Documentație Proiect P.G.

Snake Definitive Edition

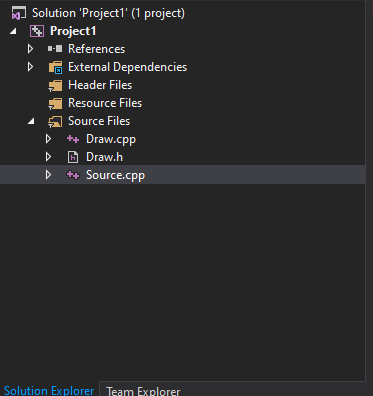
Dragoș Uleanu, I.E.S.C. T.I. Anul III

# Tema Proiectului

În cadrul lucrării practice pentru cursul de Proiectare Grafică, ne-am propus realizarea unui joc clasic utilizând openGl și alt librării asociate acestuia, precum și limbajul de programare C++. Proiectul a fost realizat în VisualStudio 2017.

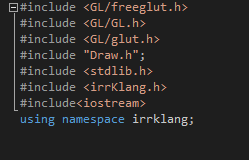
# Documentarea Codului:

Codul este structurat în 2 fișiere .cpp și un header (.h) ce conțin în integritate codul necesar funcționării aplicației. Acestea sunt denumite sugestiv Draw.cpp (responsabil de partea grafică a aplicației), Draw.h (partajează funcțiile grafice către main), Source.cpp (responsabil de logica aplicației).

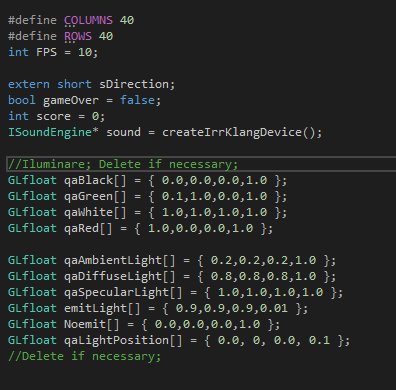


1. **Source.cpp**

Source.cpp este inima aplicației, fisierul .cpp care conține și funcția main, precum și celelalte funcții legate de utilizarea OpenGL;

Librăriile utilizate pentru realizarea acestei aplicații sunt:

* Freeglut.h
* GL.h
* Glut.h
* Draw.h (header făcut special)
* Stdlib.h
* irrKlang.h (pentru muzică și efecte speciale)
* iostream
* namespace-ul specific irrklang;

Cele două constante COLUMNS și ROWS reprezintă numărul de rânduri și de coloane ale matricei ce reprezintă tabla de joc.

FPS este viteza de deplasare care reprezintă numarul de cadre pe secundă.

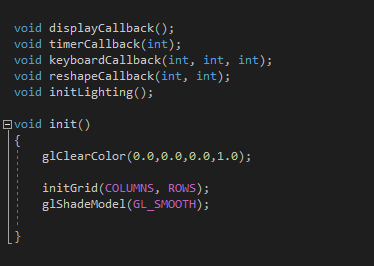
gameOver devine true în momentul în care una din condițiile de pierdere se adeverește.

Score reține scorul.

Sound se folosește de clasa ISoundEngine și de constructorul specific bibliotecii irrKlang pentru a reda fișiere audio

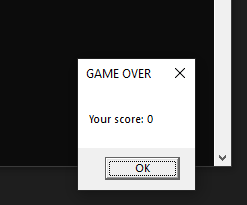
Urmează toate variabilele de tip GLFloat necesare iluminării (culorile de baza, negru, verde etc) precum și tipurile de iluminare.

În cele ce urmează vă vom prezenta funcțiile utilizate în Source.cpp.



