



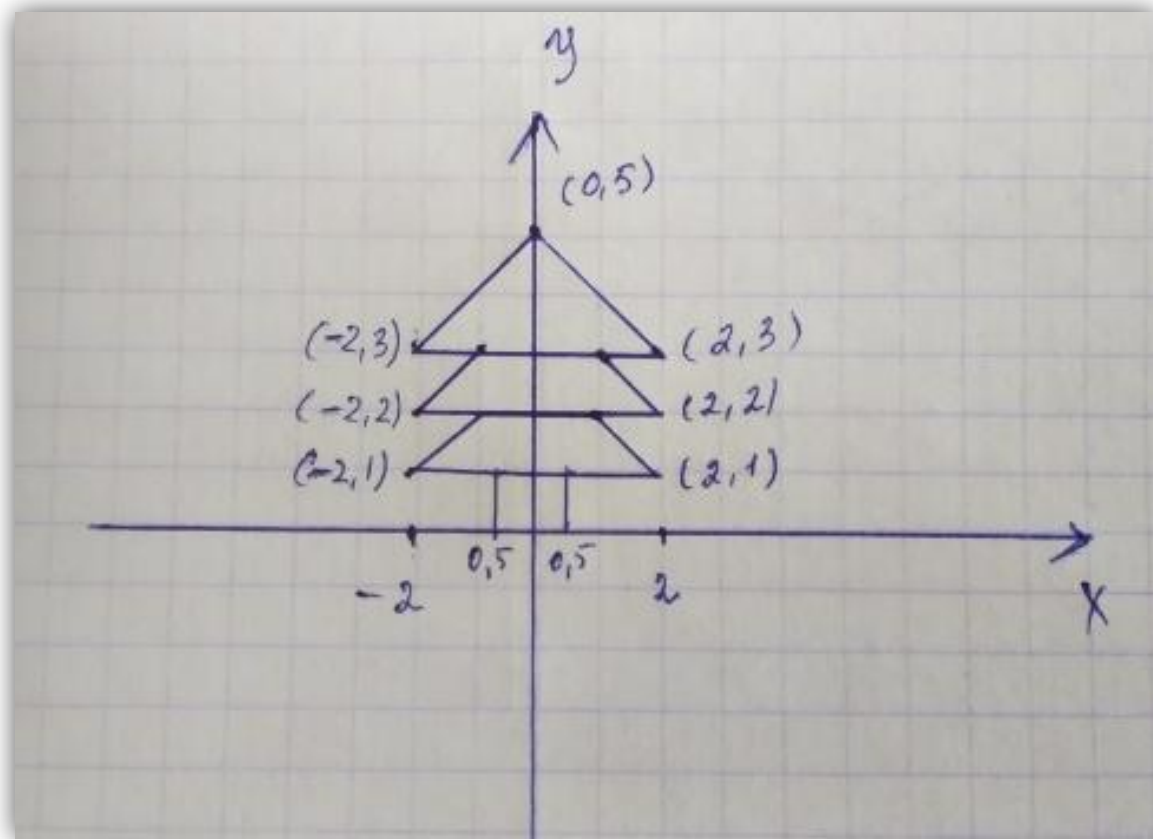
# Урок №3

Графические примитивы OpenGL,  
создание графических примитивов,  
трансформация графических примитивов

# Технология создания графического примитива

- Откройте проект из урока №2 (рисовали оси координат);
- Придумайте графический примитив из стандартных графических примитивов OpenGL. В моем случае это будет дерево – стилизованная ель;

- Изобразите стилизованное дерево на обычном листе бумаги. Лучше, если это будет листок в клетку. Определитесь с его первоначальными размерами. В нашем случае габариты ели 4x5;



- Разбейте рисунок на выпуклые полигоны. В моем случае получилось три четырехугольника и один треугольник;
- Можно приступить к написанию кода:

```
//дерево - ель 4x5
void tree1()
{
//ствол
glColor3ub(200,100,50); //цвет ствола
glBegin(GL_QUADS);
    glVertex2f(-0.5,0);
    glVertex2f(0.5,0);
    glVertex2f(0.5,1);
    glVertex2f(-0.5,1);
glEnd();
```

```
//крона
```

```
glColor3ub(0,200,0); //цвет крон
```

```
glBegin(GL_QUADS);
```

```
ы
```

```
    glVertex2f(-2,1);
```

```
    glVertex2f(2,1);
```

```
    glVertex2f(1,2);
```

```
    glVertex2f(-1,2);
```

```
    glVertex2f(-2,2);
```

```
    glVertex2f(2,2);
```

```
    glVertex2f(1,3);
```

```
    glVertex2f(-1,3);
```

```
glEnd();
```

```
glBegin(GL_TRIANGLES);
```

```
    glVertex2f(-2,3);
```

```
    glVertex2f(2,3);
```

