

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №5

ОСНОВЫ НАЛОЖЕНИЯ ТЕКСТУР В OPENGL

Цели: формирование практических навыков по работе с текстурами средствами OpenGL, их наложению на освещенные объекты подверженные проекционному сокращению.

Задачи: понимать принципы наложения растровых изображений на геометрические объекты, уметь реализовывать наложение текстур с использованием возможностей OpenGL, научиться использовать наложение множественных текстур, уметь создавать фотореалистичные сцены (корректное освещение), на которых присутствуют текстурированные геометрические объекты.

Выполнение:

1. Ознакомиться с теоретическим материалом.
2. Выполнить основные задания.
3. Предоставить отчет, по каждому заданию содержащий: формулировку задания, исходный код программы, скриншоты работающей программы (один или несколько, если необходимо).
4. Ответить на вопросы преподавателя.

В Листинге 1 представлен код программы, в котором происходит отрисовка текстуры и наложение ее на квадрат.

Листинг 1.

```
#include "glew.h"
#include "glut.h"
#include <cstring>

unsigned int texture;
//зададим массив координат квадрата
float vertex[] = { -1,-1,0,1,-1,0,1,1,0,-1,1,0 };
//массив который хранит текстурные координаты для каждой вершины
float texCoord[] = { 0,0,1,0,1,1,0,1 };

// процедура для создания квадрата и
void display() {

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, texture);
    //Активные цвета текстуры
    glColor3f(1, 1, 1);
```

```

    glPushMatrix();
    // Установка состояние для OpenGL, означающее, что мы будем использовать
указатель на массив вершин
    glEnableClientState(GL_VERTEX_ARRAY);
    // Установка состояние для OpenGL, означающее, что мы будем использовать
указатель на массив текстурных координат
    glEnableClientState(GL_TEXTURE_COORD_ARRAY);
    // Установка массива вершинных координат квадрата
    glVertexPointer(3, GL_FLOAT, 0, vertex);
    // Установка массива текстурных координат
    glTexCoordPointer(2, GL_FLOAT, 0, texCoord);
    //Выводит примитивы по данным в массиве для квадрата
    glDrawArrays(GL_TRIANGLE_FAN, 0, 4);
    glDisableClientState(GL_VERTEX_ARRAY);
    glDisableClientState(GL_TEXTURE_COORD_ARRAY);
    glPopMatrix();
    glutSwapBuffers();
}

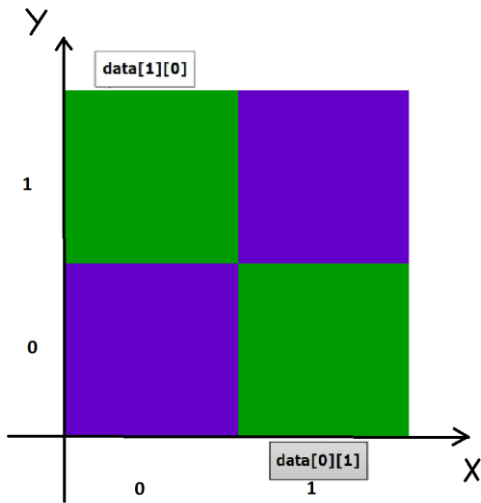
void Text_Init()
{
    // процедура для создания текстуры
    int width, height;
    width = 2;
    height = 2;
    //создаем двумерный массив 2 на 2 текселя
    struct { unsigned char r, g, b, a; } data[2][2];
    memset(data, 0, sizeof(data));
    //задаем цвет для каждого текселя структуры
    data[0][0].r = 100;
    data[0][0].b = 200;
    data[1][0].g = 155;
    data[0][1].g = 155;
    data[1][1].r = 100;
    data[1][1].b = 200;

    //Создадим имена текстур. 1-количество текстур.
    glGenTextures(1, &texture);
    //выбирает указанную текстуру как активную для наложения ее на объекты. После
этого все настройки текстуры применяются к этой активной текстуре.
    glBindTexture(GL_TEXTURE_2D, texture);
    //Основные настройки текстуры
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0,
        GL_RGBA, GL_UNSIGNED_BYTE, data);
    //Отключаем активную текстуру
    glBindTexture(GL_TEXTURE_2D, 0);
}

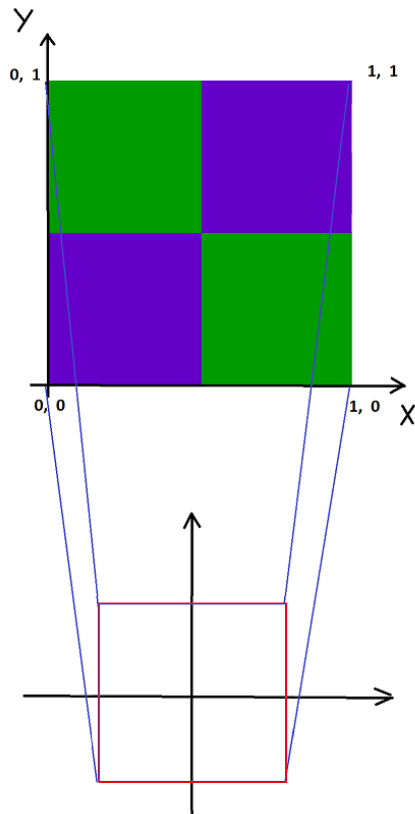
int main() {
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(600, 600);
    glutCreateWindow("Текстура-квадраты");
    Text_Init();
    glutDisplayFunc(display);
    glutMainLoop();
}

