Alte tehnici

Mihai-Sorin Stupariu

Sem. I, 2024 - 2025

Bump mapping. Normal mapping

Shadow mapping

Curbe și suprafețe Bézier

Problematizare

- Reprezentarea cât mai eficientă a unor suprafețe cu rugozitate mare.

Problematizare

- Reprezentarea cât mai eficientă a unor suprafețe cu rugozitate mare.
- Articol de referință: [Blinn, 1978]
  Alte referințe: [Bruneton & Neyret, 2012], [Kautz et al., 2001], [Heidrich & Seidel, 1999]

Problematizare

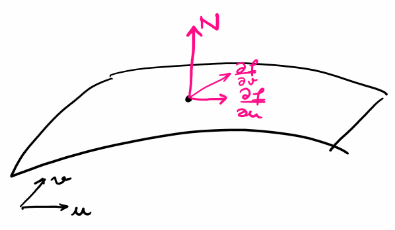
- Reprezentarea cât mai eficientă a unor suprafețe cu rugozitate mare.
- Articol de referință: [Blinn, 1978]
  Alte referințe: [Bruneton & Neyret, 2012], [Kautz et al., 2001], [Heidrich & Seidel, 1999]
- **Principiu: nu este necesar ca suprafețele propriu-zise să aibă o geometrie complicată, este suficient să fie controlat modul în care este reflectată lumina. În particular, este suficient să fie perturbați vectorii normali asociați vârfurilor.**

Problematizare

- Reprezentarea cât mai eficientă a unor suprafețe cu rugozitate mare.
- Articol de referință: [Blinn, 1978]
  Alte referințe: [Bruneton & Neyret, 2012], [Kautz et al., 2001], [Heidrich & Seidel, 1999]
- **Principiu: nu este necesar ca suprafețele propriu-zise să aibă o geometrie complicată, este suficient să fie controlat modul în care este reflectată lumina. În particular, este suficient să fie perturbați vectorii normali asociați vârfurilor.**
- Cum controlăm / implementăm modificarea vectorilor normali (teoretic / implementare)? Variante: (i) *(procedural) bump mapping*, (ii) *normal mapping*.

Bump mapping. Context

Fie $U \subset \mathbb{R}^2$ și $f : U \to \mathbb{R}^3$, $(u, v) \mapsto f(u, v)$ o suprafață parametrizată.



Conform teoriei, vectorul normal $\mathbf{n}$ la suprafață într-un punct $f(u_0, v_0)$ se calculează

$$\mathbf{n} = \frac{N}{\|N\|}, \quad N = \frac{\partial f}{\partial u}(u_0, v_0) \times \frac{\partial f}{\partial v}(u_0, v_0), \text{ pp. } N \neq 0.$$

Vectorii $\frac{\partial f}{\partial u}(u_0, v_0)$ și $\frac{\partial f}{\partial v}(u_0, v_0)$ generează planul tangent la suprafață în punctul $f(u_0, v_0)$.

Bump mapping. Ideea de lucru

1. **Introducerea unei funcții de perturbare**. Se consideră o funcție $\varphi : U \to \mathbb{R}$ care realizează o "distorsionare" cu "valori mici" ($\varphi \simeq 0$) în direcția normalei (în fiecare punct):

$$\tilde{f} = f + \varphi\,\mathbf{n}.$$

Bump mapping. Ideea de lucru

1. **Introducerea unei funcții de perturbare**. Se consideră o funcție $\varphi : U \to \mathbb{R}$ care realizează o "distorsionare" cu "valori mici" ($\varphi \simeq 0$) în direcția normalei (în fiecare punct):

$$\tilde{f} = f + \varphi\,\mathbf{n}.$$

2. **Estimarea vectorilor tangenți după distorsionare**.

$$\frac{\partial \tilde{f}}{\partial u} = \frac{\partial f}{\partial u} + \frac{\partial \varphi}{\partial u}\,\mathbf{n} + \varphi\,\frac{\partial \mathbf{n}}{\partial u} \overset{\varphi\simeq 0}{\simeq} \frac{\partial f}{\partial u} + \frac{\partial \varphi}{\partial u}\,\mathbf{n}.$$

$$\frac{\partial \tilde{f}}{\partial v} = \frac{\partial f}{\partial v} + \frac{\partial \varphi}{\partial v}\,\mathbf{n} + \varphi\,\frac{\partial \mathbf{n}}{\partial v} \overset{\varphi\simeq 0}{\simeq} \frac{\partial f}{\partial v} + \frac{\partial \varphi}{\partial v}\,\mathbf{n}.$$