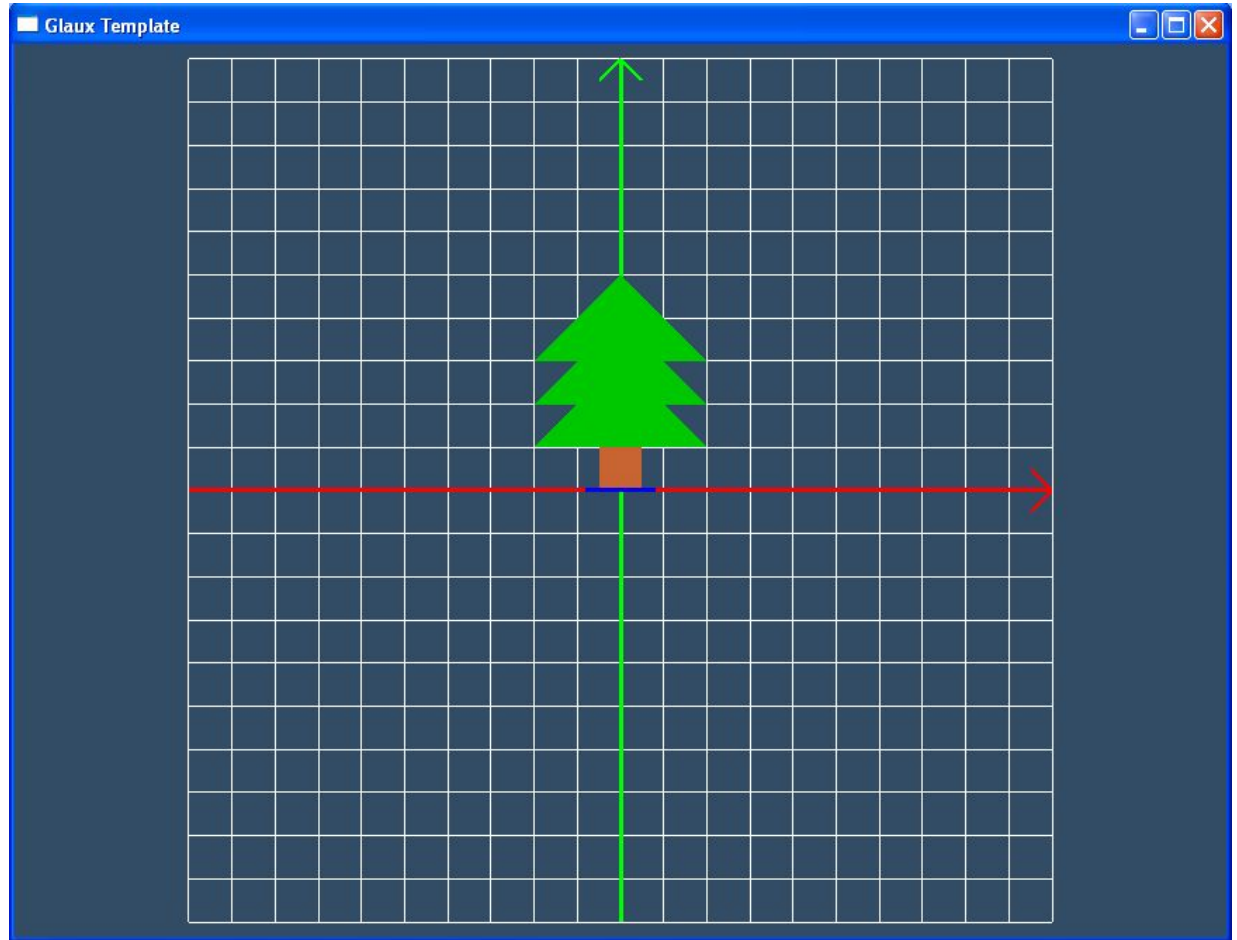
```
    glVertex2f(0,5);
```

```
glEnd();
```

```
}
```

- Вызываем функцию `tree1` в `Draw`;

```
void Draw()  
{  
    tree1();  
    osi(10);  
}
```



# Рисование примитивов с помощью функции `tree1`. Изменение примитива с помощью трансформаций

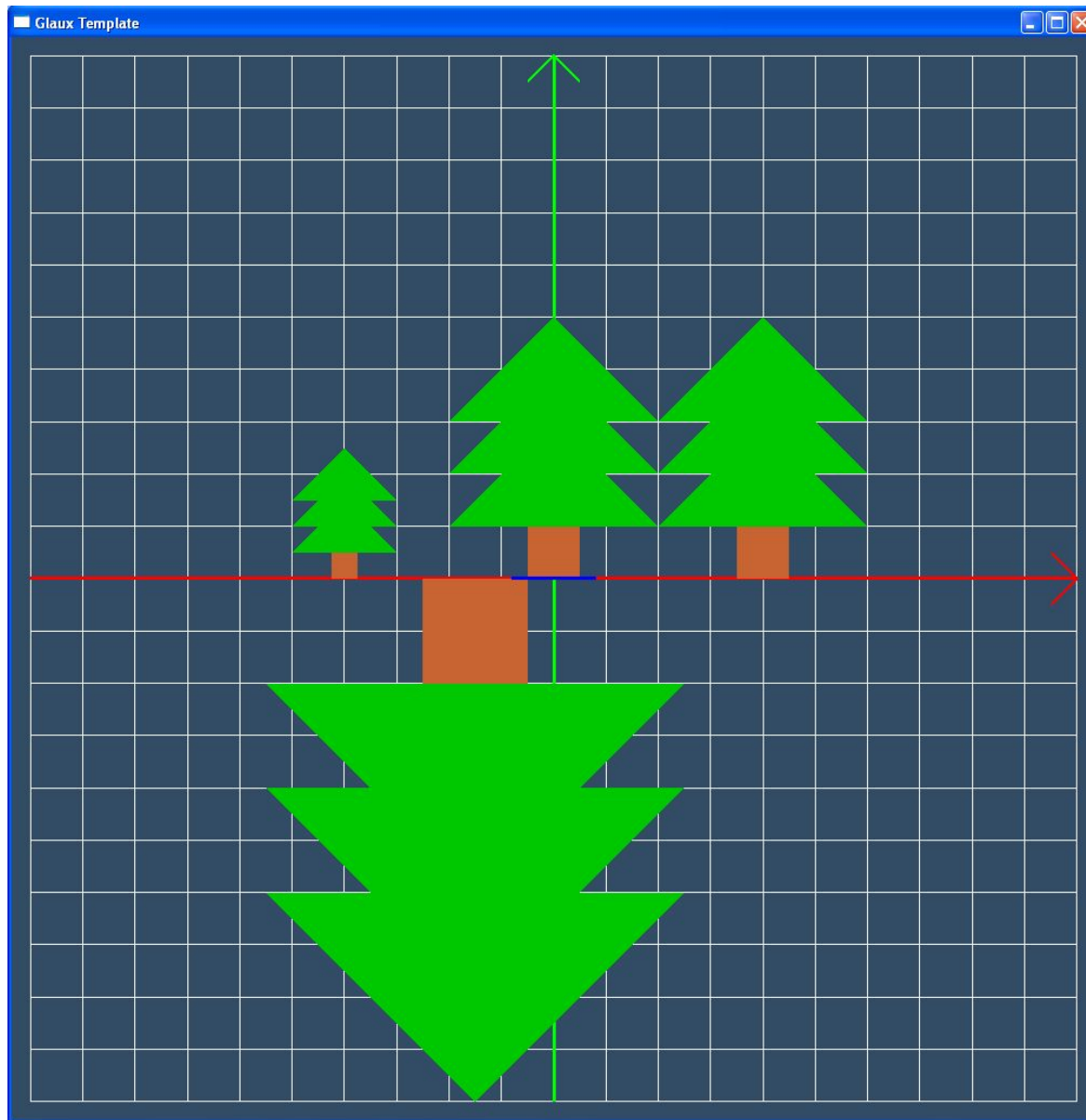
- Нарисовать дерево в точке  $(4,0,0)$ ;
- Нарисовать дерево в точке  $(-4,0,0)$ , уменьшенное в два раза;
- Нарисовать перевернутое дерево, увеличенное в два раза, в точке  $(-1.5, 0,0)$ .

