Politechnika Świętokrzyska **Wydział Elektrotechniki, Automatyki i Informatyki**	
Laboratorium: Podstawy grafiki komputerowej **Temat:** Silnik graficzny 3D	
Wykonawcy: Przemysław Kałuziński Michał Kaczor Grzegorz Kalarus	**Grupa:** 3ID13A
Data oddania: 22.01.2023	

Spis treści

Wstęp

Silnik 3D był tworzony zgodnie z założeniami. Wykoane zostały podstawowe elementy silnika. Zostało dodane rysowanie prymitywów, sześcianu oraz jego transformacje. Użytkownik może poruszać kamerą w trakcie trybu 3d, zarówno przyciskami oraz sterując myszą. Silnik 3D został napisany w OpenGL z użyciem biblioteki FreeGLUT.

Laboratorium 7:

inicjacja biblioteki odpowiedzialnej za system okienkowy i obsługę wejścia od użytkownika	`Engine::Engine(int w, int h)`
	`{`
	`this->width = w;`
	`this->height = h;`
	`glutInitWindowSize(w, h);`
	`glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);`
	`glDisable(CURSOR_SHOWING);`
	
	`glutCreateWindow("Silnik3D_Grafika");`
	`glClearColor(1.0f, 0.0f, 1.0f, 1.0f);`
	`glutReshapeFunc(resize);`
	`glutDisplayFunc(draw);`
	`glutIdleFunc(Idle);`
	`glutSpecialFunc(OnSpecialKey);`
	`glutKeyboardFunc(OnKeyBoard);`
	`glutCloseFunc(OnClose);`
	`glutPassiveMotionFunc(mouse);`
	
	`glutTimerFunc(1000 / FPS, OnTimer, 0);`
	`glutSetOption(GLUT_ACTION_ON_WINDOW_CLOSE, GLUT_ACTION_GLUTMAINLOOP_RETURNS);`
	`glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);`
	`swapPrespectiveMode();`
	`glFrontFace(GL_CCW);`
	`glEnable(GL_DEPTH_TEST);`
	`camera = Camera(w, h, glm::vec3(0.0f, 0.0f, 0.0f));`
	`}`

parametryzowanie trybu graficznego (np. w oknie, pełny ekran, wybór rozdzielczości, rzutowania itp.)	`void Engine::toggleFullScreen()`
	`{`
	`  glutFullScreenToggle();`
	`}`
	`void Engine::resize(int w, int h)`
	`{`
	`  Engine::width = w;`
	`  Engine::height = h;`
	`  camera.setWidth(w);`
	`  camera.setHeight(h);`
	`  glViewport(0, 0, w, h);`
	`}`

parametryzowanie innych rzeczy (np. liczba klatek animacji na sekundę, uruchomienie obsługi myszy/klawiatury, wielokrotne buforowanie, bufor Z itp.)	`void Engine::setFPS(int FPS)`
	`{`
	`  Engine::FPS = FPS;`
	`  glutTimerFunc(1000 / FPS, OnTimer, 0);`
	`}`

główna pętla gry korzystająca z czasomierza	`void Engine::mainLoop()`
	`{`
	`  glutMainLoop();`
	`}`

obsługa klawiatury i myszy	`void Engine::mouse(int x, int y)`
	`{`
	`  if (camera.getPerspective())`
	`  {`
	`  int x1 = glutGet(GLUT_WINDOW_X);`
	`  int y1 = glutGet(GLUT_WINDOW_Y);`
	`  pos.x -= (x - width / 2) / 1.0f;`
	`  pos.y -= (y - height / 2) / 1.0f;`
	`  SetCursorPos(width / 2 + x1, height / 2 + y1);`
	`  }`
	`  else`
	`  {`
	`  camera.setPosition({ 0,0,0 });`
	`  camera.setOrientation({ 0,0,0 });`
	`  }`
	`}`
	`void Engine::OnKeyBoard(unsigned char key, int x, int y)`
	`{`
	`  if (key == '1')`
	`  counter--;`
	`  if (key == '2')`
	`  counter++;`
	`  if (key == 'w')`
	`  pos.z += 3;`
	`  if (key == 's')`
	`  pos.z -= 3;`
	`  if (key == 'r')`
	`  {`
	`  pos.x = 0;`
	`  pos.y = 0;`
	`  pos.z = 0;`
	`  camera.setPosition(pos);`
	`  }`
	`  if (key == 0x1B)`
	`  exit(0);`
	`  camera.Inputs(key);`
	`}`

