inc. geom. în gr. pe calc

4.10.2010

OpenGL

↓

dezvoltarea unor aplicații folosind openGL

Evaluare - verificare (ult. săpt)

Punctaj: 10p of.
40p test grilă (final -> verificare)
50p pe parcursul sem.
(laborator: proiecte, testări)

Laborator: 2h l2 săpt. (si -> sp) -> 7 lab

Biblioteci utilizate de OpenGL și funcții asociate

- bibliotecă fundamentală ("core library")
  -> independentă de platforma pe care lucrăm
  -> funcțiile corespunzătoare au prefixul gl
  (ex.: glVertex(), glColor(), glBegin());
- OpenGL Utility (GLU): proceduri / funcții legate de proiecție, cuadrice, conice
  funcțiile asociate au prefixul glu
- pentru a realiza fereastra de vizualizare
  bibliotecă dependentă de sistem
  (ex.: OpenGL Utility Toolkit GLUT) -> poate interacționa cu orice sistem de operare care utilizează ferestre de vizualizare

Există și AppleGL (AGL) etc.

* → funcțiile asociate cu prefixul: glut

Primitive grafice. Atribute ale primitivelor grafice

P.g. sunt realizate cu ajutorul vârfurilor (vertex)

În OpenGL un vârf este definit printr-o funcție de tipul

glVertex $\underset{\sim}{*}$(   ) ;

↓ sufix   ⇕ coordonatele vârfului

*: 3 informații

- dimensiunea spațiului în care considerăm vârful
  $n \in \{2, 3, 4\}$

ex.: $n=2$ (2D)   (3, 8)

$n=3$ (3D)   (2, 7, 9) ; (3, 8, 0)

$n=4$ (2, 7, 9, 1) ; (3, 8, 0, 1)

⚠ intern $n=4$, implicit: a 3^a componentă este 0.0
a 4^a — " — 1.0

- tipul de date utilizat: integer, float, double
- (posibila) utilizare a formei vectoriale

Ex.:

```
gevertex2i(80,120);
    ⇕
int p[ ]={80,120};
gevertex2iv(p);
```

Vârfurilor le sunt asociate:

- coordonate (fac parte din definiție)
- culoare
- ~~dimensiune (alte date)~~

- coordonate de texturare; } -> later
- normale (iluminare)

Culoare: (precizată înainte de vârful respectiv!)

glColor*(compcolor);

* = sufix asemănător celui de la glVertex();

n=3 : R, G, B
red green blue

n=4 : R, G, B, A
↳ factorul alfa (transparență)

i (integer) : R, G, B ∈ {0, 1, ..., 255}
f, d : R, G, B ∈ [0.0, 1.0]

0 0 0 - negru
1 1 1 - alb
1 0 0 - roșu
0 1 0 - verde
0 0 1 - albastru

culori (!) - combinații de bază
(Cum obții galben?)

Regulă: vârfurile sunt utilizate pt trasarea primitivelor grafice în cadrul unor funcții de forma

glBegin(___); ← tipul primitivei
___ ← vârfuri
glEnd();

---

① Punctul / Puncte

glBegin(GL_POINTS);
← vârfurile corespunzătoare punctelor
glEnd();

Atribute ale punctului

- culoarea (dată de culoarea vârfului)
- dimensiunea glPointSize(dimens);

⚠ Aceste funcții tb. apelate înainte de apelarea primitivei glBegin(GL_POINTS); pt a avea efectul dorit

② Segmente de dreaptă / linii

GL_LINES → segmente care unesc punctul 2k+1 cu 2k+2 (le unește 2 câte 2)
nr impar: ultimul rămâne punct

GL_LINE_STRIP → linie frântă cu care punctul i este unit cu (i+1), ∀i

GL_LINE_LOOP → linie frântă închisă cu care punctul i este unit cu (i+1), ∀i și ultimul punct e unit cu primul)

Atribute:

- culoare: ⚠ când vârfurile au culori diferite, culorile punctelor unui segment sunt "calculate" folosind interpolare (afină)
modul de trasare se controlează cu funcția glShadeMod
  - glShadeModel(GL_FLAT); → NU interpolează
  - glShadeModel(GL_SMOOTH); → default

- grosimea glLineWidth(width);
  width – float

⚠ tb. apelate înainte de glBegin(GL_LINES);
etc.