## Пример функции Draw, для рисования задуманного:

```
void Draw()
{
    tree1(); //дерево в центре начала координат
    glTranslated(4, 0, 0); //перенос на вектор(4,0,0)
    tree1();
    glLoadIdentity(); //сброс текущей матрицы
    glTranslated(-4, 0, 0); //перенос на вектор(-4,0,0)
    glScaled(0.5,0.5,1); //уменьшение размеров по осям x и y
    tree1();
    glLoadIdentity(); //сброс текущей матрицы
    glTranslated(-1.5, 0, 0); //перенос на вектор(-1.5,0,0)
    glRotated(180,1,0,0); //поворот на 180 градусов, вокруг оси OX
    glScaled(2,2,1); //увеличение размеров по осям x и y
    tree1();
    glLoadIdentity(); //сброс текущей матрицы
    osi(10);
}
```



Должен получиться вот такой результат:



## Из OpenGL Red Book

У полигона две стороны или грани—лицевая и обратная, и он может быть визуализирован по-разному в зависимости от того, которая из сторон видна наблюдателю. Полигон также считается лицевым или обратным в зависимости от того, какой из граней он повернут к наблюдателю. Программно лицевой считается та грань, вершины которой обходят против часовой стрелки. По умолчанию, лицевые и обратные грани изображаются одинаково. Чтобы это изменить, а также, чтобы рисовать только вершины или границы полигона, используйте команду **glPolygonMode()**:

```
void glPolygonMode (GLenum face, GLenum mode) ;
```

Управляет режимом отображения для лицевых и обратных граней полигонов. Параметр **face** указывает, для каких граней изменяется режим отображения и может принимать значения **GL\_FRONT\_AND\_BACK** (режим меняется и для лицевых и для обратных граней), **GL\_FRONT** (только для лицевых), **GL\_BACK** (только для обратных). Параметр **mode** может быть равен **GL\_POINT**, **GL\_LINE** или **GL\_FILL** в зависимости от желаемого режима отображения: точки, линии или заливка. По умолчанию оба типа граней рисуются в виде заполненных областей пикселей (заливки, например, если вы хотите отображать лицевые грани в виде заливки, а обратные в виде линии по границе, используйте две следующие команды:

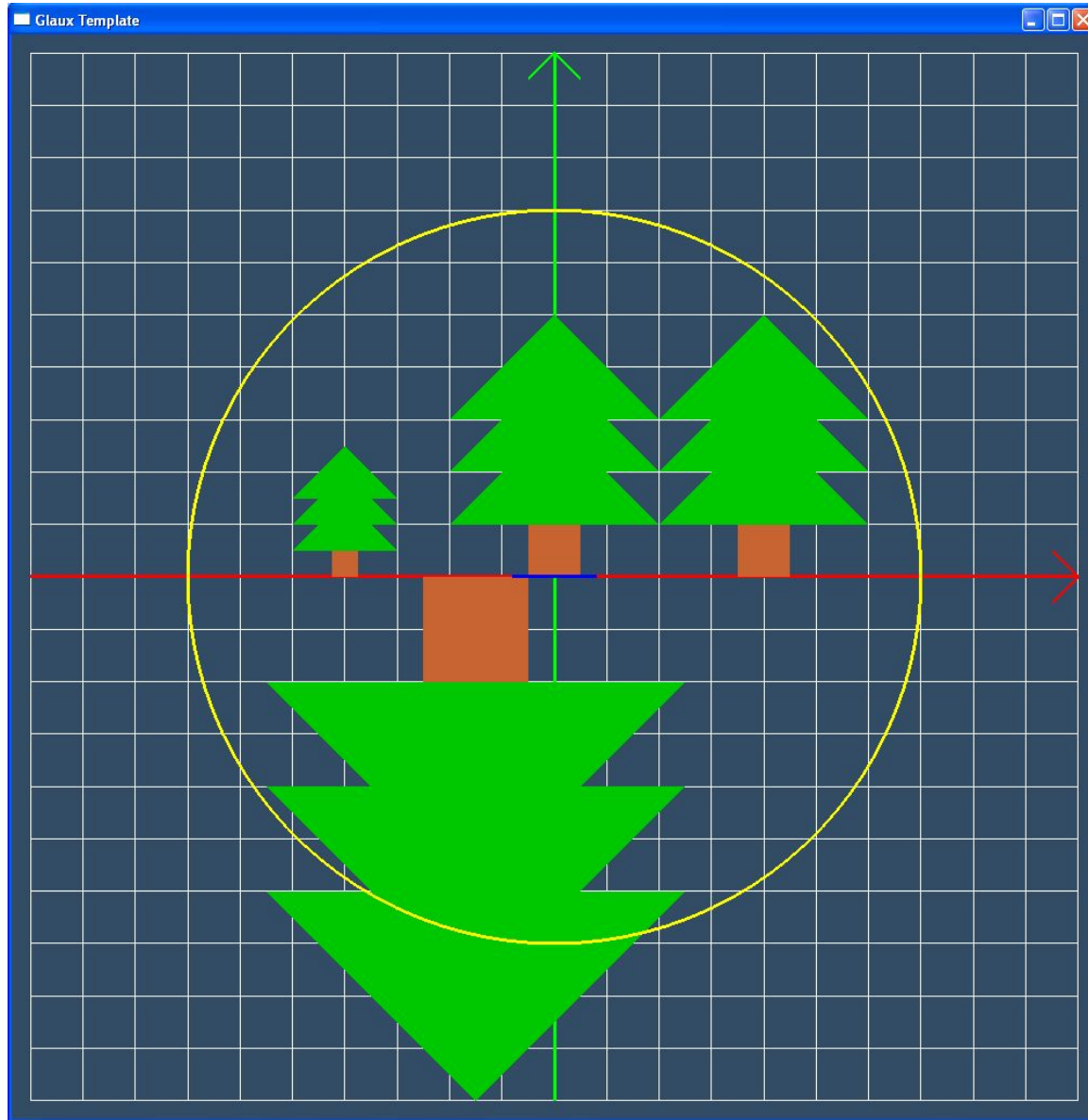
```
glPolygonMode (GL_FRONT, GL_FILL) ;
```

```
glPolygonMode (GL_BACK, GL_LINE) ;
```

