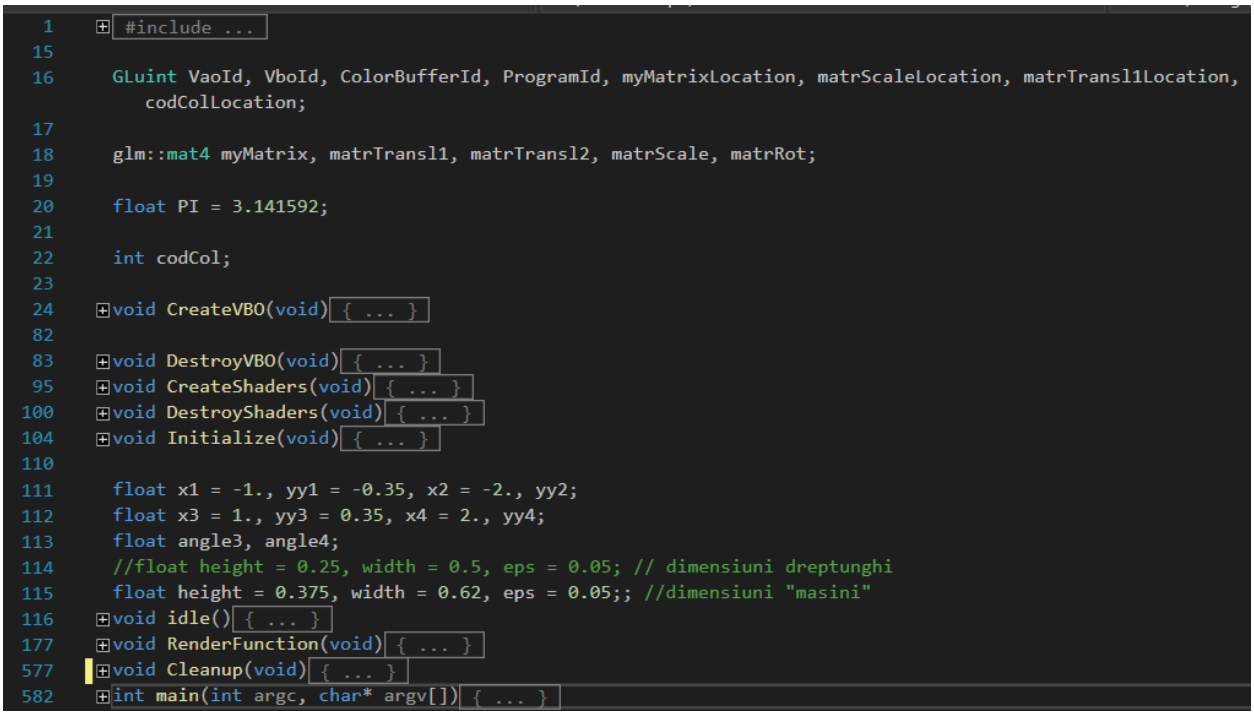
Documentatie tema 3

Lungu Andrei, grupa 332

Strucutra generala a codului:



In functia CreateVBO() initalizez varfurile celor 2 dreptunghiuri “de baza”, punctelor de delimitare dintre “benzile de pe acelasi sens” si benzilor de delimitare sensuri (se afla in jurul axei OX) si bufferele + legari VAO.
In functia DestroyVBO() distrug bufferele create anterior.

In functiile CreateShaders(), DestroyShaders() creez, respectiv distrug shaderele de varfuri si fragment ale aplicatiei.

Functia de initializare a fost folosita pentru setarea culorii de fond a ecranului si aplicarea metodelor de creare VBO si shaders.

Functia idle este cea care se ocupa de miscarea dreptunghiurilor. Cat timp dreptunghiul lent este in cadru este deplasat usor spre dreapta, respectiv stanga. Atunci cand dreptunghiul rapid este foarte aproape de cel lent intra “in depasire”, trece pe alta banda, deplasandu-se in “sus” (sau in “jos”) cu inaltimea dreptunghiului + un factor epsilon (pt a evita coliziunile), facandu-se si o usoara rotatie.

v. deplasare stanga -> dreapta

```
116  void idle()
117  {
118      //deplasare stanga --> dreapta
119      //dreptunghiul care merge lent
120      float aa = x2 - width, bb = x2 + width;
121      float cc = bb - 2 * (bb - aa) / 3, dd = bb - (bb - aa) / 3;
122      if (x1 <= 2) //inca este in cadru, il deplasez la dreapta ft usor
123      {
124          yy2 = yy1; // dreptunghiul rapid pe aceeasi banda cu cel lent
125          angle3 = 0.;
126          if (x1 > aa && x1 < bb)//intra in depasire
127          {
128              yy2 = yy2 + height + eps;//trece pe alta banda
129              if (x1 > dd && x1 < bb)
130                  //angle3 = -50.;
131                  angle3 += 0.25;
132              else if (x1 > cc && x1 < dd)
133                  angle3 = 0.;
134              else if (x1 > aa && x1 < cc)
135                  //angle3 = 50.;
136                  angle3 -= 0.25;
137          }
138          x1 += 0.0001;
139      }
140      else //il readuc in cadru
141          x1 = -1.5;
142
143      if (x2 <= 2) //dreptunghiul rapid este in cadru
144          x2 += 0.0004;
145      else
146          x2 = -1.5;
```

v. deplasare dreapta -> stanga

```
148     //deplasare dreapta --> stanga
149     float a = x4 - width, b = x4 + width;
150     float c = b - 2*(b-a)/3, d = b - (b-a)/3; // impart intervalul [a,b] in 4 : [a, c, d, b]
151     if (x3 >= -2) //inca este in cadru, il deplasez la stg ft usor
152     {
153         yy4 = yy3;
154         angle4 = 0.;
155         if (x3 > a && x3 < b)//intra in depasire
156         {
157             yy4 = yy3 - height - eps;//trece pe alta banda
158             if (x3 > d && x3 < b) //x3 in [d,b]
159                 angle4 = 50.;
160             else if (x3 > c && x3 < d) // x3 in [c, d]
161                 angle4 = 0.;
162             else if (x3 > a && x3 < c) // x3 in [a, c]
163                 angle4 = -50;
164         }
165         x3 -= 0.0001;
166     }
167     else //il readuc in cadru
168         x3 = 1.5;
169
170     if (x4 >= -2)
171         x4 -= 0.0004;
172     else
173         x4 = 1.5;
174
175     glutPostRedisplay();
176 }
```

Cu ajutorul functiei RenderFunction() desenez primitivele, aplic transformarile pentru a simula animatia (compunere de translatii si scalari sau translatii, rotatii si scalari pentru “masinile” care depasesc autovehiculul lent), fac inmultirea matricelor in programul principal si transmit shaderului de fragment culoarea pt primitiva respectiva.

Functia Cleanup este folosita pentru a distruge shaderele si vbo.

Main :

```
int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Tema 3 Lungu Andrei");
    glewInit();
    Initialize();
    glutDisplayFunc(RenderFunction);
    glutIdleFunc(idle); // pt deplasarea drept
    glutCloseFunc(Cleanup);
    glutMainLoop();
}
```