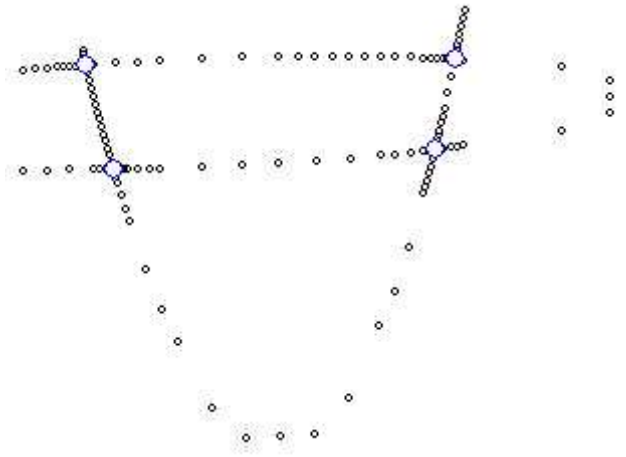
$Compute(K, i)$:
$j \leftarrow 0$, $t \leftarrow 0$
for $j \leq i$ **do**
  $c \leftarrow K[j].sub$
  $t \leftarrow t + \frac{1}{2^c}$
end
return t

Na obrázku 5.2 je výsledok algoritmu hľadajúceho prieniky dvoch Bézierových kubík.

5.2 Prienik Bézierovej kubiky a roviny

Pri počítaní viditeľnosti kriviek budeme potrebovať nájsť prieniky Bézierovej kubiky a roviny.

Je daná Bézierova kubika $B(t)$ a rovina π. Vieme, že počet prienikov kubiky $B(t)$ a roviny π nie je väčší ako počet prienikov riadiaceho polygóny



Obrázok 5.2: Prieniky dvoch Bézierových kubík. Malé body predstavujú riadiace vrcholy jednotlivých kubík, veľké body predstavujú prieniky. V okolí prienikov je väčšia hustota riadiacich vrcholov, čo je spôsobené tým, že v týchto miestach sa kubiky delili najviac.

kubiky $B(t)$ a roviny π (viď vlastnosť 5 v stati 1.3.2). Preto podobne ako pri hľadaní prieku dvoch Bézierových kriviek v rovine (stať 5.1) teraz budeme hľadať prieniky riadiaceho polygónu kubiky $B(t)$ a roviny π a kubiku $B(t)$ budeme následne prerozdeľovať.

Hľadanie prienikov polygóny p a roviny π je jednoduché. Stačí hľadať prieniky jednotlivých úsečiek polygónu p a roviny π. Prienik úsečky s rovinou zistíme tak, že do rovnice roviny dosadíme krajné body úsečky. Ak majú výsledky opačné znamienka, alebo je niektorý nulový, tak sa úsečka s rovinou preťala.

Riadiaci polygón kubiky $B(t)$ je určený jej riadiacimi vrcholmi. Pri hľadaní prienikov budeme postupovať nasledovne:

1. Pre danú Bézierovu kubiku $B_0(t)$ zistíme, či niektorá z úsečiek V_0V_1, V_1V_2, V_2V_3 (V_0, V_1, V_2 a V_3 sú riadiace vrcholy kubiky $B_0(t)$) pretína rovinu π

2. Ak áno, kubiku $B_0(t)$ rozdelíme na dve v bode $t = \frac{1}{2}$ a testujeme nové kubiky na prienik s rovinou π

3. Ak nie, skončíme.

Opäť si určíme ε, aby sme zabránili nekonečnému behu algoritmu. Tento raz budeme testovať, či dĺžka úsečiek V_0V_1, V_1V_2, V_2V_3 je menšia ako ε. Ak áno, prehlásime spoločný riadiaci vrchol práve vzniknutých kubík za prienik

Bézierovej kubiky $B(t)$ a roviny π. Tento bod vyjadríme ako $B(t_0)$, kde t_0 nájdeme pomocou funkcie $Compute(K, i)$ zo state 5.1.1.

5.3 Priesečník dvoch kruhov

Sú dané dva kruhy k_1 a k_2 ich S_1 a S_2 a ich polomery r_1 a r_2. Ideme hľadať ich priesečník.

Ak kruhy k_1 a k_2 ležia v tej istej rovine, môžu sa pretínať v ploche určenej dvomi časťami kružníc. Týmto prípadom sa však nebudeme zaoberať. Ak taký prípad nastane, budeme považovať jeden z kruhov za predný a budeme ho kresliť cez druhý kruh.

Predpokladajme teda, že kruhy k_1 a k_2 ležia v rôznych rovinách π_1 a π_2 a tieto roviny nie sú rovnobežné[1]. Priesečník kruhov k_1 a k_2 bude ležať na priesečníku rovín π_1 a π_2, ktorým je priamka p.

Nech vektor $\vec{v} = \vec{n}_1 \times \vec{n}_2$ je smerovým vektorom priamky p, vektory $\vec{n}_1$ a $\vec{n}_2$ sú normálové vektory rovín π_1 a π_2. Potrebujeme nájsť prieniky kruhov k_1 a k_2 s priamkou p.

Postup, ako nájsť prienik kruhu s priamkou si ukážeme na kruhu k_1. Prienik priamky p a kruhu k_2 sa hľadá analogicky.

Najprv musíme nájsť vektor $\vec{u}_1$ zo stredu S_1, ktorý je kolmý na priamku p: $\vec{u}_1 = \vec{n}_1 \times \vec{v}$. Nech bod X_1 je priemet stredu S_1 v smere vektora $\vec{u}_1$ na priamku p. Aby sa kruh k_1 preťal s priamkou p, musí platiť: $|S_1X_1| \leq r_1$.

Ako vyrátame vzdialenosť $|S_1X_1|$? Poznáme smerový vektor $\vec{u}_1$ úsečky S_1X_1. Z rovnice roviny π_2 si vypočítame vzdialenosť d stredu S_1 od roviny π_2. Nech bod Y_1 je priemet stredu S_1 do roviny π_2 v smere vektora $\vec{n}_2$. Bod X_1 leží v rovine π_2.

Dostávame pravouhlý trojuholník $S_1Y_1X_1$. Veľkosť hrany S_1Y_1 je d, uhol $X_1S_1Y_1$ je uhol, ktorý zvierajú vektory $\vec{u}_1$ a $\vec{n}_2$. My chceme zistiť veľkosť hrany S_1X_1. Predpokladajme, že vektory $\vec{n}_2$ a $\vec{u}_1$ sú normalizované, potom:

$$|S_1X_1| = \frac{d}{\vec{u}_1 \cdot \vec{n}_2}$$

kde $\vec{u}_1 \cdot \vec{n}_2$ je vlastne kosínus uhla $X_1S_1Y_1$.

Už poznáme vzdialenosť stredu S_1 od priamky p ($|S_1X_1|$). Ak $|S_1X_1| \leq r_1$, priamka p pretína kruh k_1.

Predpokladajme, že $|S_1X_1| < r_1$[2]. Potrebujeme nájsť úsečku A_1B_1, ktorá je prienikom kruhu k_1 a priamky p.

Vzdialenosť bodov A_1 a B_1 od stredu S_1 je r_1. Máme teda dva zhodné pravouhlé trojuholníky $S_1X_1A_1$ a $S_1X_1B_1$. Z Pytagorovej vety dostávame:

$$|X_1A_1| = |X_1B_1| = \sqrt{r_1^2 - |S_1X_1|^2}$$

[1] Ak by boli, tak by sa kruhy nemohli pretnúť a teda nemá zmysel sa touto situáciou zaoberať.