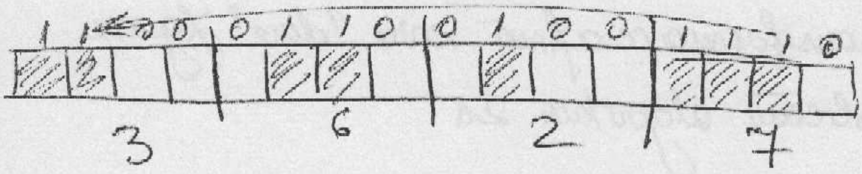
- modul de desenare (şablon/model)

⊛ este definit printr-o funcţie de forma glLineStipple(repeat_factor, pattern);

un nat / precizează de câte ori se repetă pattern

16 biţi reprezentaţi în hexazecimal)

Ex.: pattern: 0X7263



culoarea primitivei

fond

1 - pixel "ON" (cu culoarea primitivei,

0 = pixel "OF"

default: 0XFFFF

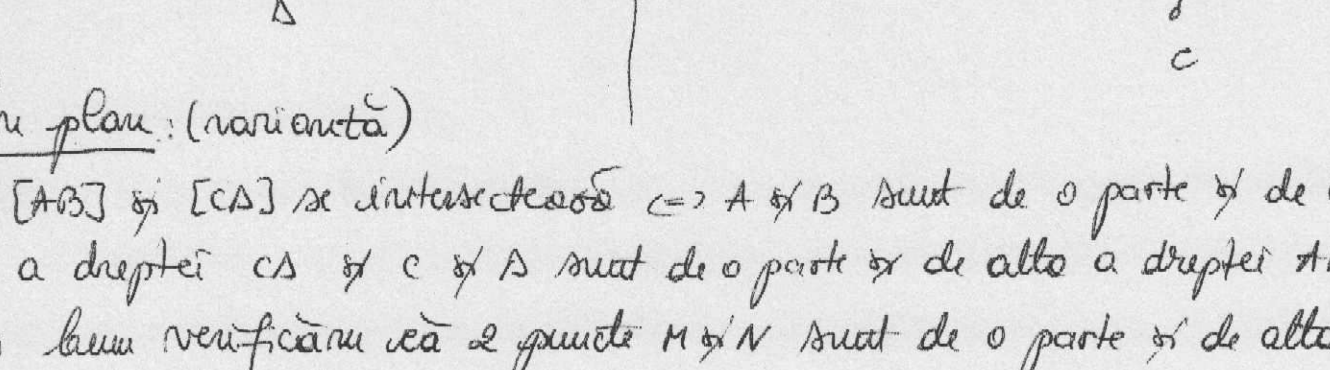
⊛ este activat / dezactivat prin:

glEnable(GL_LINE_STIPPLE);

glDisable(GL_LINE_STIPPLE);

Intersecţii de segmente

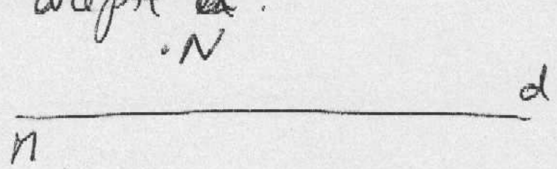
Întrebare: cum putem verifica dacă 2 segmente [AB] şi [CD] se intersectează?



În plan: (varianta)

① [AB] şi [CD] se intersectează $\Leftrightarrow$ A şi B sunt de o parte şi de alta a dreptei CD şi C şi D sunt de o parte şi de alta a dreptei AB.

② Cum verificăm că 2 puncte M şi N sunt de o parte şi de alta a unei drepte d?



Scriem ecuația implicită a dreptei d : $f(x,y)=0$

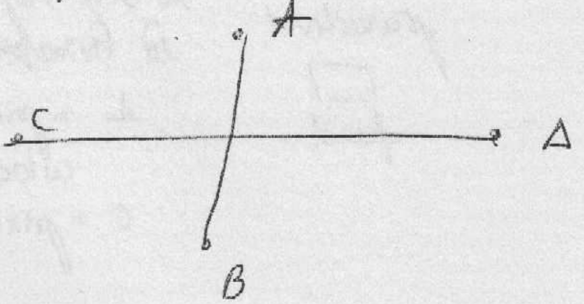
M și N sunt de o parte și de alta a lui $d \iff \boxed{f(M)\cdot f(N) < 0}$

<u>În spațiu</u>: (pp că A, B, C, D coplanare

$\longrightarrow$ de testat...)

<u>Var 1</u>: găsim o izometrie / transformare afină care "duce" figura în planul $z=0$ și aplicăm algoritmul 2D.

