|  |  |  |
| --- | --- | --- |
|  | **Universitatea *Transilvania* din Brașov**  **Facultatea de Inginerie Electrică și Știința Calculatoarelor**  **Departamentul Automatică și Tehnologia Informației** |  |

**Transformări Grafice Bidimensionale în OpenGL.**

**Practică de domeniu 2023**

|  |  |
| --- | --- |
| Cosferenț Roxana Adelina  *Gr. 4LF412, AIA II* |  |

**BRAȘOV, 2023**

Cuprins

[I.INTRODUCERE 2](#_Toc143269640)

[1. Ce este OpenGL? 2](#_Toc143269641)

[2. Gestionarea Stării și Desenarea Obiectelor Geometrice 2](#_Toc143269642)

[2.1 Ștergerea Ferestrei 2](#_Toc143269643)

[2.2 Specificarea unei Culori 2](#_Toc143269644)

[2.3 Primitivele Geometrice de Desen OpenGL 3](#_Toc143269645)

[3. Percepția Culorilor 4](#_Toc143269646)

[3.1 Computer Color 4](#_Toc143269647)

[4. Iluminarea 5](#_Toc143269648)

[5. Crearea Surselor de Lumină 6](#_Toc143269649)

[II. TEMA PROIECTULUI 7](#_Toc143269650)

[1. Introducere 7](#_Toc143269651)

[2. Codul Proiectului 8](#_Toc143269652)

[Inițializarea Coordonatelor Pătratului 9](#_Toc143269653)

[Variabile pentru Transformări 9](#_Toc143269654)

[Funcția drawSquare() 9](#_Toc143269655)

[Funcția display() 10](#_Toc143269656)

[Funcția reshape() 10](#_Toc143269657)

[Funcția keyboard() 10](#_Toc143269658)

[Funcția main() 10](#_Toc143269659)

[3. Rezultat 10](#_Toc143269660)

[4. Concluzie 11](#_Toc143269661)

[III. Bibliografie 12](#_Toc143269662)

# I.INTRODUCERE

## Ce este OpenGL?

OpenGL este o interfață software către hardware-ul grafic. Această interfață este alcătuită din aproximativ 150 de comenzi distincte pe care le utilizați pentru a specifica obiectele și operațiile necesare pentru a produce aplicații interactive tridimensionale.

OpenGL este conceput ca o interfață optimizată, independentă de hardware, care poate fi implementată pe numeroase platforme hardware diferite. Pentru a atinge aceste calități, nu sunt incluse în OpenGL comenzi pentru efectuarea sarcinilor de gestionare a ferestrelor sau pentru obținerea intrărilor de la utilizator; în schimb, trebuie să lucrați prin sistemul de gestionare a ferestrelor care controlează hardware-ul specific pe care îl utilizați. Similar, OpenGL nu furnizează comenzi de nivel înalt pentru a descrie modele de obiecte tridimensionale. Astfel de comenzi ar putea permite specificarea de forme relativ complexe precum automobile, părți ale corpului, avioane sau molecule. Cu OpenGL, trebuie să construiți modelul dorit folosind un set mic de primitive geometrice - puncte, linii și poligoane.

În unele implementări (cum ar fi cea cu X Window System), OpenGL este proiectat să funcționeze chiar și dacă computerul care afișează grafica pe care o creați nu este același cu cel pe care rulează programul vostru grafic. Acest lucru poate fi cazul dacă lucrați într-un mediu de calculatoare în rețea, în care multe computere sunt conectate între ele printr-o rețea digitală. În această situație, computerul pe care rulează programul vostru și care emite comenzile de desenare OpenGL este numit client, iar computerul care primește acele comenzi și realizează desenarea este numit server. Formatul pentru transmiterea comenzilor OpenGL (numit protocol) de la client la server este întotdeauna același, astfel încât programele OpenGL pot funcționa pe o rețea chiar și dacă clientul și serverul sunt computere diferite. Dacă un program OpenGL nu rulează pe o rețea, atunci există doar un singur computer, care este în același timp client și server.