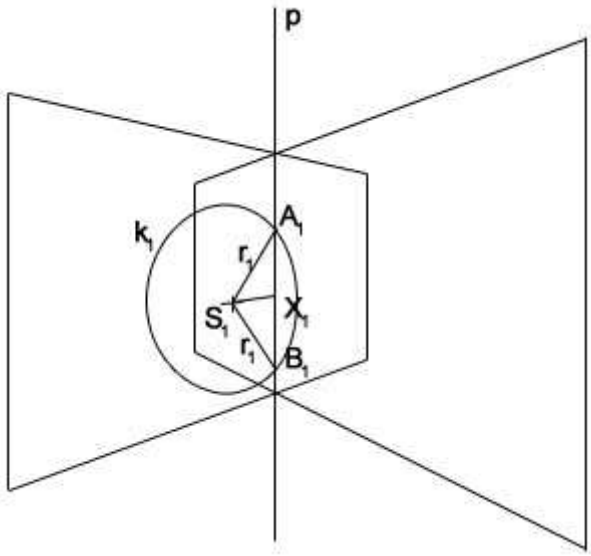
[2] Ak $|S_1X_1| = r_1$, tak prienikom priamky p a kruhu k_1 je jeden bod.

Body A_1 a B_1 zostrojíme nasledovne:

$$\begin{aligned} A_1 &= X_1 + |X_1A_1| \cdot \vec{v} \\ B_1 &= X_1 - |X_1B_1| \cdot \vec{v} \end{aligned}$$

pričom predpokladáme, že vektor $\vec{v}$ je normalizovaný.

Situácia je ilustrovaná na obrázku 5.3.



Obrázok 5.3: Priesečník kruhu k_1 s priamkou p, ktorá je prienikom dvoch rovín. S_1, r_1 - stred a polomer kruhu k_1; X_1 kolmý priemet bodu S_1 na priamku p; A_1B_1 - úsečka, ktorá je priesečníkom kruhu k_1 a priamky p.

Analogicky zostrojíme aj úsečku A_2B_2, ktorá je prienikom kruhu k_2 a priamky p.

Teraz, keď máme úsečky A_1B_1 a A_2B_2, nájdeme ich prienik. Ten bude zároveň prienikom kruhov k_1 a k_2. Stačí, ak zistíme, či niektorý z bodov A_1, B_1 neleží na úsečke A_2B_2 alebo či niektorý z bodov A_2, B_2 neleží na úsečke A_1B_1.

5.4 Priesečník gule a kruhu

Priesečníkom gule a kruhu je krivka na povrchu gule, ležiaca v jednej rovine.

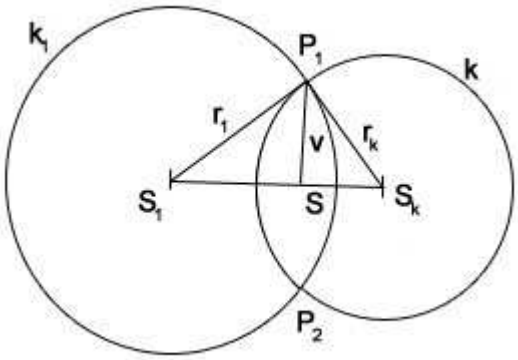
Je daná guľa g so stredom v S_g a polomerom r_g. Ďalej je daný kruh k so stredom v S_k, polomerom r_k, ležiaci v rovine π, ktorá je určená stredom S_k a normálovým vektorom $\vec{n}$.

Najprv nájdeme prienik gule g s rovinou π. Do rovnice roviny π dosadíme stred S_g gule g[3] a dostaneme tak vzdialenosť d stredu S_g od roviny π. Ak $|d| < r_g$, prienikom gule g a roviny π je kružnica k_1, ktorá má stred v bode $S_1 = S_g + d \cdot \vec{n}$. Polomer r_1 kružnice k_1 dostaneme nasledovne:

$$r_1 = \sqrt{r_g^2 - d^2}$$

Teraz máme v rovine π kruh k a kružnicu k_1. Ich priesečník je zároveň priesečníkom gule g a kruhu k.

Podobne ako pri hľadaní prieniku dvoch gúľ (stať 5.8), nájdeme body P_1 a P_2, ktoré tvoria prienik kružníc k a k_1 (obr. 5.4).



Obrázok 5.4: Priesečník kruhu k so stredom S_k a polomerom r_k a kružnice k_1 so stredom S_1 a polomerom r_1. Priesečník tvorí časť kružnice k_1 od bodu P_2 po bod P_1.

$$\begin{aligned} S &= S_1 + \frac{r_k^2 - r_1^2 - |S_1S_k|^2}{2 \cdot |S_1S_k|^2} \cdot (S_1 - S_k) \\ \vec{v} &= \vec{n} \times (S_1 - S_2) \\ a &= \sqrt{r_1^2 - |S_1S|^2} \\ P_1 &= S + a \cdot \frac{\vec{v}}{|\vec{v}|} \\ P_2 &= S - a \cdot \frac{\vec{v}}{|\vec{v}|} \end{aligned}$$

Časť kružnice k_1 od bodu P_2 po bod P_1 tvorí prienik kruhu k a kružnice k_1. Túto krivku si reprezentujeme ako krivku K zloženú s Bézierových kubík (sta 3.1.1)

Krivka K, ktorú sme dostali, je prienikom gule g a kruhu k. Pri vykresľovaní však budeme potrebovať len viditeľnú časť krivky K. Preto ešte musíme nájsť prienik krivky K s kontúrou gule. Kontúra c gule g je kružnica ležiaca

[3]Vektor $\vec{n}$ musí byť normalizovaný.

v nejakej rovine π_c (viď stať 3.3.4). Pre nájdenie prienikov krivky K a kontúry c stačí, ak nájdeme prieniky krivky K a roviny π_c (vid stať 5.2). Obe krivky (K aj c) totiž ležia na povrchu gule.

Nech body P_1 a P_2 sú prieniky krivky K a roviny π_c. Prieniky budú nanajvýš dva, lebo krivka K je časť kružnice a kružnica sa s rovinou pretína nanajvýš v dvoch bodoch[4]. Body P_1 a P_2 delia krivku K na tri časti: K_1, K_2 a K_3. Pre krivku K_1 zistíme, či sa nachádza v tom istom polpriestore určenom rovinou π_c ako stred premietania C[5]. Ak áno, krivka K_1 je viditeľná, krivka K_2 nie je viditeľná, krivka K_3 je viditeľná. Ak nie, krivka K_1 nie je viditeľná, krivka K_2 je viditeľná, krivka K_3 nie je viditeľná.

5.5 Priesečník valca a kruhu

Podobne ako u prieniku guľa a kruhu, aj tu tvorí priesečník krivka na povrchu valca ležiaca v jednej rovine.

Je daný valec c so stredmi podstáv S_1 a S_2 a polomerom r. Ďalej je daný kruh k so stredom v S_k, polomerom r_k, ležiaci v rovine π, ktorá je určená stredom S_k a normálovým vektorom $\vec{n}$.

Predstavme si, že rovina π pretína valec len v plášti. V tomto prípade tvorí priesečník elipsa e[6]. Stred S elipsy e leží na priamke S_1S_2 (obr. 5.5).

Elipsa e je určená stredom S, polomerom r, polomerom d a vektormi osí $\vec{u}$ a $\vec{w}$ (obr. 5.6). Polomer r je polomerom valca. Vektor $\vec{u} = \vec{n} \times \vec{v}$ a vektor $\vec{w} = \vec{u} \times \vec{n}$. Ešte nám zostáva nájsť stred S a polomer d.

Stred S leží sa prieniku priamky S_1S_2 a roviny π. Jeho polohu vyrátame nasledovne:

$$S = S_2 - i \cdot a \cdot \frac{\vec{v}}{|\vec{v}|}$$

kde $\vec{v} = S_1 - S_2$, a je orientovaná vzdialenosť bodu S_2 od roviny π, $i = -1$ ak $\vec{n} \cdot \vec{v} < 0$, $i = 1$ ak $\vec{n} \cdot \vec{v} > 0$.