# Графические примитивы: окружность, круг, эллипс, кольцо

## ■ Окружность:

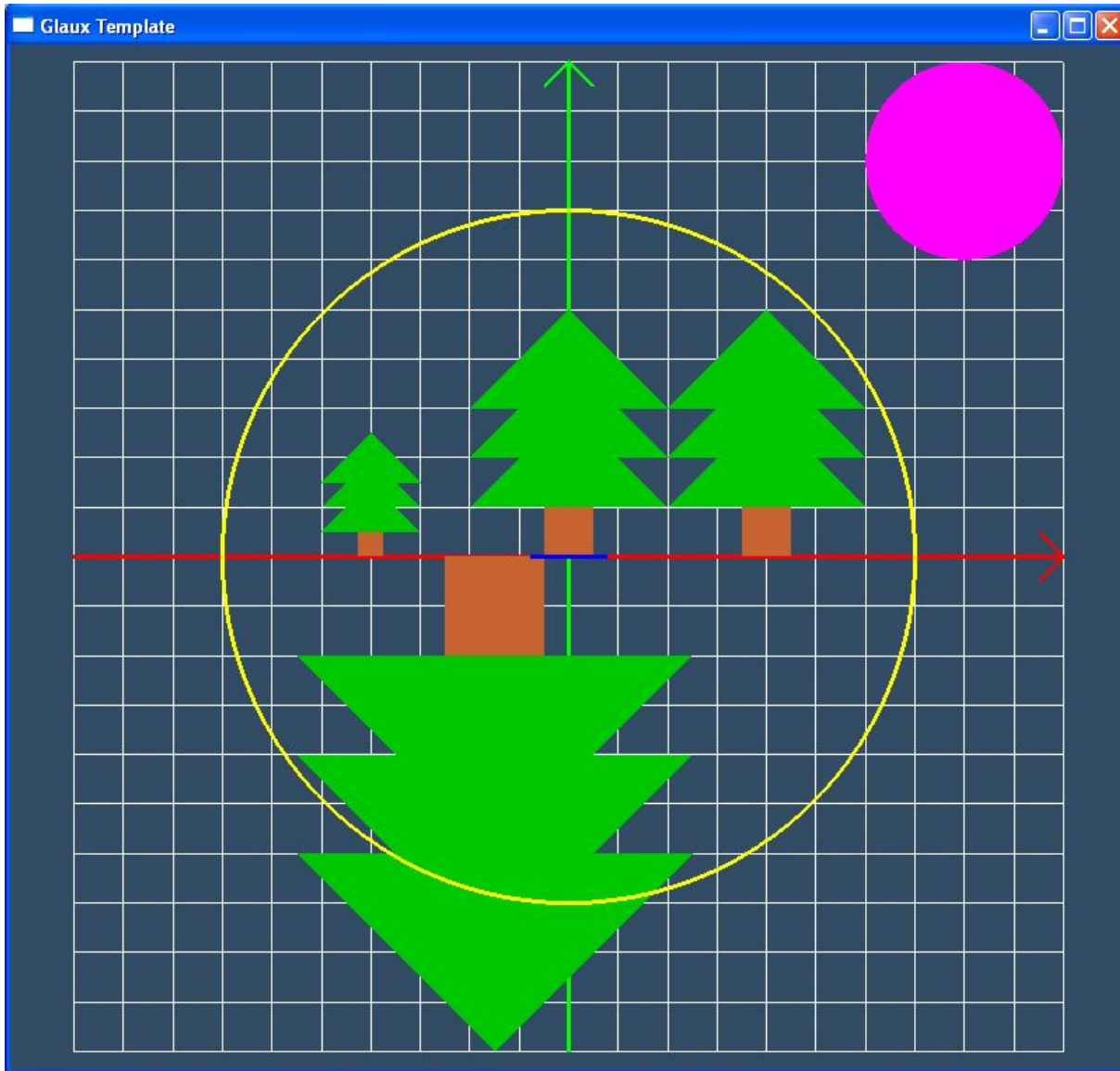
```
void DrawCircle(double x, double y, double radius,
double r, double g, double b, float size=1.0f)
{
// передаются координаты центра, радиус и цвет
окружности, толщина линии
int n = 100; // количество точек
glLineWidth(size);
glBegin(GL_LINE_LOOP);
    glColor3d(r, g, b);
    for (int i = 0; i < n; i++)
        glVertex2d(x+radius*cos(i*2*3.14159/n),
                    y+radius*sin(i*2*3.14159/n));
glEnd();
}
```



```
DrawCircle(0,0,7,1,1,0,3);
```