Bump mapping. Ideea de lucru

1. **Introducerea unei funcții de perturbare**. Se consideră o funcție $\varphi : U \to \mathbb{R}$ care realizează o "distorsionare" cu "valori mici" ($\varphi \simeq 0$) în direcția normalei (în fiecare punct):
$$\tilde{f} = f + \varphi\,\mathbf{n}.$$
2. **Estimarea vectorilor tangenți după distorsionare**.
$$\frac{\partial \tilde{f}}{\partial u} = \frac{\partial f}{\partial u} + \frac{\partial \varphi}{\partial u}\,\mathbf{n} + \varphi\,\frac{\partial \mathbf{n}}{\partial u} \overset{\varphi \simeq 0}{\simeq} \frac{\partial f}{\partial u} + \frac{\partial \varphi}{\partial u}\,\mathbf{n}.$$
$$\frac{\partial \tilde{f}}{\partial v} = \frac{\partial f}{\partial v} + \frac{\partial \varphi}{\partial v}\,\mathbf{n} + \varphi\,\frac{\partial \mathbf{n}}{\partial v} \overset{\varphi \simeq 0}{\simeq} \frac{\partial f}{\partial v} + \frac{\partial \varphi}{\partial v}\,\mathbf{n}.$$
3. **Estimarea vectorului normal după distorsionare**.
$$\tilde{N} = \frac{\partial \tilde{f}}{\partial u} \times \frac{\partial \tilde{f}}{\partial v} \simeq \left(\frac{\partial f}{\partial u} + \frac{\partial \varphi}{\partial u}\,\mathbf{n}\right) \times \left(\frac{\partial f}{\partial v} + \frac{\partial \varphi}{\partial v}\,\mathbf{n}\right) =$$
$$= \frac{\partial f}{\partial u} \times \frac{\partial f}{\partial v} + \frac{\partial f}{\partial u} \times \frac{\partial \varphi}{\partial v}\,\mathbf{n} + \frac{\partial \varphi}{\partial u}\,\mathbf{n} \times \frac{\partial f}{\partial v} + \frac{\partial \varphi}{\partial u}\,\mathbf{n} \times \frac{\partial \varphi}{\partial v}\,\mathbf{n} = N + D.$$
Vectorul $D = \frac{\partial f}{\partial u} \times \frac{\partial \varphi}{\partial v}\,\mathbf{n} + \frac{\partial \varphi}{\partial u}\,\mathbf{n} \times \frac{\partial f}{\partial v}$ s.n. *displacement vector*.

De la teorie la implementare. *Normal mapping*

- **Inițial:** f (obiectul) și φ (*bump function*) definite pe același domeniu ([Blinn, 1978])

De la teorie la implementare. *Normal mapping*

- **Inițial:** f (obiectul) și φ (*bump function*) definite pe același domeniu ([Blinn, 1978])
- **Idee:** “separarea” *bump function* / a normalelor de suprafața pe care este randată

De la teorie la implementare. *Normal mapping*

- **Inițial:** f (obiectul) și φ (*bump function*) definite pe același domeniu ([Blinn, 1978])
- **Idee:** "separarea" *bump function* / a normalelor de suprafața pe care este randată
- **Practic:** cum se rețin normalele? - folosirea texturilor Tutoriale: learnopengl, opengl-tutorial

Normal mapping. Principii

1. Un vector normal are trei componente, ca o imagine color de tip RGB. Astfel, o colecție de vectori normali pentru o primitivă poate fi stocată într-un fișier de tip imagine (*normal map*), care conține vectorii normali. Practic: pot fi utilizate texturile, transmise către shader folosind variabile de tip *uniform sampler*.

Normal mapping. Principii

1. Un vector normal are trei componente, ca o imagine color de tip RGB. Astfel, o colecție de vectori normali pentru o primitivă poate fi stocată într-un fișier de tip imagine (*normal map*), care conține vectorii normali. Practic: pot fi utilizate texturile, transmise către shader folosind variabile de tip *uniform sampler*.
2. Într-un fișier de tip *normal map* nu sunt reținute normalele propriu-zise, ci deviația acestora față de direcția "teoretică" a normalelor pentru primitiva randată. Pe componentele x, y sunt variații față de verticala, iar pe componenta z este valoarea 1. Două consecințe: (i) aspect vizual cu tonuri de albastru, (ii) este necesară raportarea la **geometria primitivei**. Aceasta implică o **schimbare de coordonate**.

Normal mapping. 1. RGB $\leftrightarrow$ Normal

Componentele unui cod RGB (de tip `float`) au valori în intervalul $[0.0, 1.0]$. Componentele unui vector normal au valori în intervalul $[-1.0, 1.0]$. Este necesară aplicarea unor transformări de corespondență între cele două intervale.

