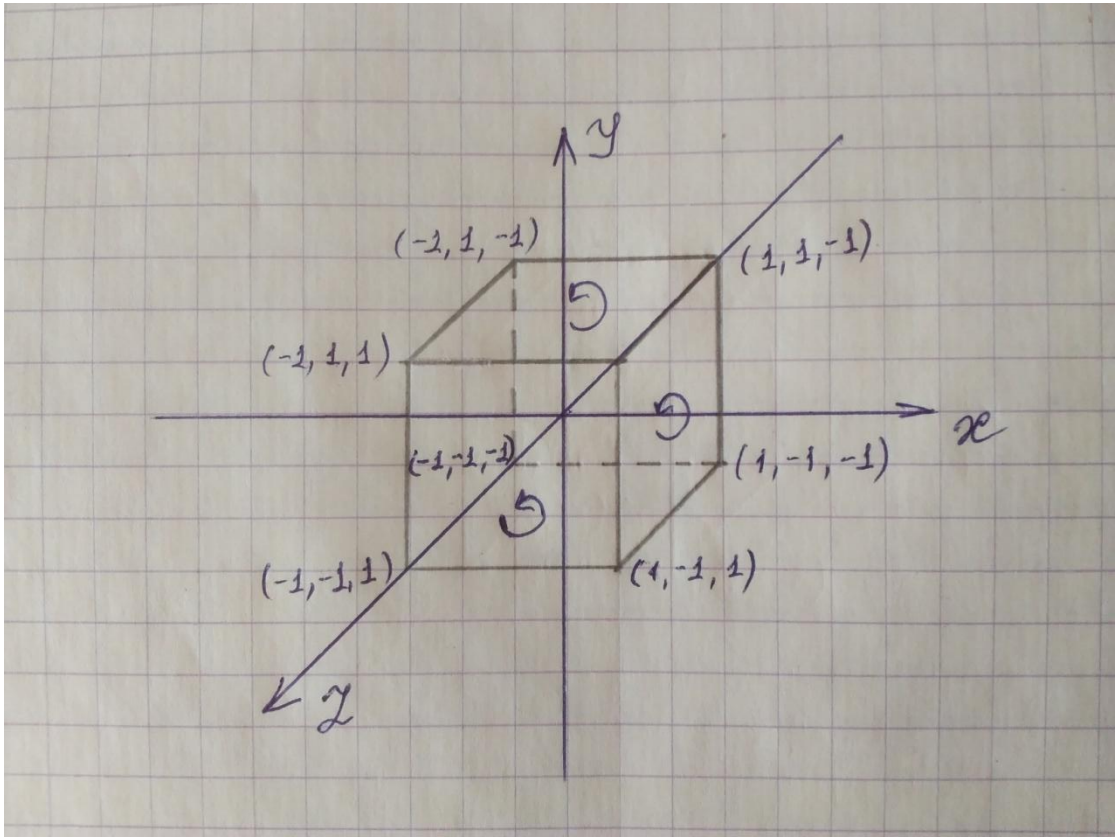
Создаем 3D-модель

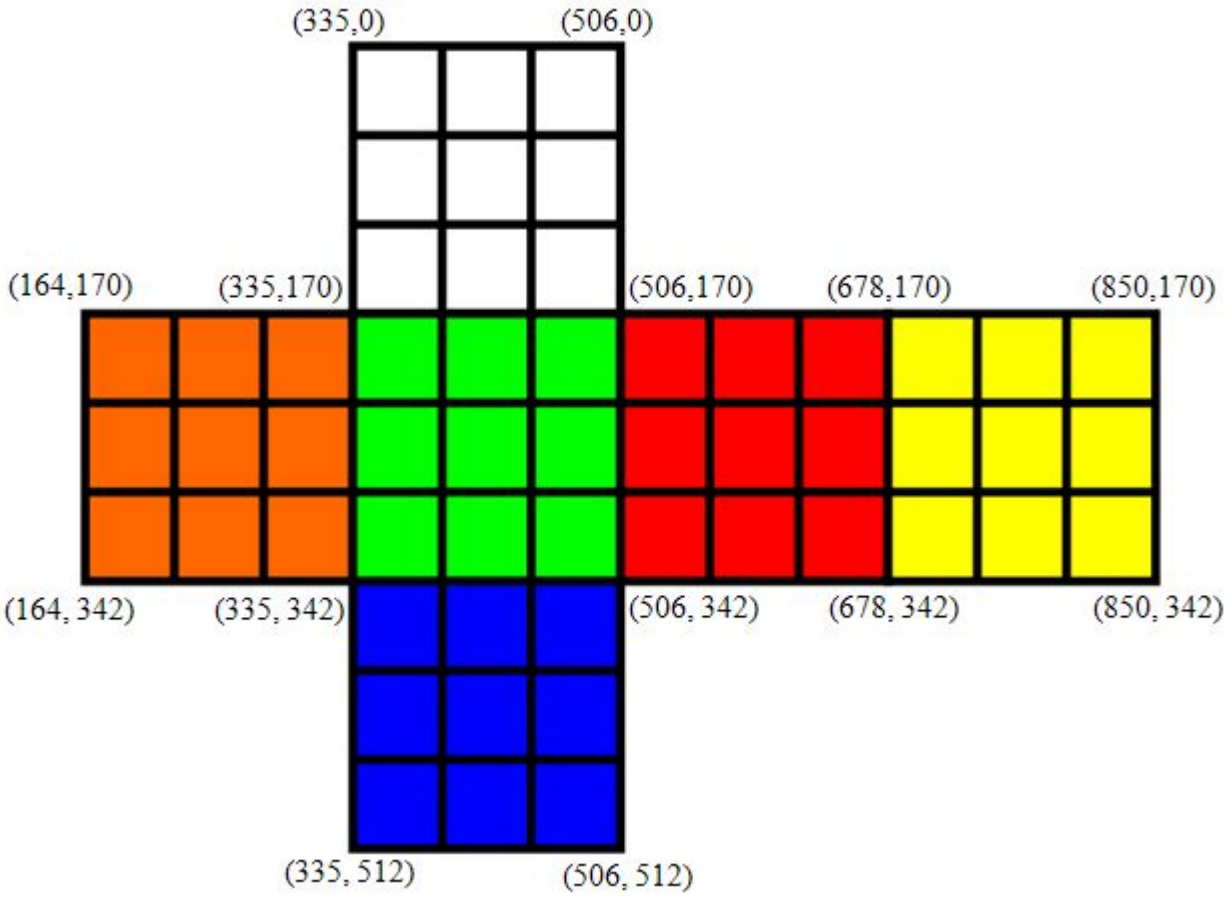
(кубик с текстурой кубика Рубика)

Создадим 3D-модель куба, центр которого находится в начале координат, а длина грани равна 2 единицам (так проще вычислять координаты всех вершин).



Обход вершин будем делать против часовой стрелки, чтобы грани куба были лицевыми.

Подготовим текстуру (текстура прикреплена ниже) размером 512x1024 пикселей в формате BMP (Paint вам в помощь). Находим координаты вершин будущей текстуры (опять Paint вам в помощь).



Приступаем к написанию программы.

```
#include <iostream>
#include <math.h>
#include <windows.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include "glaux.h"
using namespace std;

#pragma comment (lib,"opengl32.lib")
#pragma comment (lib,"glu32.lib")
#pragma comment (lib, "glaux.lib")

//#pragma comment(lib, "legacy_stdio_definitions.lib")//
раскоментировать для версий выше VS12
```

```
GLuint   texture[1];// Массив для хранения индексов текстур
//Функция для загрузки текстур
void LoadBMP(GLuint* texture, const char* filename)
{
    AUX_RGBImageRec* txt;
    txt = auxDIBImageLoad(filename);
    glGenTextures(1, texture);
