Primitive grafice

Mihai-Sorin Stupariu

Sem. al II-lea, 2022 - 2023

Spațiul culorilor

- Culorile sunt obținute combinând intensitățile de pe trei canale: R (red); G (green); B (blue) – Cubul RGB.

Spațiul culorilor

- Culorile sunt obținute combinând intensitățile de pe trei canale: R (red); G (green); B (blue) – Cubul RGB.
- În OpenGL o culoare este indicată prin funcția

```
glColor* (  );
```

Spațiul culorilor

- Culorile sunt obținute combinând intensitățile de pe trei canale: R (red); G (green); B (blue) – Cubul RGB.
- În OpenGL o culoare este indicată prin funcția

```
glColor* (  );
```

Sufixul * poate indica:

Spațiul culorilor

- Culorile sunt obținute combinând intensitățile de pe trei canale: R (red); G (green); B (blue) – Cubul RGB.
- În OpenGL o culoare este indicată prin funcția

```
glColor* (  );
```

Sufixul * poate indica:

- dimensiunea n a spațiului de culori în care lucrăm, $n = 3$ (RGB) sau $n = 4$ (RGBA); A=factorul "alpha", legat de opacitate/transparență.

Spațiul culorilor

- Culorile sunt obținute combinând intensitățile de pe trei canale: R (red); G (green); B (blue) – Cubul RGB.
- În OpenGL o culoare este indicată prin funcția

```
glColor* (  );
```

Sufixul * poate indica:

- dimensiunea n a spațiului de culori în care lucrăm, $n = 3$ (RGB) sau $n = 4$ (RGBA); A=factorul "alpha", legat de opacitate/transparență.
- tipul de date utilizat, care poate fi:
  `i` (integer) ($\texttt{i} \in \{0, 1, \ldots, 255\}$)
  `f` (float)
  `d` (double) ($\texttt{f}, \texttt{d} \in [0.0, 1.0]$)

Spațiul culorilor

- Culorile sunt obținute combinând intensitățile de pe trei canale: R (red); G (green); B (blue) – Cubul RGB.
- În OpenGL o culoare este indicată prin funcția

```
glColor* (  );
```

Sufixul * poate indica:

- dimensiunea n a spațiului de culori în care lucrăm, $n = 3$ (RGB) sau $n = 4$ (RGBA); A=factorul "alpha", legat de opacitate/transparență.
- tipul de date utilizat, care poate fi:
  `i` (integer) (`i` $\in \{0, 1, \ldots, 255\}$)
  `f` (float)
  `d` (double) (`f`, `d` $\in [0.0, 1.0]$)
- (opțional) posibila formă vectorială, indicată prin sufixul `v`.

Vârfuri - funcții pentru indicare

Primitivele grafice sunt trasate cu ajutorul **vârfurilor** (*entități abstracte, a nu fi confundate cu punctele!*). În OpenGL un vârf este definit cu ajutorul funcției

```
glVertex* (  );
```

Cu ajutorul sufixului * se indică:

- dimensiunea n a spațiului în care considerăm vârful, $n \in \{2, 3, 4\}$ (în cazul $n = 2$, intern, a treia coordonată este 0, a patra egală cu 1; similar, pentru $n = 3$ a patra coordonată este egală cu 1)
- tipul de date utilizat, care poate fi:
  - `i` (integer)
  - `s` (short)
  - `f` (float)
  - `d` (double)
- (opțional) posibila formă vectorială, indicată prin sufixul `v`.

Caracteristici ale vârfurilor

- Unui vârf îi sunt asociate:

Caracteristici ale vârfurilor

- Unui vârf îi sunt asociate:
  - coordonate (fac parte din definiție),

Caracteristici ale vârfurilor

- Unui vârf îi sunt asociate:
  - coordonate (fac parte din definiție),
  - o culoare (v. secțiunea Culori...),

Caracteristici ale vârfurilor

- Unui vârf îi sunt asociate:
  - coordonate (fac parte din definiție),
  - o culoare (v. secțiunea Culori...),
  - o normală (legată de funcții de iluminare),

Caracteristici ale vârfurilor

- Unui vârf îi sunt asociate:
  - coordonate (fac parte din definiție),
  - o culoare (v. secțiunea Culori...),
  - o normală (legată de funcții de iluminare),
  - coordonate de texturare

Caracteristici ale vârfurilor

- Unui vârf îi sunt asociate:
  - coordonate (fac parte din definiție),
  - o culoare (v. secțiunea Culori...),
  - o normală (legată de funcții de iluminare),
  - coordonate de texturare
- Pentru o anumită caracteristică este considerată valoarea *curentă* a respectivei caracteristici. Altfel spus, ea trebuie indicată în codul sursă *<u>înaintea</u> vârfului* (principiul *mașinii de stare*).