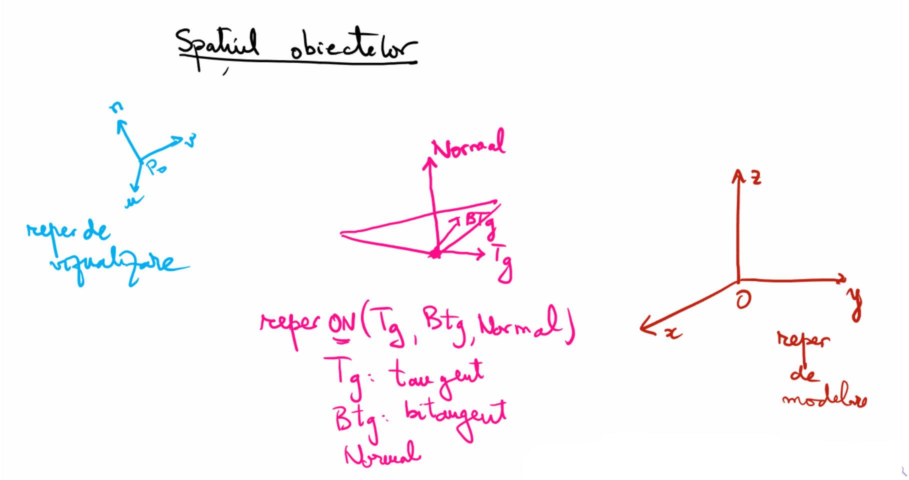
$$RGB = 0.5 Normal + 0.5$$

$$Normal_{aux} = 2RGB - 1, \; Normal = \frac{Normal_{aux}}{\|Normal_{aux}\|}.$$

De exemplu, codul RGB $(0.5, 0.35, 0.3)$ conduce la vectorul $Normal_{aux} = (0.0, -0.3, -0.4)$, iar $Normal = (0.0, -0.6, -0.8)$.

Normal mapping. 2. Schimbare de coordonate

Ideea de bază: normalele utilizate reprezintă "devierea" față de vectorul $(0, 0, 1)$, i.e. planul procesat la momentul respectiv este cel orizontal. Prin urmare, este necesară o schimbare de coordonate (un nou reper), relevante pentru vârfurile considerate.



Normal mapping. 2. Schimbare de coordonate

Folosind vectorii tangent, bitangent, normal asociați unui vârf (se presupune că au normă 1 și sunt perpendiculari 2×2) se construiește o matrice ortogonală (matricea TBN, care are acești vectori pe linii), apoi se poate calcula normala "perturbată" pentru un vârf dat. Există diverse moduri în care poate fi utilizată această matrice.

Height mapping

Sunt utilizate *height maps* pentru a perturba pozițiile vârfurilor.
Practic: o imagine (textură) care conține elevații/altitudini (*heights*), de obicei o scară de gri, cu valori între 0 și 1.
Pot fi utile pentru generarea terenurilor.
Exemplu

Principiu și etape

- **Principiu:** Obiectele care nu pot fi văzute de sursa de lumină sunt în umbră. Informația referitoare la obiectele vizibile din poziția sursei de lumină este stocată în buffer-ul de adâncime (*depth buffer*).

Principiu și etape

- **Principiu:** Obiectele care nu pot fi văzute de sursa de lumină sunt în umbră. Informația referitoare la obiectele vizibile din poziția sursei de lumină este stocată în buffer-ul de adâncime (*depth buffer*).
- **Etapa 1.** Scena este "desenată" din poziția sursei de lumină. Buffer-ul de adâncime (*depth buffer*) conține, pentru fiecare pixel, distanța dintre sursa de lumină și cel mai apropiat obiect (pe baza *Hidden Surface Removal algorithm*). Informația este stocată într-un buffer dedicat sau într-o textură (*shadow buffer / shadow texture*).

Principiu și etape

- **Principiu:** Obiectele care nu pot fi văzute de sursa de lumină sunt în umbră. Informația referitoare la obiectele vizibile din poziția sursei de lumină este stocată în buffer-ul de adâncime (*depth buffer*).
- **Etapa 1.** Scena este "desenată" din poziția sursei de lumină. Buffer-ul de adâncime (*depth buffer*) conține, pentru fiecare pixel, distanța dintre sursa de lumină și cel mai apropiat obiect (pe baza *Hidden Surface Removal algorithm*). Informația este stocată într-un buffer dedicat sau într-o textură (*shadow buffer* / *shadow texture*).
- **Etapa 2.** Redarea propriu-zisă a scenei. Pentru fiecare pixel p se extrage valoarea z_p din buffer-ul/textura dedicat(ă) umbrei. Dacă pentru obiectul desenat (corespunzător pixelului) distanța până la sursa de lumină este mai mare decât z_p, atunci obiectul respectiv este în umbră și este desenat cu culoarea umbrei (eventual se aplică termenul ambiental al modelului de iluminare).

