|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кафедра |  | O7 |  | Информационные системы и программная инженерия |
|  |  | шифр |  | наименование кафедры, по которой выполняется работа |
| Дисциплина |  | Компьютерная геометрия и графика | | |
|  |  | наименование дисциплины | | |

|  |  |  |
| --- | --- | --- |
| ЛАБОРАТОРНАЯ РАБОТА | 7 |  |
|  | номер (при наличии) |  |
| Текстуры: режимы фильтрации,  режимы взаимодействия текстуры с накладываемым объектом. | | |
| Автоматическая генерация текстурных координат. | | |
|  | | |

при наличии указать тему лабораторной работы и (или) номер варианта

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ОБУЧАЮЩИЙСЯ** | | | | | | |  |
| группы | | | |  | | И913Б |  |
|  |  | Кульга Н.И | | | | |  |
| подпись |  | фамилия и инициалы | | | | |  |
|  | | | | |  |  |  |
| дата сдачи | | | | |  |  |  |
| **ПРОВЕРИЛ** | | | | | | |  |
|  | | | | | | | |
| ученая степень, ученое звание, должность | | | | | | | |
|  |  | Мажайцев Е.А | | | | | |
| подпись |  | фамилия и инициалы | | | | | |
| Оценка / балльная оценка | | |  | | | |  |
|  | | | | |  |  |  |
| дата проверки | | | | |  |  |  |

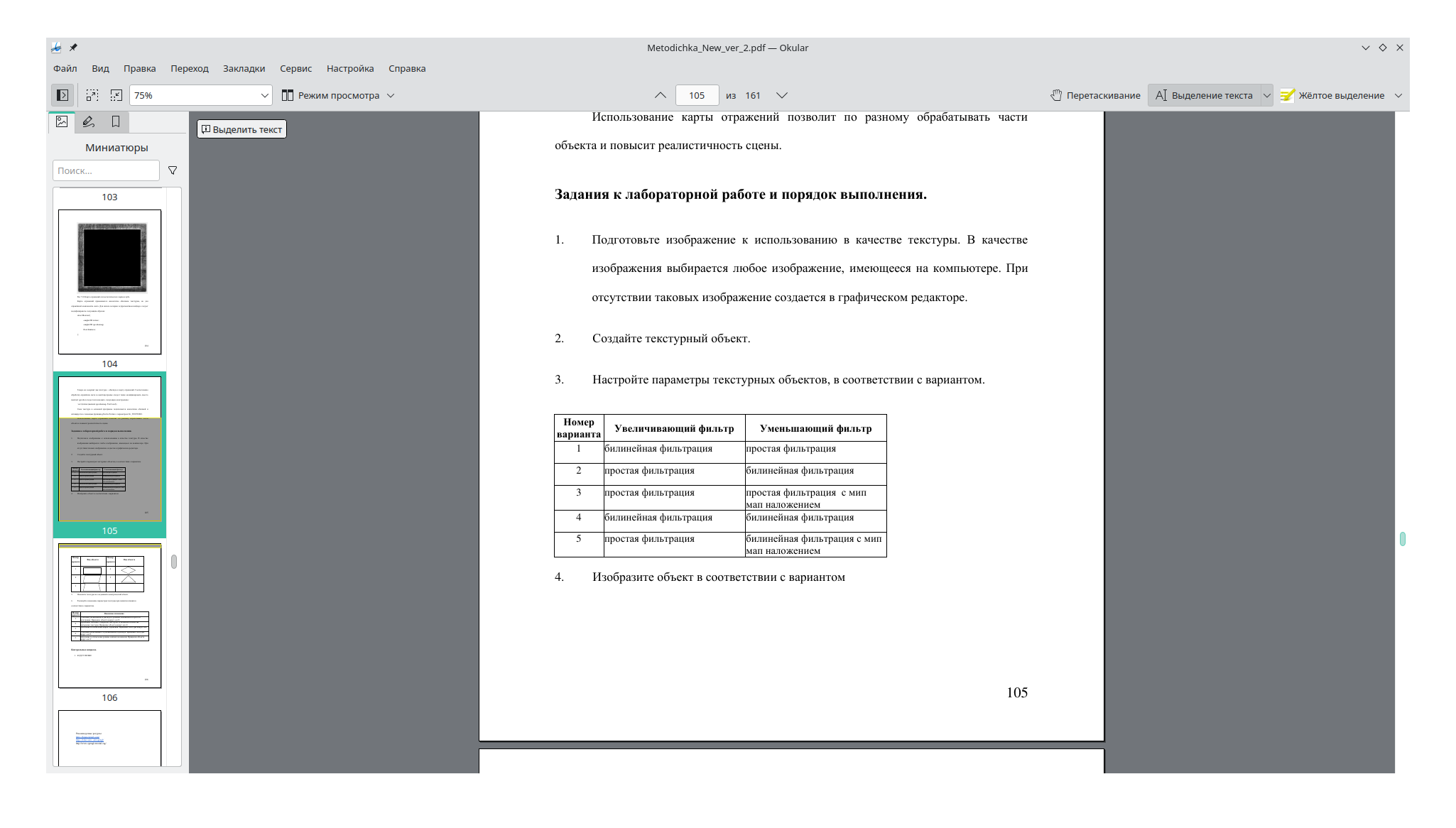
Задание:

Задание 1.

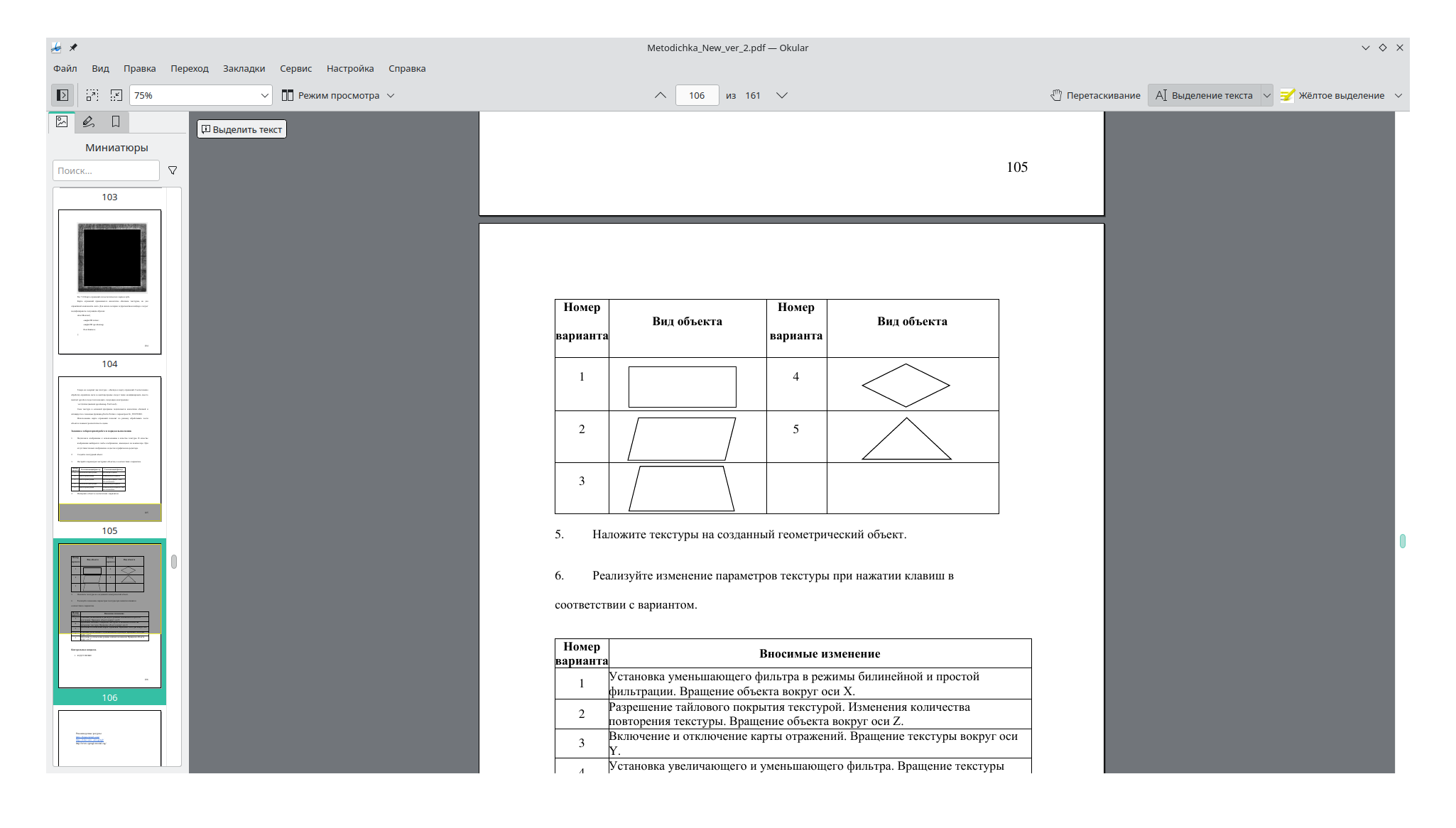
1. Подготовьте изображение к использованию в качестве текстуры. В качестве изображения выбирается любое изображение, имеющееся на компьютере. При отсутствии таковых изображение создается в графическом редакторе.