<u>Var 2</u>:



$AB = \{(1-t)A + tB \mid t \in \mathbb{R}\}$ = dreapta AB

$CD = \{(1-s)C + sD \mid s \in \mathbb{R}\}$ = dreapta CD

(?) găsim s și $t \in \mathbb{R}$ a.î. $(1-t_0)A + t_0 B = (1-s_0)C + s_0 D$

(trecând în coordonate: găsim 3 ecuații cu 2 necunoscute s_0 și t_0)

- dacă sistemul e compatibil determinat soluția e unică
  $s_0, t_0 \rightarrow$ punctul de $\cap$ al dreptelor
- dacă $s_0, t_0 \in [0,1]$, se $\cap$ chiar <u>segmentele</u>

③ Poligoane

Funcții cu ajutorul cărora pot fi desenate poligoane (inclusiv interiorul). Ce este interiorul? Cum se colorează / umple? Putem reprezenta "cât mai variat" aceste poligoane?

a) glBegin(GL_TRIANGLES);
-- șir de vârfuri $p_1 p_2 \ldots$
glEnd();

desenează Δ
$p_1 p_2 p_3$
$p_4 p_5 p_6$
—
$p_{3k+1}\, p_{3k+2}\, p_{3k+3}$
(pot exista vârfuri care nu apar în niciun triunghi)

triunghiuri care nu au laturi în comun

b) glBegin(GL_TRIANGLE_STRIP);
-- șir de vârfuri -- $p_1 - p_n$
glEnd();

desenează Δ:
$\Delta p_1 p_2 p_3$
$\Delta p_2 p_3 p_4$
$\Delta p_4 p_5 p_6$ etc

triunghiuri care au în comun muchii (care?)

c) glBegin(GL_TRIANGLE_FAN);
-- șir de vârfuri $p_1 \ldots p_n$
glEnd();

desenează
$\Delta p_1 p_2 p_3$
$\Delta p_1 p_3 p_4$
$\Delta p_1 p_k p_{k+1}$

... au în comun vârful p_1

d) glBegin(GL_QUADS);
-- șir de vârfuri $p_1 \ldots p_n$
glEnd();

patrulatere
$p_1 p_2 p_3 p_4$
$p_5 p_6 p_7 p_8$

e) glBegin(GL_QUAD_STRIP)
-- șir de vârfuri
glEnd();

$p_1 p_2 p_3 p_4$

—

f) glBegin(GL_POLYGON);
-- nr de vârfuri

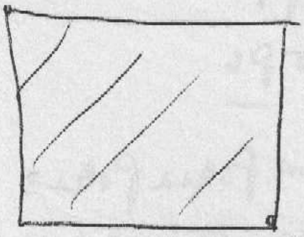
glEnd();

⚠ vezi comentarii ulterioare

Există o funcție specială pt desenarea dreptunghiurilor 2D:

glRect*(x_1, y_1, x_2, y_2);

(x_1, y_1)



(x_2, y_2)

Reguli referitoare la vârfurile ce determină un poligon (pt ca GL_POLYGON să producă efectul dorit)

- cel puțin 2, distincte între ele 2x2

1) punctele tb să fie situate în același plan (coplanare)

Condiția de coplanaritate pt n puncte $p_1, p_2, \dots, p_n$ din $\mathbb{R}^3$:

$$\text{rang}\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ x_{p_1} & x_{p_2} & x_{p_3} & \dots & x_{p_n} \\ y_{p_1} & y_{p_2} & y_{p_3} & \dots & y_{p_n} \\ z_{p_1} & z_{p_2} & z_{p_3} & \dots & z_{p_n} \end{pmatrix} = 3$$

Alternativ:

vectorial:

- se formează vectorii

$\overrightarrow{p_1p_2} = p_2 - p_1$

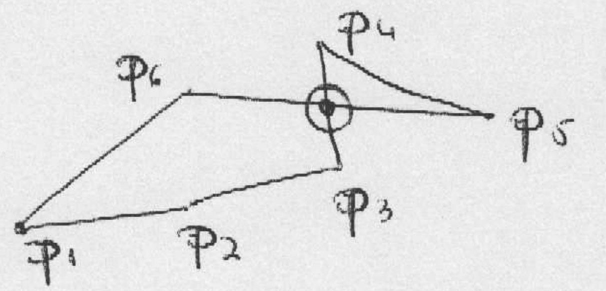
$\overrightarrow{p_1p_3} = p_3 - p_1$

$\dots$

$\overrightarrow{p_1p_n} = p_n - p_1$

- punctele $P_1, P_2, \dots P_n$ sunt coplanare $\Leftrightarrow$ spațiul generat de vectorii indicați are dimensiunea $\underline{2}$.

2)



Ordinea vârfurilor să fie indicată a.î. muchiile poligonului să nu se intersecteze (decât muchiile succesive în vârfuri)

→ linia poligonală care ia naștere să nu aibă autointersecții

De fapt: ce tb verificat?