Shadow mapping - etapa 1

- Camera este mutată în poziția sursei de lumină - este necesară o matrice de vizualizare-proiecție adecvată (`ShadowMVP`). Poziția observatorului: sursa de lumină, este ales un punct de referință adecvat.

Shadow mapping - etapa 1

- Camera este mutată în poziția sursei de lumină - este necesară o matrice de vizualizare-proiecție adecvată (`ShadowMVP`). Poziția observatorului: sursa de lumină, este ales un punct de referință adecvat.
- Se are în vedere copierea buffer-ului de adâncime într-o textură (variante folosind `glCopyTexImage2D( )` sau `glFrameBufferTexture( )` - în acest din urmă caz nu este necesară copierea buffer-ului într-o textură, existând una atașată). Valorile din buffer-ul de adâncime sunt accesate în funcția `glTexImage2D`, folosind pentru format opțiunea `GL_DEPTH_COMPONENT`.

Shadow mapping - etapa 1

- Camera este mutată în poziția sursei de lumină - este necesară o matrice de vizualizare-proiecție adecvată (`ShadowMVP`). Poziția observatorului: sursa de lumină, este ales un punct de referință adecvat.
- Se are în vedere copierea buffer-ului de adâncime într-o textură (variante folosind `glCopyTexImage2D( )` sau `glFrameBufferTexture( )` - în acest din urmă caz nu este necesară copierea buffer-ului într-o textură, existând una atașată). Valorile din buffer-ul de adâncime sunt accesate în funcția `glTexImage2D`, folosind pentru format opțiunea `GL_DEPTH_COMPONENT`.
- Este apelată o funcție de desenare (e.g. `glDrawArrays()`), fiind activat testul de adâncime, dar fiind apelat `glDrawBuffer(GL_NONE)`. În shader-ul de vârfuri: aplicată transformarea de mai sus. În shader-ul de fragment: nimic. După desenare, se revine la varianta implicită pentru buffer-ul de cadru (funcții adecvate, de exemplu `glDrawBuffer(GL_FRONT)`).

Shadow mapping - etapa 2

- Pentru fiecare vârf procesat trebuie stabilit dacă este în umbră sau nu. În shader-ul de vârfuri: aplicată o transformare folosind o matrice adecvată (`ShadowMVP2`, obținută trecând de la $[-1, 1]$ la $[0, 1]$), rezultatul fiind transmis în shader-ul de fragment. În acesta din urmă, folosind funcția `textureProj` din GLSL se stabilește dacă un pixel este sau nu în umbră.

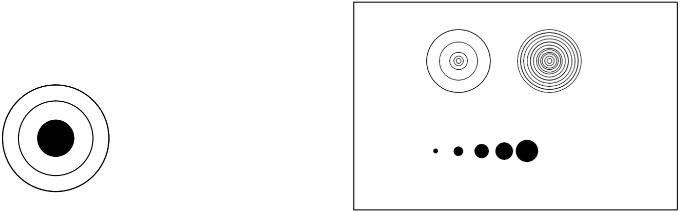
Shadow mapping - etapa 2

- Pentru fiecare vârf procesat trebuie stabilit dacă este în umbră sau nu. În shader-ul de vârfuri: aplicată o transformare folosind o matrice adecvată (`ShadowMVP2`, obținută trecând de la $[-1, 1]$ la $[0, 1]$), rezultatul fiind transmis în shader-ul de fragment. În acesta din urmă, folosind funcția `textureProj` din GLSL se stabilește dacă un pixel este sau nu în umbră.
- În programul principal sunt realizați pașii obișnuiți pentru desenare. În plus, este utilizată o variabilă uniformă de tip *sampler*, numită `sampler2DShadow`, care este atașată unei texturi de tip umbră.

Shadow mapping - etapa 2

- Pentru fiecare vârf procesat trebuie stabilit dacă este în umbră sau nu. În shader-ul de vârfuri: aplicată o transformare folosind o matrice adecvată (`ShadowMVP2`, obținută trecând de la $[-1, 1]$ la $[0, 1]$), rezultatul fiind transmis în shader-ul de fragment. În acesta din urmă, folosind funcția `textureProj` din GLSL se stabilește dacă un pixel este sau nu în umbră.
- În programul principal sunt realizați pașii obișnuiți pentru desenare. În plus, este utilizată o variabilă uniformă de tip *sampler*, numită `sampler2DShadow`, care este atașată unei texturi de tip umbră.
- *Shadow mapping* poate produce artefacte, există diverse îmbunătățiri

Motivație: cum reprezentăm elementele grafice?