2. Создайте текстурный объект.

3.Настройте параметры текстурных объектов, в соответствии с вариантом.

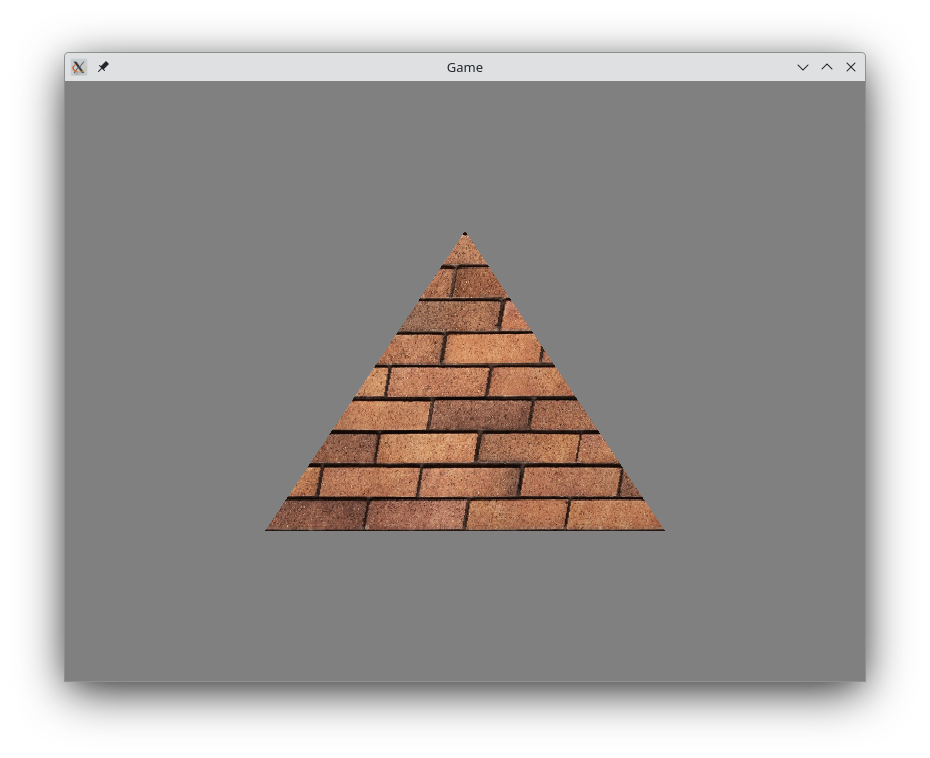
Результат работы программы:

4. Изобразите объект в соответствии с вариантом



5. Наложите текстуры на созданный геометрический объект.

6. Реализуйте изменение параметров текстуры при нажатии клавиш в соответствии с вариантом.