Polomer d vypočítame pomocou polomeru r a uhla α, ktorý zvierajú vektory $\vec{n}$ a $\vec{v}$.

$$\alpha = \arccos \frac{|\vec{n} \cdot \vec{v}|}{|\vec{n}||\vec{v}|}$$

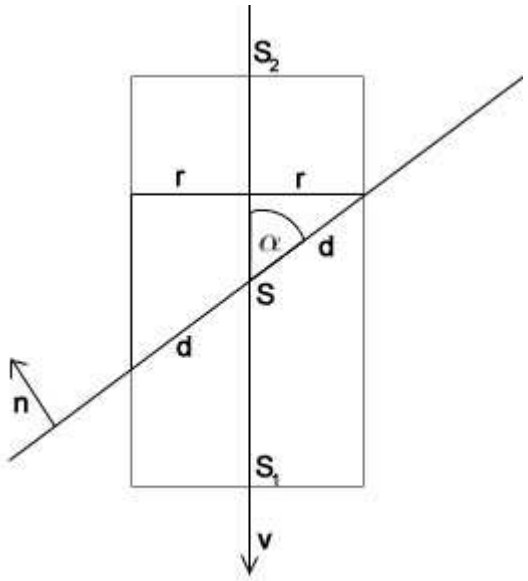
$$d = \frac{r}{\sin \alpha}$$

Našli sme teda elipsu e. Avšak na začiatku sme predpokladali, že rovina pretína iba plášť valca. Čo sa stane, ak rovina pretne aj niektorú podstavu, prípadne obe.

[4] ak kružnica neleží v danej rovine

[5] stačí jeden bod krivky K_1 dosadiť do rovnice roviny π_c.

[6] Ak je vektor $\vec{n}$ rovnobežný s vektorom $\vec{v} = S_1 - S_2$, priesečníkom je kružnica. Ak je vektor $\vec{n}$ kolmý na vektor $\vec{v}$, priesečníkom môže byť obdĺžnik.



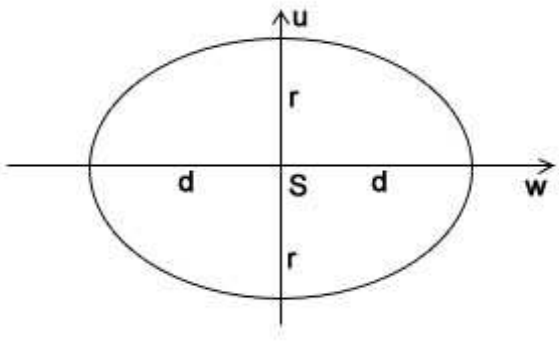
Obrázok 5.5: Kolmý priemet valca c a roviny π v smere kolmom na vektory $\vec{n}$ a $\vec{v}$. $\vec{n}$ - normálový vektor roviny π; $\vec{v} = S_1 - S_2$; S_1, S_2 - stredy podstáv valca c; r - polomer valca c; α - uhol zvierajúci vektory $\vec{n}$ a $\vec{v}$; S - stred elipsy, ktorá je prienikom roviny π a valca c.

Z elipsy najskôr spravíme krivku k_e, ktorú tvoria pospájané Bézierove kubiky (stať 3.1.2). Potom nájdeme priesečníky krivky k_e s rovinami π_1 a π_2, v ktorých ležia podstavy valca (stať 5.2). Krivka k_e sa nám tak rozloží na viacero menších kriviek podľa toho, koľko raz sa krivka k_e preťala s rovinami π_1 a π_2. Z týchto kriviek zoberieme len tie, ktoré ležia medzi rovinami π_1 a π_2[7] a ich zjednotenie označíme K.

Teraz máme krivku K, ktorá je prienikom roviny π a valca c. Zostáva nám už len spraviť prienik krivky K s kruhom k.

Kružnicu k si aproximujeme pomocou Bézierových kubík (stať 3.1.1). Túto aproximáciu si označíme ako K_k. Krivky K a K_k ležia v jednej rovine. Prienik Bézierových kubík ležiacich v jednej rovine už vieme spraviť (stať 5.1). Časť krivky K nachádzajúca sa v kruhu k je priesečníkom valca v a kruhu k.

[7]Zoberieme nejaký vnútorný bod krivky a ten dosadíme do rovníc rovín π_1 a π_2. Tak zistíme, ktoré krivky sa nachádzajú medzi týmito rovinami.



Obrázok 5.6: Elipsa e, ktorá je prienikom roviny π a valca c. Polomery elipsy sú r (to je aj polomer valca) a d. S - stred elipsy, leží na priamke S_1S_2. $\vec{u} = \vec{n} \times \vec{v}$; $\vec{w} = \vec{u} \times \vec{n}$; $\vec{n}$ - normálový vektor roviny π; $\vec{v} = S_1 - S_2$.

5.6 Priesečník kužeľa a kruhu

Je daný kužeľ c so stredom podstavy S_c, vrcholom V_c a polomerom podstavy r_c. Ďalej je daný kruh k so stredom v S_k, polomerom r_k, ležiaci v rovine π, ktorá je určená stredom S_k a normálovým vektorom $\vec{n}$.

Podobne ako pri hľadaní prieniku valca a kruhu, najprv nájdeme prienik kužeľa c s rovinou π. Prienikom kužeľa a roviny môže byť elipsa, parabola alebo hyperbola. Pekný dôkaz aj s popisom konštrukcie je v [Tul02].

5.7 Priesečník mešu a kruhu

Je daný meš m a kruh k so stredom S a polomerom r, ležiaci v rovine π.

V stati o mešoch (stať 4.3) sme si ukázali, ako spraviť prienik trojuholníka s rovinou. Teraz pre každý trojuholník mešu zistíme, či sa pretína s rovinou π, v ktorej leží kruh k. Ak sa trojuholník pretína s rovinou π a vzdialenosť tohto prieniku (úsečky) od stredu S je menšia ako r, trojuholník sa z kruhom k pretína. V tomto prípade rozdelíme trojuholník podľa roviny π.

5.8 Priesečník dvoch gúľ

Priesečníkom dvoch gúľ je kružnica. Jej stred leží na priamke určenej stredmi pretínajúcich sa gúľ.

Najprv si uvedieme podmienky, kedy sa dve gule pretínajú, potom si dokážeme, že ich prienikom je kružnica[8] a nakoniec si ukážeme, ako túto kružnicu nájsť.

[8] alebo jeden bod.

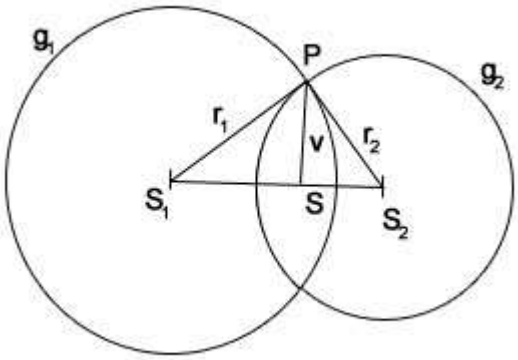
Sú dané dve gule: guľa g_1 so stredom S_1 a polomerom r_1 a guľa g_2 so stredom S_2 a polomerom r_2. Nech $d = |S_1S_2|$ je vzdialenosť stredov gúľ g_1 a g_2. Gule g_1 a g_2 sa pretínajú práve vtedy, keď platí nasledovné:

$$r_1 + r_2 \geq d \quad \wedge \quad d + r_1 \geq r_2 \quad \wedge \quad d + r_2 \geq r_1 \tag{5.1}$$

Pre body priesečníku gulí g_1 a g_2 platí, že sú od stredu S_1 vzdialené r_1 a od stredu S_2 vzdialené r_2. Tento priesečník definujeme nasledovne:

$$M = \{P;\, |PS_1| = r_1,\, |PS_2| = r_2\}$$

Ak platí podmienka (5.1), tak body S_1, S_2, P ($P \in M$) tvoria zhodné trojuholníky. Všetky tieto trojuholníky majú spoločnú hranu S_1S_2. Ľahko vidieť, že body množiny M tvoria kružnicu so stredom S ležiacim na prieniku hrany S_1S_2 a výšky v trojuholníka S_1S_2P z bodu P. Polomer kružnice je dĺžka výšky v. Kružnica leží v rovine určenej bodom S a vektorom $\vec{u} = S_1 - S_2$ (obr. 5.7).



Obrázok 5.7: Priesečník dvoch gúľ. g_1 - guľa so stredom S_1 a polomerom r_1; g_2 - guľa so stredom S_2 a polomerom r_2; Priesečník gúľ g_1 a g_2 je kružnica so stredom S, polomerom $|v|$, P leží na tejto kružnici, v je výška trojuholníka S_1S_2P.