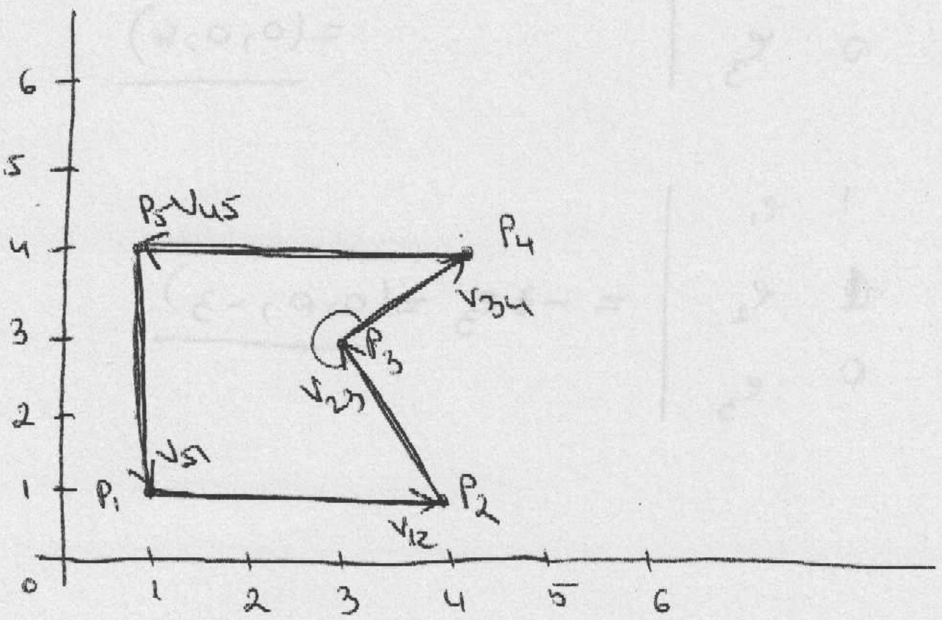
- Fie $P_1, P_2, \dots, P_n$ vârfuri (în această ordine!)
- se formează muchiile $[P_1P_2], [P_2P_3], \dots, [P_kP_{k+1}]$ (cu convenția $P_{n+1} = P_1$)
- Tb. ca 2 muchii distincte să nu aibă puncte comune (cu excepția muchiilor succesive care au în comun un vârf)

⇝ problema revine la a testa intersecții de segmente (v. curs anterior)

3) Vârfurile / punctele să genereze un poligon convex

(?) criteriu care să precizeze dacă un poligon este convex / concav.

Exemplu



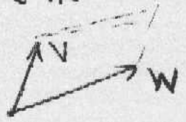
$P_1 = (1,1)$
$P_2 = (4,1)$
$P_3 = (3,3)$
$P_4 = (4,4)$
$P_5 = (1,4)$

$\Rightarrow$ poligon concav

Considerăm toate aceste puncte în $\mathbb{R}^3$, având ultima coordonată egală cu 0 și considerăm vectorii determinați de perechi de puncte succesive: $v_{12}, v_{23}, v_{34}, v_{45}, v_{51}$.

(<u>Remember</u> Produs vectorial în $\mathbb{R}^3$

$v, w \in \mathbb{R}^3$



Se calculează formal astfel:

$$\underbrace{\begin{vmatrix} v_1 & w_1 & e_1 \\ v_2 & w_2 & e_2 \\ v_3 & w_3 & e_3 \end{vmatrix}}_{} \longrightarrow \text{vector din } \mathbb{R}^3$$

comp lui v în baza canonică

vectorii bazei canonice

componentele lui w în baza canonică)

$V_{12} = \overrightarrow{P_1P_2} = P_2 - P_1 = (3, 0, 0)$

$V_{23} = P_3 - P_2 = (-1, 2, 0)$

$$\underline{V_{12} \times V_{23}} = \begin{vmatrix} 3 & -1 & e_1 \\ 0 & 2 & e_2 \\ 0 & 0 & e_3 \end{vmatrix} \overset{\text{dezv. după col 3}}{=\!=\!=} 0 \cdot e_1 + 0 \cdot e_2 + 6 e_3 = \underline{(0, 0, 6)}$$

$V_{34} = (1, 0, 0)$

$$\underline{V_{23} \times V_{34}} = \begin{vmatrix} -1 & 1 & e_1 \\ 2 & 1 & e_2 \\ 0 & 0 & e_3 \end{vmatrix} = -3 e_3 = \underline{(0, 0, -3)}$$

$$\underline{v_{34} \times v_{45}} = \begin{vmatrix} 1 & -3 & e_1 \\ 1 & 0 & e_2 \\ 0 & 0 & e_3 \end{vmatrix} = \underline{(0,0,3)}$$

$$\underline{v_{45} \times v_{51}} = \begin{vmatrix} -3 & 0 & e_1 \\ 0 & -3 & e_2 \\ 0 & 0 & e_3 \end{vmatrix} = \underline{(0,0,9)}$$

Concluzia. Un poligon 2D este convex $\Leftrightarrow$ toate produsele vectoriale de forma $v_{i\,i+1} \times v_{i+1\,i+2}$ $(i+1 \in \{2,3,\dots,n,1\})$ cu convenție au ultima componentă pozitivă sau negativă (tb. să aibă același semn)

gta.math.unibuc.ro → Academic - Stupi - CGGC

Poligon din $\mathbb{R}^2$:
- convex $\Leftrightarrow$ toate vf au același "semn" (ultima comp a prod vect)
- concav: cum stabilim vârfurile "concave"?