    glBindTexture(GL_TEXTURE_2D, *texture);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexImage2D(GL_TEXTURE_2D, 0, 3, txt->sizeX, txt->sizeY, 0,
    GL_RGB, GL_UNSIGNED_BYTE, txt->data);
    delete txt;
}
```

```
void CALLBACK resize(int width,int height)
{
          glViewport(0, 0, width, height);
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     gluPerspective(45.0, (GLfloat)width / height, 1.0, 1000.0);
          gluLookAt( 0,0,10, 0,0,0, 0,1,0 );
     glMatrixMode(GL_MODELVIEW);
}
```

```
float tx1=0,ty1=0,tx2=0,ty2=0,tx3=0,ty3=0,tx4=0,ty4=0;// Переменные(глобальные)
для хранения текстурных координат(будут вычисляться для каждой грани)

//Функция рисования грани
void DrawQuadT(float x1, float y1, float z1, float x2, float y2, float z2,
float x3, float y3, float z3, float x4, float y4, float z4)
{
    glEnable(GL_TEXTURE_2D);//Разрешение на использование текстур
    glBindTexture(GL_TEXTURE_2D, texture[0]);//Выбор текстуры
    glBegin(GL_QUADS);
      glTexCoord2f(tx1, ty1); glVertex3f(x1, y1, z1);
      glTexCoord2f(tx2, ty2); glVertex3f(x2, y2, z2);
      glTexCoord2f(tx3, ty3); glVertex3f(x3, y3, z3);
      glTexCoord2f(tx4, ty4); glVertex3f(x4, y4, z4);
    glEnd();
    glDisable(GL_TEXTURE_2D);//Запрет на использование текстур
}
```

