Tony Diep
u0934661
01/26/2017
2420
Design Document For N_Body Program
Ed Stephen Ancajas

I pledge that the work done here was my own and that I have learned how to write this program, such that I could throw it out and restart and finish it in a timely manner. I am not turning in any work that I cannot understand, describe, or recreate. I further acknowledge that I contributed substantially to all code handed in and vouch for it's authenticity. (Tony Diep)

Program Design

In our N-Body program, we analyzed the time the program took calculate the frames for each Flotsam plotted on the Star_Field. It seemed clear that as we increase the amount of flotsams drawn onto the Star_Field, the amount of time to calculate the frame for each Flotsam increases over time. So there is a linear (perhaps somewhat quadratic) relationship between the amount of flotsams and the time to calculate the frame. We were surprised that the graph had some quadratic pattern but this is likely because we used a computer with a high processor (Intel i7) and with good amount of RAM, so drawing immense amounts of flotsams (even including the vectors for each of them) isn't so much problematic. Our initial design when approaching this project was to first get the Geometry_Vector class working and running because it is essentially the mathematical "workhorse" for the program. The vectors are best represented as a component that contains a x and y coordinate and every Flotsam carries those values.

Once we were able to complete and test the Geometry_Vector class, we moved on to getting the Satellites (Planets, Sun, Star, and Flotsams) to show onto the Star_Field. Although it was certainly a challenging process, we were able to fix major and minor bugs in our program. Most of these bugs usually occur when we forget to distinguish between the component as a

whole versus the graphics object that represented each satellite. At this point, we started the simulations and studied the time to calculate the frame for each satellite. The time to calculate was not linear and in fact took longer to calculate. We used the vectors to configure the random

locations frequently in order to best represent the movement in the solar system.

We spent over 15 hours working on this assignment, which is about the expected

amount of time for us to complete the assignment. We were at first startled about the assignment since in class when the sample program was demonstrated because it had a lot of complexity. However, I think after the countless hours of spending on this assignment, we realized some parts of the assignments are not that intimidating. Definitely the most

time-consuming part of the assignment is getting the planets to show up and move around. This

is because we forget to implement the paintComponent() method in each of the children classes since they have their own drawings. Also, we forgot to use the setLocation(), setSize(), and update_screen_coordinates() methods. We also saw that we overriden some of those methods when they shouldn't be. We forget to keep track of the previous changes in our code, which makes it more difficult to fix our programs' bugs. We do believe that an extension for this assignment would be nice because there is a lot of mathematical computations and it seemed overwhelming.

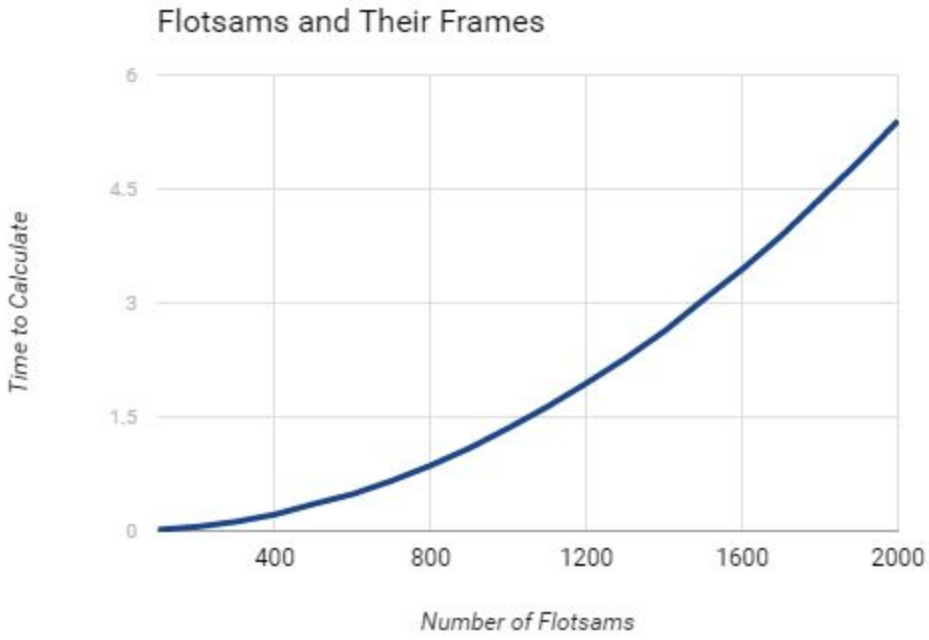
This project does involve a lot of inheritance and polymorphism, which are two key components of object-oriented programming (OOP). By doing this, we save a huge ton of code because it allows us to take advantage of already implemented methods (in which Java most likely has them built in). We use the keyword "instanceof" to give us control as to when certain

satellites should take place in the frame at a given moment. For example, when working on the supernova aspect of the star, we need to only make this happen when a star is clicked as opposed to a planet, flotsam, etc.

Although this project has been time-consuming and mostly frustrating, I have learned many things about GUI's. The project does teach a lesson about programming in that we do not have to be physicists or astronomers to create a simulation like this. If we had the time to do go

and beyond on this project, we would definitely work on the saucer and maybe an alien class to make things interesting.

Time Analysis



In our N-Body program, we analyzed the time the program took calculate the frames for each Flotsam plotted on the Star_Field. It seemed clear that as we increase the amount of flotsams drawn onto the Star_Field, the amount of time to calculate the frame for each Flotsam increases over time. So there is a linear (perhaps somewhat quadratic) relationship between the amount of flotsams and the time to calculate the frame. We were surprised that the graph had some quadratic pattern but this is likely because we used a computer with a high processor (Intel i7) and with good amount of RAM, so drawing immense amounts of flotsams (even including the vectors for each of them) isn't so much problematic. Thus, as each Flotsam was added to the program, some spot in memory in our laptop is reserved for each of the different Flotsams. We can also say that the size of the flotsams have some contributing factor because the bigger the Flotsams (and along with how many are drawn), requires more memory from the

laptop. There is a slight issue in that our program is creating a lot of Geometry_Vector objects than necessary because all of the tools are already there. When creating a vector, there is no need to create more instance variables that represent a x coordinate and y coordinate.

It is also wise to note that the laptop does not only draw the flotsams onto the frame but also executes the repetitive calls of such methods like paintComponent() and update_screen_coordinates() (which is called iteratively in the Star_Field class). This means the program creates a bunch of stack frames and it is important for the laptop to be able to handle them. This will affect the time to calculate the frames for each flotsam as well as the laptop speed when doing the task.