Textures

How to add detail to a surface…

Gives fine details to our otherwise flat polygons.

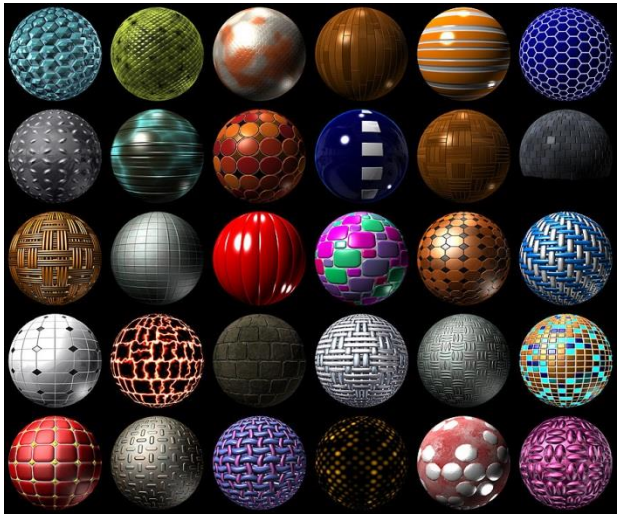
Can be used for visualizing complex objects
Without needing to model them with triangles.

A sky can be applied as a texture to a sphere
and will look good.

Texture sources

- Stored images
  - BMP,JPG,PNG,GIF….
  - Raster/Vector (SVG)

- Procedural textures
  - Created by mathematical equation.
  - Works great on modern shader hardware.

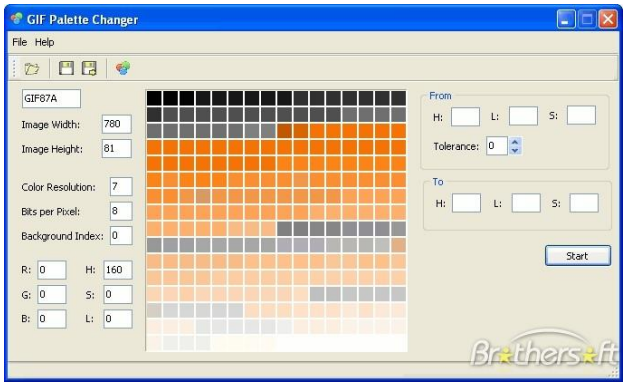


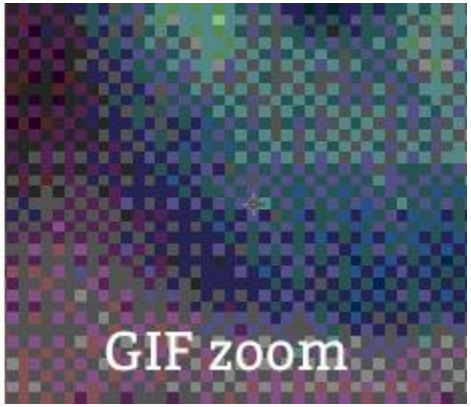
Raster formats

- Compressed – Non compressed (BMP)
- Lossless (PNG) – Lossy (JPEG)

