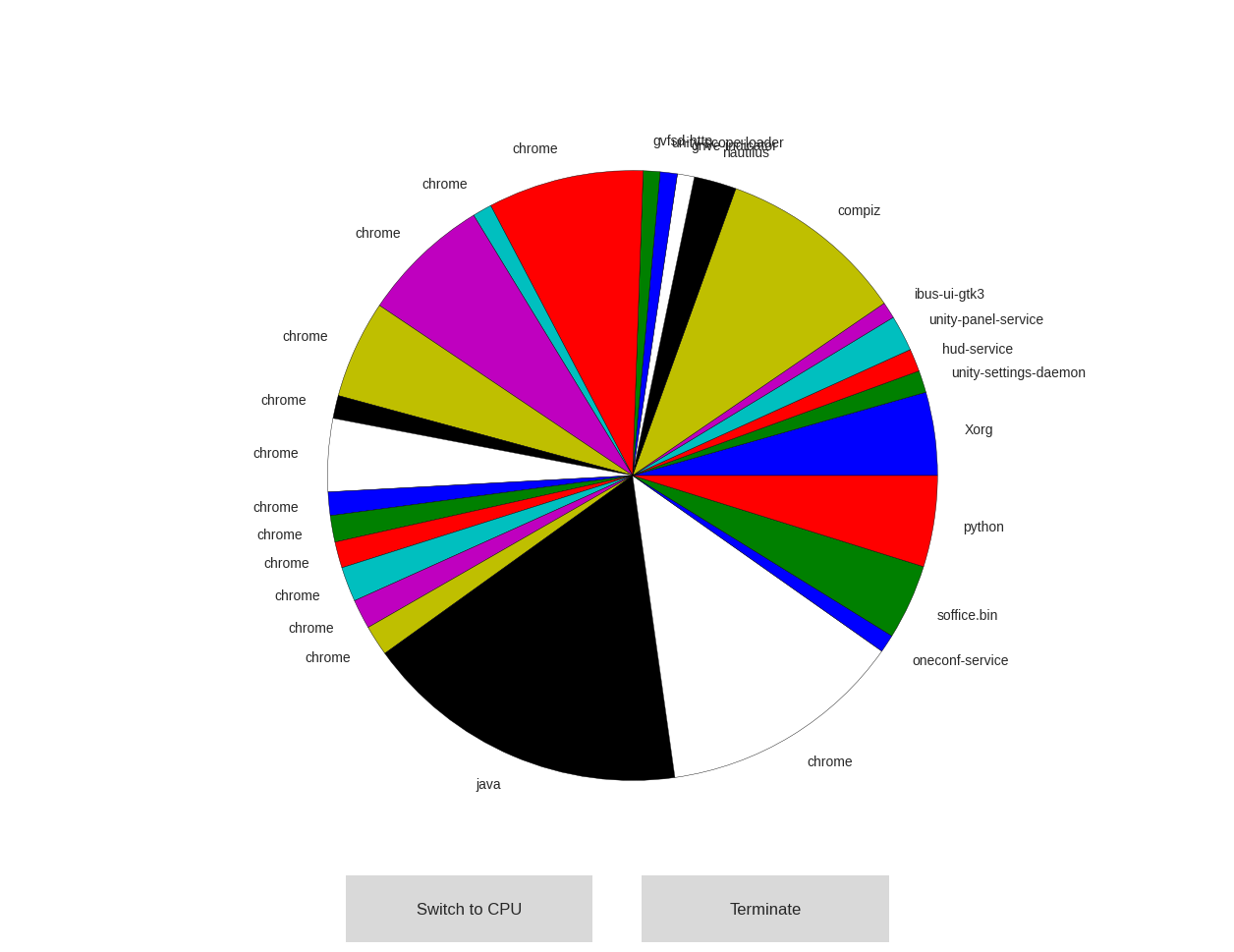
David Abrahams & Tom Heale

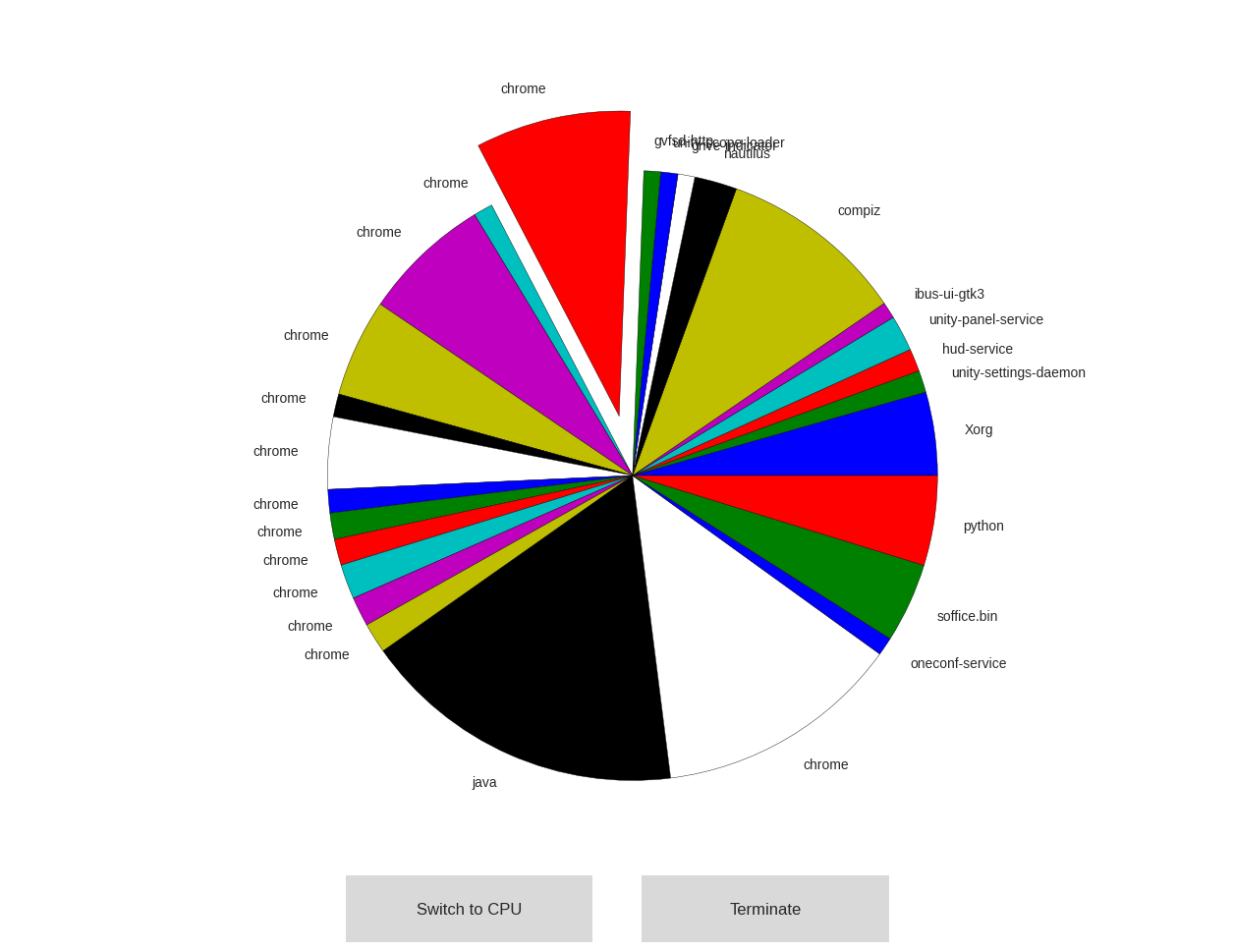
**Project Overview:**

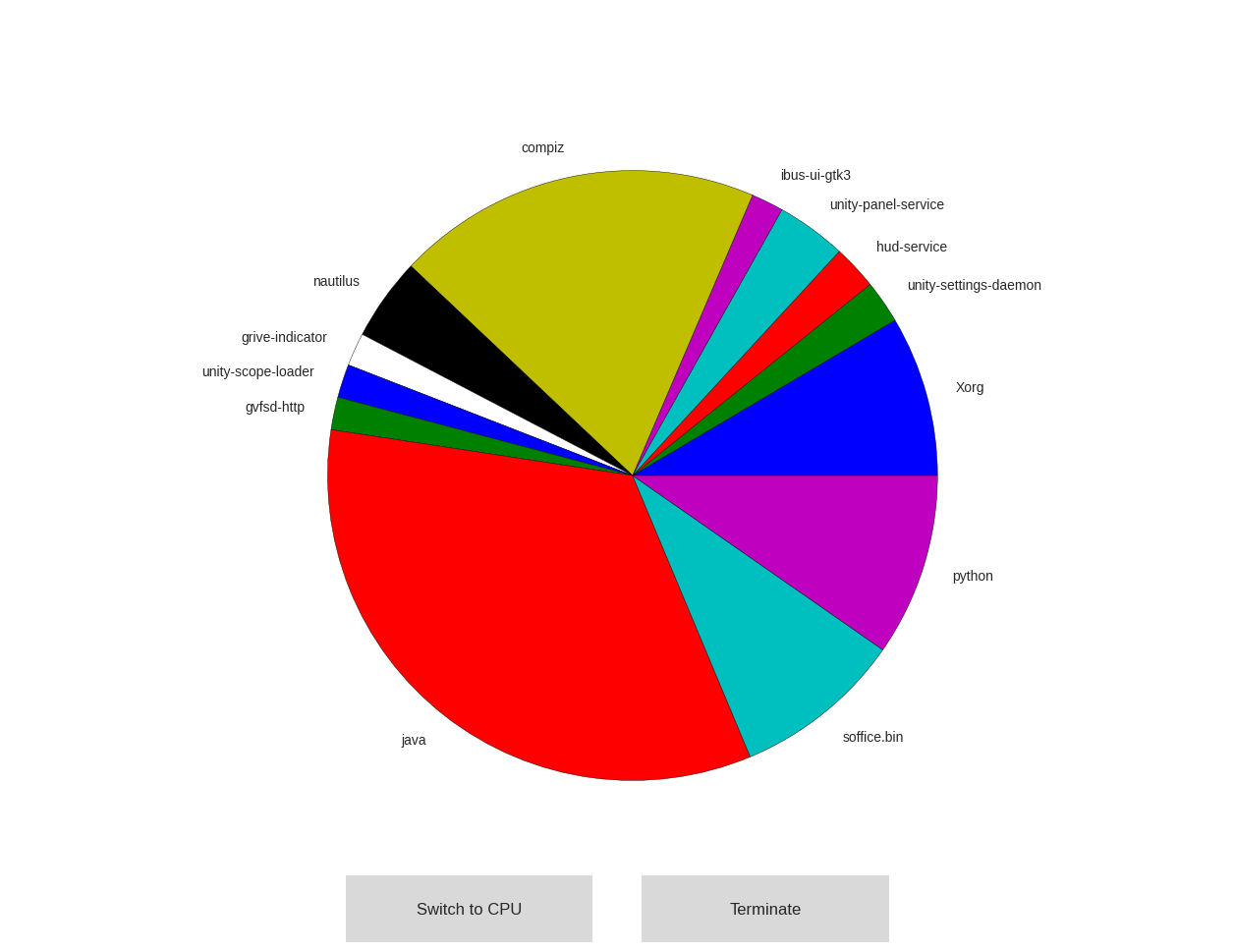
We created a simple visual task manager for Linux in python. The user can view processes' CPU and RAM usage, and terminate selected processes. The task manager updates in real time.

