Mihoc Tudor Dan
MLR5060 - Grafică pe calculator
February 24, 2022

Laboratorul 1

Rezolvați exercițiile de mai jos. Termenul de predare este Laboratorul 2. Se poate întârzia cu o penalizare de 2 puncte până la Laboratorul 3.

Exercițiul 1. Instalați API-ul OpenGL

* **Ubuntu**
  Se instalează pachetele necesare introducând în terminal comanda:

  sudo apt-get install libglew-dev freeglut3-dev mesa-common-dev libglu1-mesa-dev

  Se poate folosi orice IDE (pentru C++) pentru a edita, compila și executa sursele. Nu uitați să faceți legătura între proiectele voastre și bibliotecile de funcții încărcate.
* **Windows**
  1) Instalați și configurați MSYS.
  2) Instalați Pacman în MSYS2.
     ```
     pacman -Syu
     ```
  3) Instalați compilatorul MinGW folosind CLI.
     ```
     pacman -S mingw-w64-x86_64-toolchain
     ```
  4) Instalați biblioteca freeglut.
     ```
     pacman -S mingw-w64-x86_64-freeglut
     ```
  5) Instalați biblioteca glew.
     ```
     pacman -S mingw-w64-x86_64-glew
     ```
  6) Instalați biblioteca glfw.
     ```
     pacman -S mingw-w64-x86_64-glfw
     ```
  7) Instalați biblioteca GLAD. Pentru asta urmați următorii pași:
     - folosiți serviciul web pentru GLAD de la adresa glad.dav1d.de:
     - alegeți ca limbaj de programare C++
     - la secțiunea API alegeți o versiune de OpenGL (minim 3.3)
     - alegeti ca profil Core și bifati opțiunea Generate a loader option.
     - ignorați partea cu extensii (deocamdată) și apăsați butonul Generate pentru a obține fișierele necesare.
     - copiați ambele foldere (glad și KHR) în include (unde e mingw-ul instalat de exemplu) și adăugați fișierul glad.c la proiectul vostru.
  8) Instalați biblioteca GLM pentru funcțiile matematice.
     ```
     pacman -S mingw-w64-x86_64-glm
     ```
  9) Instalați Code Blocks (sau alt IDE) și configurați-l astfel încât să folosească compilatorul proaspăt instalat. Nu uitați să faceți legătura între proiectele voastre și bibliotecile de funcții încărcate.
     ```
     -lfreeglut -lglew32 -lopengl32 -lglfw3 -lglu32
     ```

Pentru alte variante de a configura un sistem pentru programare grafică puteți urmări instrucțiunile din [1] sau de pe site-ul autorului acestei cărți learnopengl.com.

Exercițiul 2. Realizați o aplicație simplă ce deschide o fereastră si o colorează în roșu. În Listing 1 aveți un exemplu de cod. Modificați culoarea de fundal cu cea cerută.

```
#include <GLFW/glfw3.h>
int main(void)
{
    GLFWwindow* window;
    /* Initialize the library */
    if (!glfwInit())
    return -1;
    /* Create a windowed mode window and its OpenGL context */
    window = glfwCreateWindow(640, 480, "Hello World", NULL, NULL);
    if (!window)
    {
        glfwTerminate();
        return -1;
    }
    /* Make the windows context current */
    glfwMakeContextCurrent(window);

    /* Loop until the user closes the window */
    while (!glfwWindowShouldClose(window))
    {
        /* Render here */
        glClearColor(0.5f, 0.3f, 0.3f, 1.0f);
        glClear(GL_COLOR_BUFFER_BIT);
        /* Swap front and back buffers */
        glfwSwapBuffers(window);
        /* Poll for and process events */
        glfwPollEvents();
    }
    glfwTerminate();
    return 0;
}
```

Listing 1. Codul sursă în C++ pentru a crea o fereastră folosind opengl.

Exercițiul 3. În Listing 2 aveți un exemplu de cod cum să desenați o linie cu ajutorul algoritmului lui Bresenham [2], punct cu punct, în *the fixed pipeline*.

!!! Observați că exemplul e **pur didactic** - *the fixed pipeline* este depreciată, nu se mai utilizează. Însă până vom învața metode mai moderne o vom utiliza.

Folosind acest exemplu, desenați un cerc în această fereastră folosind algoritmul lui Bresenham pentru cerc.

```
#include <GLFW/glfw3.h>
#include <GL/glut.h>
void init(void)
{
  glClearColor(1.0,1.0,1.0,0.0);
  glMatrixMode(GL_PROJECTION);
  gluOrtho2D(0.0,400.0,0.0,400.0);
}
void setPixel(GLint x,GLint y)
```

```
{
  glBegin(GL_POINTS);
     glVertex2i(x,y);
  glEnd();
}

void line()
{
  int x0 = 50, y0=50, xn = 300, yn = 150, x, y;
  int dx, dy,   //deltas
  pk,   //decision parameter
    k;    //looping variable

  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f( 1 ,0, 0);
  setPixel(x0, y0); //plot first point

  // difference between starting and ending points
  dx = xn - x0;
  dy = yn - y0;
  pk = 2 * dy - dx;
  x = x0; y = y0;

  for ( k = 0; k < dx-1; ++k ) {
    if ( pk < 0 ) {
      pk = pk + 2 * dy;     //calculate next pk
          //next pixel: (x+1, y )
    } else {
                //next pixel: (x+1, y+1)
      pk = pk + 2*dy - 2*dx;    //calculate next pk
      ++y;
    }
    ++x;
    setPixel( x, y );
  }

  glFlush();
}


int main(void)
{
    GLFWwindow* window;
    /* Initialize the library */
    if (!glfwInit())
    return -1;
    /* Create a windowed mode window and its OpenGL context */
    window = glfwCreateWindow(400, 400, "Bresenham's Line algorithm, works only
   for |m| < 1", NULL, NULL);
    if (!window)
    {
        glfwTerminate();
        return -1;
    }
    /* Make the windows context current */
    glfwMakeContextCurrent(window);

    /* set up the initial conditions (color of the background), projection mode,
    */
    init();
```

```
68
69      /* Loop until the user closes the window */
70      while (!glfwWindowShouldClose(window))
71      {
72          /* Render here */
73          line();
74          /* Swap front and back buffers */
75          glfwSwapBuffers(window);
76          /* Poll for and process events */
77          glfwPollEvents();
78      }
79      glfwTerminate();
80      return 0;
81  }
```

Listing 2. Codul sursă în C++ pentru a crea o fereastră folosind opengl.

References

[1] Joey de Vries. *Learn OpenGL - graphics programming*. Kendall & Wells, 2020.
[2] JavaTpoint. www.javatpoint.com.