```

Обратите внимание каким образом формируется массив с текстурными координатами data. Первый индекс – это координата Y, а вторая – X.



Для точной передачи изображения текстуры необходимо понимать как текстурные координаты сопоставляются с экранными.



Рассмотрим вариант загрузки текстуры из файла. Для этого скачаем сам файл картинки в любом формате, а затем установим библиотеку `stb_image`. Для этого подключим файл `stb_image.h`. Не забудьте поместить файл с картинкой в папку с проектом.

Листинг 2

```
#define STB_IMAGE_IMPLEMENTATION
#include "glew.h"
```

```

#include "glut.h"
#include "stb_image.h"
#include <cstring>

unsigned int texture;

float vertex[] = { -1,-1,0,1,-1,0,1,1,0,-1,1,0 };

//массив который хранит текстурные координаты для каждой вершины
float texCoord[] = { 1,1,1,0,0,0,0,1 };

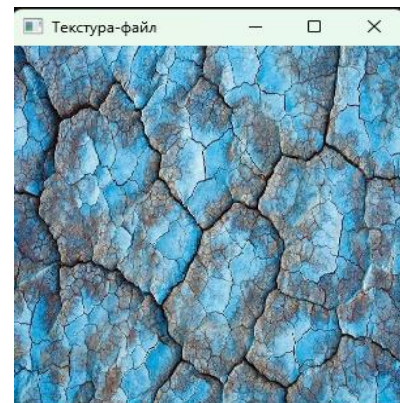
void display() {

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, texture);
    glColor3f(1, 1, 1);
    glPushMatrix();
        glEnableClientState(GL_VERTEX_ARRAY);
        glEnableClientState(GL_TEXTURE_COORD_ARRAY);
        glVertexPointer(3, GL_FLOAT, 0, vertex);
        glTexCoordPointer(2, GL_FLOAT, 0, texCoord);
        glDrawArrays(GL_TRIANGLE_FAN, 0, 4);
        glDisableClientState(GL_VERTEX_ARRAY);
        glDisableClientState(GL_TEXTURE_COORD_ARRAY);
    glPopMatrix();
    glutSwapBuffers(); }

void Game_Init()
{
    int width, height, cnt;
    //загрузим данные картинки
    unsigned char* data = stbi_load("1.jpeg", &width, &height, &cnt, 0);
    glGenTextures(1, &texture);
    glBindTexture(GL_TEXTURE_2D, texture);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0, cnt==4 ?
        GL_RGBA: GL_RGB , GL_UNSIGNED_BYTE, data);
    glBindTexture(GL_TEXTURE_2D, 0);
    stbi_image_free(data);
}

int main() {
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutCreateWindow("Текстура-файл");
    glutInitWindowSize(500, 500);
    Game_Init();
    glutDisplayFunc(display);
    glutMainLoop();
}

```



Множественная текстура.

Применение *сокращённой* (или *множественной*) *текстуры* (mipmapping) является мощной техникой наложения текстуры, позволяющей повысить и производительность визуализации, и визуальное качество сцены.