5.8.1 Konštrukcia priesečníka dvoch gúľ

Ukázali sme si, že priesečníkom dvoch gúľ je kružnica (resp. jeden bod). Teraz si uvedieme postup, ako ju nájsť. Znovu si zadefinujme priesečník gúľ g_1 a g_2, ale tento raz ako kružnicu k so stredom S, polomerom r, ležiacu v rovine definovanej bodom S a normálovým vektorom $\vec{n}$:

$$k = (S, r, \vec{n})$$

Vektor $\vec{n}$ poznáme. Je to vektor $\vec{n} = S_1 - S_2$, body S_1 a S_2 sú stredy gúľ g_1 a g_2. Potrebujeme nájsť stred S a vypočítať veľkosť polomeru r.

Vieme, že stred S leží na priamke S_1, S_2[9]. Označme $d = |S_1S_2|$, $d_1 = |S_1S|$ a uhol $\alpha = \angle PS_1S_2$. Z kosínusovej vety dostaneme:

$$\cos\alpha = \frac{r_2^2 - r_1^2 - d^2}{2r_1 d}$$

Ďalej platí:

$$\cos\alpha = \frac{d_1}{r_1}$$

a teda:

$$d_1 = \frac{r_2^2 - r_1^2 - d^2}{2d} \tag{5.2}$$

Stred S vyjadríme takto:

$$S = S_1 + \frac{d_1}{d} \cdot (S_2 - S_1)$$

a po dosadení (5.2):

$$S = S_1 + \frac{r_2^2 - r_1^2 - d^2}{2d^2} \cdot (S_2 - S_1)$$

Polomer $r = r_1 \cdot \sin\alpha$

Podobne ako pri hľadaní priesečníka gule a kruhu (stať 5.4), aj teraz potrebujeme zistiť viditeľnosť kružnice k_1 vzhľadom na gule g_1 a g_2. Kružnicu k_1 si najprv reprezentujeme ako krivku K zloženú z Bézierových kubík (stať 3.1.1) a potom vyrátame jej prienik s rovinami, v ktorých ležia kontúry gúľ g_1 a g_2 (stať 3.3.4). Dostaneme dve krivky zložené z Bézierových kubík. Obe predstavujú časť priesečníka gúľ g_1 a g_2, jedna krivka s vyriešenou viditeľnosťou vzhľadom na guľu g_1 druhá krivka s vyriešenou viditeľnosťou vzhľadom na guľu g_2.

5.9 Priesečník trojuholníka a gule

Je daná guľa g so stredom S a polomerom r. Ďalej je daný trojuholník ABC ležiaci v rovine π.

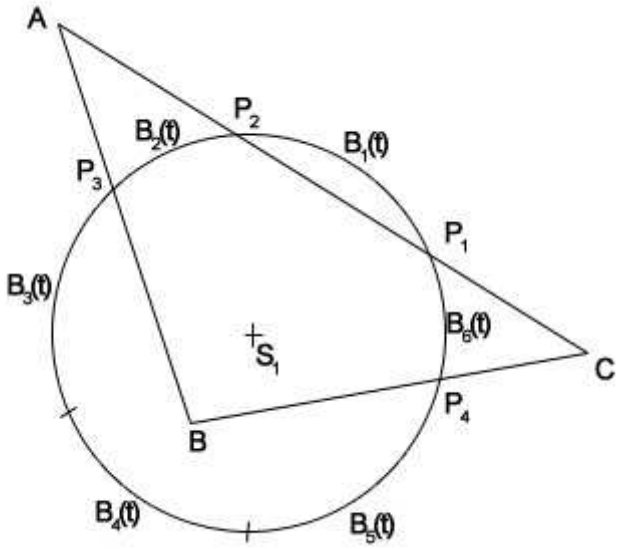
Najskôr nájdeme kružnicu k_1 so stredom S_1 a polomerom r_1, ktorá je prienikom gule g a roviny π (viď stať 5.4). Pre každú hranu trojuholníka ABC nájdeme jej prienik s kružnicou k_1. Nájdenie prieniku priamky a kruhu sme už mali (stať 5.3).

Nech teda body $P_1, P_2, \ldots, p_n$ sú prieniky hrán trojuholníka ABC a kružnice k_1. Zoraďme tieto body podľa ich polárnych súradníc vzhľadom na

[9]Stred S nemusí nutne ležať medzi bodmi S_1 a S_2.

stred kružnice k_1: $P_{i_1}, P_{i_2}, \ldots, p_{i_n}$, kde $(i_1, i_2, \ldots, i_n)$ je permutácia prvkov $(1, 2, \ldots, n)$.

Kružnicu k_1 si teraz reprezentujme ako krivku K zloženú z Bézierových kubík, pričom body $P_{i_1}, P_{i_2}, \ldots, p_{i_n}$ sú krajné riadiace vrcholy niektorých z týchto Bézierových kubík (stať 3.1.1) (obr. 5.8). Bézierova kubika nie je schopná reprezentovať celú kružnicu, ale len jej časť, preto musí byť kružnica k_1 rozdelená aj na niektorých ďalších miestach.



Obrázok 5.8: Kružnica k_1 so stredom S_1 reprezentovaná ako postupnosť Bézierových kubík $B_1(t), B_2(t), \ldots, B_6(t)$, pričom priesečníky P_1, P_2, P_3, P_4 trojuholníka ABC a kružnice k_1 sú krajné riadiace vrcholy niektorých Bézierových kubík.

5.10 Priesečník gule a mešu

Je daná guľa g so stredom S a polomerom r. Ďalej je daný meš m, ktorého steny sú trojuholníky, ktoré sa nepretínajú.

Nájdeme jeden trojuholník $\triangle_1$ mešu m, ktorý sa pretína s guľou g (stať 5.9). Vezmeme jednu krivku K_1, ktorá tvorí priesečník trojuholníka $\triangle_1$ a gule g.

Krajné vrcholy A_1 a B_1 krivky K_1 ležia na hranách trojuholníka $\triangle_1$. Nech krajný vrchol B_1 krivky K_1 leží na hrane e_1 trojuholníka $\triangle_1$. Nájdeme trojuholník $\triangle_2$ mešu m, ktorý tiež obsahuje hranu e_1. Nájdeme krivku K_2, ktorá je priesečníkom trojuholníka $\triangle_2$ a gule q. Krivka K_2 má krajné vrcholy

$A_2 = B_1$ a B_2. Vezmeme bod B_2 a hranu e_2 trojuholníka $\triangle_2$, na ktorej leží vrchol B_2. Opäť nájdeme trojuholník $\triangle_3$ mešu m obsahujúci hranu e_2 a opäť hľadáme krivku K_3.

Skončíme, keď pre nejakú hranu e_i už neexistuje ďalší trojuholník, alebo keď nájdeme už nájdený priesečník. Výsledná krivka zložená z kriviek $K_1, K_2, \ldots, K_n$ je priesečníkom mešu m a gule g. Samozrejme meš m a guľa g môžu mať aj viac ako jeden priesečník. Nájdeme trojuholník, ktorý sme ešte neprehľadávali a ktorý sa pretína s guľou g a algoritmus opakujeme.

5.11 Priesečník gule a valca

Je daná guľa g so stredom S a polomerom r_g a valec c so stredmi podstavy S_1 a S_2 a polomerom r_c.

Ukážeme si algoritmus, ktorý použijeme pri hľadaní prieniku medzi guľou, valcom a kužeľom. Je založený na tom, že vždy spravíme rez objektov nejakou rovinou π a v tejto rovine hľadáme priesečníky prienikov objektov s rovinou π. T.j. označme K_1 krivku predstavujúcu prienik roviny π s prvým objektom . Ďalej označme K_2 krivku predstavujúcu prienik roviny π s druhým objektom. Prienikom objektov v rovine π je prienik kriviek K_1 a K_2.

Krivku K predstavujúcu prienik objektov budeme aproximovať Bézierovými kubikami. Použijeme vrcholy, ktoré dostaneme hľadaním prienikov objektov v rezoch. V týchto bodoch tiež poznáme smerový vektor dotyčnice.

