TGVE – **T**0neg0d **G**UI **V**isual **E**ditor
last updated: 06/21/14

The purpose of this document is merely to lay out the design of this module and some key points that it should have as far as functionality. So, in a nutshell, it's going to be designed to behave similarly as the swing gui editor that comes with a jMonkeyEngine standalone installation. (The swing gui editor was formerly called project Matisse for the NetBeans environment.) This is so that the learning curve for using this module won't be as steep as if we had to trailblaze a new work flow specific to the task; a lot of that has already been accomplished by the awesome crew behind the swing GUI editor.

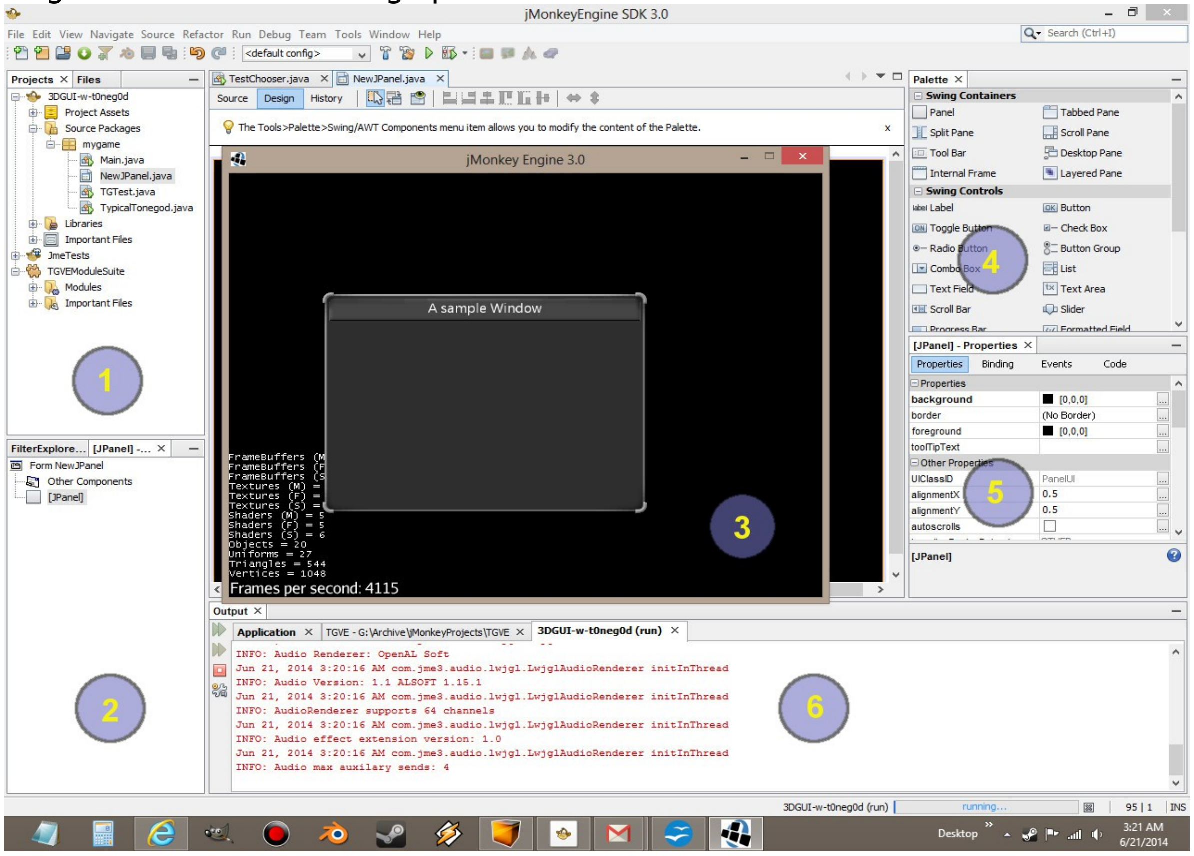
Oh yeah, and this tool is for making GUI/2D interfaces for video games.

So, lets copy the behaviors and program this for this type of scenario/work flow sample:
End-user has created their jMonkeyEngine Application/SimpleApplication project, and they are ready to make a new GUI.

[new file wizard]
End-User names the GUI, specifies folders for assets and code (default for code: %projectDir/%srcDir/%guiName and default for graphics and sounds %assets/gui/%guiName), sets the base resolution for rendering, and selects the mode of how this GUI should operate: as a standard GUI (Overlay for the screen), or targeting an offscreen texture for usage on a geometry.

