Algoritmul Z-buffer pentru eliminarea zonelor acoperite

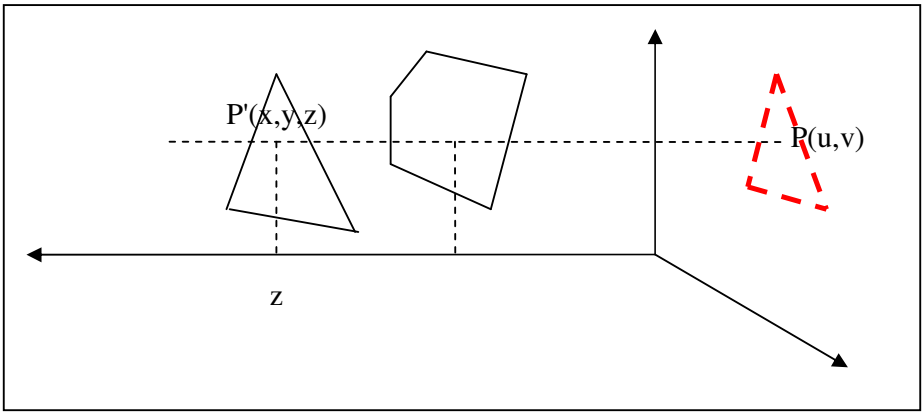
Presupuneri inițiale:

- observatorul este pe axa Oz
- planul de proiecție este xOy

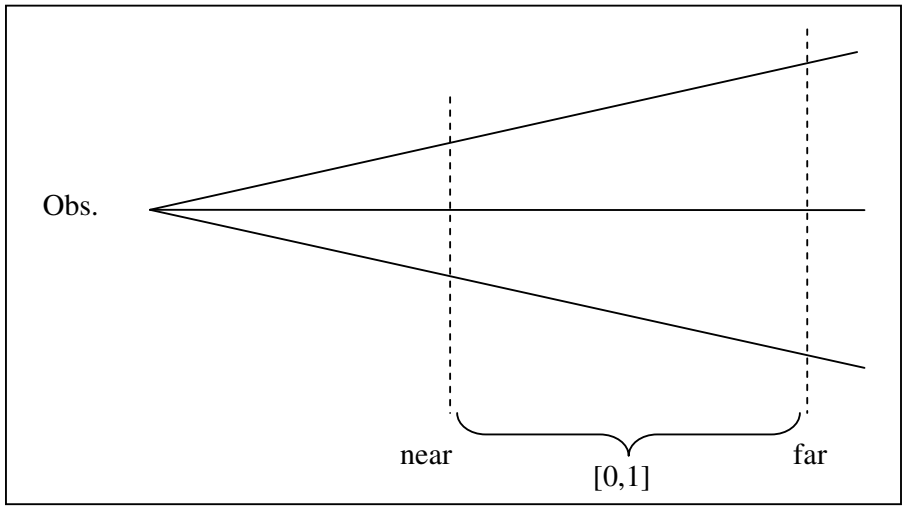
Desenul se realizează într-o zonă de pe ecran (poate fi întregul ecran).
Necesar de memorie pentru algoritm: pentru fiecare pixel din zona (de pe ecran) unde se desenează se alocă un număr real => un algoritm mare consumator de memorie. Fie A această matrice (zonă de memorie). Valoarea A(u,v) reprezintă cota z a ultimului punct desenat în pixelul (u,v).

Algoritmul Z-buffer:

1. Se inițializează fiecare element din matricea A la o valoare foarte îndepărtată de observator (cota **z** a unui punct aflat foarte departe de observator, deci a unui punct aflat după toate punctele obiectului care trebuie desenat).
2. Ecranul (zona de desenare) se inițializează cu o anumită culoare (fondul ecranului) corespunzătoare valorii inițiale a elementelor matricei A.
3. Se iau pe rând toate poligoanele (fețele) obiectului grafic (presupunem că obiectul grafic este delimitat de o mulțime de poligoane). Pentru fiecare poligon, care trebuie colorat cu o culoare c, se execută paşii următori:
   a. Se determină (nu se desenează) reprezentarea (proiecția) acestui poligon pe ecran. Această proiecție este formată dintr-o mulțime de puncte (pixeli).
   b. Pentru fiecare punct P(u,v) de pe ecran, aflat pe frontiera sau interiorul acestei proiecții
      - Se determină punctul P'(x,y,z) (un punct dintr-o infinitate de puncte) de pe poligonul curent care se proiectează în P(u,v). Se compară valoarea z cu valoarea A(u,v).
      - Dacă punctul P' (care se desenează în P(u,v)) este mai apropiat de observator decât ultimul punct desenat anterior în P, deci dacă valoarea z > A(u,v), atunci pixelul (u,v) se colorează cu o culoare c asociată punctului P' şi A(u,v)=z.



Observație. Volumul de vedere este normalizat astfel încât pe axa Oz valorile din volumul de vedere vor avea valorile z în intervalul [0,1] (sau [-1,1]). In acest caz valoarea inițială pentru elementele din matricea A poate fi 1.



Există mai multe **variante** ale acestui algoritm, care folosesc mai puțină memorie, dar timpul de calcul este mai mare:
1. Algoritmul Z-buffer pentru o linie, în care matricea A se reduce la o singură linie. In acest caz toate poligoanele obiectului grafic se parcurg pentru fiecare linie.
2. Fereastra de pe ecran se divide în diverse subzone, iar matricea A corespunde la o astfel de zonă. Pentru fiecare subzonă se parcurg toate poligoanele obiectului grafic.

Comenzi utile în OpenGL:

- **glEnable**(GL_DEPTH_TEST) - permite folosirea bufferul de adâncime (z-bufferul). La fiecare desenare cu OpenGL se va folosi algoritmul z-buffer;
- **glDisable**(GL_DEPTH_TEST) - interzice folosirea bufferul de adâncime. Din acest moment fiecare comandă de desenare a unei primitive se va executa fără teste cu privire la vizibilitatea acestei primitive, deci se vede efectul comenzilor de desenare în ordinea în care se execută.
- **glClear**(GL_DEPTH_BUFFER_BIT) - şterge bufferul de adâncime (z-bufferul)
- **glDepthFunc**(***functie***):
  La determinarea unei valori z care trebuie comparată cu valoarea care există în buffer se foloseşte o funcție, precizată de această comandă. Un anumit punct se desenează dacă trece de testul:

[valoare din buffer] [operator relațional precizat de funcție] [valoare z curentă pt. punct]

Functile posibile sunt:

- GL_LESS - punctul se desenează dacă e mai aproape de observator
- GL_NEVER - punctul nu se desenează
- GL_GREATER - punctul se desenează dacă e mai departe de observator
- GL_EQUAL - operatorul pentru comparație: =
- GL_NOTEQUAL - operatorul pentru comparație: <>
- GL_GEQUAL - operatorul pentru comparație: >=
- GL_LEQUAL - operatorul pentru comparație: <=
- GL_ALWAYS - testul trece indiferent de valoare (identic cu comanda **glDisable**(GL_DEPTH_TEST)