Grafică vectorială și grafică rasterială

Comentarii:

(i) Fonturile sunt de fapt elemente grafice.

(ii) În proiectare este nevoie de forme cât mai variate, fie la nivel de schiță, fie într-un stadiu mai avansat de proiectare.

Scop: Cum generăm elementele de grafică vectorială? (**Curbe Bézier**).

Mecanism

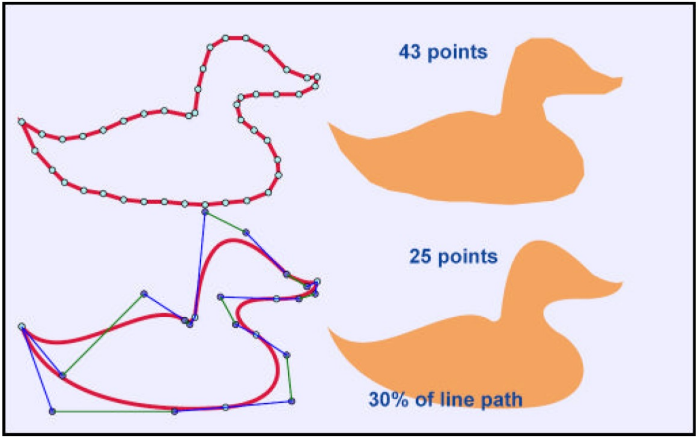
Input: O mulțime de puncte (poligon de control)
Output: Curba reprezentată

Curbe de interpolare:
Exemplu

Curbe Bézier:
https://javascript.info/bezier-curve;
https://www.jasondavies.com/animated-bezier/

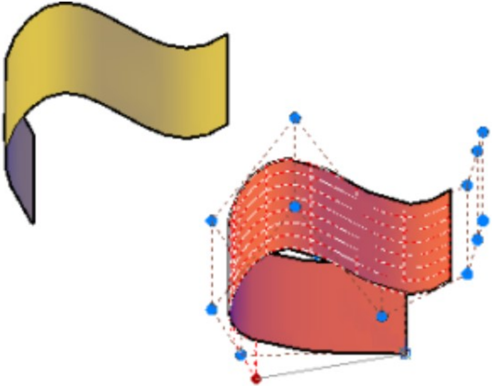
Mecanism

Input: O mulțime de puncte (poligon de control)
Output: Curba reprezentată



Sursa: Duce et al, SVG tutorial

Același principiu funcționează și pentru suprafețe



Sursa: Knowledge Autodesk

Generalități

- Dat un poligon de control $(\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_n)$, putem construi o curbă polinomială care să interpoleze aceste puncte.
- Unele proprietăți ale curbelor de interpolare (de exemplu, faptul că nu sunt incluse în acoperirea convexă a punctelor poligonului de control) fac ca acestea să nu fie practice în aplicații legate de grafica pe calculator.
- În anii '60, independent unul de celălalt, Paul de Casteljau și Pierre Bézier au investigat curbele asociate poligoanelor de control dintr-o altă perspectivă. Chiar dacă proprietatea de interpolare nu este verificată, sunt alte proprietăți geometrice remarcabile care s-au dovedit a fi foarte utile în inginerie și, ulterior, în CAGD: curbele Bézier (sau, mai precis, reprezentarea Bézier a curbelor polinomiale). La fel ca și curbele de interpolare, curbele Bézier pot fi construite folosind fie metode de natură geometrică (algoritmul de Casteljau), fie utilizând un aparat algebric (forma Bernstein).

Algoritmul de Casteljau pentru cazul $n = 2$

Fie $\mathbf{b}_0$, $\mathbf{b}_1$ și $\mathbf{b}_2$ trei puncte necoliniare. Pentru $t \in \mathbb{R}$ se construiesc punctele

$$\begin{aligned} \mathbf{b}_0^1(t) &= (1-t)\mathbf{b}_0 + t\mathbf{b}_1, \\ \mathbf{b}_1^1(t) &= (1-t)\mathbf{b}_1 + t\mathbf{b}_2, \\ \mathbf{b}_0^2(t) &= (1-t)\mathbf{b}_0^1(t) + t\mathbf{b}_1^1(t). \end{aligned}$$

Punctul $\mathbf{b}_0^2(t)$ descrie, când t variază în $\mathbb{R}$, o parabolă, mai precis parabola care trece prin punctele $\mathbf{b}_0$ și $\mathbf{b}_2$ și ale cărei tangente în aceste puncte sunt dreptele $\mathbf{b}_0\mathbf{b}_1$, respectiv $\mathbf{b}_2\mathbf{b}_1$. Pentru $t \in [0,1]$ se obține arcul acestei parabole care unește punctele $\mathbf{b}_0$ și $\mathbf{b}_2$.