obsługa czyszczenia ekranu do zadanego koloru	`void Engine::setBackGroundColor(glm::vec4 color)`
	`{`
	`  glClearColor(color.r, color.g, color.b, color.a);`
	`}`

obsługa zmiany aktywnego rzutowania	`void Engine::swapPrespectiveMode()`
	`{`
	`  if (camera.getPerspective())`
	`  swapToOrtogonal();`
	`  else`
	`  swapToPerspectiv();`
	`}`

zamknięcie gry (deinicjacja biblioteki odpowiedzialnej za system okienkowy, sprzątanie pamięci itp.)	`void Engine::OnClose()`
	`{`
	
	`}`

Laboratorium 8:

Zaimplementuj w silniku 3D stosowne klasy, które umożliwią łatwe rysowanie prymitywów i obiektów wspomnianych w sekcji 4. Użyj tablic wierzchołków i kolorów.	`/** \brief Klasa Drawer`
	` *`
	` * Klasa Drawer jest klasa sluzaca do rysowania prymitywow.`
	` *`
	` */`
	`class Drawer`
	`{`
	`public:`
	`  static void drawTriangles(glm::vec3 tab[], glm::vec3 color[], int n = 4);`
	`  static void drawTriangles(glm::vec3 tab[], glm::vec3 color, int n = 4);`
	`  static void drawTrianglesStrip(glm::vec3 tab[], glm::vec3 color[], int n = 4);`
	`  static void drawTrianglesStrip(glm::vec3 tab[], glm::vec3 color, int n = 4);`
	`  static void drawTrianglesFan(glm::vec3 tab[], glm::vec3 color[], int n = 4);`
	`  static void drawTrianglesFan(glm::vec3 tab[], glm::vec3 color, int n = 4);`
	`  static void drawLines(glm::vec3 tab[], glm::vec3 color[], int size, int n);`
	`  static void drawLines(glm::vec3 tab[], glm::vec3 color, int size, int n);`
	`  static void drawLinesLoop(glm::vec3 tab[], glm::vec3 color[], int size, int n = 4);`
	`  static void drawLinesLoop(glm::vec3 tab[], glm::vec3 color, int size, int n = 4);`
	`  static void drawPoints(glm::vec3 tab[], glm::vec3 color[], int size, int n);`
	`  static void drawPoints(glm::vec3 tab[], glm::vec3 color, int size, int n);`
	`  static void drawCube(glm::vec3 tab[], glm::vec3 cubeNorm[], glm::vec3 color[], int index[]);`
	`  static void drawCube(glm::vec3 tab[], glm::vec3 cubeNorm[], glm::vec3 color, int index[]);`
	`  static void drawCubeLines(glm::vec3 tab[], glm::vec3 cubeNorm[], glm::vec3 color, int index[]);`
	`  static void drawCubeWithLines(glm::vec3 tab[], glm::vec3 cubeNorm[], glm::vec3 color[], int index[]);`
	`};`

Zaimplementuj w silniku 3D klasę reprezentującą sześcian, który będzie obiektem indeksowanym. Użyj tablic wierzchołków,	`/** \brief Klasa Cube`
	` *`
	` * Klasa Cube jest klasa szescianu.`
	` *`
	` */`
	`class Cube :`

ścian, normalnych i kolorów.	`public Object`
	`{`
	`glm::mat4 matrix;/**< Matrix przechowujacy matrix szescianu */`
	`glm::vec3 points[8];/**< Tablica wektorow punktow szescianu */`
	`glm::vec3 norms[36];/**< Tablica wektorow norm szescianu */`
	`glm::vec3 colors[8];/**< Tablica wektorow kolorow szescianu */`
	`static int index[];/**< Statyczna tablica liczb calkowitych przechowujaca indexy */`
	`float r;/**< Zmienna zmiennoprzecnikowa uzywana w rotacji */`
	`public:`
	`Cube(float x, float y, float z);`
	
	`void translate(glm::vec3 p);`
	
	`void rotate(float degree, glm::vec3 p);`
	
	`void scale(glm::vec3 p);`
	
	`void draw(glm::mat4 view);`
	`};`

Laboratorium 9:

