**PyPlanetarium Design**

“oh my God! – *it’s full of stars!*”

– Dave Bowman

Arthur C. Clarke, *2001: A Space Odyssey*

I decided to create a planetarium/starmap app by taking real star data based on a location and date/time that the user sets, and drawing the stars in an interactive environment that the user can explore as they wish. I wanted to make this somewhat educational as well, so the user can also opt to take a simple “quiz” in which they attempt to draw the Big Dipper, then the Little Dipper, and can receive hints on what to draw next.

**Code Design**

I used Pygame for all the visual items, and I used a library called PyEphem to do the astronomy calculations and get the stars’ azimuth and altitude. I implemented my own Star class, which stores the name and the PyEphem celestial Body object, and do my calculations for each star within that class. To map the stars from their azimuth/altitude coordinates to a 2D surface, I considered the azimuth/altitude as polar coordinates, converted them to Cartesian coordinates, and stored these coordinates in terms of a “bigger” screen, of which the screen the user sees is a small window. The “bigger” screen is a square, and its side length is always twice the current zoom number.

Other classes I have created include a Line class that stores the two stars it connects, a Hint class for purposes of displaying hints on the screen, and a Button class with many different kinds of subclasses for all of the different buttons I use. These include ModeButtons for switching game modes, DrawButtons for all of the different drawing tools, MenuButtons for long lists that the user selects from, and a NowButton, HintButton, ToggleButton, and TimeButton, each with their own specific uses (clicking on “Now”, the hint button, toggling realtime on/off, and selecting the date and time).

For the actual game class itself I used Lukas’s Pygame framework from his optional lecture. Most of the functions fall under these categories:

* Update/Calculate: updates the data in some way, whether it be to accommodate new info or to calculate new info.
* Draw: draws everything onto the screen.
* Check/MousePressed: what happens when you click the mouse. Usually called “[mode]MousePressed” or “check[Items]” or “check[Mode]”
* Init: initialize functions, called when the class is created
* Load: loads in information from a file. I use these for loading in from star databases (with a PyEphem function), from user save files, or from constellation files included in the package.
* Base: all the event handlers, timerFired, and redrawAll from the framework that make the game actually run.

**User Interface**

Fun fact: Some of the fonts I used include Gill Sans and Eurostile. This is because in 2OO1: A Space Odyssey, the iconic title screen is in Gill Sans, and Eurostile Bold Extended is the typeface for Hal 9000’s interface. (I also used Monospace for all of the game-rendered text because Pygame rendered it better than the other two.)

* Many of the applications I studied from the Internet had click-and-drag and scroll-to-zoom as their main methods of getting around their starmaps, so I implemented a similar tactic on my own app.
* I figured icons would look nice, so I used Google Images to find some nice draw icons.
* I kept the interface fairly basic, as I am not a designer.
* A few of the apps I found (Stellarium, for instance) didn’t explain any of the buttons or what they did whatsoever, so I opted to include a couple of help screens so that people could figure out what they were doing.
* A lot of the icons I saw on other apps didn’t make visual sense, so I just used either text or the most generic stock icons I could find for all of my buttons.
* The “quiz” mode includes a hint button – Kosbie suggested this to me, and some other apps I found had this feature as well.
* Stellarium didn’t have anything educational – so I included an educational bit (also at the suggestion of Kosbie).
* In an effort to keep things simple, I simplified selection of features on the options screen. You need to click on the boxes and use the arrow keys to change the numbers or the city.
* There is an upper-bound and lower-bound zoom limit, and the boundaries of the starmap’s bounding box cannot be crossed by click-and-drag. This is to prevent people getting “lost in space,” as it were.
* For simplicity, save and save image just write over any files in the folder named savedata.txt and screenshot.png, respectively.
* I haven’t seen anything with a screenshot feature built in, so I decided to add that. Save/load seem relatively unutilized features as well.
* During the User-a-thon, someone suggested to me to make drawing lines between stars easier by prioritizing the stars themselves over their labels (as you can click on either the star or its label to start/finish drawing a line). I implemented this feature, and it’s a lot easier to draw lines to the stars you want now.

I hope to continue to add more features even after the project is over!