Zense Project Report

Simulation of motion under constant gravitational acceleration and collisions

Author – Hemanth Chitti

IMT2018028

Date – January 11th 2019

Idea

What I hoped to achieve in this project was an accurate simulation of motion of a projectile in a box under gravity on different planets, with the implicit assumption of motion close to the surface only. It would also entail elastic collisions with the walls of the box, without air resistance or buoyancy.

Technology used and implementation details

Having learnt Python in the first semester, I decided to code my simulation too in Python. To this end, I set out to use the module Pygame.

At first, I wanted to directly write kinematic equations for the x and y coordinates which we had learned in 11th class. However, I realized that this wasn't feasible, as I would have to account for collisions with walls as well and the equation would have to be transformed. I was able to get another way to do it, however – we didn't have to have an equation for the entire curve, we could just examine how the x and y coordinates changed over small distances, and the entire motion would end up being traced.

I had wanted to use a circle as the projectile, for then that would be an actual ball. However, the centre of a circle cannot assume non-integer coordinates, and thus the motion would not be as

smooth. Then I decided to draw a square as the projectile, as the coordinates of the top left corner of a square can assume float values also.

At first, I had been using a big square, of side length 50, but then I was suggested to use a smaller square, of side 5, as that would allow for the motion to be displayed better. After being able to view the projectile motion, I then implemented elastic collisions with the walls. I had the sides of the screen as the walls, ceiling, and floor.

To implement the condition for different planets, all I had to do was manipulate the gravity variables. I used the formula for g, GM/R^2, where G is the universal gravitational constant, M is mass of planet, R is radius of planet and got values for M and R of different planets from this website :https://www.universetoday.com/34024/mass-of-the-planets/. I then created lists to store the different variables, and then created a list to store the different values of acceleration due to gravities near surfaces of the different planets.

I had wanted to trace out the curve fully at first, which could be done using a function to light up a pixel. This could be accomplished by lighting up the current position of the projectile at every point. However, my previous algorithm for moving the projectile involved filling the screen with colour , which would thus erase the previous projectile. This would erase the lighted up pixel too, so the solution seemed to be storing all the pixels of the current positions in a list and lighting them all up in every iteration. But this takes too many iterations, and slows down the program considerably after a point till the point where you cannot see the projectile moving.

Then , I accidentally found a solution to this problem.

By drawing a black screen and then blitting the main screen onto it, I was able to get a thick white line on the screen wherever the projectile moved. In fact, as my initial plan of generating at least a small tail of 500 pixels had involved colours like red and blue, it led to a beautiful synthesis of colour against the dark background. Thus though I

had only wanted to simulate projectile motion, I ended up generating beautiful patterns!

Now, I had initially had an idea of choosing between the different planets based on user input, but a better suggestion I got was this – toggling between the planets at will. The game would start off on Mercury, and the user would provide the input based on 1-indexing of the list of planets, and using this as an index (after converting to 0- indexing)for the list of g, I could get the different values for the planets. This made my initial idea of simulation more interesting.

What I wanted to do was change the background image with the planets, but this would affect the tracing of the curve, so I instead printed a statement when the planet changed, and kept the beautiful (in my eyes, with white curves on a dark background) tracing.

This is how I implemented and improved on my idea of simulating projectile motion.

Images and Videos

A video of the simulation can be found here :

https://www.youtube.com/watch?v=qmsw5fd4iTs&feature=youtu.be

Future scope of project

I hope to be able to extend my project to be able to simulate varying accelerations of gravities with heights, and also add in varying densities of air. I also hope to be able to add in general relativity and thus simulate motion at very high speeds or near regions of high gravity. Though for now, it

was just a fun exercise to toggle between planets with collisions, I would want to make it a hyper-realistic description later on.

Experience of doing project

Doing this project was a mixed bag for me. It took me time to get started, as I had a different idea at the start of not actually moving the ball and instead making a sort of Desmos program where moving the ball along the curve with arrow keys

would display the coordinates . However, I abandoned the idea later on.

At first, I was ready to do my project using Pygame library, but then I found out about Visual Python, another module which could display 3D motion, and which also had many inbuilt functionalities, especially for tracing. I tried to use it , but then it was tricky to manipulate the camera action (Visual Python moves objects by moving the camera). Soon, I was able to see that I could achieve my objective of tracing even in Pygame, so I decided to go back to Pygame and finish it.

I believe that I now have actual interest in Game Development and Simulations after this project and the one that I had to submit for the last semester's end term, as it was great that I could just come back after class and keep working on this. My initial idea evolved quite a bit as well, and some which were not feasible also had to be dropped. It was fun to be hacking away, looking for a solution to my problems, either by thinking about it myself or asking my friends. Especially

interesting is how I started off with no clear idea of how to proceed but then slowly kept making my program better and better , and then managed to build something.

It also was something I used to think about in the middle of a boring class, thinking about whether it was feasible to add a feature or trying to mathematically derive an equation which would help me. So all in all, while very frustrating and despairing to work with at times, I can say this project was a great experience to have.