Palette format

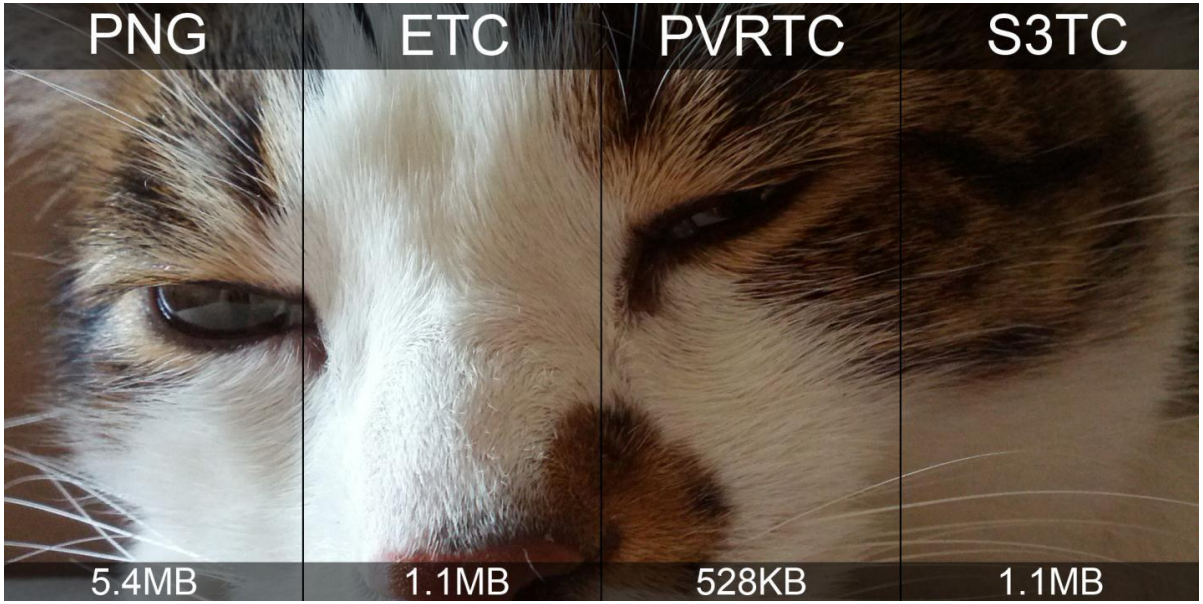
- Gif images
- 256-color palette





OpenGL image formats

- Textures are unpacked to BMP-byte array
- Alternative, compressed textures.
- ETC,PVRTC, S3TC