## ■ Круг:

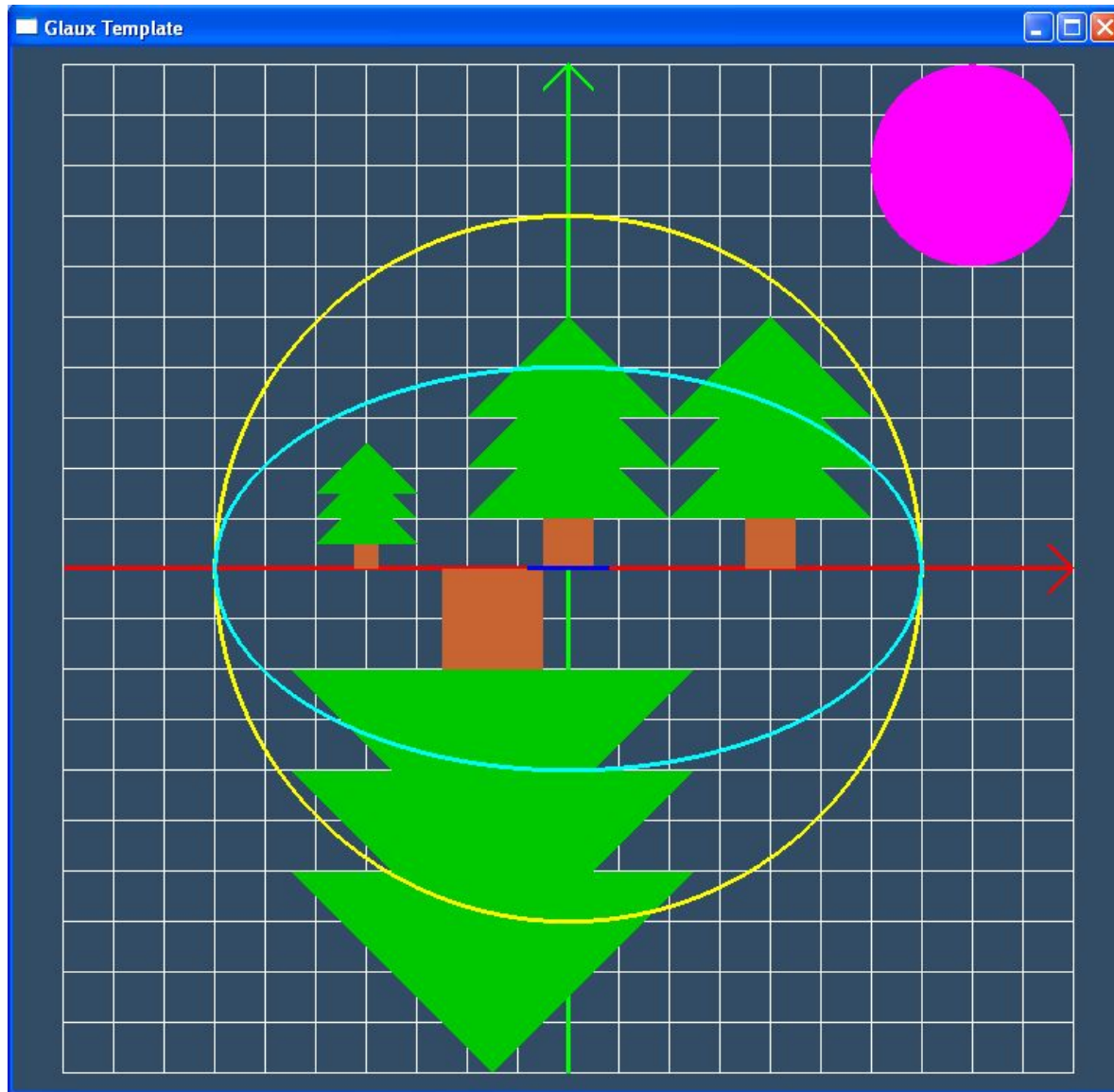
```
void DrawRound(double x, double y, double radius, double
r, double g, double b)
{
// передаются координаты центра, радиус и цвет круга
int n = 100; // количество точек
glBegin(GL_POLYGON);
    glColor3d(r, g, b);
for (int i = 0; i < n; i++)
    glVertex2d(x + radius*cos(i * 2 * 3.14159 / n),
              y + radius*sin(i * 2 * 3.14159 / n));
glEnd();
}
```



`DrawRound ( 8 , 8 , 2 , 1 , 0 , 1 ) ;`

## ■ Эллипс:

```
void DrawEllipse(double r1, double r2, double r, double g,
double b, float size=1.0f)
{
// передаются полуоси, цвет эллипса, толщина линии
int n = 100; // количество точек
glLineWidth(size);
glBegin(GL_LINE_LOOP);
    glColor3d(r, g, b);
    for (int i = 0; i < n; i++)
        glVertex2d(r1*cos(i*2*3.14159/n), r2*sin(i*2*3.14159/n));
glEnd();
}
```



```
DrawEllipse (7,4,0,1,1,3) ;
```



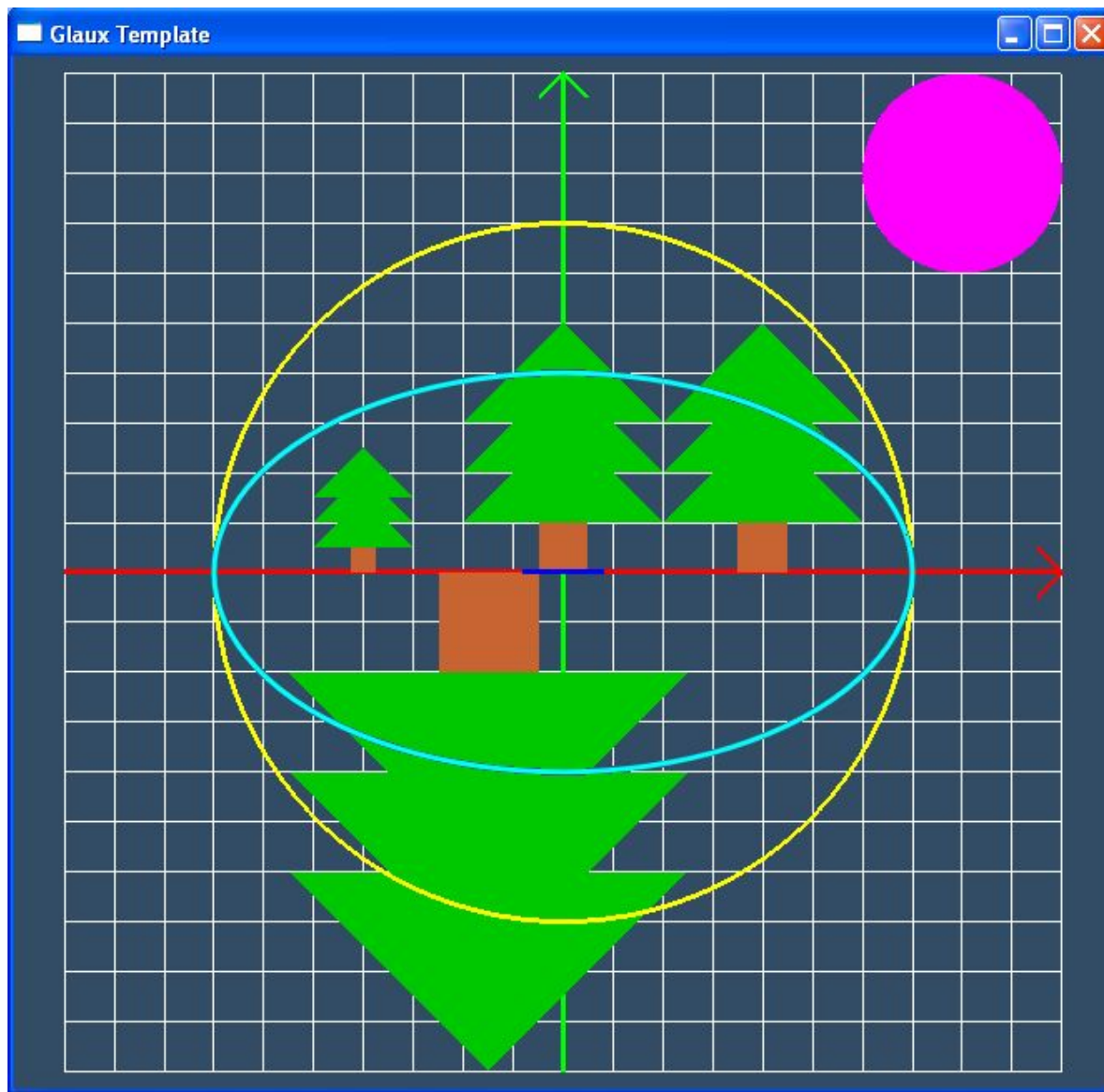
Линии также можно делать более гладкими. Для получения эффекта сглаживания необходимо его разрешить:

```
void DrawEllipse(double r1, double r2, double r, double g,
    double b, float size=1.0f)
{
    // передаются координаты центра, радиус и цвет окружности,
    // толщина линии
    int n = 100; // количество точек
    glEnable( GL_LINE_SMOOTH ); // разрешаем сглаживание линий
    glLineWidth(size);
    glBegin( GL_LINE_LOOP );
        glColor3d(r, g, b);
        for (int i = 0; i < n; i++)
            glVertex2d(r1*cos(i*2*3.14159/n), r2*sin(i*2*3.14159/n));
    glEnd();
    glDisable( GL_LINE_SMOOTH ); // восстанавливаем режим по
    // умолчанию
}
```

