Project Elements Checklist Form

This form must be completed and appended to your final project write-up or essay.

Synthesis of Lecture Material:
Please complete the table below, to summarize how your project synthesises material from at least 2 lectures. You may earn bonus points for including material from a 3rd lecture.

Lecture Number/Topic	Concepts included in the Project	Location in the Project/Essay/Write-Up
Lecture 8 and lab 3	Light behaving as a wave or ray, depending on the experiment	The simulation simulates and collects data as light travels as a ray
Lecture 7	Light having a constant speed and gravity have a calculatable constant	To find how much light rays change using general relativity, these constants are needed
Lecture 2	Light diminishes in intensity at the pace of the inverse square law	The simulation simulates and collects data as light travels and diminishes in intensity
Lecture 9	Light "bends" in trajectory around objects of large mass due to time dilation, and can be calculated using general relativity equations (also found in this lecture)	The simulation simulates and calculates this concept
(Optional Bonus)		

My write up includes all of these aswell, since my write up explains the simulation

Quantitative analysis:

a) Briefly describe the quantitative analysis component of your project:

Seeing as my project does multiple calculations (calculates light intensity, light ray position based on a trajectory, and light trajectory change due to time dilation)
just for one light ray and can calculate many light rays, my project is entirely quantitative analysis; there is an output.

b) Where is the quantitative component(s) found, within your project, essay, and/or write-up?

The file named "main.py" has my code that shows how the simulation runs and the calculations at work, but also, my write up goes through a small
sample run to show the equations and their results in a digestible manner on page 4-5.
There is also a flow chart of how my code works and the calculations used on page 4 of my write up.

1. The Project & its Connection to the Course

My project is to build and test a program that simulates light rays in an input environment; including their dissipation in light intensity due to the inverse square law, and their change in direction as objects of large mass cause the light rays to "bend" due to time dilation, also known as gravity lensing. The project will simulate this on a two dimensional plain using a simple cartesian plane.

How to use the simulation and build test input environments can be found in the file path where this file is found, in the readMe.txt file.

The following is an example of a possible input environment and the following simulation, with the yellow circle being a light source, the orange circle being a collection point, and the rest of the circles being planets; the simulation could simulate the following environment

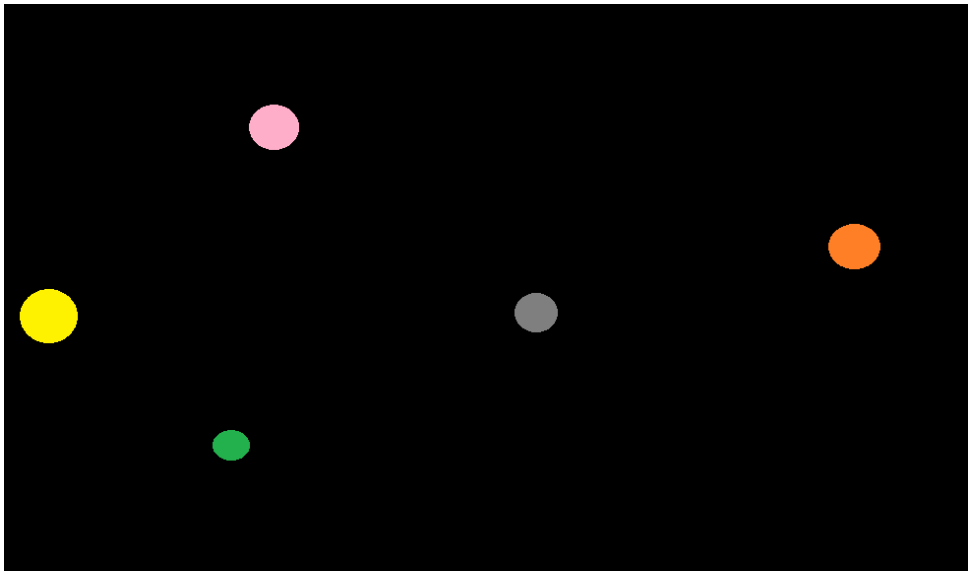

Figure 1.1
and then simulate the trajectory and intensity of the emitted light rays.