## Gestionarea Stării și Desenarea Obiectelor Geometrice

## 2.1 Ștergerea Ferestrei

Desenarea pe un ecran de calculator este diferită față de desenarea pe hârtie în sensul că hârtia începe albă, iar tot ce trebuie să faceți este să desenați imaginea. Pe un computer, memoria care conține imaginea este de obicei umplută cu ultima imagine pe care ați desenat-o, deci de obicei trebuie să o ștergeți într-o culoare de fundal înainte de a începe să desenați noua scenă. Culoarea pe care o utilizați pentru fundal depinde de aplicație. Pentru un procesor de texte, s-ar putea să ștergeți cu alb (culoarea hârtiei) înainte de a începe să desenați textul. Dacă desenați o vedere dintr-o navă spațială, ștergeți cu negrul spațiului înainte de a începe să desenați stelele, planetele și navele spațiale extraterestre. Uneori s-ar putea să nu fie nevoie să ștergeți ecranul deloc; de exemplu, dacă imaginea reprezintă interiorul unei camere, fereastra grafică întreagă este acoperită pe măsură ce desenați toate pereții.

## 2.2 Specificarea unei Culori

Cu OpenGL, descrierea formei unui obiect care se desenează este independentă de descrierea culorii sale. Ori de câte ori se desenează un anumit obiect geometric, acesta este desenat folosind schema de colorare specificată în prezent. Schema de colorare poate fi la fel de simplă ca "desenați totul în roșu de camion de pompieri", sau la fel de complicată ca "presupuneți că obiectul este făcut din plastic albastru, că există un reflector galben orientat într-o anumită direcție și că există o lumină generală roșiatic-maronie în restul locurilor." În general, un programator OpenGL stabilește mai întâi culoarea sau schema de colorare și apoi desenează obiectele. Până când culoarea sau schema de colorare este schimbată, toate obiectele sunt desenate în acea culoare sau folosind acea schemă de colorare. Această metodă ajută OpenGL să obțină o performanță mai bună la desenare decât ar rezulta dacă nu ar urmări culoarea curentă.

De exemplu, pseudocodul:

|  |
| --- |
| ***Exemplul1*** |
| set\_current\_color(red);  draw\_object(A);  draw\_object(B);  set\_current\_color(green);  set\_current\_color(blue);  draw\_object(C); |

Desenează obiectele A și B în roșu și obiectul C în albastru. Comanda de pe a patra linie care setează culoarea curentă la verde este inutilizată. Pentru a seta o culoare, utilizați comanda glColor3f(). Aceasta primește trei parametri, toți numere în virgulă mobilă între 0.0 și 1.0. Parametrii sunt, în ordine, componente roșu, verde și albastru ale culorii. Puteți considera aceste trei valori ca specificând o "amestecare" de culori: 0.0 înseamnă să nu utilizați deloc acea componentă, iar 1.0 înseamnă să utilizați cât mai mult din acea componentă. Astfel, codul glColor3f(1.0, 0.0, 0.0); face cel mai strălucitor roșu pe care sistemul îl poate desena, fără componente verzi sau albastre. Toate zerourile fac negru; în contrast, toate unitățile fac alb. Setarea tuturor celor trei componente la 0.5 rezultă în gri (la jumătatea distanței între negru și alb). Iată opt comenzi și culorile pe care le-ar stabili.

|  |
| --- |
| ***Comenzi*** |
| glColor3f(0.0, 0.0, 0.0); negru  glColor3f(1.0, 0.0, 0.0); roșu  glColor3f(0.0, 1.0, 0.0); verde  glColor3f(1.0, 1.0, 0.0); galben  glColor3f(0.0, 0.0, 1.0); albastru  glColor3f(1.0, 0.0, 1.0); magentă  glColor3f(0.0, 1.0, 1.0); cyan  glColor3f(1.0, 1.0, 1.0); alb |

## 2.3 Primitivele Geometrice de Desen OpenGL

Acum că ați văzut cum să specificați vârfurile, totuși trebuie să știți cum să spuneți OpenGL să creeze un set de puncte, o linie sau un poligon din acele vârfuri. Pentru a face acest lucru, încadrați fiecare set de vârfuri între un apel către glBegin() și un apel către glEnd(). Argumentul transmis către glBegin() determină ce fel de primitivă geometrică este construită din vârfuri. De exemplu, Exemplul 2 specifică vârfurile pentru poligonul prezentat în Figura 1

|  |
| --- |
| ***Exemplul2*** |
| glBegin(GL\_POLYGON);  glVertex2f(0.0, 0.0);  glVertex2f(0.0, 3.0);  glVertex2f(4.0, 3.0);  glVertex2f(6.0, 1.5);  glVertex2f(4.0, 0.0);  glEnd(); |