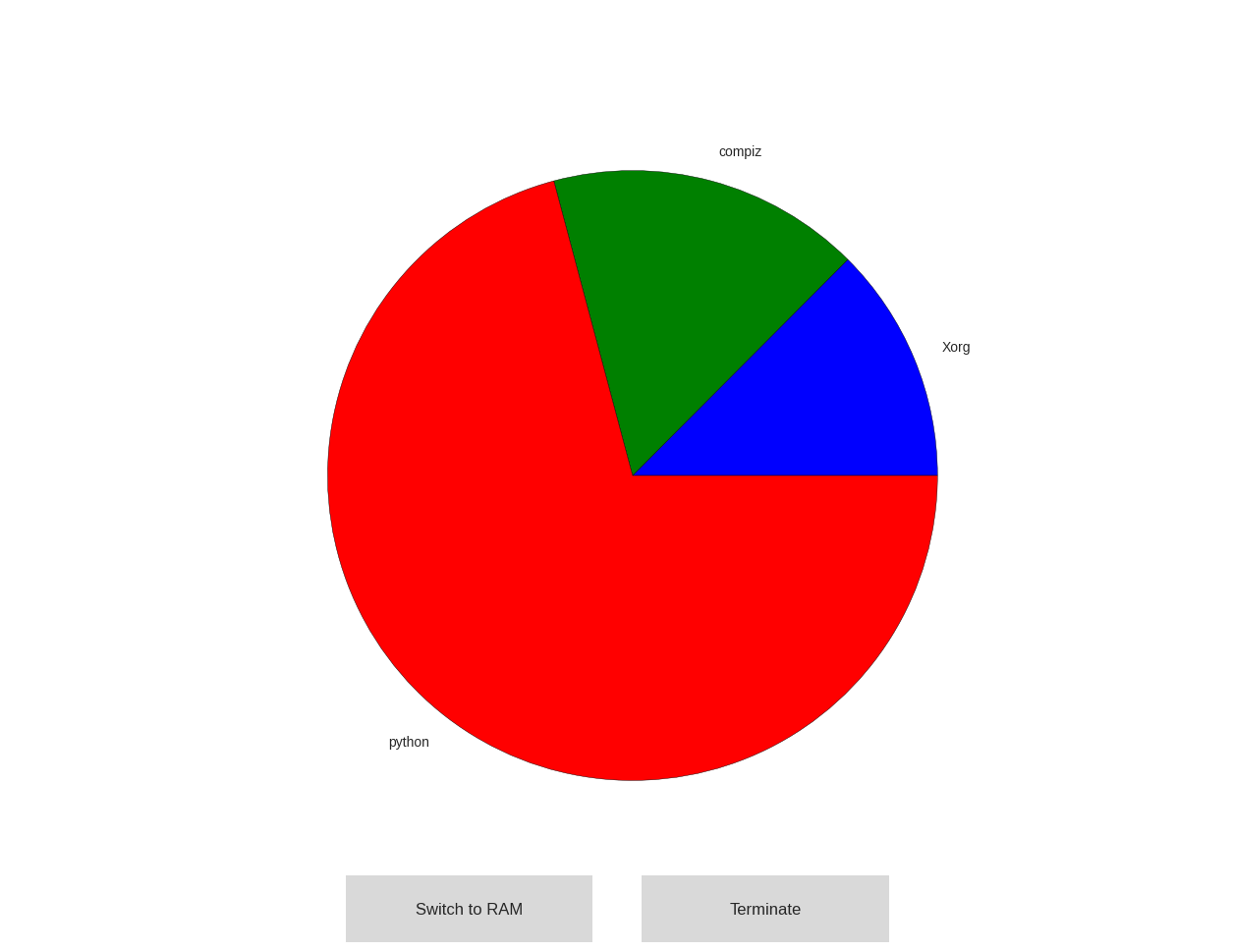
**Results:**

Our program provides similar functionality to Windows Task Manager. A pie chart allows the user to view which processes are the most “heavy” on the computer. Below are all the processes my computer is currently running:

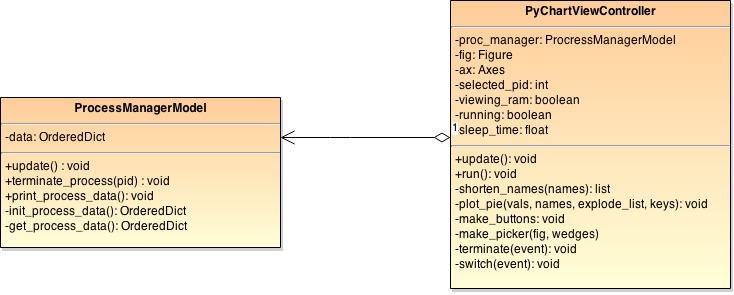
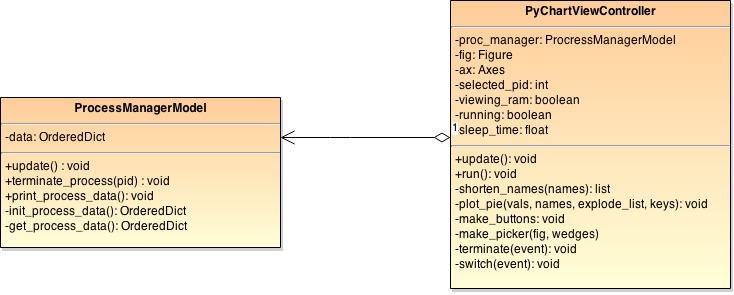
If I wanted to close one of my chrome processes, I would simply click on it in the pie chart, and that process would be selected:

Then, to close the processes, I would click terminate, and the process would end. In this case, closing the selected chrome process caused chrome to crash and end all chrome processes:

The pie chart updates real time, so if you open or close processes while viewing the window, the wedges in the pie chart will change size according to RAM/CPU usage. Finally, the user can click “Switch to CPU” if they would like to view CPU usage. However, this is less useful because the ProcessManager itself is very processor intensive, as it is constantly updating and checking which processes are running. Therefore, generally the ProcessManager will be using most of the CPU:



**Implementation:**



Our program is broken up into two separate classes, ProcessManagerModel. and PyChartViewController.