В некоторых случаях этого недостаточно, поэтому в функцию RunOpenGL добавим следующий код:

```
glEnable( GL_BLEND ); //разрешаем смешивание при  
наложении  
glBlendFunc( GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA  
); //устанавливаем наиболее подходящие параметры  
смешивания
```

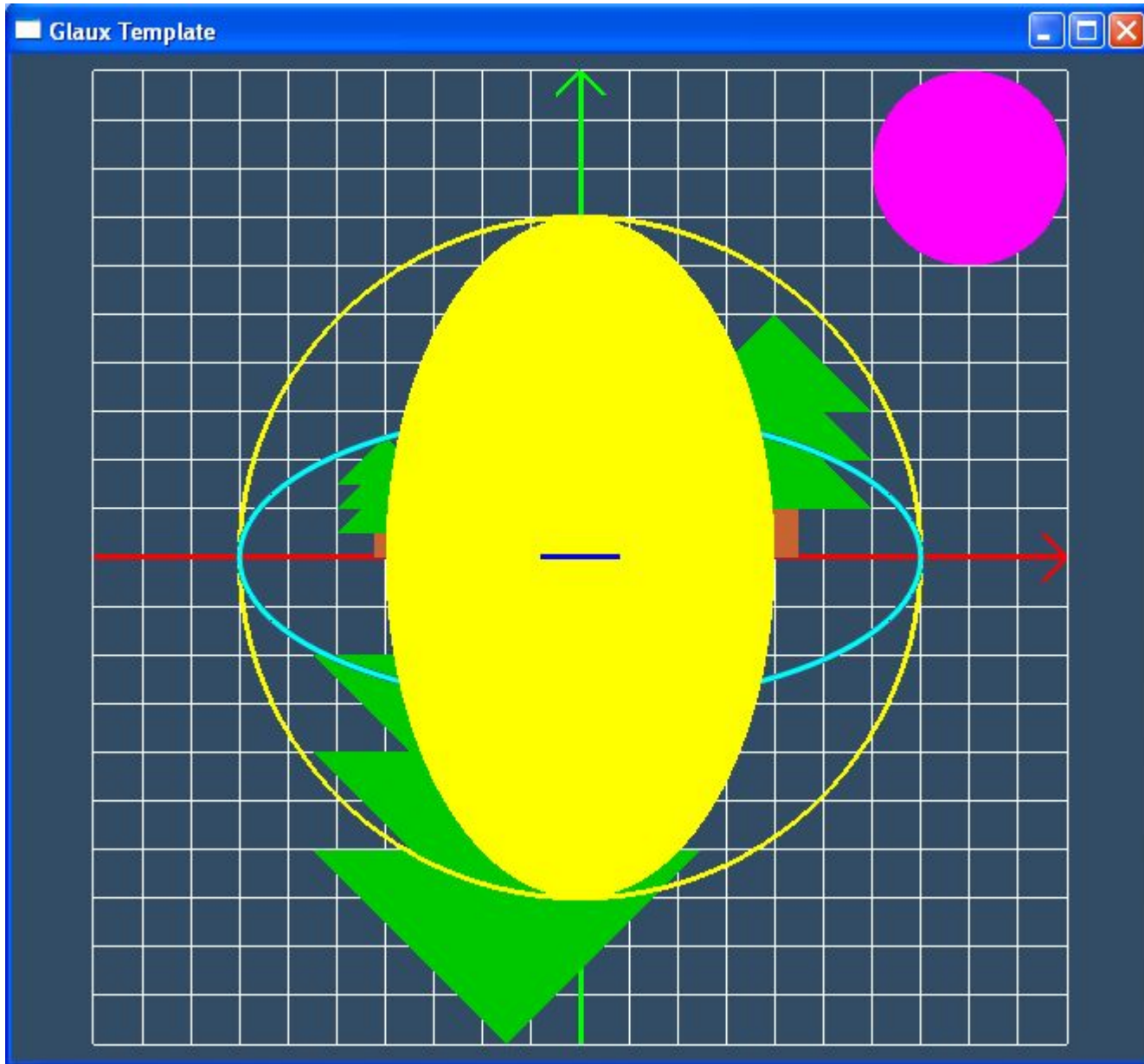
Этот код должен выполняться один раз.