A black and white image of a hexagon

Description automatically generatedFigura1

## Percepția Culorilor

## 3.1 Computer Color

Pe un ecran de calculator color, hardware-ul determină fiecare pixel de pe ecran să emită cantități diferite de lumină roșie, verde și albastră. Aceste valori sunt numite R, G și B. Adesea sunt împachetate împreună (uneori cu o a patra valoare, numită alfa sau A), iar valoarea împachetată se numește valoarea RGB (sau RGBA). Informația de culoare la fiecare pixel poate fi stocată fie în modul RGBA, în care valorile R, G, B și posibil A sunt păstrate pentru fiecare pixel, fie în modul de index de culoare, în care este stocat un singur număr (numit index de culoare) pentru fiecare pixel. Fiecare index de culoare indică o intrare într-o tabelă care definește un set specific de valori R, G și B. O astfel de tabelă se numește hartă de culori.

Valorile R, G și B pot varia de la 0.0 (niciunul) la 1.0 (intensitate maximă). De exemplu, R = 0.0, G = 0.0 și B = 1.0 reprezintă cel mai strălucitor albastru posibil. Dacă R, G și B sunt toate 0.0, pixelul este negru; dacă toate sunt 1.0, pixelul este desenat în cel mai strălucitor alb care poate fi afișat pe ecran. Amestecul de verde și albastru creează nuanțe de cyan. Albastru și roșu se combină pentru magenta. Roșu și verde creează galben. Pentru a vă ajuta să creați culorile dorite din componentele R, G și B, priviți cubul de culoare prezentat în Placă 12. Axele acestui cub reprezintă intensitățile roșu, albastru și verde. O versiune alb-negru a cubului este prezentată în Figura 2.

Comenzile pentru a specifica o culoare pentru un obiect (în acest caz, un punct) pot fi la fel de simple ca acestea:

|  |
| --- |
| ***Exemplul3*** |
| glColor3f (1.0, 0.0, 0.0); /\* culoarea curenta RGB e rosu: \*/  /\* total rosu, nu verde, nu albastru. \*/  glBegin (GL\_POINTS);  glVertex3fv (point\_array);  glEnd (); |

A diagram of a cube with different colors

Description automatically generatedFigura 2

## Iluminarea

A close-up of a sphere

Description automatically generated

Figura 3 : O Sferă Iluminată și o Sferă Neiluminată

După cum puteți observa, o sferă neiluminată nu arată diferit de un disc bidimensional. Acest lucru demonstrează cât de importantă este interacțiunea dintre obiecte și lumină în crearea unei scene tridimensionale.

Cu OpenGL, puteți manipula iluminarea și obiectele într-o scenă pentru a crea o varietate de efecte. Acest capitol începe cu o introducere în eliminarea suprafețelor ascunse. Apoi, explică cum să controlați iluminarea într-o scenă, discută modelul conceptual OpenGL de iluminare și descrie în detaliu cum să setați numeroasele parametri de iluminare pentru a obține anumite efecte. Spre sfârșitul capitolului, sunt prezentate calculele matematice care determină modul în care iluminarea afectează culoarea

Când desenați o scenă compusă din obiecte tridimensionale, unele dintre ele ar putea oculta întreg sau părți din altele. Schimbarea punctului de vedere poate schimba relația de ascundere. De exemplu, dacă vizionați scena din direcția opusă, orice obiect care era anterior în fața altuia se află acum în spatele acestuia.

Pentru a desena o scenă realistă, aceste relații de ascundere trebuie menținute. Să presupunem că codul vostru funcționează în felul următor:

|  |
| --- |
| ***Exemplul4*** |
| while (1) {  get\_viewing\_point\_from\_mouse\_position();  glClear(GL\_COLOR\_BUFFER\_BIT);  draw\_3d\_object\_A();  draw\_3d\_object\_B();  } |

Pentru unele poziții ale mouse-ului, obiectul A ar putea să oculteze obiectul B. Pentru altele, s-ar putea să fie invers. Dacă nu se face nimic special, codul anterior desenează întotdeauna obiectul B în al doilea rând (și, astfel, deasupra obiectului A), indiferent de poziția de vizualizare selectată. Într-un scenariu nefavorabil, dacă obiectele A și B se intersectează astfel încât o parte a obiectului A să oculte o parte a obiectului B și o parte a lui B să oculte A, schimbarea ordinei de desenare nu furnizează o soluție.