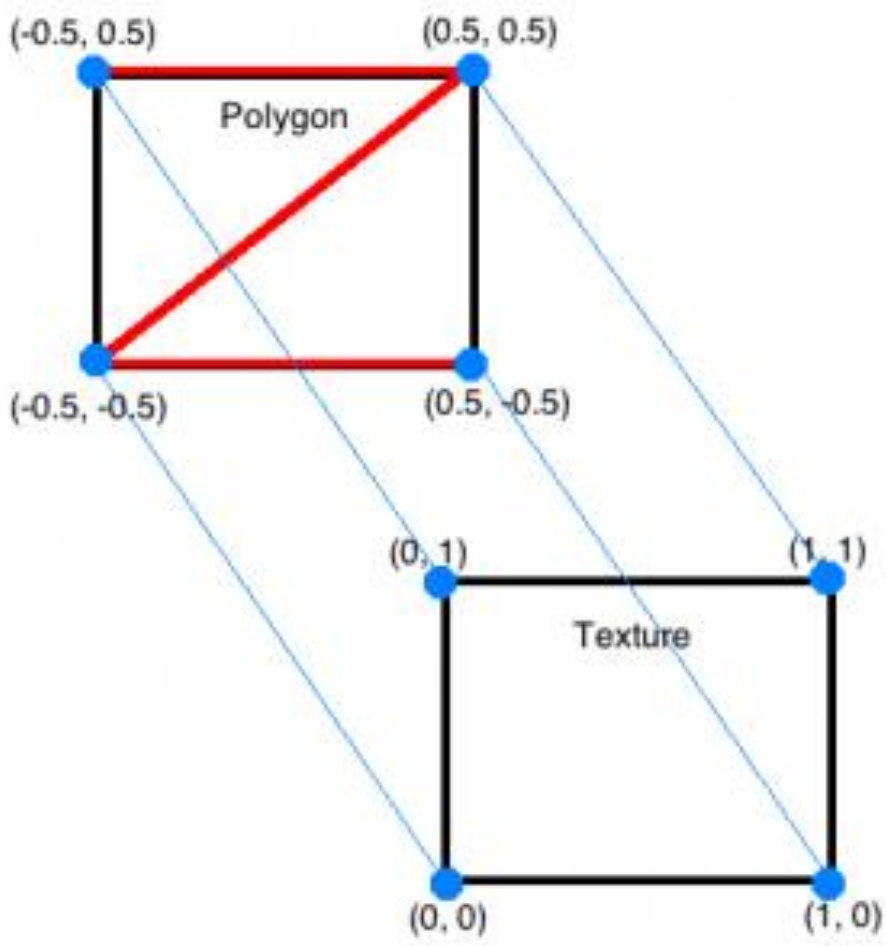


Compressed Textures

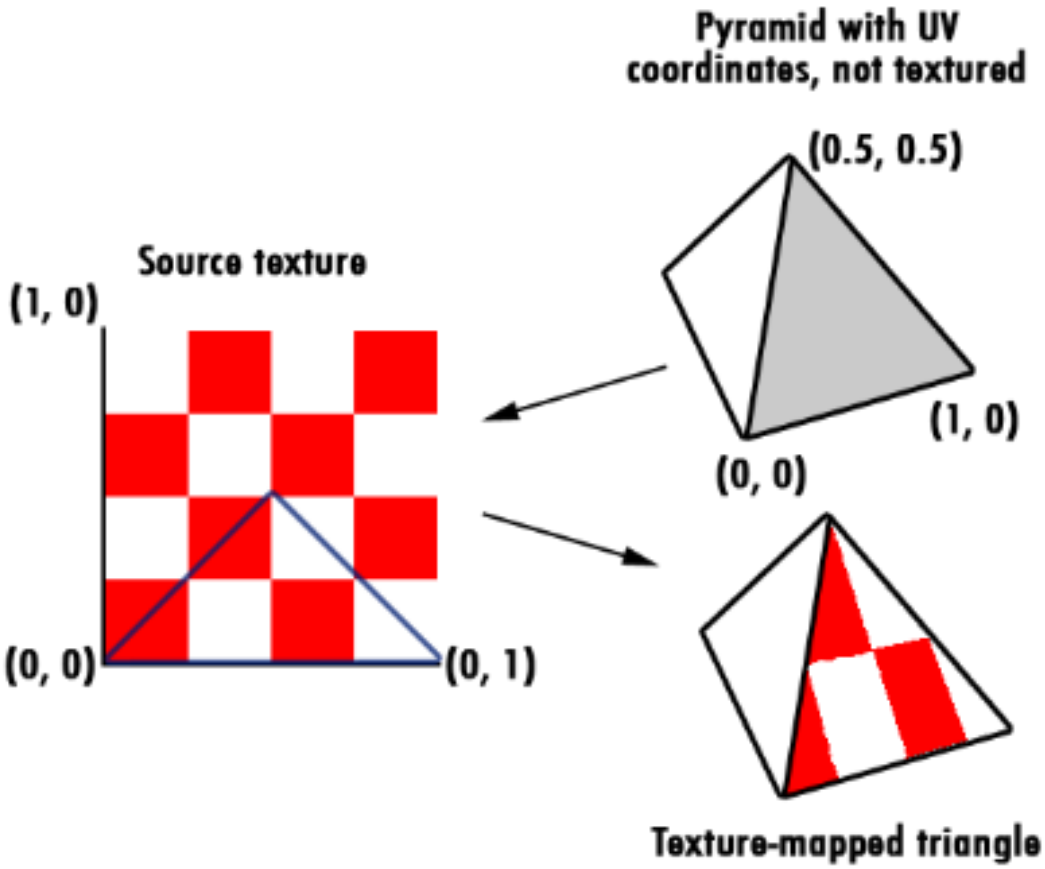
- ETC - http://en.wikipedia.org/wiki/Ericsson_Texture_Compression
- S3TC - http://en.wikipedia.org/wiki/S3_Texture_Compression
- 3Dc - http://en.wikipedia.org/wiki/3Dc (for normal maps)