Punctele intermediare pot fi scrise într-un tablou triunghiular, numit **schemă de Casteljau**. Considerăm, de exemplu, $n = 2$ și fixăm $t_0 \in [0,1]$. Schema de Casteljau corespunzătoare are forma

$$\begin{array}{lll} \mathbf{b}_0 & & \\ \mathbf{b}_1 & \mathbf{b}_0^1(t_0) & \\ \mathbf{b}_2 & \mathbf{b}_1^1(t_0) & \mathbf{b}_0^2(t_0) \end{array} \qquad (1)$$

Exemplu

Considerăm punctele

$$\mathbf{b}_0 = (0, 6), \quad \mathbf{b}_1 = (6, 6), \quad \mathbf{b}_2 = (6, 0).$$

Pentru $t = \frac{1}{3}$ avem

$$\mathbf{b}_0^1\left(\frac{1}{3}\right) = \frac{2}{3}\mathbf{b}_0 + \frac{1}{3}\mathbf{b}_1 = (2, 6),$$
$$\mathbf{b}_1^1\left(\frac{1}{3}\right) = \frac{2}{3}\mathbf{b}_1 + \frac{1}{3}\mathbf{b}_2 = (6, 4),$$
$$\mathbf{b}_0^2\left(\frac{1}{3}\right) = \frac{2}{3}\mathbf{b}_0^1 + \frac{1}{3}\mathbf{b}_1^1 = \left(\frac{10}{3}, \frac{16}{3}\right).$$

Schema de Casteljau asociată este

$$\begin{array}{lll} (0, 6) & & \\ (6, 6) & (2, 6) & \\ (6, 0) & (6, 4) & \left(\frac{10}{3}, \frac{16}{3}\right). \end{array}$$

Algoritmul de Casteljau, forma generală

Fie $\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$. Pentru $t \in \mathbb{R}$ se notează $\mathbf{b}_i^0(t) := \mathbf{b}_i$ $(i = 0, \ldots, n)$ și se definesc punctele

$$\mathbf{b}_i^r(t) := (1-t)\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t), \quad \begin{cases} r = 1, \ldots, n \\ i = 0, \ldots, n-r \end{cases} \tag{2}$$

Punctul $\mathbf{b}_0^n(t)$ descrie, când t variază, o curbă, notată cu $\mathbf{b}^n$. Punctele $\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n$ se numesc **puncte de control** ale curbei $\mathbf{b}^n$, iar poligonul determinat de acestea se numește **poligon de control**.

Analog cazului $n = 2$, punctele intermediare pot fi scrise într-un tablou triunghiular, numit **schemă de Casteljau**.

Exemplu

Considerăm punctele

$$\mathbf{b}_0 = (1, -2), \quad \mathbf{b}_1 = (3, 2), \quad \mathbf{b}_2 = (3, -2), \quad \mathbf{b}_3 = (-3, -2).$$

Schema de Casteljau corespunzătoare acestor puncte și valorii $t_0 = \frac{1}{2}$ a parametrului este

$$\begin{array}{llll} (1, -2) & & & \\ (3, 2) & (2, 0) & & \\ (3, -2) & (3, 0) & (\frac{5}{2}, 0) & \\ (-3, -2) & (0, -2) & (\frac{3}{2}, -1) & (2, -\frac{1}{2}). \end{array}$$

Varianta algebrică ($n = 2$)

Date $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$, prin calcul direct se obține

$$\mathbf{b}_0^2(t) = (1-t)^2\mathbf{b}_0 + 2t(1-t)\mathbf{b}_1 + t^2\mathbf{b}_2.$$

Polinoamele care apar în această scriere sunt **polinoamele Bernstein de grad 2**

$$B_0^2(t) = (1-t)^2, \quad B_1^2(t) = 2t(1-t), \quad B_2^2(t) = t^2,$$

deci

$$\mathbf{b}_0^2(t) = \sum_{i=0}^{2} B_i^2(t)\mathbf{b}_i.$$

Comentarii

- Polinoamele Bernstein de gradul 2 formează o **bază** în spațiul vectorial al polinoamelor de grad mai mic sau egal cu 2, deci **orice** curbă parametrizată polinomial (cu grad ≤ 2) poate fi scrisă folosind polinoame Bernstein.

Comentarii

- Polinoamele Bernstein de gradul 2 formează o **bază** în spațiul vectorial al polinoamelor de grad mai mic sau egal cu 2, deci **orice** curbă parametrizată polinomial (cu grad ≤ 2) poate fi scrisă folosind polinoame Bernstein.
- De fapt: curbele polinomiale de grad mai mic sau egal cu 2 sunt curbele construite folosind algoritmul de Casteljau (sau folosind forma Bernstein) pentru $n = 2$.