Nech vrchol P leží na prieniku objektov o_1 a o_2, t.j. $P \in K$. Smerový vektor $\vec{u}$ dotyčnice krivky K v bode P má tvar:

$$\vec{u} = \vec{n}_1 \times \vec{n}_2$$

kde vektor $\vec{n}_1$ je normálový vektor v bode P vzhľadom na povrch objektu o_1 a vektor $\vec{n}_2$ je normálový vektor v bode P vzhľadom na povrch objektu o_2.

Na zostrojenie krivky K interpolujúcej takéto vrcholy P použijeme interpoláciu opísanú v stati 3.2.

Krivka K je aproximáciou prieniku objektov o_1 a o_2.

5.12 Priesečník valca a mešu

Opäť tak ako pri hľadaní priesečníku gule a mešu, aj teraz nájdeme priesečník valca a trojuholníka mešu a budeme hľadať prieniky susedných trojuholníkov s valcom.

5.13 Priesečník kužeľa a mešu

Budeme postupovať podobne ako pri hľadaní priesečníku gule a mešu a valca a mešu. Nájdeme trojuholník, ktorý pretína kužeľ a budeme postupovať po

susedoch tohto trojuholníka a tak skladať krivku K zloženú z Bézierových kubík. Algoritmus na nájdenie priesečníku kužeľa a trojuholníka je v [Ebe02]

Algoritmus 6: Priesečníky dvoch Bézierových kriviek B_1, B_2

Data: $K_1 = (B_1)$, $K_2 = (B_2)$, $B_1.sub = 0$, $B_2.sub = 0$, $res = ()$
Result: poľe res

$cut_1 \leftarrow 0$, $cut_2 \leftarrow 0$, $i \leftarrow 0$, $j \leftarrow 0$
$all = false$
while *!all* **do**
    $enough \leftarrow false$
    while *!enough* **do**
        $cut \leftarrow true, i \leftarrow cut_1$
        for $i \leq cut_1 + 1 \ \wedge \ cut$ **do**
            $j \leftarrow 0$
            for $j < Count(K_2) \ \wedge \ cut$ **do**
                if $Rect(K_1[i]$ *sa pretína s* $Rect(K_2[j])$ **then**
                    Prerozdeľ K_1 a K_2
                end
            end
        end
        $i \leftarrow st_1$, $j \leftarrow 0$
        for $i \leq Count(K_1) \ \wedge \ cut$ **do**
            for $j < Count(K_2) \ \wedge \ cut$ **do**
                if $Rect(K_1[i]$ *sa pretína s* $Rect(K_2[j])$ **then**
                    Prerozdeľ K_1 a K_2
                end
            end
        end
        if *cut* **then**
            $enough \leftarrow true$, $all \leftarrow true$
        end
    end
end

Algoritmus 7: Prerozdeľ K_1 a K_2

Prerozdeľ K_1 **a** K_2:
$(B_1, B_2) \leftarrow Subdivide(K_1[i], \frac{1}{2})$
$Insert(K_1, B_1, i)$, $Insert(K_1, B_2, i+1)$
$(B_1, B_2) \leftarrow Subdivide(K_2[j], \frac{1}{2})$
$Insert(K_2, B_1, j)$, $Insert(K_2, B_2, j+1)$
$cut \leftarrow false$, $cut_1 = i$, $cut_2 = j$
if *strany* $Rect(K_1[i]$ *a stany* $Rect(K_2[j]) < \varepsilon$ **then**
    $res = res \cup (Compute(K_1, i), Compute(K_2, j))$
    $enough \leftarrow true$, $cut_1 \leftarrow cut_1 + 2$, $st_1 \leftarrow cut_1$
end

Kapitola 6

Globálna viditeľnosť

6.1 Rozdelenie Coonsových záplat pozdĺž krivky

Ako sme už povedali, geometrické objekty, ktoré premietame máme v rovine reprezentované ako plochy zložené z Coonsových záplat (Coonsove meše). Priesečníky telies máme reprezentované ako krivky, ktoré sú po častiach Bézierovými kubikami. Ich obraz v perspektívnom premietaní si tiež reprezentujeme ako postupnosť Bézierových kubík.

Aby sa objekty vykreslili správne, potrebujeme Coonsov meš m rozdeliť pozdĺž krivky K predstavujúcej priesečník objektov. Napr. ak sa v scéne pretnú dve gule, nájdeme Coonsove meše pre každú guľu, zostrojíme priesečník gúľ, meše a priesečník premietneme a v rovine rozdelíme premietnuté meše pozdĺž premietnutého priesečníku.

Coonsove meše, ktoré budeme používať majú hraničné krivky v tvare Bézierovej kubiky. Krivka K reprezentujúca priesečník dvoch telies je poskladaná z Bézierových kubík. Pri riešení prieniku Coonsových záplat a krivky K budeme teda hľadať priesečníky Bézierových kubík (stať 5.1).

Elementárne Coonsove plochy, z ktorých je Coonsov meš zložený budeme nazývať záplaty. Bézierove kubiky tvoriace krivku K budeme nazývať segmenty krivky K. Algoritmus riešiaci popísaný problém bude pracovať nasledovne:

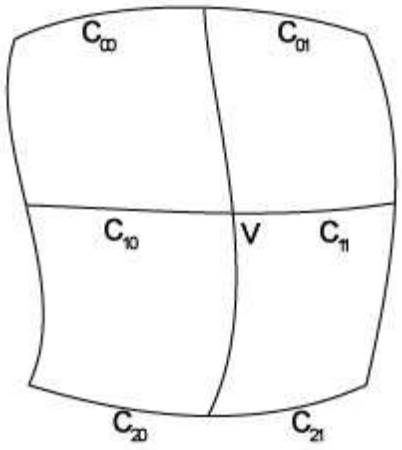
1. Nájdeme prieniky krivky K s hraničnými krivkami Coonsovho mešu m.
2. Zoberieme každú záplatu mešu m, ktorá sa preťala s krivkou K a skontrolujeme
   (a) počet prienikov hraničných kriviek záplaty a krivky K
   (b) počet segmentov ležiacich v záplate
3. Ak je počet prienikov hraničných kriviek záplaty a krivky K alebo počet segmentov ležiacich v záplate väčší ako dva, záplatu pre-

rozdelíme. Vzniknú tak štyri nové záplaty, pre ktoré musíme opäť počítať prieniky s krivkou K.

4. Ak je pre všetky záplaty počet prienikov hraničných kriviek záplaty s krivkou K a počet segmentov krivky K ležiacich v záplate správny, záplaty rozdelíme pozdĺž krivky K.

6.1.1 Prerozdelenie Coonsovej záplaty

Prerozdelením Coonsovej záplaty urobíme z jednej Coonsovej záplaty štyri menšie, ktoré spolu definujú rovnakú množinu bodov ako pôvodná Coonsova záplata. Prerozdelenie sa urobí tak, že sa prerozdelia hraničné Bézierove kubiky v bode $t = t_0$. Vyrátame bod $V = X(t_0, t_0)$, kde $X(u, v)$ je rovnica Coonsovej záplaty. Pomocou derivácii vo vrchole V nájdeme zostávajúce štyri hraničné krivky (obr. 6.1)



Obrázok 6.1: Prerozdelenie Coonsovej záplaty. Krivky C_{00}, C_{01}, C_{10}, C_{11}, C_{20}, C_{21} sú nové hraničné krivky v smere u. Podobne vznikli nové krivky aj v smere v. Vrchol $V = X(t_0, t_0)$.

Pre krivky $C_{00}, C_{01}, C_{20}, C_{21}$ poznáme ich riadiace polygóny, lebo vznikli prerozdelením Bézierových kriviek. Ukážeme si, ako nájsť riadiaci polygón krivky C_{10}.