Varianta 1. cu un algoritm eficient (e.g. Graham's Scan) - determinăm frontiera acoperirii convexe ⇝ v.f convexe

Varianta 2. Presupunem că $P_1, P_2, \dots, P_r$ au același semn după redenumire
$Q_1, Q_2, \dots, Q_e$ ⇝ au semn contrar.
Vârfurile concave sunt în acoperirea convexă a mulțimii determinate de celelalte vârfuri

În general, date niște puncte $A_1, A_2, \dots, A_n$ și M din $\mathbb{R}^3$ (presupuse coplanare) cum testăm dacă M este în interiorul acoperirii convexe $Conv(A_1, A_2, \dots, A_n)$

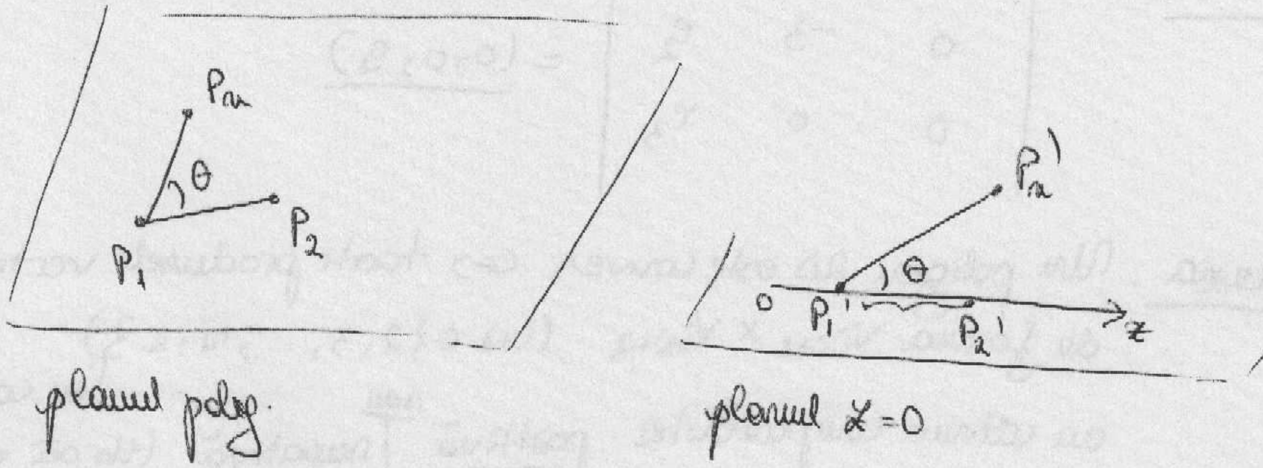
Varianta 1: cu arii

Varianta 2: M coplanar cu $A_1, A_2, \dots, A_n \Rightarrow M = \alpha_1 A_1 + \alpha_2 A_2 + \dots + \alpha_n A_n$ (combinație afină)
$\alpha_1 + \alpha_2 + \dots + \alpha_n = 1$

$M \in \text{Conv}(A_1, \ldots, A_n) \Leftrightarrow \exists$ o combinație afină care să fie convexă

Testarea concavității pt un poligon din $\mathbb{R}^3$

Metoda 1. găsim o transformare euclidiană, adică o izometrie a.î. poligonul inițial să fie transformat într-un poligon situat în planul $z = 0$



găsim o izometrie a.î. $P_1 \longmapsto P_1' = (0,0,0)$

$P_2 \longmapsto P_2' = (\|P_1P_2\|, 0, 0)$

$P_n \longmapsto P_n'$ a.î. $\widehat{P_2 P_1 P_n} \equiv \widehat{P_2' P_1' P_n'}$ și

$\|P_n P_1\| = \|P_n' P_1'\|$

Metoda 2. Generalizăm în spațiu metoda din plan

Exemplu.
$P_1 = (2,-1,1)$
$P_2 = (0,1,2)$
$P_3 = (1,0,3)$
$P_4 = (1,0,2)$

$\overrightarrow{P_1P_2} = (-2,2,1)$
$\overrightarrow{P_1P_3} = (-1,1,2)$
$\overrightarrow{P_1P_4} = (-1,1,1)$