1. init() reprezintă funcția de inițiere ce cuprinde atât funcția de curățare a culorii din buffer precum și funcțiile initGrid (inițializează matricea) și glShadeModel (aplică un model de Shader)
2. displayCallBack()
3. void displayCallback()
4. {
5. glMatrixMode(GL\_MODELVIEW); //delete if necessary;
6. glClear(GL\_COLOR\_BUFFER\_BIT);
7. glLoadIdentity(); //delete if necessary;
8. drawGrid();
9. drawSnake();
10. drawFood();
11. glutSwapBuffers();
12. if (gameOver)
13. {
14. char score1[10];
15. \_itoa\_s(score, score1, 10);
16. char message[50] = "Your score: ";
17. strcat\_s(message, score1);
18. sound->stopAllSounds();
19. sound->play2D("media/gameOver.wav");
20. MessageBox(NULL, message, "GAME OVER", 0);
22. exit(0);
23. }
24. }

Funcția displayCallback întoarce callback-ul necesar funcției glutDisplayFunc() din main. În interiorul funcției displayCallback se apelează funcția de desenare a matricei, a șarpelui, a mâncării, precum și funcțiile asociate iluminării. Tot aici se generează o fereastră nouă pentru terminarea jocului.

Tot aici se poate remarca obiectul sound care generează fișierul audio *gameOver.wav* în momentul în care if-ul se adeverește.

1. reshapeCallback()

void reshapeCallback(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0.0, COLUMNS, 0.0, ROWS,-1.0,1.0);

glMatrixMode(GL\_MODELVIEW);

}

Această funcție servește rolul de callback pentru glutReshapeFunc; În interiorul ei se fase trecerea de la o matrice la alta și se stabilesc axele de coordonate în sistemul de axe xOy. Funcția primeste doi parametri de tip int , w și h care reprezintă lungimea și lațimea ferestrei.

1. timerCallBack()
2. void timerCallback(int)
3. {
4. glutPostRedisplay();
5. glutTimerFunc(1000 / FPS, timerCallback,0);
6. }

Această funcție servește rolul de callback către glutTimerFunc() și are rolul de a stabili viteza de deplasare a șarpelui raportată la numărul de cadre pe secundă

4. keyboardCallback()

void keyboardCallback(int key, int, int)

{

switch (key)

{

case GLUT\_KEY\_UP:

if (sDirection != DOWN)

{

sDirection = UP;

}

break;

case GLUT\_KEY\_DOWN:

if (sDirection != UP)

{

sDirection = DOWN;

}

break;

case GLUT\_KEY\_RIGHT:

if (sDirection != LEFT)

{

sDirection = RIGHT;

}

break;

case GLUT\_KEY\_LEFT:

if (sDirection != RIGHT)

{

sDirection = LEFT;

}

break;

case GLUT\_KEY\_F2:

initLighting();

break;

case GLUT\_KEY\_F1:

exit(0);

break;

}

}

Această funcție trimite un CallBack către glutSpecialFunc(). Am preferat folosirea acestei functii pentru a evita necesitatea de atribuire a comenzilor către taste specifice, în cazul acesta existând deja taste special alese, printre care enumerăm sagețile, atât de necesare unui joc clasic. Identificarea tastei apăsate se face printr-un switch case, ieșirea din joc este pusă pe tasta F1, iluminarea pe F2.

5. initLighting()

void initLighting()

{

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, qaAmbientLight);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, qaDiffuseLight);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, qaSpecularLight);

glLightfv(GL\_LIGHT0, GL\_POSITION, qaLightPosition);

}

Aceasta funcție realizează iluminarea conform variabilelor legate de lumină. Funcția se apelează doar în cazul apăsării tastei F2.

6. main()

int main(int argc, char \*\*argv)

{

glutInit(&argc, argv); //initializare Glut

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE); //initializare Display

glutInitWindowSize(500, 500);

glutCreateWindow("Snake Definitive Edition");

glutDisplayFunc(displayCallback); //callback display

glutReshapeFunc(reshapeCallback);

glutTimerFunc(0, timerCallback, 0);

glutSpecialFunc(keyboardCallback);

init();

sound->play2D("media/sd1.mp3", true);

glutMainLoop();

return 0;

}

În funcția main sunt apelate toate funcțiile necesare realizării logicii jocului. Se inițializează prin Glut-ul, se inițializează display-ul, fereastra la dimensiunile 500x500 și se denumește ”Snake Definitive Edition”.