Рисунок 1 — Результат выполненияTimes New Roman

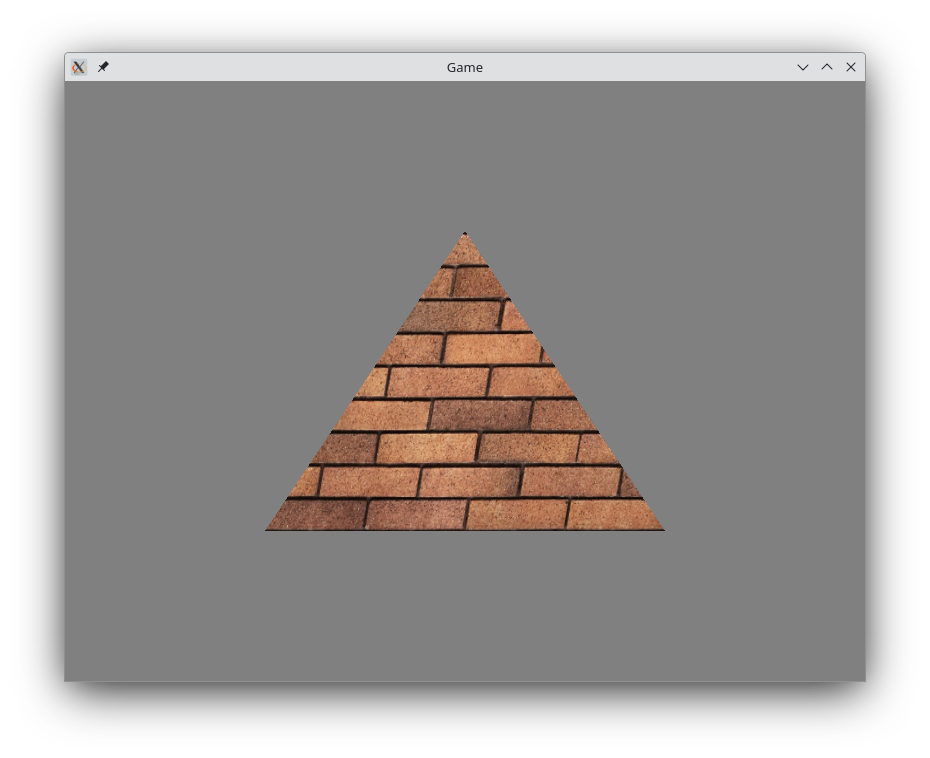


Рисунок 2 — Результат выполнения

Код программы:

package lr7.models;  
  
import lr7.textures.ModelTexture;  
*/\*\*  
 \* Класс Текстурированной модели.  
 \* Этот класс объединит данные модели и текстуры.  
 \*/*public class TexturedModel {  
  
 */\*\* загруженная модель \*/* private RawModel model;  
   
 */\*\* загруженная текстура \*/* private ModelTexture texture;  
  
 public TexturedModel(RawModel model, ModelTexture texture) {  
 this.model = model;  
 this.texture = texture;  
 }  
  
 */\*\*  
 \* Получение данных модели  
 \** ***@return*** *модель  
 \*/* public RawModel getRawModel() {  
 return model;  
 }  
  
 */\*\*  
 \* Получение данных текстуры  
 \** ***@return*** *текстуру  
 \*/* public ModelTexture getTexture() {  
 return texture;  
 }  
}

package lr7.textures;  
  
*/\*\*  
 \* Класс для текстурирования нашей модели  
 \*/*public class ModelTexture {  
  
 */\*\* Идентификатор текстуры \*/* private int textureId;  
  
 public ModelTexture(int textureId) {  
 this.textureId = textureId;  
 }  
  
 public int getId() {  
 return textureId;  
 }  
}

package lr7.textures;  
  
import org.lwjgl.opengl.GL11;  
import org.lwjgl.opengl.GL30;  
import org.lwjgl.stb.STBImage;  
import org.lwjgl.system.MemoryStack;  
  
import java.nio.ByteBuffer;  
import java.nio.IntBuffer;  
  
import static org.lwjgl.opengl.GL11.\*;  
import static org.lwjgl.opengl.GL11C.*glTexParameteri*;  
import static org.lwjgl.opengl.GL43.*GL\_TEXTURE\_2D*;  
  
public class Texture {  
  
 private final int id;  
  
 public Texture(int id) {  
 this.id = id;  
  
 }  
  
 public Texture(String fileName, boolean bilinearFilteringWithMipMapOverlay)throws Exception{  
 //this.id=loadTexture(fileName) или как сделано, создаём объект исполььзуя конструктор выше через this  
 this(*loadTexture*(fileName,bilinearFilteringWithMipMapOverlay));  
 }  
  
 private static int loadTexture(String fileName, boolean bilinearFilteringWithMipMapOverlay)throws Exception{  
 int width, height;  
 // Будет содержать декодированное изображение (так как каждый пиксель  
 // использует четыре байта, его размер будет 4 ширины по высоте)  
 ByteBuffer buf;  
  
