Paul Kratt
2/14/2008
CS421
Final Project Report

Executive summary

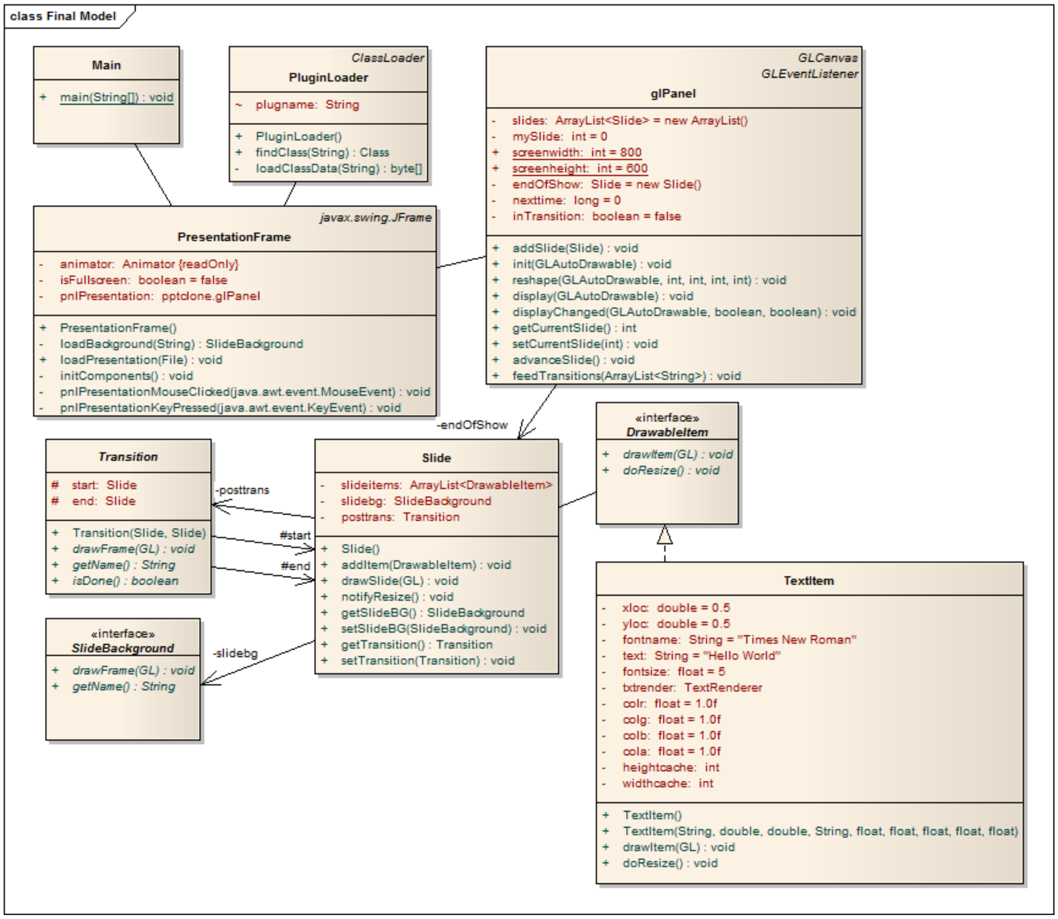
The purpose of this project was to develop an application that utilizes 3D graphics, meaningful user interaction, and applies an advanced graphic concept discussed during the course. A 3D-accelerated presentation tool was developed, which desired to maintain audience interest in presentations with visual splendor.

Background

This program was written out of a desire for an affordable method of creating presentations with dynamic backgrounds. Several plug-ins for PowerPoint exist that are capable of this, but they cost upwards of $100. Since a 3D Presentation tool fell within the scope of the project requirements, it was chosen for implementation. Java was used as the language of implementation, with the JOGL library for 3D graphics.

Design

The very first project deliverable developed was a UML design. Although some functions were added to the design later on as well a PluginLoader class, the overall class structure did not change during the development of the project. The final UML diagram is presented below.



class Final Model
Main
+ main(String[]) : void
ClassLoader
PluginLoader
~ plugname: String
+ PluginLoader()
+ findClass(String) : Class
- loadClassData(String) : byte[]
GLCanvas
GLEventListener
glPanel
- slides: ArrayList<Slide> = new ArrayList()
- mySlide: int = 0
+ screenwidth: int = 800
+ screenheight: int = 600
- endOfShow: Slide = new Slide()
- nexttime: long = 0
- inTransition: boolean = false
+ addSlide(Slide) : void
+ init(GLAutoDrawable) : void
+ reshape(GLAutoDrawable, int, int, int, int) : void
+ display(GLAutoDrawable) : void
+ displayChanged(GLAutoDrawable, boolean, boolean) : void
+ getCurrentSlide() : int
+ setCurrentSlide(int) : void
+ advanceSlide() : void
+ feedTransitions(ArrayList<String>) : void
javax.swing.JFrame
PresentationFrame
- animator: Animator {readOnly}
- isFullscreen: boolean = false
- pnlPresentation: pptclone.glPanel
+ PresentationFrame()
- loadBackground(String) : SlideBackground
+ loadPresentation(File) : void
- initComponents() : void
- pnlPresentationMouseClicked(java.awt.event.MouseEvent) : void
- pnlPresentationKeyPressed(java.awt.event.KeyEvent) : void
-endOfShow
«interface»
DrawableItem
+ drawItem(GL) : void
+ doResize() : void
Transition
# start: Slide
# end: Slide
+ Transition(Slide, Slide)
+ drawFrame(GL) : void
+ getName() : String
+ isDone() : boolean
-posttrans
#start
#end
Slide
- slideitems: ArrayList<DrawableItem>
- slidebg: SlideBackground
- posttrans: Transition
+ Slide()
+ addItem(DrawableItem) : void
+ drawSlide(GL) : void
+ notifyResize() : void
+ getSlideBG() : SlideBackground
+ setSlideBG(SlideBackground) : void
+ getTransition() : Transition
+ setTransition(Transition) : void
«interface»
SlideBackground
+ drawFrame(GL) : void
+ getName() : String
-slidebg
TextItem
- xloc: double = 0.5
- yloc: double = 0.5
- fontname: String = "Times New Roman"
- text: String = "Hello World"
- fontsize: float = 5
- txtrender: TextRenderer
- colr: float = 1.0f
- colg: float = 1.0f
- colb: float = 1.0f
- cola: float = 1.0f
- heightcache: int
- widthcache: int
+ TextItem()
+ TextItem(String, double, double, String, float, float, float, float, float)
+ drawItem(GL) : void
+ doResize() : void