Nech V_{00}, V_{01}, V_{02}, V_{03} sú riadiace vrcholy krivky C_{00} a V_{20}, V_{21}, V_{22}, V_{23} sú riadiace vrcholy krivky C_{20}. Riadiace vrcholy krivky C_{10} označme V_{10}, V_{11}, V_{12}, V_{13}. Platí:

$$
\begin{aligned}
V_{10} &= D_0(t_0) \\
V_{11} &= (1-t_0)(V_{10} + (V_{01} - V_{00})) + t_0(V_{10} + (V_{21} - V_{20})) \\
V_{12} &= (1-t_0)(V + (V_{02} - V_{03})) + t_0(V + (V_{22} - V_{23})) \\
V_{13} &= V
\end{aligned}
$$

Analogicky to platí aj pre ostatné novo vzniknuté hraničné krivky.

6.1.2 Prieniky Coonsových záplat s krivkou K

Krivka K, ako sme už povedali, je postupnosť za sebou idúcich Bézierových kubík. Pri hľadaní prienikov krivky K a Coonsových záplat budeme postupovať nasledovne.

Najprv zistíme, v ktorej záplate sa nachádza vrchol V_0, ktorý je prvým riadiacim vrcholom prvého segmentu krivky K. Potom nájdem prvý prienik krivky K s hraničnou krivkou záplaty. Zapíšem si údaje o prieniku. Záplate zväčším počítadlo prienikov. Podľa toho, v ktorom segmente sa krivka K preťala s hraničnou krivkou záplaty, zvýšim počítadlo segmentov nachádzajúcich sa v záplate. V hľadaní prienikov pokračujem v ďalšej záplate, ktorej hraničná krivka je rovnaká ako tá, s ktorou sa práve preťala krivka K.

6.1.3 Štruktúra Coonsovho mešu

Trochu si priblížime štruktúru Coonsovho mešu. Aby sme nemuseli počítať priesečníky jednej hraničnej krivky viackrát, meš obsahuje pole, v ktorom má uložené všetky hraničné Bézierove krivky. Coonsove záplaty neobsahujú samotné kubiky, ale len odkazy na ne.

Odkaz na hraničnú krivku obsahuje pointer na Bézierovu kubiku $B(t)$ plus dva parametre t_1 a t_2, ktoré predstavujú začiatok a koniec krivky. T.j. hraničnú krivku tvorí krivka $B(t)$, kde $t \in \langle t_0, t_1 \rangle$.

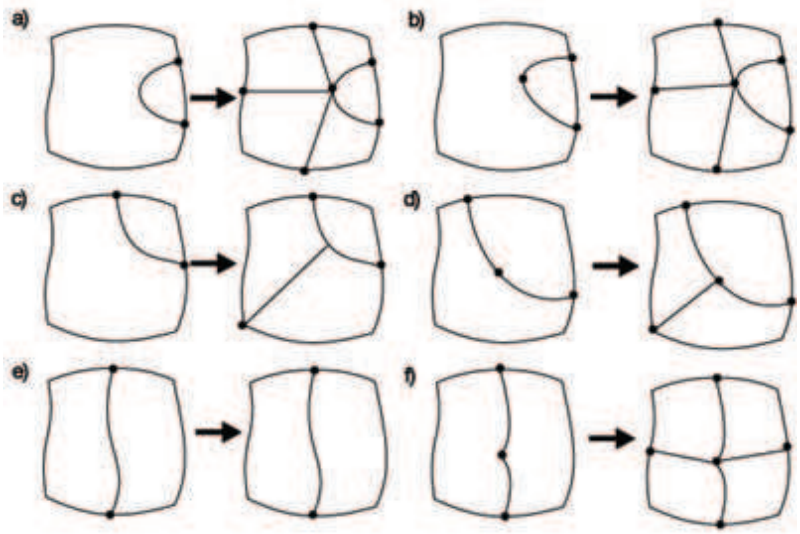
Táto reprezentácia je užitočná po prerozdelení Coonsovej záplaty, kedy sa nemusí vytvárať nová Bézierova kubika, ale sa len zaznamená jej počiatok a koniec. Vďaka tomu, že sa pointer na hraničnú krivku nezmení a zmenia sa len parametre t_1 a t_2, nestratí sa informácia o tom, ktoré záplaty sú susedné.

6.1.4 Prerozdelenie Coonsovej záplaty pozdĺž krivky K

Na obrázku 6.2 vidíme možné prieniky krivky K a Coonsovej záplaty po prerozdelení záplaty tak, aby spĺňali podmienky o počte prienikov (stať 6.1). Pre prípady a), c), e) platí, že počet segmentov krivky K v Coonsovej záplate je jedna. V prípadoch b), d), f) sa ležia v Coonsovej záplate dva segmenty krivky K.

Segmenty krivky K musia tvoriť hraničné krivky Coonsových záplat, lebo práve krivkou K vedie priesečník premietaných objektov. Na obráz-

ku 6.2 je ku každému prípadu aj prerozdelenie Coonsovej záplaty na nové Coonsove záplaty.



Obrázok 6.2: Coonsove záplaty v rôznych prienikoch s krivkou K a následné prerozdelenie záplat.

6.2 Globálna viditeľnosť Coonsových záplat

Pri riešení viditeľnosti Coonsových záplat použijeme rovnaký princíp ako pri riešení viditeľnosti trojuholníkových mešov. Najprv odstránime záplaty, ktoré sú zakryté inými objektmi a potom určíme poradie vykresľovania jednotlivých záplat, čím opäť využijeme jednoduchosť maliarovho algoritmu.

Oproti trojuholníkovým mešom to máme trochu zjednodušené v tom, že už nemusíme riešiť lokálnu viditeľnosť. Záplaty toho istého objektu sa totiž neprekrývajú. Pri objektoch, ktoré sú v rovine (priemetni) reprezentované ako Coonsove meše, sme lokálnu viditeľnosť riešili už pri premietaní.

Coonsovu záplatu označíme za neviditeľnú, ak ani jeden z jej riadiacich vrcholov nie je viditeľný. Inak je záplata viditeľná. Táto metóda nie je presná, ale je rýchla. Rovnakým spôsobom sme určovali aj viditeľnosť trojuholníkov v trojuholníkovom meši.

Vrchol V je viditeľný, ak bod P predstavujúci vzor vrcholu V v priestore[1] je viditeľný. A bod P je viditeľný, ak úsečka CP, kde C je stred premietania, nepretína žiaden objekt v priestore.

[1] $V = f(P)$, kde f je perspektívna projekcia.

Budeme teda testovať prieniky úsečiek s objektmi v priestore ("Ray Casting", čiže vysielanie lúča).

V nasledujúcich kapitolách budeme používať toto označenie:

- Vrchol V je vrchol v priemetni, pre ktorý zisťujeme, či je viditeľný, alebo neviditeľný.
- Bod P je bod v priestore, ktorého obraz je vrchol V v premietaní f.
- Bod C je stred premietania f.

6.2.1 Viditeľnosť bodu P vzhľadom na trojuholník

Je daný trojuholník $\triangle$ ležiaci v rovine π. Viditeľnosť bodu P vzhľadom na trojuholník sme už preberali v časti o mešoch (stať 4.1.1). Len pre pripomenutie si načrtneme postup.

Pri riešení viditeľnosti už máme objekty scény premietnuté. Preto najprv zistíme, či vrchol V sa nachádza v priemete trojuholníka $\triangle$. Ak áno, tak pomocou rovnice roviny π zistíme, či sa body P a C nachádzajú v tom istom polpriestore definovanom rovinou π, alebo nie. Ak áno, bod P je viditeľný, inak je neviditeľný.

6.2.2 Viditeľnosť bodu P vzhľadom na guľu

Je daná guľa g so stredom S a polomerom r.

Bod P je neviditeľný práve vtedy, ak je leží vo vnútri gule g, alebo ak je za guľou g. Preto najprv otestujeme vzdialenosť bodu P od stredu S. Ak je menšia ako r, bod P je neviditeľný.