Funcții pentru primitive

Vârfurile sunt utilizate pentru trasarea *primitivelor grafice*.

- O funcție de tipul `glVertex ( )` poate fi apelată într-un cadru de tip

```
glBegin (*);

glEnd;
```

(unde * reprezintă tipul de primitivă generat);

Tipuri de primitive

- Puncte: `GL_POINTS`

Tipuri de primitive

- Puncte: `GL_POINTS`
- Segmente de dreaptă: `GL_LINES`, `GL_LINE_STRIP`, `GL_LINE_LOOP`

Tipuri de primitive

- Puncte: `GL_POINTS`
- Segmente de dreaptă: `GL_LINES`, `GL_LINE_STRIP`, `GL_LINE_LOOP`
- Triunghiuri: `GL_TRIANGLES`, `GL_TRIANGLE_STRIP`, `GL_TRIANGLE_FAN`

Tipuri de primitive

- Puncte: `GL_POINTS`
- Segmente de dreaptă: `GL_LINES`, `GL_LINE_STRIP`, `GL_LINE_LOOP`
- Triunghiuri: `GL_TRIANGLES`, `GL_TRIANGLE_STRIP`, `GL_TRIANGLE_FAN`
- Dreptunghiuri: `GL_QUADS`, `GL_QUAD_STRIP`

Tipuri de primitive

- Puncte: `GL_POINTS`
- Segmente de dreaptă: `GL_LINES`, `GL_LINE_STRIP`, `GL_LINE_LOOP`
- Triunghiuri: `GL_TRIANGLES`, `GL_TRIANGLE_STRIP`, `GL_TRIANGLE_FAN`
- Dreptunghiuri: `GL_QUADS`, `GL_QUAD_STRIP`
- Poligoane (convexe!): `GL_POLYGON`

Puncte - reprezentare

Sunt randate puncte într-un cadru de tipul

```
glBegin (GL_POINTS);
...  // lista de varfuri
glEnd;
```

Exemplu. Următoarele două secvențe de cod sursă sunt echivalente, având ca efect desenarea punctelor de coordonate (50,100), (70,80) și (90,150).

(i)

```
glBegin(GL_POINTS);
glVertex2i (50, 100);
glVertex2i (70, 80);
glVertex2i (90, 150);
glEnd;
```

(ii)

```
int p1[ ] = {50, 100};
int p2[ ] = {70, 80};
int p3[ ] = {90, 150};

glBegin(GL_POINTS);
glVertex2iv (p1);
glVertex2iv (p2);
glVertex2iv (p3);
glEnd;
```

Puncte - atribute ale punctelor

- Culoarea (dată de culoarea vârfului)

Puncte - atribute ale punctelor

- Culoarea (dată de culoarea vârfului)
- Dimensiunea

```
glPointSize (dimens);
```

Această funcție trebuie apelată înainte de `glBegin (GL_POINTS);` pentru a avea efectul dorit.

Puncte - atribute ale punctelor

- Culoarea (dată de culoarea vârfului)
- Dimensiunea

```
glPointSize (dimens);
```

Această funcție trebuie apelată înainte de `glBegin (GL_POINTS);` pentru a avea efectul dorit.

- **Comentariu:** Modalitatea de reprezentare a punctelor poate fi controlată folosind

```
glEnable(GL_POINT_SMOOTH);
..// desenarea varfurilor
glDisable(GL_POINT_SMOOTH);
```

Segmente de dreaptă - reprezentare (I)

Un singur segment de dreaptă, determinat de vârfurile de coordonate (x_1, y_1), (x_2, y_2) (vom lua $x_1, x_2, y_1, y_2 \in \mathbb{Z}$) poate fi trasat utilizând următoarea secvență de cod sursă:

```
glBegin (GL_LINES);
  glVertex2i (x1, y1);
  glVertex2i (x2, y2);
glEnd;
```

Segmente de dreaptă - reprezentare (IIa)

De asemenea, pentru trasarea segmentelor de dreaptă mai pot fi utilizate și constantele simbolice `GL_LINES`, `GL_LINE_STRIP`, `GL_LINE_LOOP`. Să considerăm o mulțime de puncte având coordonatele (pe care le presupunem numere întregi) $P_1 = (x_1, y_1), P_2 = (x_2, y_2), \ldots, P_n = (x_n, y_n)$.