Eliminarea părților obiectelor solide care sunt ascunse de altele se numește eliminare de suprafață ascunsă.

Cel mai simplu mod de a realiza eliminarea suprafețelor ascunse este să utilizați bufferul de adâncime (uneori numit z-buffer).

Un buffer de adâncime funcționează prin asocierea unei adâncimi sau distanțe față de planul de vizualizare (de obicei planul de tăiere aproape) cu fiecare pixel de pe fereastră. Inițial, valorile de adâncime pentru toți pixelii sunt setate la cea mai mare distanță posibilă (de obicei planul de tăiere departe) folosind comanda glClear() cu GL\_DEPTH\_BUFFER\_BIT. Apoi obiectele din scenă sunt desenate în orice ordine.

Pentru a utiliza bufferul de adâncime, trebuie să activați bufferul de adâncime. Acest lucru trebuie făcut doar o dată. Înainte de a desena, de fiecare dată când desenați scena, trebuie să ștergeți bufferul de adâncime și apoi să desenați obiectele din scenă în orice ordine.

Pentru a converti exemplul de cod anterior pentru a efectua eliminarea suprafețelor ascunse, modificați-l în următorul mod:

|  |
| --- |
| ***Exemplul5*** |
| glutInitDisplayMode (GLUT\_DEPTH | .... );  glEnable(GL\_DEPTH\_TEST);  ...  while (1) {  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);  get\_viewing\_point\_from\_mouse\_position();  draw\_3d\_object\_A();  draw\_3d\_object\_B();  } |

Argumentul pentru glClear() șterge atât bufferul de adâncime, cât și bufferul de culoare.

## Crearea Surselor de Lumină

În OpenGL, puteți crea surse de lumină pentru a simula interacțiunea obiectelor cu lumina. Aceasta vă permite să adăugați efecte de iluminare realiste la scenele 3D și să evidențiați detaliile obiectelor. Sursa de lumină emite lumină în direcții diferite și poate avea diverse proprietăți, cum ar fi culoarea, intensitatea și poziția.

Pentru a crea o sursă de lumină, trebuie să specificați tipul sursei de lumină (cum ar fi sursa direcțională, punctuală sau reflectată) și proprietățile ei asociate.

Sursele de lumină au mai multe proprietăți, cum ar fi culoarea, poziția și direcția. Următoarele secțiuni explică cum să controlați aceste proprietăți și cum arată lumina rezultată. Comanda folosită pentru a specifica toate proprietățile luminilor este “glLight\*()”; aceasta primește trei argumente: pentru a identifica sursa de lumină a cărei proprietate este specificată, proprietatea și valoarea dorită pentru acea proprietate.

Creează sursa de lumină specificată de "light”, care poate fi „GL\_LIGHT0”, ”GL\_LIGHT1”, ... , sau `GL\_LIGHT7`. Caracteristica luminii care se setează este definită de „pname”, care specifică un parametru numit (vezi Tabelul 1). `param` indică valorile la care se setează caracteristica `pname`; este un pointer către un grup de valori în cazul folosirii versiunii vectoriale, sau valoarea în sine dacă se folosește versiunea non-vectorială. Versiunea non-vectorială poate fi folosită doar pentru a seta caracteristici ale luminii cu o singură valoare.

|  |  |  |
| --- | --- | --- |
| Numele parametruui | Default Value | Definitie |
| GL\_AMBIENT | (0.0, 0.0, 0.0, 1.0) | intensitatea RGBA ambientală a luminii |
| GL\_DIFFUSE | (1.0, 1.0, 1.0, 1.0) | intensitatea difuză a RGBA a luminii |
| GL\_SPECULAR | (1.0, 1.0, 1.0, 1.0) | intensitatea RGBA speculară a luminii |
| GL\_POSITION | (0.0, 0.0, 1.0, 0.0) | (x, y, z, w ) poziția luminii |
| GL\_SPOT\_DIRECTION | (0.0, 0.0, -1.0) | (x, y, z ) direcția reflectoarelor |
| GL\_SPOT\_EXPONENT | 0.0 | exponent de lumina reflectoarelor |
| GL\_SPOT\_CUTOFF | 180.0 | unghiul de întrerupere a reflectoarelor |
| GL\_CONSTANT\_ATTENUATION | 1.0 | factor de atenuare constantă |
| GL\_LINEAR\_ATTENUATION | 0.0 | factor de atenuare liniară |
| GL\_QUADRATIC\_ATTENUATION | 0.0 | factorul de atenuare quadratică |