- http://www.opengl.org/wiki/Image_Format#Compressed_formats

Texture Coordinates

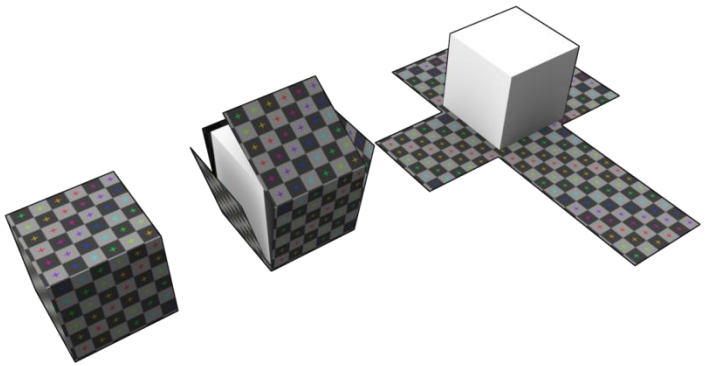


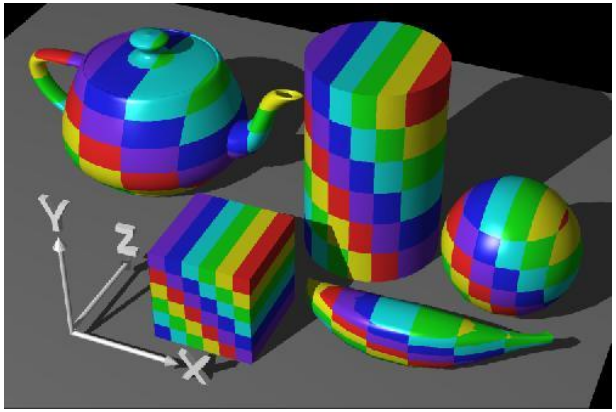
Texture Coordinates

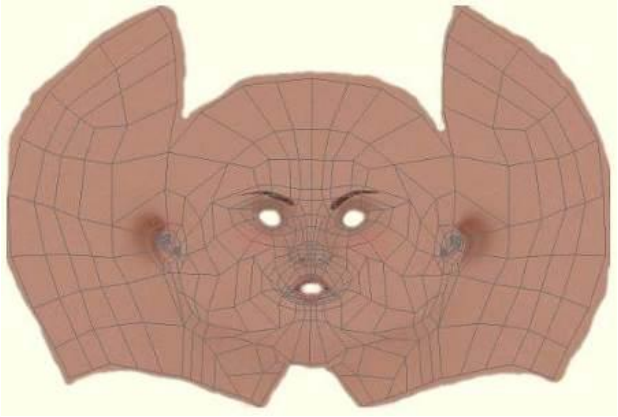


More Complex Texture Coordinates

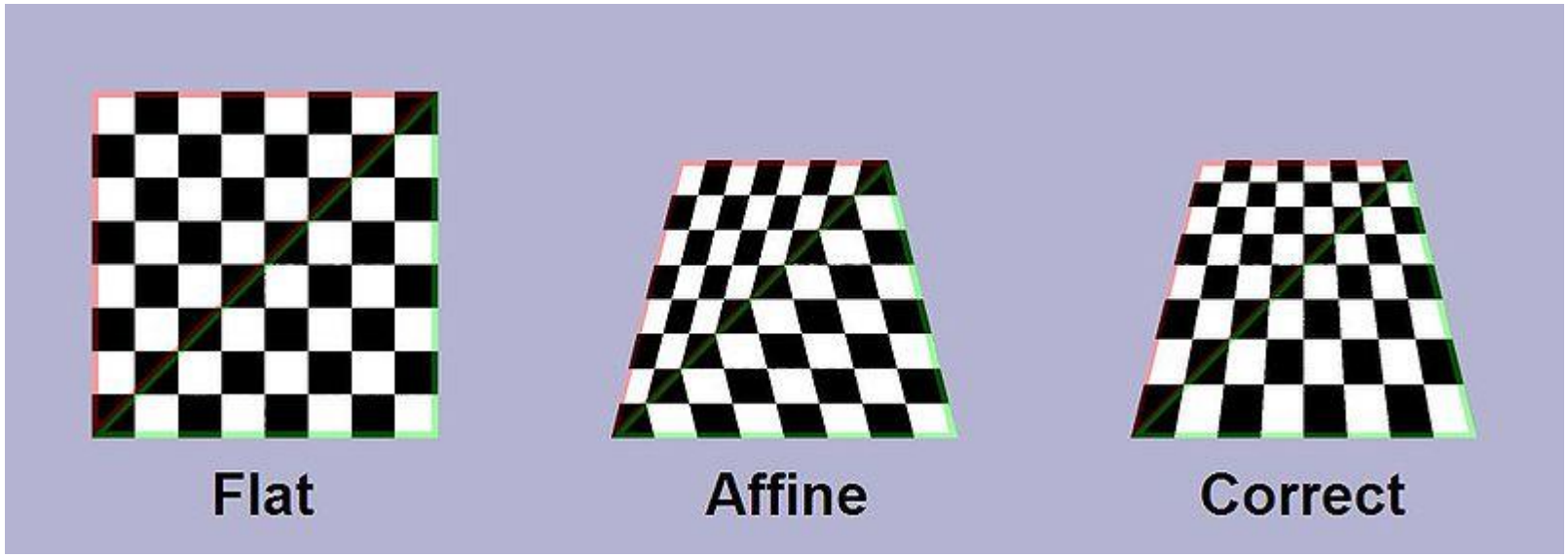
UVW mapping, going from 2D to 3D.