ProcessManagerModel is a class that can both access information about the computer's process and terminate them. ProcessManagerModel makes heavy use of the psutil library to access and control current processes. The only functions outside classes must use are update() and terminate\_process(). Outside classes can then access data, an OrderedDict that maps from {pid:(process\_name, cpu\_percent, ram\_usage\_mb)}. update() works by first remove the processes from data that are no longer running. It then adds all new processes on to the end of data. terminate\_process(pid) uses psutil to delete the process of a given pid.

PyChartViewController is a class that uses matplotlib.pyplot to display current information and allow the user to use the “Controller.” PyChartViewController contains a ProcessManagerModel object that is uses to access and control data. PyChartViewController runs a function update() in a loop, which calls update() on the ProcessManagerModel, and then replots the information on the screen. When the user clicks on parts of the pie chart, the PyChartViewController uses a dictionary it creates to map from the Wedge object to a process pid. When the user clicks terminate, PyChartViewController calls terminate\_process(pid) on its ProcessManagerModel.

An interesting design decision we made was making ProcessManagerModel contain an OrderedDict rather than a dict. This is because we knew we would plotting the processes in a pie chart, and we wanted the order to stay the same on each loop iteration. Because dicts are unordered, the processes could jump around the pie chart with each loop iteration if we used a regular dict.