***Tabel 1: Valori implicite pentru pname parameter of glLight\****

Notă: Valorile implicite enumerate pentru GL\_DIFFUSE și GL\_SPECULAR în Tabelul 5-1 se aplică doar pentru GL\_LIGHT0. Pentru alte lumini, valoarea implicită este (0.0, 0.0, 0.0, 1.0) atât pentru GL\_DIFFUSE cât și pentru GL\_SPECULAR.

Exemplul 6 arată cum să folosiți funcția glLight\*():

|  |
| --- |
| ***Exemplul6: Definirea culorilor și a poziției pentru o sursă de lumină*** |
| GLfloat light\_ambient[] = { 0.0, 0.0, 0.0, 1.0 };  GLfloat light\_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };  GLfloat light\_specular[] = { 1.0, 1.0, 1.0, 1.0 };  GLfloat light\_position[] = { 1.0, 1.0, 1.0, 0.0 };  glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);  glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);  glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular);  glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position); |

Matricile sunt definite pentru valorile parametrilor, iar glLightfv ( ) este apelat în mod repetat pentru a seta diverși parametri. În acest exemplu, primele trei apeluri către glLightfv ( ) sunt de prisos, deoarece sunt utilizate pentru a specifica valorile implicite pentru parametrii GL\_AMBIENT, GL\_DIFFUSE și GL\_SPECULAR. Nu uitați să porniți fiecare lumină cu glEnable ( ). Toți parametrii pentru glLight \* ( ) și valorile lor posibile sunt explicate în secțiunile următoare. Acești parametri interacționează cu cei care definesc modelul de iluminare generală pentru o anumită scenă și proprietățile materialului unui obiect.

# II. TEMA PROIECTULUI

## Introducere

În lumea graficii computerizate, iluminarea joacă un rol esențial în crearea scenei vizuale. Definirea și configurarea surselor de lumină în OpenGL permit controlul asupra modului în care obiectele interacționează cu lumina, determinând astfel aspectul și atmosfera unei scene. Acest aspect este crucial pentru a crea scene realiste și impresionante în aplicațiile 3D. Proiectul nostru implică crearea unei ferestre OpenGL în care vom desena un pătrat simplu. Dar aici vine partea interesantă: vom putea transforma acest pătrat într-o varietate de moduri, de la rotații și scalări până la translații. Mai mult, toate aceste transformări vor fi gestionate printr-o interacțiune ușoară cu tastatura.

## Codul Proiectului

Codul oferă o structură clară pentru crearea fereastră OpenGL și adăugarea obiectelor. Iată pașii principali din cod:

**1.Inițializare și Afișare:** Am început prin a inițializa fereastra OpenGL folosind biblioteca GLUT. De asemenea, am definit coordonatele colțurilor pătratului.

**2.Funcție de Desenare:** Am creat funcția drawSquare() care utilizează coordonatele pătratului pentru a desena acest obiect folosind glBegin(GL\_QUADS) și glEnd().

**3.Reconfigurare:** Am dezvoltat funcția reshape() pentru a reconfigura volumul de vizualizare atunci când fereastra este redimensionată.

**4.Interacțiunea Tastaturii:** Prin funcția keyboard(), am permis interacțiunea cu tastatura. Fiecare tasta ('r', 's', 't', 'u') declanșează transformări specifice asupra pătratului, urmate de reafisarea scenei.