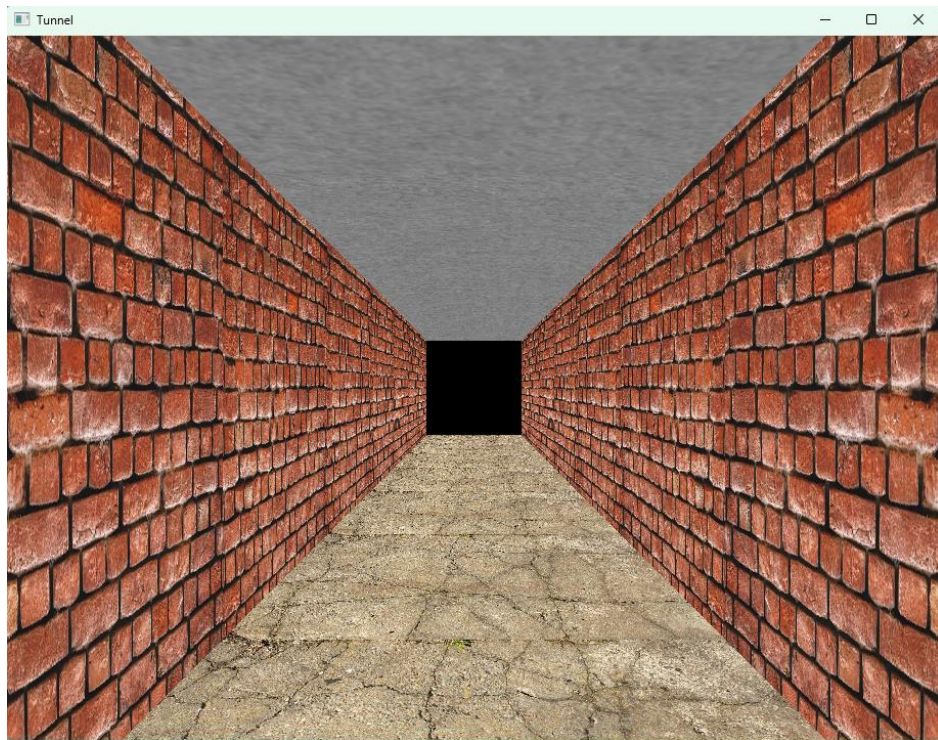
Функция **gluBuild2DMipmaps** создает 2-D MIP-карты.

Эта текстура добавляет новый прием к двум базовым режимам текстурной фильтрации GL_NEAREST и GL_LINEAR, предоставляя четыре перестановки варианта фильтрации множественной текстуры, перечисленных в таблице.

GL_NEAREST	Фильтрация по ближайшему "соседу" на основном уровне текстуры
GL_LINEAR	Линейная фильтрация на основном уровне текстуры
GL_NEAREST_MIPMAP_NEAREST	Выбор ближайшего уровня текстуры и выполнение фильтрации по ближайшему «соседу»
GL_NEAREST_MIPMAP_LINEAR	Выполнение линейной интерполяции между уровнями текстуры и выполнение фильтрации по ближайшему "соседу"
GL_LINEAR_MIPMAP_NEAREST	Выбор ближайшего уровня текстуры и выполнение линейной фильтрации
GL_LINEAR_MIPMAP_LINEAR	Выполнение линейной интерполяции между уровнями текстуры и выполнение линейной фильтрации, также называется <i>трилинейной фильтрацией</i> или <i>трилинейным множественным отображением</i>

Рассмотрим программу, которая иллюстрирует множественное отображение и различные режимы фильтрации текстуры множественного отображения. Нажимая клавиши со стрелками вверх и вниз, вы перемещаете точку наблюдения назад-вперед по туннелю, а контекстное меню (вызывается щелчком правой кнопки мыши) позволяет переключаться между шестью различными режимами

фильтрации и сравнивать их влияние на визуализацию изображения. Исходный код программы приводится в листинге 3.