• Putem desena segmentele de forma $[P_{2k+1}P_{2k+2}]$ (în cazul în care n este impar, ultimul punct nu este unit cu nimeni) folosind instrucțiunile

```
glBegin (GL_LINES);
  glVertex2i (x1, y1);
  glVertex2i (x2, y2);
  .........
  glVertex2i (xn, yn);
glEnd;
```

Segmente de dreaptă - reprezentare (IIb)

De asemenea, pentru trasarea segmentelor de dreaptă mai pot fi utilizate și constantele simbolice `GL_LINES`, `GL_LINE_STRIP`, `GL_LINE_LOOP`. Să considerăm o mulțime de puncte având coordonatele (pe care le presupunem numere întregi) $P_1 = (x_1, y_1), P_2 = (x_2, y_2), \ldots, P_n = (x_n, y_n)$.

- Linia frântă ce unește punctele $P_1, P_2, \ldots, P_n$ este trasată astfel:

```
glBegin
(GL_LINE_STRIP);
  glVertex2i (x1, y1);
  glVertex2i (x2, y2);
  .........
  glVertex2i (xn, yn);
glEnd;
```

Segmente de dreaptă - reprezentare (IIc)

De asemenea, pentru trasarea segmentelor de dreaptă mai pot fi utilizate și constantele simbolice `GL_LINES`, `GL_LINE_STRIP`, `GL_LINE_LOOP`. Să considerăm o mulțime de puncte având coordonatele (pe care le presupunem numere întregi) $P_1 = (x_1, y_1), P_2 = (x_2, y_2), \ldots, P_n = (x_n, y_n)$.

• Linia poligonală închisă determinată de punctele $P_1, P_2, \ldots, P_n$ (poate conține autointersecții!) este desenată folosind comenzile:

```
glBegin
(GL_LINE_LOOP);
  glVertex2i (x1, y1);
  glVertex2i (x2, y2);
  .........
  glVertex2i (xn, yn);
glEnd;
```

Segmente de dreaptă - atribute ale segmentelor de dreaptă

- **1. Culoarea** (dată de culorile extremităților segmentului). Modul de trasare se controlează cu

```
glShadeModel (...);
```

, fiind posibile două variante.

Segmente de dreaptă - atribute ale segmentelor de dreaptă

- **1. Culoarea** (dată de culorile extremităților segmentului). Modul de trasare se controlează cu

```
glShadeModel (...);
```

fiind posibile două variante.

(i) **Interpolarea culorilor.**

```
glShadeModel (GL_SMOOTH);
```

Când vârfurile A și B ale unui segment $[AB]$ au culori diferite, culorile pixelilor sunt determinate folosind o regulă de tip "gradient" (interpolare afină). Fie $P \in [AB]$,

$$P = (1-\alpha)A + \alpha B, \quad \alpha \in [0,1]$$

un punct de pe segmentul $[AB]$. Fie $c_A, c_B \in [0,1]^3$ (sau $c_A, c_B \in [0,1]^4$) culorile lui A, respectiv B. Culoarea lui P este dată de

$$c_P = (1-\alpha)c_A + \alpha c_B.$$

Segmente de dreaptă - atribute ale segmentelor de dreaptă

- **1. Culoarea** (dată de culorile extremităților segmentului). Modul de trasare se controlează cu

```
glShadeModel (...);
```

fiind posibile două variante.

(i) **Interpolarea culorilor.**

```
glShadeModel (GL_SMOOTH);
```

Când vârfurile A și B ale unui segment $[AB]$ au culori diferite, culorile pixelilor sunt determinate folosind o regulă de tip "gradient" (interpolare afină). Fie $P \in [AB]$,

$$P = (1-\alpha)A + \alpha B, \quad \alpha \in [0,1]$$

un punct de pe segmentul $[AB]$. Fie $c_A, c_B \in [0,1]^3$ (sau $c_A, c_B \in [0,1]^4$) culorile lui A, respectiv B. Culoarea lui P este dată de

$$c_P = (1-\alpha)c_A + \alpha c_B.$$

(ii) **Utilizarea unei singure culori.**

```
glShadeModel (GL_FLAT);
```

Segmente de dreaptă - atribute ale segmentelor de dreaptă

- **2. Lățimea**

```
glLineWidth (width);
```

Segmente de dreaptă - atribute ale segmentelor de dreaptă

- **2. Lățimea**

```
glLineWidth (width);
```

- **3. Modul de desenare (șablon, model)**

  Este definit prin

```
glLineStipple(repeatfactor, pattern);
```

  și activat / dezactivat prin

```
glEnable (GL_LINE_STIPPLE);
.........................
glDisable (GL_LINE_STIPPLE);
```