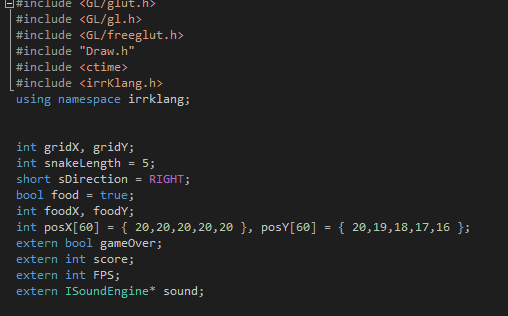
Se pot observa și funcțiile care necesită callback-urile menționate mai devreme.

După toate aceste apeluri se apelează și funcția init(); partea logică este rezolvată.

Folosindu-ne de un obiect sound, pe fundal se va repeta melodia „Welcome to the East” aparținând coloanei sonore a jocului Metin 2. Un deliciu rafinat pentru nostalgici.

Se declanșează evenimentul glut iar la final se returnează 0.

**2.Draw.cpp**



Librăriile utilizate sunt aceleași ca în cazul Source.cpp.

Variabilele utilizate reprezintă:

gridX,gridY – coordonatele colțurilor matricei.

snakeLength – dimensiunea inițială a șarpelui

sDirecțion – direcția de deplasare inițială

food – dacă există mâncare plasată deja pe tablă

foodX,foodY – coordonatele mâncării

posX,posY – coordonatele pătrățelelor ce alcătuiesc șarpele.

variabilele specifice Source.cpp „pasate” prin pointerul extern pentru a putea fi utilizate în Draw.cpp;

void initGrid(int x, int y)

{

gridX = x;

gridY = y;

}

Acționează ca un constructor pentru generarea Grid-ului.

void unit(int x, int y)

{

if (x == 0 || y == 0 || x == gridX - 1 || y == gridY - 1)

{

glLineWidth(3.0);

glColor3f(1.0, 0.0, 0.0);

}

else

{

glLineWidth(1.0);

glColor3f(1.0, 1.0, 1.0);

}

//glLineWidth(1.0);

//glColor3f(1.0, 1.0, 1.0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(x, y);

glVertex2f(x + 1, y);

glVertex2f(x + 1, y + 1);

glVertex2f(x, y + 1);

glEnd();

}

Funcția generează un pătrat la coordonatele x și y cu adăugarea că dacă pătratul trebuie generat drept contur, atunci acesta să fie roșu. De asemenea, pătratele roșii au o grosime a laturii mai mare decât cele normale. Am ales folosirea comenzii glVertex pentru a avea un control mai mare asupra coordonatelor unde se desenează.

void drawGrid()

{

for (int x = 0; x < gridX; x++)

{

for (int y = 0; y < gridY; y++)

{

unit(x, y);

}

}

}

Această funcție desenează pătrate până la coordonatele gridX și gridY;

void drawFood()

{

if (food == true)

{

random(foodX, foodY);

}

food = false;

glColor3f(1.0, 1.0, 1.0);

glRectf(foodX, foodY, foodX + 1, foodY + 1);

}

Această funcție generează un pătrățel de culoare albă, considerat mâncare dacă și numai dacă nu există deja pe tablă o „mâncare”. După generare, variabila booleana food devine false. De asemenea functia drawFood() apelează random(foodX,foodY) o funcție făcută special, utilizând o formulă de calcul pentru a face mâncarea să fie cu adevărat aleator plasată.

void drawSnake()

{

for (int i = snakeLength - 1; i > 0; i--)

{

posX[i] = posX[i - 1];

posY[i] = posY[i - 1];

}

if (sDirection == UP)