Forma algebrică a curbelor Bézier - cazul general

Pentru $n \in \mathbb{N}$ fixat, **polinoamele Bernstein de grad** n, $B_0^n(t), B_1^n(t), \dots, B_n^n(t)$ sunt definite prin

$$B_i^n(t) = C_n^i t^i (1-t)^{n-i}, \quad i \in \{0, \dots, n\},$$

unde $C_n^i = \frac{n!}{i!(n-i)!}$. Prin convenție, se poate defini $B_i^n(t) = 0$, dacă $i \notin \{0, \dots, n\}$.

Fapt: Dat un poligon de control $(\mathbf{b}_0, \mathbf{b}_1, \dots, \mathbf{b}_n)$ și $\mathbf{b}_0^n(t)$ punctul contruit cu algoritmul de Casteljau pentru un $t \in \mathbb{R}$, are loc relația

$$b_0^n(t) = \sum_{k=0}^{n} B_k^n(t)\mathbf{b}_k.$$

Aceasta este forma Bernstein a curbei Bézier asociate poligonului $(\mathbf{b}_0, \mathbf{b}_1, \dots, \mathbf{b}_n)$.

Proprietăți elementare

Fie $(\mathbf{b}_0, \ldots, \mathbf{b}_n)$ un poligon de control din $\mathbb{R}^m$ și $\mathbf{b} : [0, 1] \to \mathbb{R}^m$ curba Bézier asociată.

Proprietăți elementare

Fie $(\mathbf{b}_0, \dots, \mathbf{b}_n)$ un poligon de control din $\mathbb{R}^m$ și $\mathbf{b} : [0,1] \to \mathbb{R}^m$ curba Bézier asociată.

(i) $\mathbf{b}$ este o curbă polinomială, având gradul mai mic sau egal cu n;

(ii) curba $\mathbf{b}$ interpolează extremitățile poligonului de control, i.e. $\mathbf{b}(0) = \mathbf{b}_0$, $\mathbf{b}(1) = \mathbf{b}_n$; în particular, dacă poligonul de control este închis, curba Bézier asociată este închisă;

(iii) **proprietatea acoperirii convexe**: punctele curbei Bézier $\mathbf{b}$ se află în acoperirea convexă a punctelor de control;

Proprietăți elementare

Fie $(\mathbf{b}_0, \ldots, \mathbf{b}_n)$ un poligon de control din $\mathbb{R}^m$ și $\mathbf{b} : [0,1] \to \mathbb{R}^m$ curba Bézier asociată.

(i) $\mathbf{b}$ este o curbă polinomială, având gradul mai mic sau egal cu n;

(ii) curba $\mathbf{b}$ interpolează extremitățile poligonului de control, i.e. $\mathbf{b}(0) = \mathbf{b}_0$, $\mathbf{b}(1) = \mathbf{b}_n$; în particular, dacă poligonul de control este închis, curba Bézier asociată este închisă;

(iii) **proprietatea acoperirii convexe**: punctele curbei Bézier $\mathbf{b}$ se află în acoperirea convexă a punctelor de control;

(iv) **invarianță afină**: dacă $\tau : \mathbb{R}^m \to \mathbb{R}^m$ este o aplicație afină, atunci curba Bézier asociată poligonului de control date de $(\tau(\mathbf{b}_0), \ldots, \tau(\mathbf{b}_n))$ este curba $\tau(\mathbf{b}^n)$;

(v) **(Invarianța la combinații baricentrice)**: fie $(\mathbf{b}_0, \ldots, \mathbf{b}_n)$, respectiv $(\widetilde{\mathbf{b}}_0, \ldots, \widetilde{\mathbf{b}}_n)$ două poligoane de control și $\mathbf{b}$, respectiv $\widetilde{\mathbf{b}}$ curbele Bézier corespunzătoare. Pentru orice $\alpha \in \mathbb{R}$, curba Bézier asociată poligonului de control $((1-\alpha)\mathbf{b}_0 + \alpha\widetilde{\mathbf{b}}_0, \ldots, (1-\alpha)\mathbf{b}_n + \widetilde{\mathbf{b}}_n)$ este curba $(1-\alpha)\mathbf{b} + \alpha\widetilde{\mathbf{b}}$.

De reținut!

- Orice curbă Bézier este definită/controlată de un poligon de control, acesta este "memorat/stocat" și determină geometria curbei.

De reținut!

- Orice curbă Bézier este definită/controlată de un poligon de control, acesta este "memorat/stocat" și determină geometria curbei.
- Pentru construcția/randarea curbelor Bézier este folosit algoritmul de Casteljau sau reprezentarea cu polinoame Bernstein.

De reținut!

