Chapter 3. Рисуем простые объекты

3.1 Общие положения

Точки, линии, треугольники, четырехугольники, многоугольники - простые объекты, из которых состоит любые сложные фигуры. В предыдущей главе мы рисовали сферу, конус и тор. OpenGL непосредственно не поддерживает функций для создания таких сложных объектов, т.е. таких функций нет в opengl32.dll. Эти функции есть в библиотеках утилит glu32.dll, и устроены они следующим образом. Для того чтобы нарисовать сферу функция auxSolidSphere использует функции из библиотеки glu32.dll, а те в свою очередь, используют базовую библиотеку opengl32.dll и из линий или многоугольников строят сферу. Примитивы создаются следующим образом:

```
glBegin(GLenum mode);  // указываем, что будем рисовать
  glVertex[2 3 4][s i f d](...); // первая вершина
  ...                           // тут остальные вершины
  glVertex[2 3 4][s i f d](...); // последняя вершина
glEnd();                        // закончили рисовать примитив
```

Сначала вы говорите, что будете рисовать - glBegin с соответствующим параметром. Возможные значения mode перечислены ниже в таблице. Далее вы указываете вершины, определяющие объекты указанного типа. Обычно вы будете задавать вершину одним из четырех способов.

```
glVertex2d(x,y);   // две переменных типа double
glVertex3d(x,y,z);  // три переменных типа double
glVertex2dv(array); // массив из двух переменных типа double
glVertex3dv(array);  // массив из трех переменных типа double
```

И наконец, вы вызываете glEnd, чтобы указать, что вы закончили рисовать объекты типа, указанного в glBegin. Далее мы подробно разберем создание всех примитивов.

Значение mode	Описание
GL_POINTS	Каждый вызов glVertex задает отдельную точку.
GL_LINES	Каждая пара вершин задает отрезок.
GL_LINE_STRIP	Рисуется ломаная.
GL_LINE_LOOP	Рисуется ломаная, причем ее последняя точка соединяется с первой.
GL_TRIANGLES	Каждые три вызова glVertex задают треугольник.
GL_TRIANGLE_STRIP	Рисуются треугольники с общей стороной.
GL_TRIANGLE_FAN	Тоже самое, но по другому правилу соединяются вершины, ряд ли понадобится.
GL_QUADS	Каждые четыре вызова glVertex задают четырехугольник.
GL_QUAD_STRIP	Четырехугольники с общей стороной.
GL_POLYGON	Многоугольник.

3.2 Точки

Вы можете нарисовать столько точек, сколько вам нужно. Вызывая glVertex3d, вы устанавливает новую точку. При создании точек вы можете изменить следующие параметры. Вы можете вызывать glColor3d внутри glBegin/glEnd. Размер точки можно устанавливать с помощью функции:

```
void glPointSize(GLfloat size)
```

Режим сглаживания можно устанавливать вызовом функции

```
glEnable(GL_POINT_SMOOTH)
```

Отключается соответственно вызовом glDisable() с этим параметром. Последние функции - glPointSize и glEnable/glDisable надо вызывать вне glBegin/glEnd, иначе они будут проигнорированы. Функции glEnable/glDisable включают/выключают множество опций, но вы должны учитывать, что некоторые опции влекут за собой большие вычисления и, следовательно, изрядно затормаживают ваше приложение, поэтому без надобности не стоит их включать. Очевидно, что совершенно не к чему включать освещение, наложение текстуру и сглаживания цветов при рисовании точек. Пока вы с этими возможностями OpenGL не познакомились, поэтому запомните это на будущее.

```
// рисуем точки
  glPointSize(2);
  glBegin(GL_POINTS);
    glColor3d(1,0,0);
    glVertex3d(-4.5,4,0); // первая точка
    glColor3d(0,1,0);
    glVertex3d(-4,4,0);   // вторая точка
    glColor3d(0,0,1);     // третья
    glVertex3d(-3.5,4,0);
  glEnd();
  glPointSize(5);
  glBegin(GL_POINTS);
    glColor3d(1,0,0);
    glVertex3d(-2,4,0); // первая точка
    glColor3d(0,1,0);
    glVertex3d(-1,4,0);   // вторая точка
    glColor3d(0,0,1);     // третье
    glVertex3d(0,4,0);
  glEnd();
  glPointSize(10);
  glEnable(GL_POINT_SMOOTH);
  glBegin(GL_POINTS);
    glColor3d(1,0,0);
    glVertex3d(2,4,0); // первая точка
    glColor3d(0,1,0);
    glVertex3d(3,4,0);   // вторая точка
    glColor3d(0,0,1);    // третья
    glVertex3d(4,4,0);
  glEnd();
  glDisable(GL_POINT_SMOOTH);
```

3.3 Линии

Для линий вы также можете изменять ширину, цвет, размер сглаживание. Если вы зададите разные цвета для начала и конца линии, то ее цвет будет переливающемся. OpenGL по умолчанию делает интерполяцию. Так же вы можете рисовать прерывистые линии, делается это путем наложения маски при помощи следующей функции:

```
void glLineStipple(GLint factor, GLushort pattern   );
```

Второй параметр задает саму маску. Например, если его значение равно 255(0x00FF), то чтобы вычислить задаваемую маску воспользуемся калькулятором. В двоичном виде это число выглядит так: 0000000011111111, т.е. всего 16 бит. Старшие восемь установлены в ноль, значит тут линии не будет. Младшие установлены в единицу, тут будет рисоваться линия. Первый параметр определяет, сколько раз повторяется каждый бит. Скажем, если его установить равным 2, то накладываемая маска будет выглядеть так:

```
0000000000000011111111111111111
```