Функция рисования кубика. Каждую грань рисуем отдельно, перед гранью вычисляем текстурные координаты, вектор нормали к поверхности грани. Текстурные координаты и координаты вершин грани обходим против часовой стрелки.

```
//функция рисования кубика с текстурой
void DrawRubik()
{
    //1 – передняя грань
    glNormal3d(0,0,1);
    tx1 = 335/1024.f;
    ty1 = (512-342)/512.f;
    tx2 = 506/1024.f;
    ty2 = ty1;
    tx3 = tx2;
    ty3 = (512-170)/512.f;
        tx4 = tx1;
    ty4 = ty3;
    DrawQuadT(-1,-1,1, 1,-1,1, 1,1,1, -1,1,1);
```

```
    //2 – левая грань
glNormal3d(-1,0,0);
tx1 = 164/1024.f;
ty1 = (512-342)/512.f;
tx2 = 335/1024.f;
ty2 = ty1;
tx3 = tx2;
ty3 = (512-170)/512.f;
    tx4 = tx1;
ty4 = ty3;
     DrawQuadT(-1,-1,-1, -1,-1,1, -1,1,1, -1,1,-1);

//3 – правая грань
glNormal3d(1,0,0);
tx1 = 506/1024.f;
ty1 = (512-342)/512.f;
tx2 = 678/1024.f;
ty2 = ty1;
tx3 = tx2;
ty3 = (512-170)/512.f;
    tx4 = tx1;
ty4 = ty3;
DrawQuadT(1,-1,1, 1,-1,-1, 1,1,-1, 1,1,1);
```

```
//4 – задняя грань
glNormal3d(0,0,-1);
tx1 = 676/1024.f;
ty1 = (512-342)/512.f;
tx2 = 850/1024.f;
ty2 = ty1;
tx3 = tx2;
ty3 = (512-170)/512.f;
tx4 = tx1;
ty4 = ty3;
DrawQuadT(1,-1,-1, -1,-1,-1, -1,1,-1, 1,1,-1);

//5 – верхняя грань
glNormal3d(0,1,0);
tx1 = 335/1024.f;
ty1 = (512-170)/512.f;
tx2 = 506/1024.f;
ty2 = ty1;
tx3 = tx2;
ty3 = 1;
tx4 = tx1;
ty4 = ty3;
DrawQuadT(-1,1,1, 1,1,1, 1,1,-1, -1,1,-1);
```

```
    //6 - нижняя грань
    glNormal3d(0,-1,0);
    tx1 = 335/1024.f;
    ty1 = 0;
    tx2 = 506/1024.f;
    ty2 = ty1;
    tx3 = tx2;
    ty3 = (512-342)/512.f;
    tx4 = tx1;
    ty4 = ty3;
    DrawQuadT(-1,-1,1, 1,-1,1, 1,-1,-1, -1,-1,-1);
}

//Функция Draw рисует вращающийся кубик
void Draw()
{
   glLoadIdentity();
   glRotated(GetTickCount() % 36000 / 10.f, 1, 1, 1);
   DrawRubik();
}

void CALLBACK display(void)
{
  glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
  Draw();//Вызываем рисование вращающегося кубика в display
  auxSwapBuffers();
}
```

```
void main()
{
    auxInitPosition( 0, 0, 600, 600);
    auxInitDisplayMode( AUX_RGB | AUX_DEPTH | AUX_DOUBLE );
    auxInitWindow( "Glaux Template" );
    auxReshapeFunc(resize);
    auxIdleFunc(display);
    glClearColor(0.2f, 0.3f, 0.4f, 0.0f);//Цвет фона окна
    glEnable(GL_DEPTH_TEST);
    LoadBMP(&texture[0], "rubik.bmp");//Загрузка текстуры
    auxMainLoop(display);
}
```

В данной программе нет освещения. Кубик будет выглядеть так, как показано на рис.1. На рис. 2 уже имеется один источник света, расположенный на оси X.