Texture coords interpolation



Affine texture mapping directly interpolates a texture coordinate u_α between two endpoints u_0 and u_1:

$$u_\alpha = (1-\alpha)u_0 + \alpha u_1 \text{ where } 0 \leq \alpha \leq 1$$

Perspective correct mapping interpolates after dividing by depth z, then uses its interpolated reciprocal to recover the correct coordinate:

$$u_\alpha = \frac{(1-\alpha)\frac{u_0}{z_0} + \alpha\frac{u_1}{z_1}}{(1-\alpha)\frac{1}{z_0} + \alpha\frac{1}{z_1}}$$

All modern 3D graphics hardware implements perspective correct texturing.

http://en.wikipedia.org/wiki/Texture_mapping

Drawing triangle with texture coordinates.

```
FloatBuffer vBuffer = Buffers.newDirectFloatBuffer(9);
vBuffer.put(-0.5f).put(-0.5f).put(0.0f);
vBuffer.put(+0.5f).put(-0.5f).put(0.0f);
vBuffer.put(+0.5f).put(+0.5f).put(0.0f);
vBuffer.flip();

FloatBuffer tBuffer = Buffers.newDirectFloatBuffer(6);
tBuffer.put(0.0f).put(1.0f);
tBuffer.put(1.0f).put(1.0f);
tBuffer.put(1.0f).put(0.0f);
tBuffer.flip();

gl.glEnableClientState(GL2.GL_VERTEX_ARRAY);
gl.glEnableClientState(GL2.GL_TEXTURE_COORD_ARRAY);

gl.glTexCoordPointer(2, GL2.GL_FLOAT, 8, tBuffer);
gl.glVertexPointer(3, GL2.GL_FLOAT, 12, vBuffer);
gl.glDrawArrays(GL2.GL_TRIANGLES, 0, 3);

gl.glDisableClientState(GL2.GL_TEXTURE_COORD_ARRAY);
gl.glDisableClientState(GL2.GL_VERTEX_ARRAY);
```

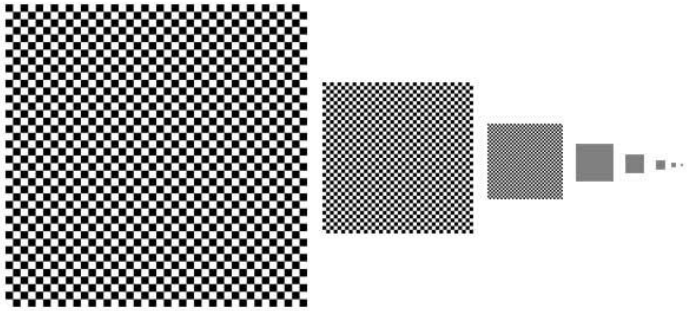
Loading texture Jogl way

```
BufferedImage img = null;
   try {
      img = ImageIO.read(new File("textur.jpg"));
   } catch (IOException e) {
   }

   Texture texture = AWTTextureIO.newTexture(drawable.getGLProfile(), img, true);
   texture.enable(gl);
   texture.bind(gl);
   gl.glTexParameterf(GL.GL_TEXTURE_2D, GL.GL_TEXTURE_MIN_FILTER,
GL.GL_LINEAR_MIPMAP_LINEAR);
   gl.glTexParameterf(GL.GL_TEXTURE_2D, GL.GL_TEXTURE_MAG_FILTER,
GL.GL_NEAREST);
```