Ak je vzdialenosť bodu P od stredu S väčšia ako r, vyrátame vzdialenosť d priamky CP od stredu S:

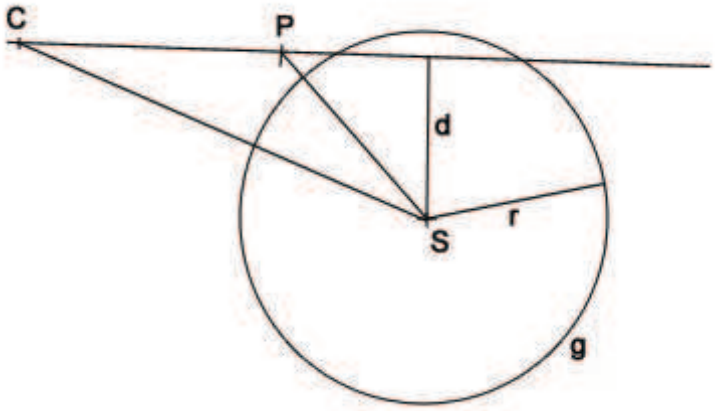
$$d = |CS| \cdot \sin \alpha$$

Uhol α je uhol, ktorý zvierajú vektory $\vec{u} = S - C$ a $\vec{v} = P - C$. Vieme, že $\sin^2 \alpha = 1 - \cos^2 \alpha$ a $\cos \alpha = \frac{\vec{v} \cdot \vec{u}}{|\vec{v}||\vec{u}|}$. A teda

$$d = |CS| \cdot \sqrt{1 - \left(\frac{\vec{v} \cdot \vec{u}}{|\vec{v}||\vec{u}|} \right)^2}$$

Ak $d > r$, bod P je viditeľný. Inak potrebujeme zistiť, či je bod P pred alebo za guľou g. To zistíme tak podľa uhla $\beta = \angle CPS$, ktorý vieme vyrátať pomocou vektorov $C - P$ a $S - P$. Ak je uhol $\beta > \frac{\pi}{2}$, bod P je pred guľou g a teda je viditeľný[2]. Inak je bod P zakrytý guľou g a teda neviditeľný (obr. 6.3).

[2] Je to naozaj tak, lebo možnosť, že sa bod P nachádza vo vnútri gule g sme už vylúčili.



Obrázok 6.3: Viditeľnosť bodu P vzhľadom na guľu g. S, r - stred a polomer gule g; C - stred premietania.

6.2.3 Viditeľnosť bodu P vzhľadom na valec

Je daný valec c so stredmi podstáv S_1 a S_2 a polomerom r.

Ak vzdialenosť d priamok CP a S_1S_2 je väčšia ako r, priamka CP sa nepreťala s valcom c a bod P je viditeľný.

Inak nájdeme prienik Q priamky CP a valca c. Ak $CQ \geq CP$, bod P je viditeľný, inak je bod P neviditeľný.

Ako nájsť bod Q? Ak sa stred premietania C nachádza medzi rovinami podstáv valca c, testujeme len prienik s plášťom. Inak testujeme aj prienik s podstavou ležiacou v rovine, ktorá je bližšie k bodu C.

Prienik s podstavou sa urobí tak, že sa nájde prienik s rovinou π, v ktorej podstava leží a potom sa testuje vzdialenosť tohto prieniku od stredu podstavy. Ak je menšia ako r, bod Q je prienikom podstavy a priamky CP.

Prienik s plášťom overíme tak, vypočítame vzdialenosť a bodu P od priamky S_1S_2. Ak bod P leží v tom istom polpriestore definovanom priamkou S_1S_2 a vektorom $\vec{u} = (P - C) \times (S_1 - S_2)$ a súčasne vzdialenosť a je väčšia ako r, bod P je viditeľný, inak je neviditeľný.

6.2.4 Viditeľnosť bodu P vzhľadom na kužeľ

Je daný kužeľ c so stredom podstavy S, polomerom podstavy r a vrcholom V_c. Opäť najprv nájdeme prienik Q priamky CP s kužeľom c a potom porovnáme vzdialenosti $|CP|$ a $|CQ|$.

Algoritmus na hľadanie prieniku kužeľa s priamkou je v [Ebe00].

6.3 Globálna viditeľnosť trojuholníkového mešu

Rovnako ako pri lokálnej viditeľnosti mešov, aj teraz budeme určovať viditeľnosť vrcholov jednotlivých trojuholníkov mešu. Na určenie ich viditeľnosti použijeme rovnaký postup ako pri určovaní viditeľnosti vrcholov v Coonsoných záplatách (stať 6.2).

6.4 Určenie poradia vykresľovania

V tejto časti za budeme zaoberať určením poradia vykresľovania tak, aby sa záplaty a trojuholníky, ktoré sú prekryté inými objektmi, vykreslili pred týmito objektmi. Toto je princíp maliarovho algoritmu. V predchádzajúcich kapitolách sme trojuholníky a Coonsove záplaty rozdeľovali, aby sa nepretínali. Vďaka tomu môžeme princíp maliarovho algoritmu použiť.

Použijeme algoritmus veľmi podobný tomu, ktorý sme použili pri určovaní poradia vykresľovania mešu (stať 4.1.2). Premietnutú 2D scénu si rozdelíme do obálok. Pod obálkou rozumieme nejakú oblasť scény, napr. obdĺžnikovú. Nájdeme príslušnosť jednotlivých trojuholníkov a Coonsových záplat do týchto obálok pomocou ich vrcholov, pre ktoré nájsť príslušnosť do obálok nie je problém.

Pre jednoduchosť nazvime elementmi trojuholníky a Coonsove záplaty, ktoré nám v priemetni reprezentujú zobrazené objekty.

Pre každý element zistíme, s ktorými ďalšími elementmi sa pretína. Testovať budeme len elementy patriace do jednej a tej istej obálky. Ak sa totiž elementy nenachádzajú v rovnakých obálkach, tak sa nemôžu pretnúť. Týmto eliminujeme počet zbytočných testov a algoritmus tak urýchlime.

Predpokladajme, že máme dva elementy, ktoré sa preťali. To znamená, že časti objektov v 3D scéne, ktoré prislúchajú týmto elementom, sa prekrývajú. Opäť si zadefinujeme binárnu reláciu $>$. Pre elementy e_1 a e_2 vzťah $e_1 > e_2$ znamená, že časť objektu o_1 v 3D prislúchajúca elementu e_1 je bližšie k stredu premietania ako časť objektu o_2 v 3D prislúchajúca elementu e_2 a teda element e_1 sa bude vykresľovať po elemente e_2, aby ho prekryl.

Nad množinou všetkých elementov sa nám vďaka relácii $>$ vytvoril orientovaný graf. Na ňom urobíme rovnaký algoritmus ako v stati 4.1.2. Takto rozhodneme o poradí vykresľovania elementov.

Pre zrýchlenie algoritmu môžeme takúto modifikáciu. Najskôr si každý objekt v 3D scéne obalíme 3D obálkou, ktorá bude konvexná. Pre každú dvojicu objektov zistíme, či sa ich obálky preťali. Ak nie, tak poradie vykresľovania týchto dvoch objektov určíme na úrovni celého objektu a nemusíme ju určovať na úrovni elementov. Ak sa obálky preťali, viditeľnosť riešime na úrovni elementov.

Kapitola 7

Záver

V našej práci sme sa pokúšali ponúknuť návod, ako zobrazovať 3D scénu použitím perspektívneho premietacia z $\mathbb{R}^3$ do $\mathbb{R}^2$ tak, aby sa zachovalo čo najviac vektorovej informácie a výsledný obrázok bol škálovateľný. Využili sme pri tom viacero aproximačných metód na reprezentáciu objektov, aby sa dali riešiť problémy viditeľnosti a pretínania objektov. Použili sme niektoré postupy urýchľujúce spracovávanie scény ako napr. rozdelenie scény do obálok.

Metódy opisované v tejto práci sme implementovali. Výsledky tejto implementácie si ukážeme v nasledujúcej kapitole.

V práci je vidieť to, že nie vždy ide všetko. Pre nás bolo dôležité zachovanie vektorovej informácie a preto kvalita obrázkov vygenerovaných našou implementáciou nie je taká ako kvalita obrázkov, ktoré vznikli použitím inej metódy zobrazovania, ktorej výsledkom sú bitmapy (napr. raytracing).