Zaimplementuj klasę reprezentującą obserwatora, która umożliwi łatwe konfigurowanie i zmianę pozycji kamery w scenie 3D.	`/** \brief Klasa Camera`
	` *`
	` * Klasa Camera sluzy za widok uzytkownika.`
	` *`
	` */`
	`class Camera`
	`{`
	`glm::mat4 view;/**< Matrix przechowujacy wspolrzedne widoku */`
	`glm::mat4 proj;/**< Matrix przechowujacy wspolrzedne projekcji */`
	`glm::vec3 Position = { 0, 0, 0 };/**< Wektor przechowujacy pozycje */`
	`glm::vec3 Orientation = glm::vec3(0.0f, 0.0f, 1.0f);/**< Wektor przechowujacy orientacje */`
	`glm::vec3 Up = glm::vec3(0.0f, 1000.0f, 1000.0f);/**< Wektor przechowujacy gdzie znajduje sie gora */`
	`float FOVdeg = 60;/**< Zmienna zmiennoprzecinkowa przechowujaca kat widzenia */`
	`float nearPlane = 1;/**< Zmienna zmiennoprzecinkowa przechowujaca najblizszy punkt */`
	`float farPlane = 1000;/**< Zmienna zmiennoprzecinkowa przechowujaca najdalszy punkt */`
	`int width;/**< Zmienna calkowita przchowujaca szerokosc */`
	`int height;/**< Zmienna calkowita przchowujaca wysokosc */`
	`bool perspective = false;/**< Zmienna bool przechowujaca rodzaj perspektywy */`
	`float speed = 2.5f;/**< Zmienna zmiennoprzecinkowa przechowujaca predkosc poruszania kamery */`
	`float sensitivity = 100.0f;/**< Zmienna zmiennoprzecinkowa przechowujaca czulosc */`
	`public:`
	`Camera(int width, int height, glm::vec3 position);`
	`void Matrix();`

	`void Inputs(int key);`
	`void setWidth(int width);`
	`void setHeight(int height);`
	`void setFOVdeg(int FOVdeg);`
	`void changePerspective();`
	`bool getPerspective();`
	`void setPosition(glm::vec3 Position);`
	`void setView(glm::mat4 v);`
	`glm::vec3 getPosition();`
	`glm::mat4 getView();`
	`glm::mat4 getProjection();`
	`glm::vec3 getOrientation();`
	`void setOrientation(glm::vec3 orient);`
	`};`

Zaimplementuj hierarchię klas dla obiektów gry analogiczną do tej zaproponowanej w zadaniach z instrukcji nr 4.	`class Cube :` `public Object`

Rozszerz funkcjonalność klasy reprezentującej sześcian o możliwość wykonywania na tym obiekcie transformacji geometrycznych 3D przedstawionych w tej instrukcji. Wpleć tę klasę w opracowaną hierarchię klas.	`/** \brief Metoda transalte`
	` *`
	` * Metoda translokuje szescian.`
	` *`
	` * \param[in] p przekazuje wketor wspolrzednych`
	` *`
	` */`
	`void Cube::translate(glm::vec3 p)`
	`{`
	`glm::mat4 m`
	`= {`
	`1,0,0,p.x,`
	`0,0,0,p.y,`
	`0,0,0,p.z,`
	`0,0,0,1 };`
	`matrix *= m;`
	`}`
	
	`/** \brief Metoda rotate`
	` *`
	` * Metoda rotuje szescian.`
	` *`
	` * \param[in] degree przekazuje kat`
	` * \param[in] p przekazuje wketor wspolrzednych`
	` *`
	` */`
	`void Cube::rotate(float degree, glm::vec3 p)`
	`{`
	`glm::mat4 m =`
	`glm::rotate<float>(glm::radians(degree), p);`
	`matrix *= m;`
	
	`}`
	
	`/** \brief Metoda scale`
	` *`
	` * Metoda skaluje szescian.`
	` *`
	` * \param[in] p przekazuje wketor wspolrzednych`
	` *`
	` */`
	`void Cube::scale(glm::vec3 p)`
	`{`
	`glm::mat4 m = glm::scale<float>(p);`
	`matrix *= m;`
	`}`

Laboratorium 10:

Nie wykonano

Laboratorium 11:

Zespół wybrał laboratorium 11, aby być z niego zwolnionym.

Laboratorium 12:

Wykonano sprawozdanie w PDF oraz opisano kod w Doxygenie.

Wnioski

Zrealizowaliśmy większość założeń projektowych. Dzięki silnikowi 3D nabyliśmy podstawową wiedzę z operacji na obiektach trójwymiarowych.