Acești vectori sunt liniar dependenți (de ce? -> determinant = 0) $\Rightarrow$
$\Rightarrow$ punctele sunt coplanare

$$P_1P_2 \times P_2P_3 = (-2,2,1) \times (1,-1,1) = \begin{vmatrix} -2 & 1 & e_1 \\ 2 & -1 & e_2 \\ 1 & 1 & e_3 \end{vmatrix} = (3,3,0)$$

$\overrightarrow{P_2P_3} \times \overrightarrow{P_3P_4} = (1,-1,1) \times (0,0,-1) = \begin{vmatrix} 1 & 0 & e_1 \\ -1 & 0 & e_2 \\ 1 & -1 & e_3 \end{vmatrix} = (1,+1,0)$

$\overrightarrow{P_3P_4} \times \overrightarrow{P_4P_1} = (-1,-1,0)$

$\overrightarrow{P_4P_1} \times \overrightarrow{P_1P_2} = (4,4,0)$

(Obs! vectorii $\overrightarrow{P_1P_2} \times \overrightarrow{P_2P_3}$, etc sunt coliniari $\to$ legat de coplanaritatea punctelor)

Putem "grupa" vârfurile $\begin{cases} P_1, P_2, P_3 \\ P_4 \end{cases}$

Afirm $P_4 \in \mathrm{Conv}(P_1, P_2, P_3)$

Determinăm $\alpha_1, \alpha_2, \alpha_3$ cu $\alpha_1+\alpha_2+\alpha_3 = 1$ a.î. $P_4 = \alpha_1 P_1 + \alpha_2 P_2 + \alpha_3 P_3$

Idee de lucru: - scriem o relație vectorială între $\overrightarrow{P_4P_1}$, $\overrightarrow{P_4P_2}$ și $\overrightarrow{P_4P_3}$ (dependență liniară)

$\overrightarrow{P_4P_1} = (1,-1,-1)$

$\overrightarrow{P_4P_2} = (-1,1,0)$

$\overrightarrow{P_4P_3} = (0,0,1)$

Relația: $\overrightarrow{P_4P_1} + \overrightarrow{P_4P_2} + \overrightarrow{P_4P_3} = 0$

$\Longrightarrow$ împărțim convenabil a.î. suma coef $= 1$

$\frac{1}{3}\overrightarrow{P_4P_1} + \frac{1}{3}\overrightarrow{P_4P_2} + \frac{1}{3}\overrightarrow{P_4P_3} = 0 = \overrightarrow{P_4P_4}$

$\xrightarrow{\text{devectorializare}} \underline{P_4 = \frac{1}{3}P_1 + \frac{1}{3}P_2 + \frac{1}{3}P_3}$

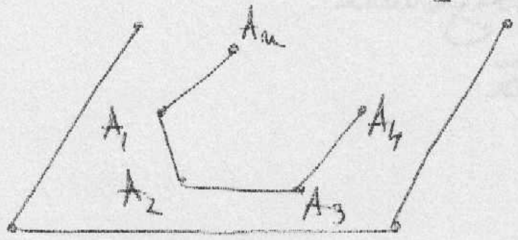
combinație afină care este convexă $\Rightarrow P_4 \in \mathrm{Conv}(P_1, P_2, P_3)$

$\Rightarrow$ poligonul $P_1P_2P_3P_4$ "concav" în P_4

Fața / spatele unui poligon convex

Vector normal la planul unui poligon convex

Fie $A_1, A_2, \ldots, A_n$ un poligon convex



Ecuația planului : $Ax + By + Cz + D = 0$

unde A, B, C, D pot fi obținuți din dezvoltarea determinantului:

$$\begin{vmatrix} x & y & z & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix} = 0$$

Cine sunt, de fapt, coeficienții A, B, C?

De fapt: $(A, B, C) - \overrightarrow{A_1A_2} \times \overrightarrow{A_2A_3}$

⚠ Oricum am alege trei puncte consecutive, vectorii obținuți sau sus au avloză sens.

Vectorul $n = \frac{1}{\sqrt{A^2+B^2+C^2}} (A, B, C)$ s.n. normala la plan

⚠ Este independent de alegerea a 3 puncte consecutive

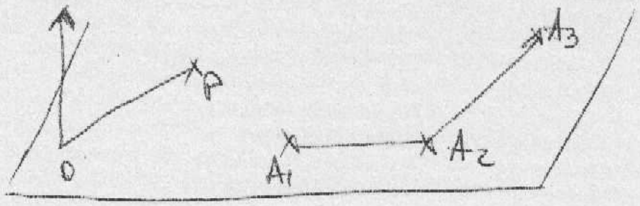
Notăm $\Pi(P) = \Pi(x, y, z) = \frac{1}{\sqrt{A^2+B^2+C^2}}(Ax + By + Cz + D)$

Def. $P = (x, y, z)$ se află în fața poligonului $\Leftrightarrow \Pi(P) > 0$

$P =$ – – spatele $\Leftrightarrow \Pi(P) \leq 0$

Ce semnificație are această definiție?