7.1 Výsledky

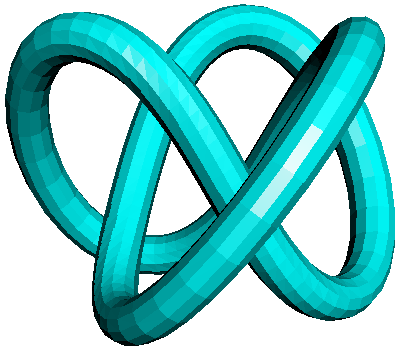
Na nasledujúcich obrázkoch sú výsledky metód opisovaných v našej práci. Na obrázku 7.1 vidieť uzol, reprezentovaný ako meš. Normály neboli zadané, preto sa použili normály rovín, v ktorých dané trojuholníky mešu ležali. Obrázok uzlu ukazuje hlavne výsledok počítania viditeľnosti.

Na obrázku 7.2 je vykreslený čajník. V tomto prípade boli zadané aj normály vo vrcholoch mešu. Preto sa povrch čajníka javí ako hladký, aj keď sú tam určité nedostatky spôsobené Gouraudovým tieňovaním.

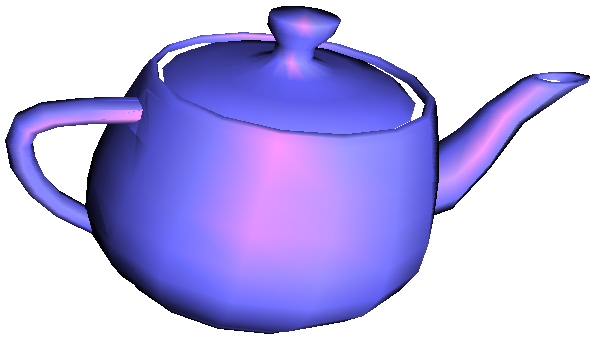
Obrázky 7.3, 7.4 a 7.5 demonštrujú použitie tieňovania pomocou Coonsových záplat. Vďaka tomu aj po zväčšení obrázkov okraje objektov zostávajú hladké, nie ako pri mešoch, keď sú vidieť ostré prechody medzi hranami (napr. obrázok čajníka).

Na obrázku 7.6 vidno výsledok metódy počítajúcej priesečníky objektov. Tri gule reprezentované mešom sa navzájom pretínajú. Steny mešov sú

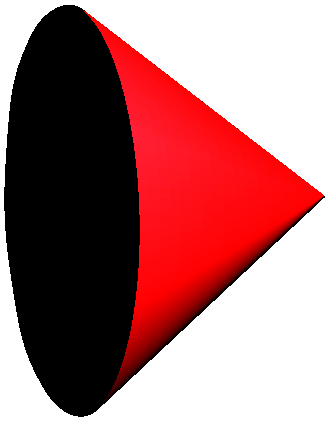
rezané podľa ich prienikov s inými stenami.



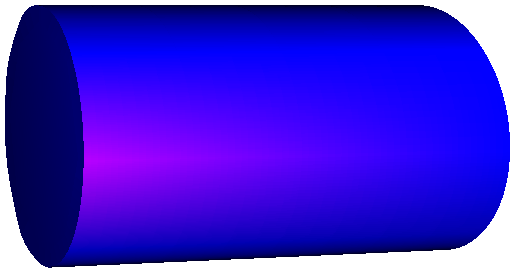
Obrázok 7.1: Uzol (meš). Je tu vidieť riešenie lokálnej viditeľnosti mešu. 2880 trojuholníkov. Čas riešenie viditeľnosti: $0,801$ sekundy.



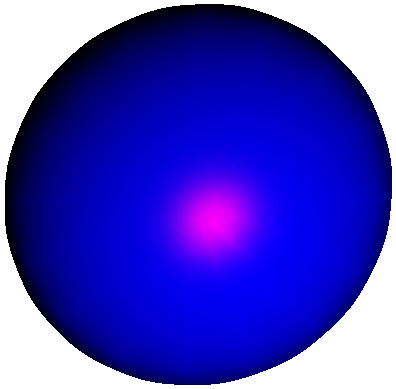
Obrázok 7.2: Čajník (meš). Oproti uzlu má definované normálu vo vrcholoch, preto vyzerá byť hladký. 1055 trojuholníkov, viditeľnosť: 0,289 sekundy



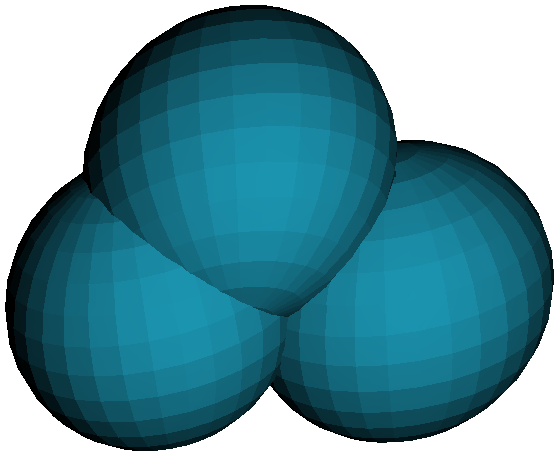
Obrázok 7.3: Kužeľ. Podstava je definovaná pomocou Coonsových plôch, preto aj po zväčšení vyzerajú jej okraj hladko. Viditeľnosť: 0,001 sekundy.



Obrázok 7.4: Valec. Podstavy sú definované pomocou Coonsových plôch ako u kužeľa.



Obrázok 7.5: Guľa. Opäť reprezentácia Coonsovými záplatami zabezpečuje hladký vzhľad okraju gule.



Obrázok 7.6: Tri gule (3 meše). Ilustrácia počítania prienikov mešov. Trojuholníky v oblasti prieniku sa delia na viaceré.

Literatúra

[Cas] Bill Casselman. *Mathematical Illustrations - A Manual of Geometry And PostScript*. World Wide Web, `http://www.math.ubc.ca/~cass/graphics/manual/`.

[Cha99] Pavel Chalmovianský. *Geometrické interpolácie*. World Wide Web, `http://fractal.dam.fmph.uniba.sk/~chalmo/lecture.html`, 1999.

[Ebe00] D. Eberly. *Intersection of a Line and c Cone*. World Wide Web, `http://www.geometrictools.com`, 2000.

[Ebe02] D. Eberly. *Intersection of a Triangle and a Cone*. World Wide Web, `http://www.geometrictools.com`, 2002.

[Eri00] Jeff Erickson. Polygon triangulation. In *CS 373: Combinatorial Algorithms*. World Wide Web, `http://compgeom.cs.uiuc.edu/~jeffe/teaching/373/notes/x07-polytri.pdf`, 2000.

[GC90] Elber Gershon and Elaine Cohen. Hidden curve removal for free form surfaces. *Computer Graphics*, 24(4), 1990.

[Goo98] Amy Gooch. Interactive non-photorealistic technical illustration. Master's thesis, The University of Utah, 1998.

[Gre02] G. Greiner. *Geometric Modeling*. World Wide Web, `http://atrey.karlin.mff.cuni.cz/projekty/vrr/doc/grafika/geometric%20modelling.pdf`, 2002.

[Inc97] Adobe Systems Incorporated. Smooth shading. Technical Report 5600, Adobe Developers Association, 1997.

[Inc99] Adobe Systems Incorporated, editor. *PostScript language reference manual third edition*. Addison-Wesley Publishing Company, 1999.

[Žár98] J. Žára. *Moderní Počítačová Grafika*. Computer Press, 1998.

[Sed03a] Thomas W. Sederberg. Bézier curves. In *Computer aided geometric design*. `http://tom.cs.byu.edu/~557/cagd.pdf`, 2003.

[Sed03b] Thomas W. Sederberg. Curve intersection. In *Computer aided geometric design*. `http://tom.cs.byu.edu/~557/cagd.pdf`, 2003.

[Tul02] S. Tuleja. *Kuželosečky*. World Wide Web, `http://vscience.euweb.cz/materialy/kuzelosecky/index.htm`, 2002.

[X3D05] *ISO/IEC 19775:200x, Extensible 3D (X3D)*. `http://www.web3d.org/x3d/specifications/X3DPublicSpecifications.zip`, 2005.