- Orice curbă Bézier este definită/controlată de un poligon de control, acesta este "memorat/stocat" și determină geometria curbei.
- Pentru construcția/randarea curbelor Bézier este folosit algoritmul de Casteljau sau reprezentarea cu polinoame Bernstein.
- Implementare: folosind *tessellation shader*. Exemplu

Extinderi

Q: Curbele Bézier sunt, de fapt, curbe polinomiale. Cum putem genera curbe cât mai complexe?

Extinderi

Q: Curbele Bézier sunt, de fapt, curbe polinomiale. Cum putem genera curbe cât mai complexe?

A:

Extinderi

Q: Curbe Bézier sunt, de fapt, curbe polinomiale. Cum putem genera curbe cât mai complexe?

A:

- Folosind mai multe puncte de control (crește gradul curbei, deci calcule mai complexe).

Extinderi

Q: Curbele Bézier sunt, de fapt, curbe polinomiale. Cum putem genera curbe cât mai complexe?

A:

- Folosind mai multe puncte de control (crește gradul curbei, deci calcule mai complexe).
- Racordând ("punând cap la cap") arce de curbă de grad mai mic. Exemplu (curbe de gradul I): graficul funcției modul $c : \mathbb{R} \to \mathbb{R},\ c(t) = (t, |t|)$. **În practică: curbe de gradul 3 (cubice).** Se ajunge la curbe polinomiale pe porțiuni (curbe spline).

Extinderi

Q: Curbele Bézier sunt, de fapt, curbe polinomiale. Cum putem genera curbe cât mai complexe?

A:

- Folosind mai multe puncte de control (crește gradul curbei, deci calcule mai complexe).
- Racordând ("punând cap la cap") arce de curbă de grad mai mic. Exemplu (curbe de gradul I): graficul funcției modul $c : \mathbb{R} \to \mathbb{R},\ c(t) = (t, |t|)$. **În practică: curbe de gradul 3 (cubice).** Se ajunge la curbe polinomiale pe porțiuni (curbe spline).
- Folosind fracții. Exemplu (curbă rațională): $c : \mathbb{B} \to \mathbb{R}^2, c(t) = \left(\frac{-t^2+1}{t^2+1}, \frac{2t}{t^2+1}\right)$. Imaginea geometrică a acestei curbe este cercul de centru O și rază 1 din care este eliminat punctul $(-1, 0)$. Se ajunge la curbe Bézier raționale.

Extinderi

Q: Curbele Bézier sunt, de fapt, curbe polinomiale. Cum putem genera curbe cât mai complexe?

A:

- Folosind mai multe puncte de control (crește gradul curbei, deci calcule mai complexe).
- Racordând ("punând cap la cap") arce de curbă de grad mai mic. Exemplu (curbe de gradul I): graficul funcției modul $c : \mathbb{R} \to \mathbb{R},\ c(t) = (t, |t|)$. **În practică: curbe de gradul 3 (cubice).** Se ajunge la curbe polinomiale pe porțiuni (curbe spline).
- Folosind fracții. Exemplu (curbă rațională): $c : \mathbb{B} \to \mathbb{R}^2, c(t) = \left(\frac{-t^2+1}{t^2+1}, \frac{2t}{t^2+1}\right)$. Imaginea geometrică a acestei curbe este cercul de centru O și rază 1 din care este eliminat punctul $(-1, 0)$. Se ajunge la curbe Bézier raționale.
- Combinând cele două idei (spline + raționale) se ajunge la o categorie mai generală, curbe NURBS - Non Uniform Rational B-Splines.

Extinderi

Q: Curbele Bézier sunt, de fapt, curbe polinomiale. Cum putem genera curbe cât mai complexe?

A:

- Folosind mai multe puncte de control (crește gradul curbei, deci calcule mai complexe).
- Racordând ("punând cap la cap") arce de curbă de grad mai mic. Exemplu (curbe de gradul I): graficul funcției modul $c : \mathbb{R} \to \mathbb{R},\ c(t) = (t, |t|)$. **În practică: curbe de gradul 3 (cubice).** Se ajunge la curbe polinomiale pe porțiuni (curbe spline).
- Folosind fracții. Exemplu (curbă rațională): $c : \mathbb{B} \to \mathbb{R}^2, c(t) = \left(\frac{-t^2+1}{t^2+1}, \frac{2t}{t^2+1}\right)$. Imaginea geometrică a acestei curbe este cercul de centru O și rază 1 din care este eliminat punctul $(-1, 0)$. Se ajunge la curbe Bézier raționale.
- Combinând cele două idei (spline + raționale) se ajunge la o categorie mai generală, curbe NURBS - Non Uniform Rational B-Splines.
- Principiile de mai sus pot fi extine în cazul suprafețelor.