|  |
| --- |
| ***COD*** |
| #include <GL/glut.h>  GLfloat squareVertices[] = {  -0.1f, -0.1f,  0.1f, -0.1f,  0.1f, 0.1f,  -0.1f, 0.1f  };  GLfloat angle = 0.0f;  GLfloat scaleX = 1.0f;  GLfloat scaleY = 1.0f;  GLfloat translateX = 0.0f;  GLfloat translateY = 0.0f;  void drawSquare() {  glBegin(GL\_QUADS);  glVertex2f(squareVertices[0], squareVertices[1]);  glVertex2f(squareVertices[2], squareVertices[3]);  glVertex2f(squareVertices[4], squareVertices[5]);  glVertex2f(squareVertices[6], squareVertices[7]);  glEnd();  }  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glMatrixMode(GL\_MODELVIEW);  glLoadIdentity();  // Aplică transformări  glTranslatef(translateX, translateY, 0.0f);  glRotatef(angle, 0.0f, 0.0f, 1.0f);  glScalef(scaleX, scaleY, 1.0f);  glColor3f(0.5f, 0.7f, 0.3f);  drawSquare();  glutSwapBuffers();  }  void reshape(int w, int h) {  glViewport(0, 0, w, h);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);  glMatrixMode(GL\_MODELVIEW);  }  void keyboard(unsigned char key, int x, int y) {  switch (key) {  case 'r':  angle += 5.0f;  break;  case 's':  scaleX += 0.1f;  scaleY += 0.1f;  break;  case 't':  translateX += 0.1f;  break;  case 'u':  translateY += 0.1f;  break;  }  glutPostRedisplay();  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(800, 600);  glutInitWindowPosition(100, 100);  glutCreateWindow("Transformări Bidimensionale în OpenGL");  glutDisplayFunc(display);  glutReshapeFunc(reshape);  glutKeyboardFunc(keyboard);  glutMainLoop();  return 0;  } |

### Inițializarea Coordonatelor Pătratului

La început, am definit coordonatele colțurilor pătratului folosind un array numit squareVertices. Fiecare colț este reprezentat de perechi de coordonate (x, y) pentru a forma pătratul.

### Variabile pentru Transformări

Am definit mai multe variabile globale care vor stoca informații despre transformările aplicate pătratului:

***angle:***Unghiul de rotație în grade.

**scaleX și scaleY:** Factorii de scalare pe axele x și y.

**translateX și translateY:** Valori pentru translație pe axele x și y.

### Funcția drawSquare()

Această funcție desenează pătratul folosind coordonatele definite anterior și glBegin(GL\_QUADS) pentru a începe definirea suprafețelor pătratului, iar glEnd() pentru a termina această definire.

### Funcția display()

Această funcție este apelată pentru a afișa obiectele pe ecran. În aceasta, se resetează matricea de modelare-vizualizare la identitate și apoi se aplică transformările: translatare, rotație și scalare în ordine.

### Funcția reshape()

Aceasta se activează atunci când fereastra este redimensionată. Ea reconfigurează volumul de vizualizare folosind glOrtho() pentru a păstra transformările într-un spațiu bidimensional.

### Funcția keyboard()

Această funcție preia tastele apăsate și efectuează transformările corespunzătoare asupra pătratului în funcție de tasta apăsată. De asemenea, reapelăm glutPostRedisplay() pentru a redesena scena cu noile transformări.

### Funcția main()

Această funcție inițializează fereastra OpenGL, stabilește funcțiile pentru afișare, redimensionare și interacțiunea tastaturii, și apoi intră în bucla principală a GLUT pentru a gestiona evenimentele.

## Rezultat

A green square on a black background

Description automatically generated

A green square on a black background

Description automatically generated

A green rectangle on a black background

Description automatically generated

## Concluzie

În concluzie, acest cod demonstrează modul în care pot fi aplicate transformări grafice bidimensionale într-o scenă OpenGL interactivă. Poți experimenta cu valorile variabilelor și cu interacțiunea tastaturii pentru a vedea cum se modifică comportamentul pătratului. Transformările grafice bidimensionale în OpenGL ne oferă posibilitatea de a controla poziția, dimensiunile și orientarea obiectelor într-un spațiu 2D. Acest proiect simplu demonstrează puterea transformărilor și modul în care pot fi folosite pentru a crea animații, jocuri și efecte vizuale captivante

# III. Bibliografie

<https://www.glprogramming.com/red/>  
<https://www.opengl.org/>

<http://nehe.gamedev.net/>  
<https://www.opengl.org/sdk/docs/tutorials/TyphoonLabs/Chapter_1.pdf>   
<https://github.com/amilajack/reading/blob/master/Graphics/The%20Book%20of%20Shaders.pdf>