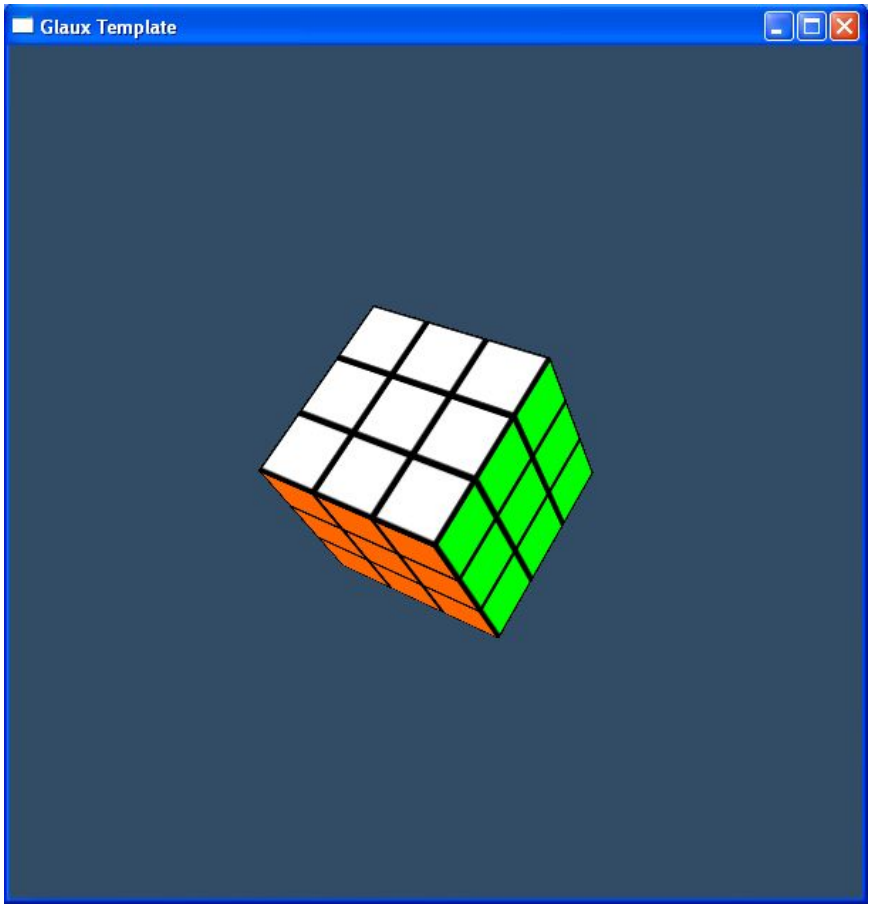


Рис.1

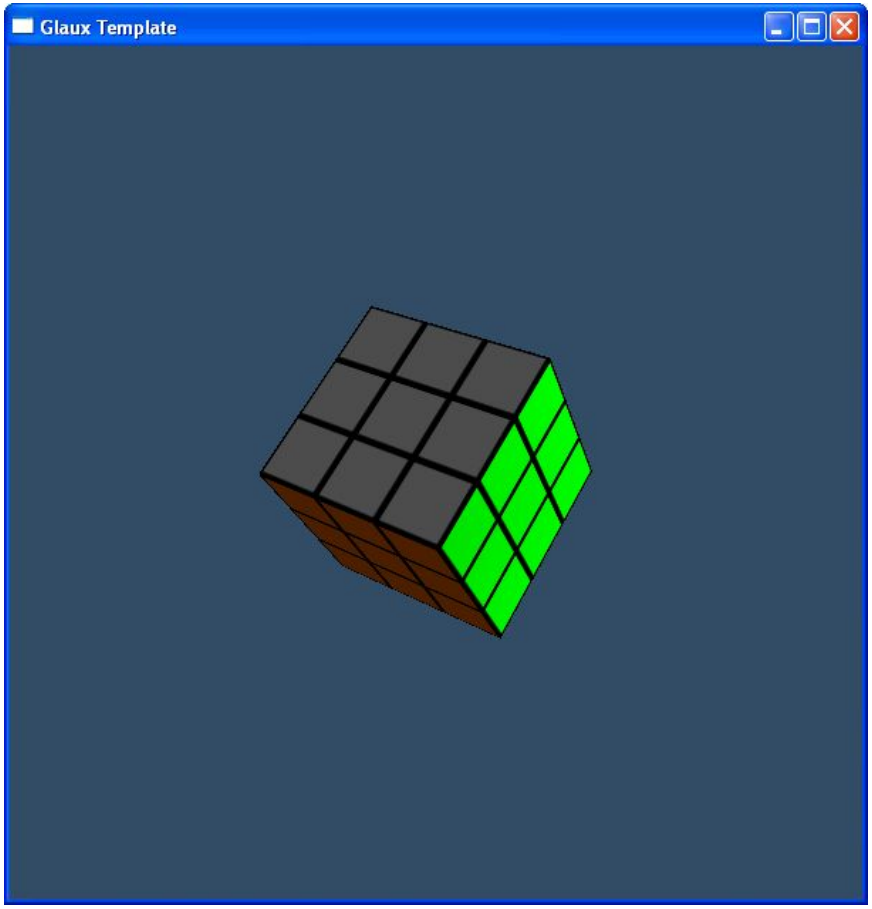


Рис.2