Линия эллипса стала более гладкой.

## ■ Эллипс с заливкой:

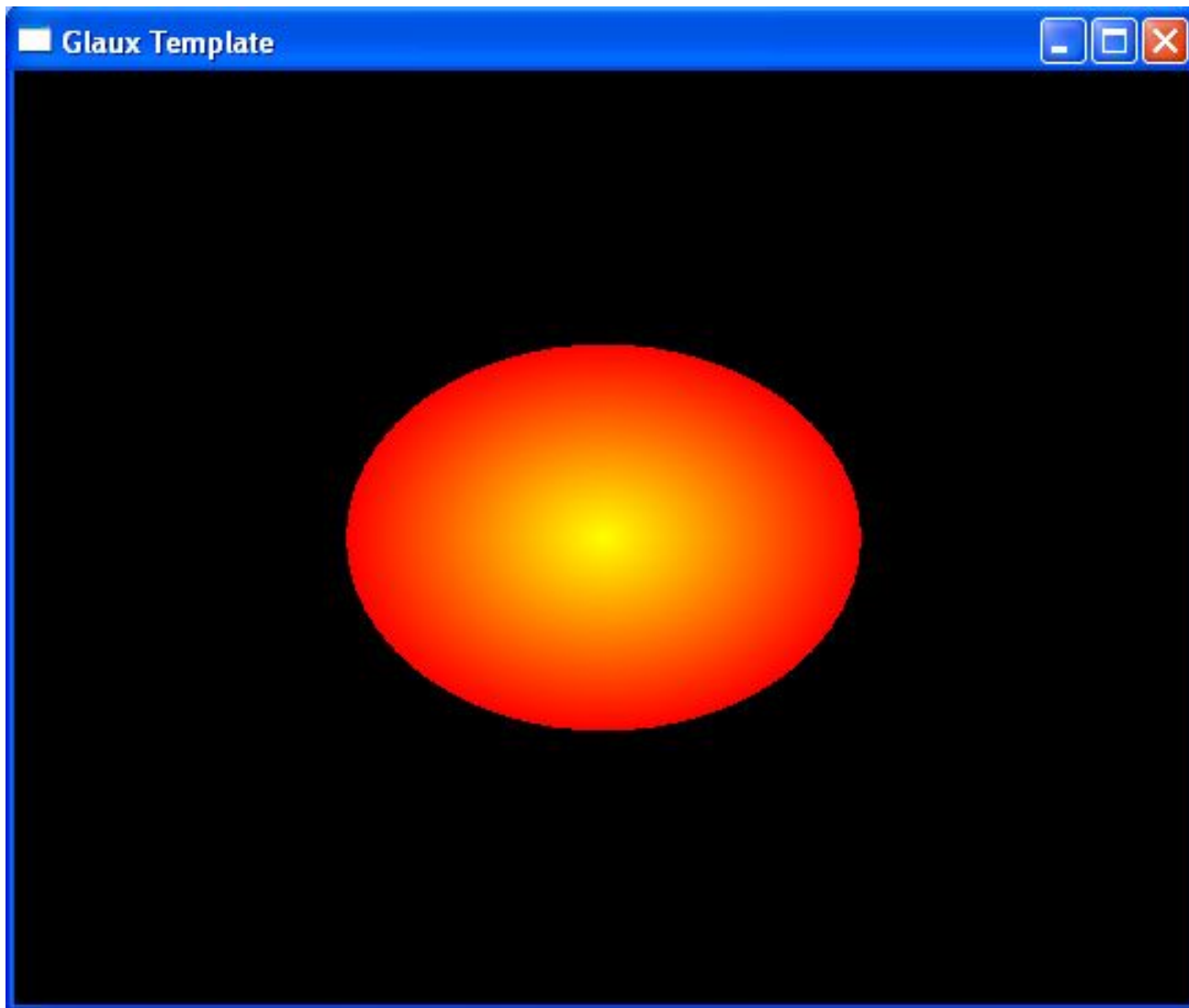
```
void DrawEllipseFill(double r1, double r2, double r, double g,
double b)
{
    // передаются полуоси, цвет заливки
    int n = 100; // количество точек
    glBegin(GL_TRIANGLE_FAN); // веер треугольников
        glColor3d(r, g, b);
        for (int i = 0; i < n; i++)
            glVertex2d(r1*cos(i*2*3.14159/n), r2*sin(i*2*3.14159/n));
    glEnd();
}
```



```
DrawEllipseFill(4,7,1,1,0);
```

## ИЛИ

```
void DrawEllipseFillGrad(double a, double b, double r1, double
g1, double b1, double r2, double g2, double b2)
{ // передаются полуоси, цвет общей точки, цвет остальных
точек
  int n = 50; // количество точек
  glBegin(GL_TRIANGLE_FAN);
    glColor3d(r1, g1, b1);
    glVertex2d(0, 0); // общая точка для всех треугольников
    glColor3d(r2, g2, b2);
    for (int i = 0; i <= n; i++)
      glVertex2d(a*cos(i*2*3.14159/n), b*sin(i*2*3.14159/n));
  glEnd();
}
```