Translatăm totul în O



P în fața poligonului $\Leftrightarrow Ax+By+Cz>0 \Leftrightarrow$ produs scalar $\langle n, \overline{OP} \rangle = (x,y,z)$

Concluzie: vectorul normal indică fața poligonului

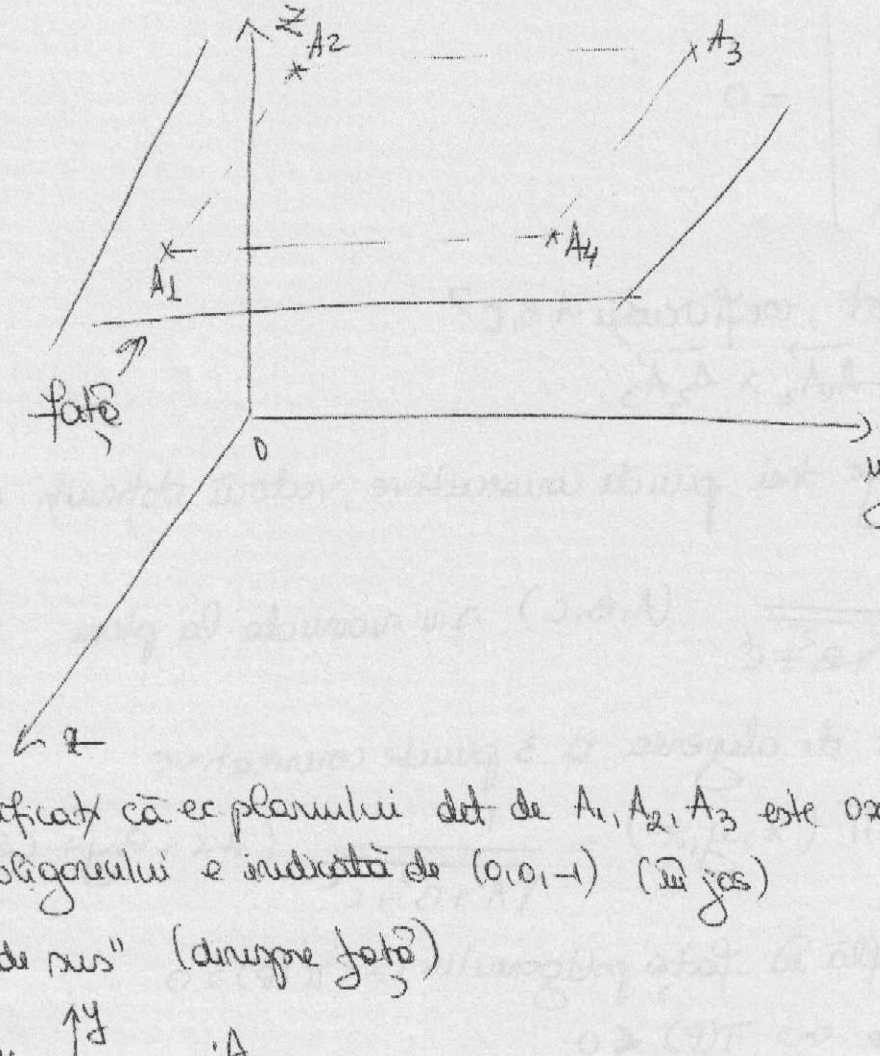
⚠ Ordinea punctelor este fundamentală

<u>Exemplu:</u> $A_1 = (100, -100, 100)$

$A_2 = (-100, -100, 100)$

$A_3 = (-100, 100, 100)$

$A_4 = (100, 100, 100)$



cu regula șurubului

Verificați că ec planului det de A_1, A_2, A_3 este $0x + 0y - 40000z + \ldots = 0 \Rightarrow$

$\Rightarrow$ fața poligonului e indicată de $(0,0,-1)$ (în jos)

