

# Unofficial LWJGL FAQ

## *Contents*

1. How to create a AWTGLCanvas ...Page 2
2. How do I create a texture loader? ...Page 3

# Unofficial LWJGL FAQ

## How to create an AWTGLCanvas

- Create a frame
- Create an AWTGLCanvas
- Add listeners to the canvas. I do this and it works.
- Add the canvas to the frame.

The AWTGLCanvas can be used as either an anonymous inner class, and inner class or a separate class.

One thing you have to do is override the “paintGL()” method.

This is where all your painting will occur.

At the end of the method, after you finish painting make sure you call “swapBuffers()” to display what you've rendered.

Whenever you want to draw something to the screen you must do it within the “paintGL()” method, anywhere else it wont work. Make sure you also call “repaint()” on the AWTGLCanvas at the end of your game loop.

*Creating and adding the canvas:*

```
//GameLoop extends AWTGLCanvas
game = new GameLoop(width, height, vsync);
frame = new JFrame("Test");

frame.add(game);

//You must request focus if you want your AWT listeners(Key/Mouse) to work
game.requestFocus();

//adding listeners to the AWTGLCanvas
inputs.addKeyListenerTo(game);
inputs.addMouseListenerTo(game);
```

*Creating the canvas:*

```
public class GameLoop extends AWTGLCanvas{
    protected void paintGL() //Overrides “paintGL()”
    {
        drawMyGameStuff();

        try
        {
            this.swapBuffers(); //show what you were drawing
        } catch (LWJGLEException e)
        {
            e.printStackTrace();
        }
    }
}
```

## Unofficial LWJGL FAQ

### How do I create a texture loader?

First you need to know how OpenGL does it. I won't display that here as it is outside the scope of this FAQ.

What I will show you is how to use LWJGL to accomplish this.

It might be easier to create a class that holds the texture ID and the width and height of the texture.

Using the texture is EXACTLY the same in LWJGL as it is in C or C++.

NOTE: All buffers are direct buffers. Which means they are native. Make sure you the ByteOrder is set to 'native'.

Here are the steps:

- Create the texture by loading it into a buffered image, just like you would normally load an image in Java.

*BufferedImage img = ImageIO.read(new File(name));*

- Get the raster from the buffered image.

*img.getRaster()*

- Get the data buffer from the raster.

*raster.getDataBuffer()*

- Get the data from the data buffer.

- Cast the data to a DataBufferByte, which is an array of type *byte*.

*byte[] tmp = (DataBufferByte)dataBuffer.getData()*

- Create a ByteBuffer, make the capacity the length of the 'data' array.

*ByteBuffer imgTransport = BufferUtils.createByteBuffer(tmp.length)*

- Put the 'data' array into the byte buffer.

*imgTransport.put(tmp);*

- Rewind the buffer, or flip it. In either case, the mark has to be 0.

*imgTransport.flip();*

- Create an empty IntBuffer with a capacity of 1. You are only going to create a single texture.

*IntBuffer intBuff = BufferUtils.createIntBuffer(1);*

## Unofficial LWJGL FAQ

- Generate the texture id by parsing the empty IntBuffer into “glGenTextures”.  
*GL11.glGenTextures(intBuff):*
- Bind the texture with “glBindTexture” by getting the id with “intBuff.get(0)”. You get the integer at the first position because A) you only have a capacity of 1 and B) you're only getting a single texture.  
*GL11.glBindTexture(GL11.GL\_TEXTURE\_2D, intBuff.get(0)):*
- Create the texture in memory by parsing in the appropriate parameters, and finally parse the ByteBuffer as the last argument in “glTexImage2D”.  
*GL11.glTexImage2D(  
    GL11.GL\_TEXTURE\_2D,  
    0,  
    img.getColorModel().hasAlpha() ? GL11.GL\_RGBA : GL11.GL\_RGB,  
    img.getWidth(),  
    img.getHeight(),  
    0,  
    img.getColorModel().hasAlpha() ? GL11.GL\_RGBA : GL11.GL\_RGB,  
    GL11.GL\_UNSIGNED\_BYTE,  
    imgTransport //byte buffer  
);*
- Set the texture filtering as either bilinear or trilinear. Which is exactly the same in C/C++ as it is in LWJGL.
- Return the texture ID to that texture with the method “intBuff.get(0)”?