The PresentationFrame class houses the main pieces of the program. It is responsible for allowing the user to load a presentation, and creating instances of Slides. The java DOM classes are used to read a presentation from XML data and populate the slides. Each slide consists of a background, an array of DrawableItems, and a transition to the next slide. All backgrounds are classes that implement the SlideBackground interface. Transitions extend an abstract transition class, and have access to before and after slides. However, transitions are responsible for all of the drawing that goes on while they run. Therefore if graphical data from either slide is to be used in a transition, the transition itself is responsible for invoking the drawing of the slides. The final component of presentations exists as the DrawableItem interface. All slides can contain any amount of DrawableItems, limited only by the system resources. The only DrawableItem in this version of the application is the TextItem, which is used to draw text to the screen.

On the technical side of things, text is rendered to the screen using the TextRenderer class that is part of JOGL. Most of the OpenGL setup relies on the default settings and camera location. If any slide element such as a background needs a change, it is up to the background to change those settings and restore them later to prevent adverse effects to other slides. Both slide transitions present in the current version of the program utilize the OpenGL display capture features to capture the current slides and perform animations with them.

User documentation

After building the application, copy the class files for all backgrounds to a folder called backgrounds in the same location as the compiled JAR. This is the location that the application will search for background plug-ins. Create XML presentation data by following the example below:

```
<?xml version="1.0" encoding="UTF-8"?>
<presentation>

<slide background="square" transition="rotate">
<text x="0.5" y="0.5" font="Arial" size="5" r="0.5" g="1.0" b="0.5"
a="1.0">Hello World</text>
<text x="0.2" y="0.6" font="Arial" size="5" r="0.5" g="1.0" b="0.5"
a="1.0">All your base are belong to us!</text>
</slide>

<slide background="none" transition="none">
<text x="0.2" y="0.5" font="Arial" size="5" r="0.9" g="1.0" b="0.9"
a="1.0">This is a sample</text>
</slide>

</presentation>
```

The following slide backgrounds are available:

- Square – A rotating gradient background of multiple colors
- Stars – Similar to the
- Floating – A rows and rows of endless washers move towards the background and disappear in the distance.
- Helixblur – A rotating touroid is surrounded by a radial blur effect. (Port of an example by Pepijn Van Eeckhoudt)
- Zoom – A circle of a random color fills the screen, “paints” the background, and shrinks back down.
- None – A black screen

The following slide transitions are available:

- Rotate – As if you were inside of a cylinder, the screen camera rotates to the right and settles on the next slide.
- Genie – The current slide is sucked towards the bottom of the screen, while the next slide zooms in from below to take its place.
- None – Go directly to the next slide. Do not pass go, do not collect $200.

After launching the program, the user is prompted with a file open box. Select the XML presentation file to load. The presentation is loaded into a resizable window by default. Click the mouse to advance slides. Use the left and right arrow keys on the keyboard to navigate forwards or backwards between slides. Pressing the "F" key on the keyboard will switch the presentation into full-screen mode on the last connected display. (In a single monitor system, the display will appear on the primary screen. On multiple monitor systems, the monitor with the highest identification value will be used.) If keyboard controls do not respond, ensure that the window has focus.

Schedule Analysis

My original proposed schedule is below:

Time period	Milestone	Product
End of Week 7 (Sunday 27th)	Rough design complete. Input file format defined.	Design documents.
Wednesday Week 8	Mid-project presentation. Basic slides with text display working. Use program to do presentation.	The presentation
End of Week 8	Plans for 4 abstract animated backgrounds using OpenGL effects, and 2 slide transitions.	Descriptions of the planned backgrounds and transitions.
End of Week 9	Slide transitions implemented. At least 2 backgrounds working.	---
Wednesday Week 10	Final presentation. All 4 backgrounds should be working.	---
Thursday Week 10	Final report completed.	Final report document, complete source code.

Through Wednesday of week 8, I was ahead of schedule. I had text display working, and one slide background already completed. However, later that week I fell behind. I never had any additional time to work on the project during the remained of that week, and my initial background ideas were not created until week 9. During the weekend after week 9, I made caught up and completed another background and the slide transitions. By Tuesday evening of week 10, all planned functionality was completed. I spent some additional time Wednesday morning and added a full-screen mode to the application. The final deliverable, this report, was not completed until Friday of Week 10. Overall, I was able to stay within the plan by a few days so I believe that it was a reasonable plan.

Conclusion

The goal of this project was to develop a 3D presentation tool, and that was accomplished successfully. No major problems were encountered during this time.