Листинг 3.

```
#define STB_IMAGE_IMPLEMENTATION
#include <C:\Users\AdminHome\Documents\GL\stb_image.h>
#include "glew.h"
#include "glut.h"
#include <math.h>

// Rotation amounts
static GLfloat zPos = -60.0f;
unsigned int texture[3];

void ProcessMenu(int value)
{
    GLint iLoop;
    for (iLoop = 0; iLoop < 3; iLoop++)
    {
        glBindTexture(GL_TEXTURE_2D, texture[iLoop]);
        switch (value)
        {
            case 0:
                glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_NEAREST);
                break;
            case 1:
                glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
                break;
            case 2:
                glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_NEAREST_MIPMAP_NEAREST);
                break;
            case 3:
```



```

        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_NEAREST_MIPMAP_LINEAR);
        break;
    case 4:
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR_MIPMAP_NEAREST);
        break;
    case 5:
    default:
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR_MIPMAP_LINEAR);
        break;
    }
}

glutPostRedisplay();
}

void SetupRC()
{
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    //разрешаем наложение текстуры
    glEnable(GL_TEXTURE_2D);
    // функции преобразования цветов источника света, цвета образа текстуры,
    //цвета вершин примитивов и цвета конфигурации текстуры для получения
    результирующего цвета
    //поверхности с наложенной на нее текстурой
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);

    int width, height, cnt;
    // генерируем 3 текстуры
    glGenTextures(3, texture);

    //текстура 1
    unsigned char* data1 = stbi_load("2.jpg", &width, &height, &cnt, 0);
    glBindTexture(GL_TEXTURE_2D, texture[0]);
    //создает все изображения MIP-карты
    gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGB, width, height, GL_RGB,
GL_UNSIGNED_BYTE, data1);
    //задаем параметры
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
    //теперь активных текстур 0
    glBindTexture(GL_TEXTURE_2D, 0);
    //освобождаем память
    stbi_image_free(data1);

    //текстура 2
    unsigned char* data2 = stbi_load("3.tga", &width, &height, &cnt, 0);
    glBindTexture(GL_TEXTURE_2D, texture[1]);
    gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGB, width, height, GL_RGB,
GL_UNSIGNED_BYTE, data2);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glBindTexture(GL_TEXTURE_2D, 0);
    stbi_image_free(data2);

    //текстура 3

```

```

        unsigned char* data3 = stbi_load("1.tga", &width, &height, &cnt, 0);
        glBindTexture(GL_TEXTURE_2D, texture[2]);
        gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGB,width, height, GL_RGB,
GL_UNSIGNED_BYTE, data3);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
        glBindTexture(GL_TEXTURE_2D, 0);
        stbi_image_free(data3);
    }

```

```

void SpecialKeys(int key, int x, int y)
{
    if (key == GLUT_KEY_UP)
        zPos += 1.0f;
    if (key == GLUT_KEY_DOWN)
        zPos -= 1.0f;
    // Refresh the Window
    glutPostRedisplay();
}

```

```

void ChangeSize(int w, int h)
{
    GLfloat fAspect;
    // Prevent a divide by zero
    if (h == 0)
        h = 1;
    // Set Viewport to window dimensions
    glViewport(0, 0, w, h);
    fAspect = (GLfloat)w / (GLfloat)h;
    // Reset coordinate system
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    // Produce the perspective projection
    gluPerspective(90.0f, fAspect, 1, 120);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

```

```

void RenderScene(void)
{
    GLfloat z;

    glClear(GL_COLOR_BUFFER_BIT );
    glEnable(GL_TEXTURE_2D);
    glPushMatrix();
    glTranslatef(0.0f, 0.0f, zPos);

    for (z = 60.0f; z >= 0.0f; z -= 10)
    {
        //пол
        glBindTexture(GL_TEXTURE_2D, texture[0]);
        glBegin(GL_QUADS);
        glTexCoord2f(0.0f, 0.0f);
        glVertex3f(-10.0f, -10.0f, z);
        glTexCoord2f(1.0f, 0.0f);
        glVertex3f(10.0f, -10.0f, z);
        glTexCoord2f(1.0f, 1.0f);
        glVertex3f(10.0f, -10.0f, z - 10.0f);
        glTexCoord2f(0.0f, 1.0f);
        glVertex3f(-10.0f, -10.0f, z - 10.0f);
    }
}