{

posY[0]++;

}

else if (sDirection == DOWN)

{

posY[0]--;

}

else if (sDirection == RIGHT)

{

posX[0]++;

}

else if (sDirection == LEFT)

{

posX[0]--;

}

for (int i = 0; i < snakeLength; i++)

{

if (i == 0)

{

glColor3f(0.0, 0.0, 1.0);

}

else

{

glColor3f(0.0, 1.0, 0.0);

}

glRectd(posX[i], posY[i], posX[i] + 1, posY[i] + 1);

}

//glColor3f(0.2, 0.0, 0.6);

//glRectd(posX, posY, posX + 1, posY + 1);

if (posX[0] == 0 || posX[0] == gridX - 1 || posY[0] == 0 || posY[0] == gridY - 1)

{

gameOver = true;

}

if (posX[0] == foodX && posY[0] == foodY)

{

sound->play2D("media/boop.wav");

score++;

snakeLength++;

FPS++;

if (score % 10 == 0)

{

sound->play2D("media/yaay.wav");

}

if (snakeLength > MAX)

{

snakeLength = MAX;

gameOver = true;

}

food = true;

}

for (int i = 1; i < snakeLength; i++)

{

if (posX[0] == posX[i] && posY[0] == posY[i])

{

gameOver = true;

}

}

}

Această funcție este folosită pentru desenarea, la fiecare pas, a șarpelui. De asemenea aici se realizează întreaga secvență de mișcare a șarpelui prin modificarea coordonatelor în funcție de direcția de deplasare. Se stabilește că primul pătrățel (capul) determină sensul de mișcare și contactul cu mâncarea/pereții (fiind reprezentat prin culoarea albastră) în timp ce restul corpului (verde) se sincronizează cu acesta. Dacă pătrățelul albastru atinge marginea gameOver devine true. La fiecare contact cu mâncarea se va da play la sunetul *boop.wav* și va crește scorul, lungimea și numarul de cadre pe secundă (viteza). Dacă scorul este multiplu de zece se va reda sunetul *yaay.wav*. Dacă lungimea șarpelui depășeșe constanta Max atunci gameOver devine true. După fiecare contact cu mâncarea food devine true și se trece la generarea unei noi locații. În momentul în care capul intersectează trunchiul se declară gameOver.

void random(int &x, int &y)

{

int maximX = gridX - 2;

int maximY = gridY - 2;

int minim = 1;

srand(time(NULL));

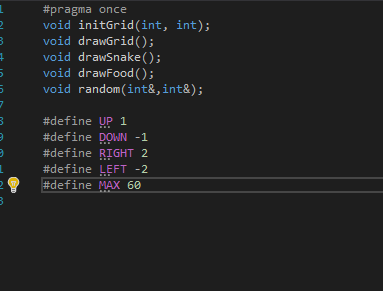
x = minim + rand() % (maximX - minim);

y = minim + rand() % (maximY - minim);

}

Această funcție se folosește de formula matematică dată pentru x și y pentru a stabili un grad cât mai mare de adevăr conceptului de „la întâmplare” omogenizând vârfurile critice ale funcției densitate de probabilitate.

**3. Draw.h**

În headerul Draw.h sunt trecute toate funcțiile din Draw.cpp cu scopul ca acestea să poată fi utilizate în orice alt .cpp adăugat proiectului.

De asemenea aici s-au declarat constantele universale ce servesc drept convenții pentru sensul de deplasare. În funcție de aceste valori, variabila sDirection decide în ce sens se deplasează șarpele.

Constanta Max servește ca limită a numărului de pătrățele ce alcătuiesc șarpele cât și ca număr maxim de cadre pe secundă (ochiul uman poate percepe între 30-60 de FPS).

# Bibliografie:

* Cursul de P.G. al Doamnei Profesor Doctor Inginer Popa Luminița.
* Learnopengl.com
* The Pentamollis Project, Youtube.
* Opengl.org
* Stackoverflow.com