(new file wizard)
The module responds by creating the appropriate template files in the folders with copies of samples to overwrite in the assets folder. It then opens up the visual editor and is ready for the user to start creation.
<considerations>
the TGVE behavior will be attached as a MultiView class to the extension of .tgve a text file with a serializedtoXML scheme, for reading/writing of the visual parts of the editor. It will be paired to a .java file with the same name. The .java/.tgve pair will cause the .tgve file to not be displayed in the project/files explorer of jMonkeyEngine. The .java file will have sections of code "colored" indicating that if they try to write in these sections of code... it's gonna be lost when they go back to the Visual design side. (I don't have enough information/experience in having the visual side respond to changes in code to make it a two way street so let's design it to just flow from the visual editor to the java code)
unpaired situations:
As a side note, once the design/code snippet setting/controls and sounds have been attached, they should be able to delete the .tgve file as a way to make the .java file behave as a normal file. If a .tgve file is renamed, the module should ask if they would like to create a new .java file with this name. If a blank .tgve file is created, the module should fire up the wizard to initialize it with a similarly named .java file (name/location of the desired src is known, just have to set the other stuff).

[MultiView Environment]
The End-user uses a bunch of tools to design the GUI … I say it like this because, honestly, we just have to make sure our tools work correctly and have lots of hints pop up so that they actually use them like we're hoping. LOL, but let's have a look at a pretty picture to give the general gist of what we're setting up.



```
FrameBuffers (M
FrameBuffers (F
FrameBuffers (S
Textures (M) =
Textures (F) =
Textures (S) =
Shaders (M) = 5
Shaders (F) = 5
Shaders (S) = 6
Objects = 20
Uniforms = 27
Triangles = 544
Vertices = 1048
Frames per second: 4115
```

```
INFO: Audio Renderer: OpenAL Soft
Jun 21, 2014 3:20:16 AM com.jme3.audio.lwjgl.LwjglAudioRenderer initInThread
INFO: Audio Version: 1.1 ALSOFT 1.15.1
Jun 21, 2014 3:20:16 AM com.jme3.audio.lwjgl.LwjglAudioRenderer initInThread
INFO: AudioRenderer supports 64 channels
Jun 21, 2014 3:20:16 AM com.jme3.audio.lwjgl.LwjglAudioRenderer initInThread
INFO: Audio effect extension version: 1.0
Jun 21, 2014 3:20:16 AM com.jme3.audio.lwjgl.LwjglAudioRenderer initInThread
INFO: Audio max auxilary sends: 4
```

(MultiView Environment)

1. the explorer window – we won't really need to worry too much about this one, with the exception that the module will need to remove the .tgve node of a paired .tgve/.java set, much like is done for .form.
2. the navigator window – a tree-view parent/child relationship view of the GUI model, code snippets and bindings to other external data objects will show up here. Mainly cause they aren't gonna show up in local graphics.
3. The editor window – lol, bad mock up time. But this will be the awt handling jME canvas that displays the design area in a shadeless format. Debug lines and boxes will be added to the elements for resizing and positioning. The functionality of this area will be further detailed later on, but there will be two design views, the first is a relative design area with a ball control at the #3 position which can be rotated to drop into a perspective based orbit camera mode so that the z-positioning of elements can be affected. The second is an orthogonal only "pixel perfect" mode based on the base resolution with magnification friendly controls and a scroll box area for panning around. There will be a "control attach" button on the tool bar for attaching code snippets to an object, and a preview button that lets them view their GUI in different resolutions and if they are doing it as a rendering for attaching to geometries, some preset typical scenes like, say hologram panels or helmet displays...

4. The palette window – predefined elements that can be dragged and dropped onto the editor window to place them into the GUI design and attach them based on the "drop point" position of the mouse.
5. The properties window – shows the list of alterable properties for the currently selected item. This will be based off of reading the currently installed t0neg0d.JAR and parsing the public variables, get/set methods, and annotations from the .class files. That way, updates shouldn't throw things off too badly.
6. The output window – meh, it's textual output, but we could probably throw a dopesheet timeline window for animations down here, and if we do it right, they could probably reuse it for core on something.

[attaching to the rest of the project]
as part of the template a few lines of commented code are put up for a copy/paste to their Application/SimpleApplication .java code to instantiate an instance of this GUI module. (this would be the suggestion for starting monkeys) Also I would recommend heavy commenting be generated along with the code, for both ourselves and the end-users, as just one of those end-user friendly things (the focus being "cheat-sheet" reminders of the conventions that @norment is trying to standardize, LOL)

erg, that's the rough draft for now. Oh, and probably the ol' gpl v2 for the license, cause honestly, I'm doing this to use for myself, random other people whose games I'll probably wanna play, and to document how to do this to increase the number of people who can use the techniques to make more cool and fun tools to actually integrate into jMonkeyEngine.

Charles Anderson/relic724
Project: Iteag