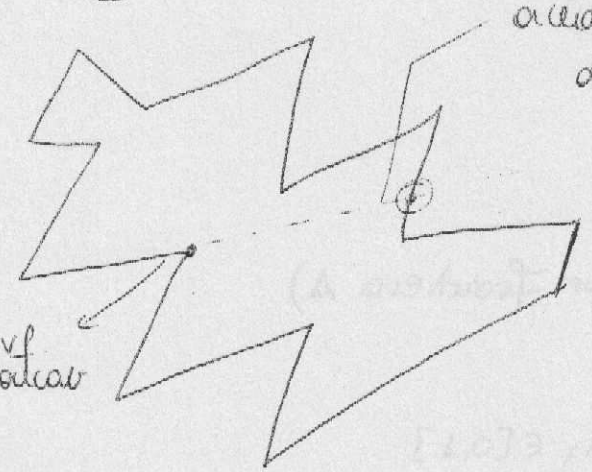
„privit de sus" (dinspre față)

Obs! Poligoanele pot fi colorate/umplute folosind sabloane
- definire un tablou de 32x32 biti
  GLubyte sablon[] = {0x70, 0xEA, ...}
- activare / aplicare / dezactivare

```
glEnable(GL_POLYGON_STIPPLE);
  glPolygonStipple(sablon);
    glBegin(GL_POLYGON);
      - - - -
    glEnd();
glDisable(GL_POLYGON_STIPPLE);
```

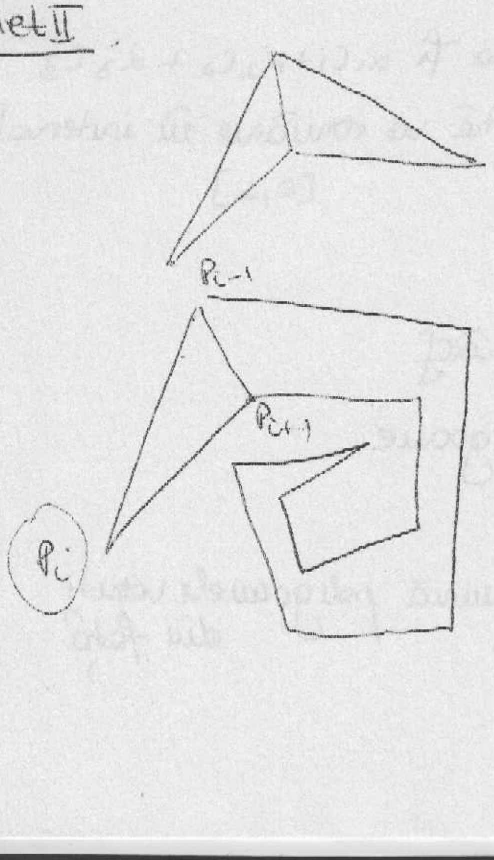
Cum reprezentăm un poligon concav?
Răspuns: Triangulându-l.

Met I.



această prelungire determină o descompunere a poligonului inițial în 2 poligoane cu un nr mai mic de concavități

Met II



- Selectăm 3 vârfuri consecutive $P_{i-1} P_i P_{i+1}$ cu P_i convex
- $P_{i-1} P_{i+1}$ este "diagonală veritabilă" (=) niciunul din celelalte vf. nu este în interiorul sau pe frontiera $\triangle P_{i-1} P_i P_{i+1}$

Orice poligon admite (cel puțin) o diagonală veritabilă.

Comentariu referitor la GL_FILL:

poligonul este "umplut" (filled)

Cum?

glShadeModel (GL_SMOOTH); (implicit)

→ combinate culorile vârfului:

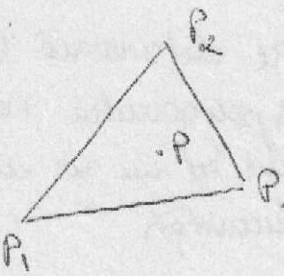
- descompunere în triunghiuri
- fiecare triunghi este colorat folosind algoritmul Gouraud

Algoritmul Gouraud pt triunghiuri

Fie P_1, P_2, P_3 vârfurile (pixel) unui triunghi colorate cu culorile c_1, c_2, c_3

(c_1, c_2, c_3 ~ RGB în $[0,1]$)

Fie P



P = punct în interiorul (sau pe frontiera Δ)

⇒ P se scrie unic

$P = \alpha_1 P_1 + \alpha_2 P_2 + \alpha_3 P_3$ cu $\alpha_1, \alpha_2, \alpha_3 \in [0,1]$

(P = combinație convexă) ⟶ culoarea sa va fi $\alpha_1 c_1 + \alpha_2 c_2 + \alpha_3 c_3$

(fiecare componentă va rămâne în intervalul $[0,1]$)

glShadeModel (GL_FLAT);

→ poligonul este umplut cu culoarea ultimului vârf

Se poate renunța la trasarea fețelor unor poligoane

glEnable (GL_CULL_FACE);
glCullFace (GL_FRONT);
... lista pol convexe
glDisable (GL_CULL_FACE);

→ elimină poligoanele văzute din față