# Elder Scrolls TODO:

1. Using obliv spiderdaedra I see skin failing to map quite badly, I presume that bones fail too in a different bug (tight shoulder thing). This skin failure is also shown in the skyrim cock ups
2. JInternalFramse for DisplayDialog to allow them to show on a full screen without jumping back or being lost on mac. The escape button exit should be on screen hud not popup because fullscreen and it don’t play nice
3. The physics synchronized calls now make the KCC pause on load of new land!
4. I should use Stack alloc in any performant areas??
5. Need to remove all prefs usage for PropertyLoader, like Dune
6. The config loader, properties loader and prefs loader system should be bought together into a mega loader, with order of load, and command line loader
7. When a card gives back 16bit depth buffer (ever?) must reduce front and back clip
8. Distant textures appear to have black back in them again? In particular Fallout, notice yellow lines on road badly minify, and distant windmill by megaton front. Dds viewer does not show, must be appearance prob? But nifskope does show it too a bit
9. The gross lods are lit up more than the close ones? Check material light values for land and lods
10. Improve distant water, make lod have a water plane, and chop that plane
11. Sounds in fallout produce errors, J3dSOUN and nifcharacter
12. I’m wondering if bsa has any memory overhead, try explorer with raw files
13. http://www.minecraftforum.net/topic/120261-131-glsl-shaders-dof-bump-mapping-waving-wheat-dynamic-shadows-and-more/
14. <http://www.jotschi.de/Uncategorized/2011/09/28/jogl-2-glsl-example.html>
15. <http://hub.jmonkeyengine.org/wiki/doku.php/sdk:plugin:shaderblow>
16. <http://en.wikipedia.org/wiki/HLSL2GLSL>
17. Varying lod fader needs knots/frames system because close things don’t need check often either, in fact lod fader is a fixed 5 frames so not easy to correct
18. Interesting code http://docs.oracle.com/javase/tutorial/extra/fullscreen/example.html

## Rift:

1. <https://github.com/38leinaD/JRift>
2. Still haven’t got GLSL FBO shader working, must use the offscreen code but make a front buffer and shade and flip like web examples do
3. NEW VERSION lotsof rework required

## Mac use:

1. I found how to force antialiasing on screen: quite strangely you have to set *j3d.implicitAntialiasing* property to true.   
   I don't fully understand why this property has to be set if you already called GraphicsConfigTemplate3D.setSceneAntialiasing(GraphicsConfigTemplate3D.PREFERRED); but it works on my computer. Miserably it has no effect for offscreen rendering.
2. <https://developer.apple.com/library/mac/qa/qa1170/_index.html>

## Could I create a binary shader?

public abstract class **Shader**

extends [NodeComponent](http://download.java.net/media/java3d/javadoc/1.4.0/javax/media/j3d/NodeComponent.html)

The Shader object is the abstract base class for programmable shader code. Currently, only text-based source code shaders are supported, so the only subclass of Shader is SourceCodeShader. We leave open the possibility for binary (object code) shaders in the future.

## I need to:

Connect sourceforge to github

Write a note on what each jar file does why it’s there

In the note find a link to the original jar file or web site if possible

Make up a list of attributions to people, and try to find licensing

Do I care about licensing in my java files? Or is that boring

# Launcher:

Also launcher and setting go hand in hand, launcher sets setting before launch, note that setting and the menu screen esc are related, and Mac requires a menu screen with exit in the Pane3D world as it won’t switch resolution

I had more ideas, if we eschew the launcher parsing bat file, we still need a way to get reliable console output on screen, possibly the launcher could have a tick box to stay resident and show console outs on a scrollable panel in a second tab? Then all output can go to a single log file nicely unless you are debugging proper.

Launcher could then include a url to get latest from and unzip over the current jar, this probably means moving the launcher into a separate jar file to not over ride itself.

So how does the launcher get updated then? I really want the launcher to get the new files down unzip everything and then somehow rename the jar and relaunch itself?

A launcher in tools for general command line script and launcher in tools3d to adf display dialog, which needs to flip over to jpanel version too, then tools3d can include jogl version option, which probably drops the bat file parse option, but given noddraw etc. that’s probably fine. If launcher cant launch put debug launcher bat file

Launcher should hand –logout to app then app knows to send all sops out to a particular file, that way boot strap can exit and doesn’t hang around.

Tools3d launcher can extends tools launch and include standard lib file of java3d and jogl and ddraw=no etc. then app can send through its own lib path files and options, each app will still have a bootstrap class invoked form the meta info but it will be cut down, but the start server code can still be invoked from it.

The boot strap then calls the launcher which then calls the main app. So launcher is the main of the main app, but then calls a separate thread???

So is bootstrap and launcher the exact something?

I have 3 types of launch

Development:

No updater required, no bootstrap required, but display resolution including full screen and antialiasing I also want to be able to optionally go jogl2 but I have no class path without boot strap, though all jars is in fact fine isn’t it?

I also possibly just want setting to go so the display stuff could be property loader - ed and recalled for straight boot up, however what’s happened on a setting change? Restart is classic

Boot strap:

Jar file only, must spawn process, want to end this process, so best to hand a log file across and get the main to pump out to log.

Display selection does not require anything but core java, so bootstrap can use it too

However boot strap needs to be a separate jar form the main which means that the selections form boot strap need to go across to the main app, but that’s just the config ini file anyway.

Boot strap also wants to update the game from a url, unzip including replace the main jar, hence it’s on a separate jar.

Command Line:

Finally I need command line gear so I can test stuff, but that’s just dev anyway.

For mac shells

http://mathiasbynens.be/notes/shell-script-mac-apps

I need to disable all extensions to the jre if I’m going to use the system JRE so here:

The first solution you could choose would be to install the files in the extensions folder of the JVM (JAVA\_HOME\lib\ext). The problem with this approach is that your application will only work until a new version of Java is installed.

So, the best approach is to install the files in your own private folder. Then, you must instruct the JVM to search for them in that location. That can be achieved by setting the **java.ext.dirs** property with the full path of that folder. You can set a Java system property in the [Edit JVM Parameters](http://www.advancedinstaller.com/user-guide/edit-jvm-parameters-dialog.html) dialog.

You will have something like: java.ext.dirs = [MyFolder\_DIR]

You will probably also want to use the native extensions installed in the "lib\ext" folder of the JVM. For this you will have to set the java.ext.dirs to a list of directories which includes your private folder but also the default one. In that case you will have to use the environment variables that the Advanced Installer [Java Launcher](http://www.advancedinstaller.com/user-guide/native-java-launcher.html) sets when it loads the application. The environment variable that you should use is **AI\_JVM\_HOME** that is set to the full path of the home dir of the loaded JVM.

So the final result will be: java.ext.dirs = %AI\_JVM\_HOME%\lib\ext;[MyFolder\_DIR]