Далее приведен исходный текст с комментариями для наглядной демонстрации что к чему.

```
glLineWidth(1);       // ширину линии
                      // устанавливаем 1
  glBegin(GL_LINES);
    glColor3d(1,0,0);    // красный цвет
    glVertex3d(-4.5,3,0); // первая линия
    glColor3d(0,1,0);
    glVertex3d(-3,3,0);  // вторая линия
    glVertex3d(-4,3.4,0);
  glEnd();
  glLineWidth(3);       // ширина 3
  glBegin(GL_LINE_STRIP); // см. ниже
    glColor3d(1,0,0);
    glVertex3d(-2.7,3,0);
    glVertex3d(-1,3,0);
    glColor3d(0,1,0);
    glVertex3d(-1.5,3.3,0);
    glColor3d(0,0,1);
    glVertex3d(-1,3.5,0);
  glEnd();
  glLineWidth(5);
  glEnable(GL_LINE_SMOOTH);
  glEnable(GL_LINE_STIPPLE); // разрешаем рисовать
                             // прерывистую линию
  glLineStipple(2,58360);    // устанавливаем маску
                             // пояснения см. ниже
  glBegin(GL_LINE_LOOP);
    glColor3d(1,0,0);
    glVertex3d(1,3,0);
    glVertex3d(4,3,0);
    glColor3d(0,1,0);
    glVertex3d(3,2.7,0);
    glColor3d(0,0,1);
    glVertex3d(2.5,3.7,0);
  glEnd();
  glDisable(GL_LINE_SMOOTH);
  glDisable(GL_LINE_STIPPLE);
```

3.4 Треугольники

Для треугольника можно задавать те же параметры, что и для линии плюс есть еще одна функция glPolygonMode. Она устанавливает опции для отрисовки многоугольника. Первый параметр может принимать значения - GL_FRONT, GL_BACK и GL_FRONT_AND_BACK. Второй параметр указывает, как будет рисоваться многоугольник. Он принимает значения - GL_POINT(рисуется только точки), GL_LINE(рисуем линии) и GL_FILL(рисуем заполненный многоугольник). Первый параметр указывает: к лицевой, тыльной или же к обеим сторонам применяется опция, заданная вторым параметром. Треугольники можно рисовать, передав GL_TRIANGLE_STRIP или GL_TRIANGLE_FAN в glBegin. В первом случае первая, вторая и третья вершины задают первый треугольник. Вторая, третья и четвертая вершина - второй треугольник. Третья, четвертая и пятая вершина - третий треугольник и т.д. Вершины n, n+1 и n+2 определят n-ый треугольник. Во втором случае первая, вторая и третья вершина задают первый треугольник. Первая, третья и четвертая вершина задают второй треугольник и т.д. Вершины 1, n+1, n+2 определят n-ый треугольник. Далее следует пример с комментариями.

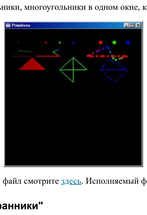
```
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL); // см. выше
  glBegin(GL_TRIANGLES);
    glColor3d(1,0,0);    // рисуем треугольник
    glVertex3d(-4,2,0);
    glVertex3d(-3,2.9,0);
    glVertex3d(-2,2,0);
  glEnd();
  glLineWidth(2);
  glPolygonMode(GL_FRONT_AND_BACK, GL_LINE); //рисуем
                          // проволочные треугольники
  glBegin(GL_TRIANGLE_STRIP); // обратите внимание на порядок
                          // вершин
    glColor3d(0,1,0);
    glVertex3d(1,2,0);
    glVertex3d(0,2.9,0);
    glVertex3d(-1,2,0);
    glVertex3d(0,1.1,0);
  glEnd();
  glEnable(GL_LINE_STIPPLE);
  glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
  glBegin(GL_TRIANGLE_FAN);
    glColor3d(0,0,1);
    glVertex3d(4,2,0);
    glVertex3d(2.6,2.8,0);
    glVertex3d(2,2,0);
    glVertex3d(3,1.1,0);
  glEnd();
  glDisable(GL_LINE_STIPPLE);
```

3.5 Четырехугольники и многоугольники

Четырехугольники рисуются вызовом функции glBegin с параметром GL_QUADS или GL_QUAD_STRIP. Для первого случая каждые четыре вершины определяют свой четырехугольник. Во втором случае рисуются связанные четырехугольники. Первая, вторая, третья и четвертая вершина определяют первый четырехугольник. Третья, четвертая, пятая и шестая вершина - второй четырехугольник и т.д. (2n-1), 2n, (2n+1) и (2n+2) вершины задают n-ый четырехугольник. Многоугольники задаются вызовом glBegin с параметром GL_POLYGON. Все вершины определяют один многоугольник. Для многоугольников можно задавать стили при помощи вышe описанной функции glPolygonMode, толщину линии, толщину точек и цвет.

3.6 Упражнение: "Примитивы"

Изобразите точки, линии, треугольники, многоугольники в одном окне, как показано ниже.



Исходный файл смотрите здесь. Исполняемый файл здесь.

3.7 Упражнение: "Многогранники"

Реализуйте проволочные многогранники с помощью проволочных треугольников, многоугольников и линий.

3.8 Резюме

Ну вот, вы еще на один шаг продвинулись в изучение библиотеки OpenGL. Теперь вы имеете представление о том, как рисовать элементарные фигуры. Из примитивов вы можете составить фигуры любой сложности.

Назад | Содержание | Вперед