Лабораторная работа №5 «Пространство»

При работе с пространством на первый план выходит сокращение расчетов в методе onPaint, поскольку он вызывается каждый раз при перерисовке окна, а трехмерные расчеты требуют гораздо больше ресурсов.

Поэтому теперь мы будем пользоваться прямой ссылкой на контекст устройства. Для этого в секции private объявим переменную DC: HDC;

Обработчик **onCreate** необходимо заменить на следующий:

```
Begin
  DC:=GetDC(Handle); //получаем прямую ссылку на контекст устройства
  SetDCPixelFormat(DC);
  hrc:=wglCreateContext(DC);
  wglMakeCurrent(DC, hrc);
  glClearColor(0.5, 0.5, 0.75, 1.0);
end;
```

Обработчик **onDestroy** становится следующим:

```
Begin
  wglMakeCurren(0,0);
  wglDeleteContext(hrc);
  ReleaseDC(Handle, DC);
  DeleteDC(DC);
End;
```

Обработчик **onResize** становится не зависимым от обработчика **onPaint**. Теперь он служит для задания видовых параметров всей сцены.

```
glViewport(0, 0, ClientWidth, ClientHeight);
glLoadIdentity; // эта команда означает «вернуть все матрицы в исходное состояние»
glFrustum(-1, 1, -1, 1, 3, 10);// видовые параметры - команда задает
                               // перспективную проекцию и расширяет область вывода
glTranslatef(0.0, 0.0, -8.0); // начальный сдвиг системы координат
glRotatef(30,1, 0, 0); //поворот системы координат, чтобы увидеть трехмерность
glRotatef(70, 0,1, 0);
InvalidateRect(Handle, nil, False);
```

Сам обработчик **onPaint** теперь служит только для отрисовки. Сначала чистится буфер кадра, затем происходит отрисовка, и содержимое буфера выводится на экран. Для работы в пространстве команда задания вершины принимает вид **glVertex3f(x,y,z)**. Для примера нарисуем куб, состоящий из 6 квадратов.

```
Begin
glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_QUADS);
  glVertex3f(1.0, 1.0, 1.0);
  glVertex3f(-1.0, 1.0, 1.0);
  glVertex3f(-l.0, -1. 0, 1.0);
  glVertex3f(l.0, -1. 0, 1.0);
glEnd;
glBegin(GL_QUADS);
  glVertex3f(l.0, 1.0, -1.0);
  glVertex3f(1.0, -1.0, -1.0);
  glVertex3f(-l.0, -1.0, -1.0),
  glVertex3f(-1.0, 1.0,-1.0);
glEnd;
glBegin(GL_QUADS);
  glVertex3f(-1.0, 1.0, -1.0);
  glVertex3f(-1.0, 1.0, -1.0);
  glVertex3f(-l.0, -1.0, 1.0);
  glVertex3f(-l.0, -1.0, 1.0);
glEnd;
glBegin(GL_QUADS);
  glVertex3f(1.0, 1.0, 1.0);
  glVertex3f(1.0, -1.0, 1.0);
  gIVertex3f(l.0, -1.0, -1.0);
  glVertex3f(1.0, 1.0,-1.0);
glEnd;
glBegin(GLQUADS);
  glVertex3f(-l. 0,1. 0, -1.0);
  glVertex3f(-1.0, 1.0, 1.0);
  glVertex3f(1.0, 1.0, 1.0);
  glVertex3f(1.0, 1.0, -1.0);
glEnd;
glBegin(GL_QUADS);
  glVertex3f(-1.0, -1.0, -1.0);
  glVertex3f(1.0, -1.0, -1.0);
  glVertex3f(1.0, -1.0, 1.0);
  glVertex3f(-1.0, -1.0, 1.0);
glEnd;
SwapBuffers(DC);
End;
```

Если перед каждой гранью куба вставить команду **glColor3f(random, random, random)**, то можно увидеть, что грани куба отображаются не корректно, а именно ближние грани не отображаются, а дальние грани (которые должны быть невидимыми) наоборот отображаются.

Чтобы исправить эту ситуацию необходимо включить работу с z-буфером или буфером глубины. Этот буфер необходим для корректного отображения пространства.

Команду **glClear(GL_COLOR_BUFFER_BIT)** необходимо заменить на

```
glClear(GL_COLOR_BUFFER_BIT or GL_DEPTH_BUFFER_BIT),
```

которая кроме буфера кадра чистит еще буфер глубины. И в обработчик **onCreate** в конец дописать команду **glEnable(GL_DEPTH_TEST)**, которая подключает к работе z-буфер. Теперь пространство передается корректно.

Задание: по нажатию кнопок осуществить вращение и перемещение полученного куба.