 // Загрузка файла текстуры  
 try(MemoryStack stack=MemoryStack.*stackPush*()) {  
 IntBuffer w = stack.mallocInt(1);  
 IntBuffer h = stack.mallocInt(1);  
 IntBuffer channels = stack.mallocInt(1);  
  
 buf = STBImage.*stbi\_load*(fileName, w, h, channels, 4);  
 if (buf == null)  
 throw new Exception("Файл текстуры [" + fileName + "] не загружен: " + STBImage.*stbi\_failure\_reason*());  
 // получение высоты и ширины картинки  
 width = w.get();  
 height = h.get();  
 }  
 //создание новой текстуры  
 int texturedId= GL11.*glGenTextures*();  
 // присоединяем текстуру  
 GL11.*glBindTexture*(GL11.*GL\_TEXTURE\_2D*,texturedId);  
  
 // Говорит OpenGL, как распаковать байты RGBA. Каждый компонент имеет размер 1 байт.  
 GL11.*glBindTexture*(GL11.*GL\_UNPACK\_ALIGNMENT*,1);  
  
 // Этот параметр в основном говорит о том, что когда пиксель рисуется  
 // без прямой связи один к одному с координатой текстуры, он выбирает  
 // ближайшую точку текстуры.  
 //glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST);  
  
 if(!bilinearFilteringWithMipMapOverlay)  
  
 *glTexParameteri*(*GL\_TEXTURE\_2D*, *GL\_TEXTURE\_MIN\_FILTER*, *GL\_LINEAR\_MIPMAP\_NEAREST*);  
 else  
 *glTexParameteri*(*GL\_TEXTURE\_2D*, *GL\_TEXTURE\_MAG\_FILTER*, *GL\_NEAREST*);  
  
 // уровень детализации текстуры  
 // GL11.glTexParameterf(GL11.GL\_TEXTURE\_2D, GL14.GL\_TEXTURE\_LOD\_BIAS, 0.5f);  
  
 // Выгрузка данных текстуры  
 // [1] GL\_TEXTURE\_2D - Определяет целевую текстуру (ее тип)  
 // [2] level - указывает номер уровня детализации. Уровень 0 - это базовый уровень изображения. Уровень n - это n-е уменьшенное изображение mipmap  
 // [3] internal format - Определяет количество цветовых компонентов в текстуре.  
 // [4] width - Определяет ширину текстурного изображения  
 // [5] height - Определяет высоту текстурного изображения  
 // [6] border - Это значение должно быть нулевым  
 // [7] format - Определяет формат данных пикселей: в данном случае RGBA.  
 // [8] type - Определяет тип данных данных пикселей. Для этого мы используем неподписанные байты.  
 // [9] data - Буфер, в котором хранятся наши данные  
 GL11.*glTexImage2D*(GL11.*GL\_TEXTURE\_2D*, 0, GL11.*GL\_RGBA*, width, height, 0,  
 GL11.*GL\_RGBA*, GL11.*GL\_UNSIGNED\_BYTE*, buf);  
  
 // Генерация Мип-карта  
 // Мип-карта - это набор изображений с уменьшенным разрешением,  
 // созданный из высокодетализированной текстуры. Эти изображения с  
 // более низким разрешением будут использоваться автоматически при  
 // масштабировании нашего объекта.  
 GL30.*glGenerateMipmap*(GL11.*GL\_TEXTURE\_2D*);  
  
 // освобождаем буфер  
 STBImage.*stbi\_image\_free*(buf);  
 return texturedId;  
 }  
  
 public void cleanup(){  
 GL11.*glDeleteTextures*(id);  
 }  
  
 public int getId(){  
 return id;  
 }  
  
}

#version 400 core  
  
// получаем текстурные координаты  
in vec2 pass\_textureCoords;  
  
out vec4 fragColor;  
  
uniform sampler2D textureSampler;  
  
void main()  
{  
 // метод texture координату точки на модели сопоставляет с координатой точи на текстуре  
 // и возвращает цвет пикселя   
 fragColor = texture(textureSampler, pass\_textureCoords);  
}

#version 400 core  
  
// получаем позиции вершин из VAO  
in vec3 position;  
// получаем текстурные координаты из VAO  
in vec2 textureCoords;  
  
// отправляем в фрагментный шейдер текстурные координаты  
out vec2 pass\_textureCoords;  
  
void main()  
{  
 // визуализируем вершину в форме (x, y, z, w)  
 gl\_Position = vec4(position, 1.0);  
  
 pass\_textureCoords = textureCoords;  
}