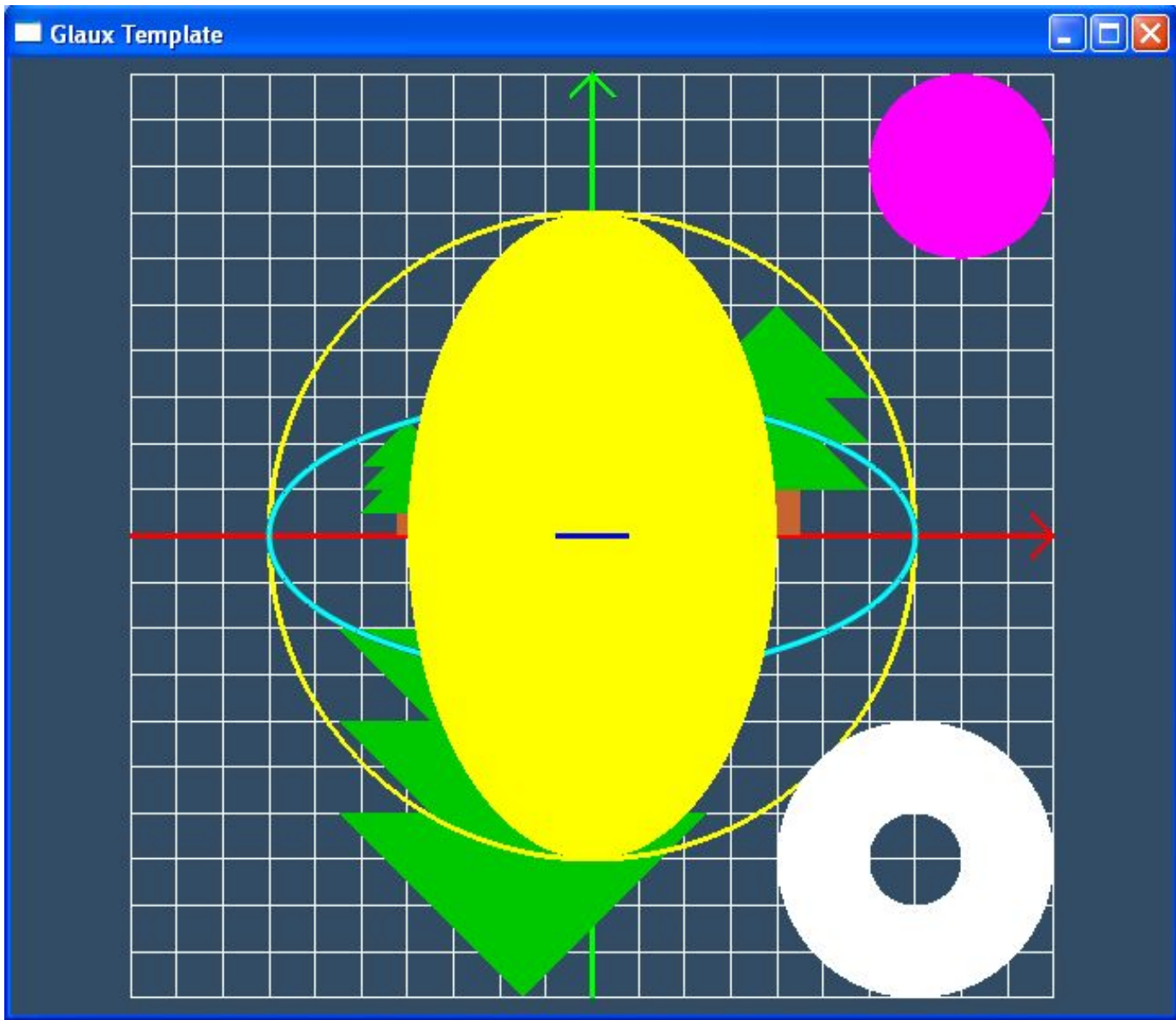


```
DrawEllipseFillGrad(8,6,1,1,0,1,0,0);
```

## ■ Кольцо:

```
void DrawRing(double x, double y, double r1, double r2,
double r, double g, double b)
{
    //передаются координаты центра, внешний радиус,
    //внутренний радиус и цвет кольца
    int n = 50; //количество точек
    glBegin(GL_TRIANGLE_STRIP); //цепочка треугольников
    glColor3d(r, g, b);
    for (int i = 0; i<=n; i++)
    {
        glVertex2d(x + r1*cos(i * 2 * 3.14159 / n),
            y + r1*sin(i * 2 * 3.14159 / n));
        glVertex2d(x + r2*cos(i * 2 * 3.14159 / n),
            y + r2*sin(i * 2 * 3.14159 / n));
    }
    glEnd();
}
```

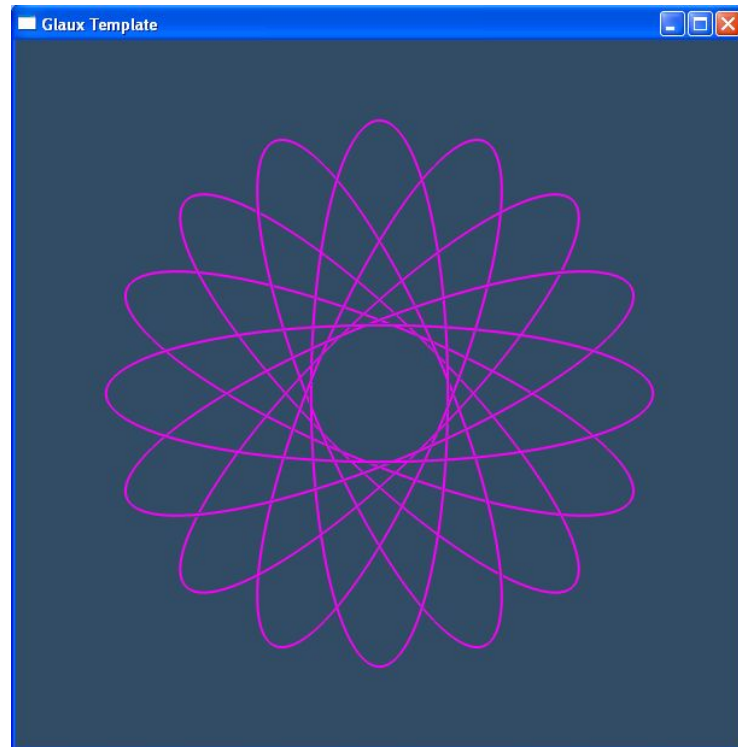




```
DrawRing(7,-7,3,1,1,1,1);
```

# Создание примитивов при помощи трансформаций

Функция рисования эллипса выводит примитив параллельно осям  $X$  и  $Y$ . Чтобы создать примитив, изображенный ниже, необходимо эллипс поворачивать вокруг оси  $Z$ .



```
void DrawOrbita(double r1, double r2, double r, double g,
double b, byte n)
{
    //r1-большая полуось, r2- малая полуось эллипсов
    glPushMatrix(); //сохранили текущую матрицу
    float dfi=360.0f/n;//n - количество эллипсов
    for(int i=0; i<n; i++)
    {
        DrawEllipse(r1,r2,r,g,b,2);
        glRotated(dfi, 0,0,1);
    }
    glPopMatrix(); //вернули текущую матрицу
}
```

Используя, выше написанные функции рисования графических примитивов, можно создавать более сложные рисунки.