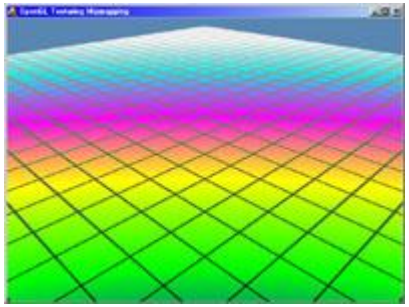
Loading bytebuffer directely to OpenGL

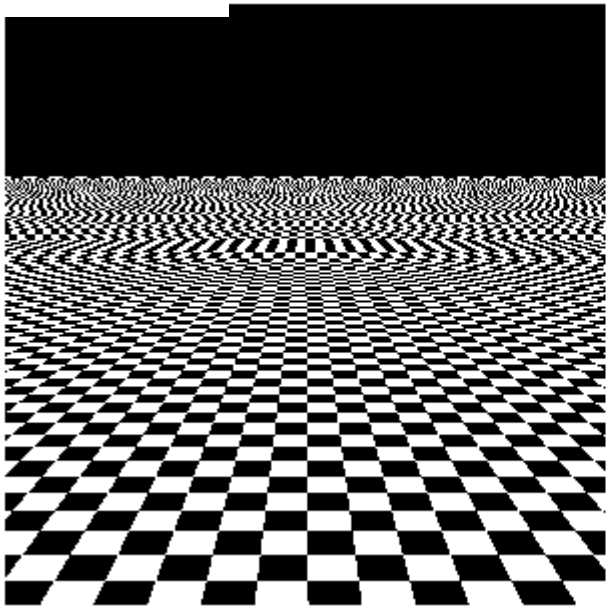
```
IntBuffer textureIDList = Buffers.newDirectIntBuffer(1);
   gl.glGenTextures(1, textureIDList);
   gl.glBindTexture(GL.GL_TEXTURE_2D,
textureIDList.get(0));

//Load image in some way

gl.glTexImage2D(GL.GL_TEXTURE_2D, 0, GL.GL_RGBA,
img.getWidth(), img.getHeight(), 0, GL.GL_RGBA,
GL.GL_UNSIGNED_BYTE, imageBytearray);
```

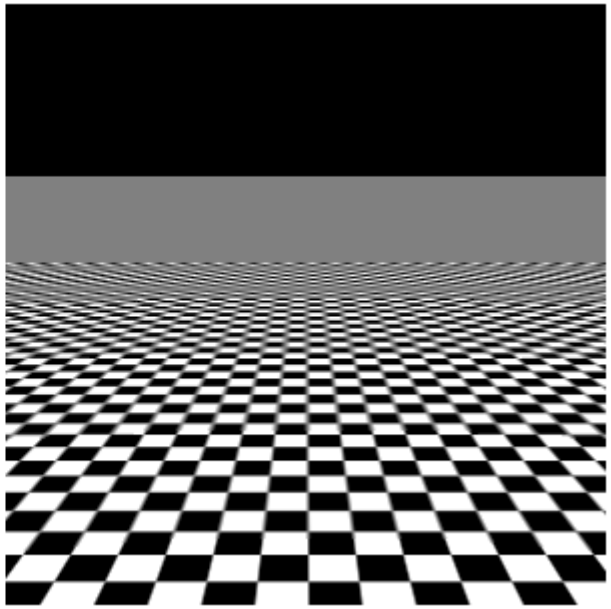
Mipmapping



Store precomputed images of half the size until
Reaching 1x1 pixel. Memoryincrease only 33%.







Texture filter combination

Filter Combination (MAG_FILTER/MIN_FILTER)	Bilinear Filtering (Near)	Bilinear Filtering (Far)	Mipmapping
GL_NEAREST / GL_NEAREST_MIPMAP_NEAREST	Off	Off	Standard
GL_NEAREST / GL_LINEAR_MIPMAP_NEAREST	Off	On	Standard
GL_NEAREST / GL_NEAREST_MIPMAP_LINEAR	Off	Off	Use trilinear filtering
GL_NEAREST / GL_LINEAR_MIPMAP_LINEAR	Off	On	Use trilinear filtering
GL_NEAREST / GL_NEAREST	Off	Off	None
GL_NEAREST / GL_LINEAR	Off	On	None
GL_LINEAR / GL_NEAREST_MIPMAP_NEAREST	On	Off	Standard
GL_LINEAR / GL_LINEAR_MIPMAP_NEAREST	On	On	Standard
GL_LINEAR / GL_NEAREST_MIPMAP_LINEAR	On	Off	Use trilinear filtering
GL_LINEAR / GL_LINEAR_MIPMAP_LINEAR	On	On	Use trilinear filtering
GL_LINEAR / GL_NEAREST	On	Off	None
GL_LINEAR / GL_LINEAR	On	On	None

http://gregs-blog.com/2008/01/17/opengl-texture-filter-parameters-explained/