```



```

        glEnd();
        //потолок
        glBindTexture(GL_TEXTURE_2D, texture[2]);
        glBegin(GL_QUADS);
        glTexCoord2f(0.0f, 1.0f);
        glVertex3f(-10.0f, 10.0f, z - 10.0f);
        glTexCoord2f(1.0f, 1.0f);
        glVertex3f(10.0f, 10.0f, z - 10.0f);
        glTexCoord2f(1.0f, 0.0f);
        glVertex3f(10.0f, 10.0f, z);
        glTexCoord2f(0.0f, 0.0f);
        glVertex3f(-10.0f, 10.0f, z);
        glEnd();
        // левая стена
        glBindTexture(GL_TEXTURE_2D, texture[1]);
        glBegin(GL_QUADS);
        glTexCoord2f(0.0f, 0.0f);
        glVertex3f(-10.0f, -10.0f, z);
        glTexCoord2f(1.0f, 0.0f);
        glVertex3f(-10.0f, -10.0f, z - 10.0f);
        glTexCoord2f(1.0f, 1.0f);
        glVertex3f(-10.0f, 10.0f, z - 10.0f);
        glTexCoord2f(0.0f, 1.0f);
        glVertex3f(-10.0f, 10.0f, z);
        glEnd();
        // правая стена
        glBegin(GL_QUADS);
        glTexCoord2f(0.0f, 1.0f);
        glVertex3f(10.0f, 10.0f, z);
        glTexCoord2f(1.0f, 1.0f);
        glVertex3f(10.0f, 10.0f, z - 10.0f);
        glTexCoord2f(1.0f, 0.0f);
        glVertex3f(10.0f, -10.0f, z - 10.0f);
        glTexCoord2f(0.0f, 0.0f);
        glVertex3f(10.0f, -10.0f, z);
        glEnd();
    }

    glPopMatrix();

    glutSwapBuffers();
}

int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Tunnel");
    glutReshapeFunc(ChangeSize);
    glutSpecialFunc(SpecialKeys);
    glutDisplayFunc(RenderScene);
    // настройки меню
    glutCreateMenu(ProcessMenu);
    glutAddMenuEntry("GL_NEAREST", 0);
    glutAddMenuEntry("GL_LINEAR", 1);
    glutAddMenuEntry("GL_NEAREST_MIPMAP_NEAREST", 2);
    glutAddMenuEntry("GL_NEAREST_MIPMAP_LINEAR", 3);
    glutAddMenuEntry("GL_LINEAR_MIPMAP_NEAREST", 4);
    glutAddMenuEntry("GL_LINEAR_MIPMAP_LINEAR", 5);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    SetupRC();
    glutMainLoop();
    return 0;
}

```

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Воспроизвести результаты, представленные в теоретическом обзоре, освоить наложение текстур и работу с множеством текстурных объектов, согласно варианту, полученному у преподавателя, наложить текстуры на объекты сцены.

Задание 1.

На основе листинга 1. Внести следующие изменения:

- 1) С помощью меню менять не менее 4 параметров в `glTexParameteri`
- 2) Добавить еще 3 текстуры и вывести их одновременно в виде:

Текстура 1	Текстура 2
Текстура 3	Текстура 4

Задание 2.

Наложить произвольную текстуру на геометрический объект.

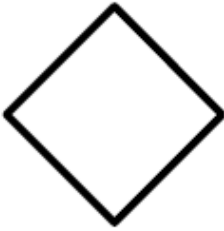
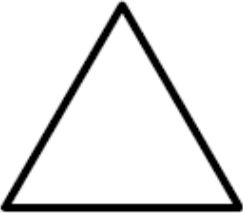
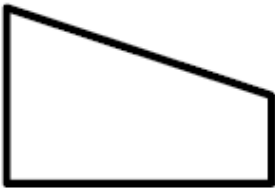


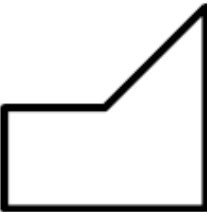




Варианты объектов:

- 1) Сфера
- 2) Тор
- 3) Куб
- 4) Цилиндр
- 5) Чайник Юта
- 6) Призма четырехгранная (не куб)
- 7) Параллелепипед
- 8) Тетраэдр
- 9) Конус
- 10) Додекаэдр

Задание 3.

Используя [Листинг 3](#) создать коридор по форме указанной в варианте. Для каждой грани использовать свою собственную текстуру т.е. отдельный файл

Варианты объектов:

1. 	6. 
2. 	7. 
3. 	8. 
4. 	9. 
5. 	10. 

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Сформулируйте понятие текселя и его характеристики.
2. Перечислите функции для загрузки текстур и параметры для их работы.
3. Оцените, какая операция является узким местом при работе с текстурами.
4. Приведите функцию, используемую для отображения текстуры на геометрические объекты.

5. Дайте определение фильтрации и поясните её роль.
6. Изложите назначение множественной (сокращенной) текстуры.
7. Покажите, по каким осям происходит фильтрация множественной текстуры.
8. Опишите работу функции генерации уровней множественной текстуры.