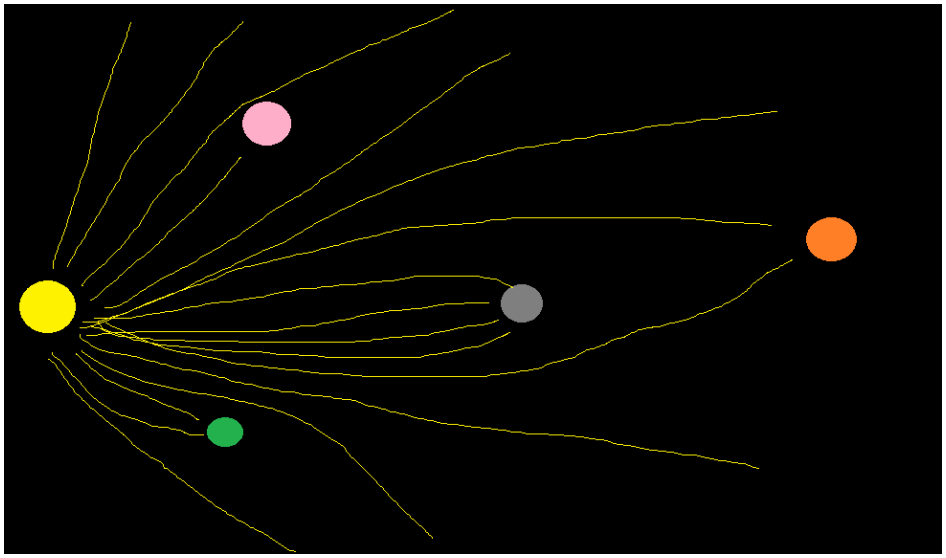

Figure 1.2
The only output, thought, would be the average light intensity collected at the orange circle.

This project relies on four keen aspects taught in the course: light's ability to act either as a wave or ray depending on what is being tested (it is being treated as a ray in this regard) seen in lecture 8 and lab 3, light having a constant speed and gravity having a calculable constant seen in lecture 7, light diminishing in intensity due to the inverse square law seen in lecture 2, and light bending in trajectory due to time dilation around objects with large mass and the general relativity equation to calculate it seen in lecture 9.

2. How the Project Actually Works

The simulation first loads the specified json file and breaks up the three possible objects (light sources, planets, and collection points) into separate data structures. Then, the angle needed to evenly simulate light rays from each source object is found by dividing 360 degrees by the number of requested light rays per source.

The program then loops through every light source, doing the following each time for each light ray:

- Finds the angle of the light source by taking the previously found angle and multiplying it by the number of light rays simulated for that light source
- Finds the starting coordinate pair for the current light ray by doing the following algorithm, (lightSourceXCoordinate+lightSourceRadius*cos(currentRaysAngle), lightSourceYCoordinate+lightSourceRadius*sin(currentRaysAngle))
- The ray's angle is changed if any objects of mass are within range using the following equations, sin(angleBetweenLightRayAndMass-currentAngle))*(angleChangeDueToGravity), angleChangeDueToGravity = ((gravitationalConstant*planetMass)/(distanceBetween(rayCoordinate, planetCoordinate)*(speedOfLight**2)))*4
- The current ray's coordinates are moved using the following equation, (currentRayXCoordinate+cos(currentRaysAngle), currentRayYCoordinate+sin(currentRaysAngle)), then distance is increased by 1 km, and the intensity of the light is diminished proportionally using the inverse square law, currentRaysLumen = currentRaysLumen/(4*pi*(distance**2))
- The previous two steps are repeated until the light is collected, runs into a planet or source, or fades below 1 Lumen
- The collected intensities for each collection point is then averaged by the number of rays that made it to the collection point

Below is a flow chart for better understanding.

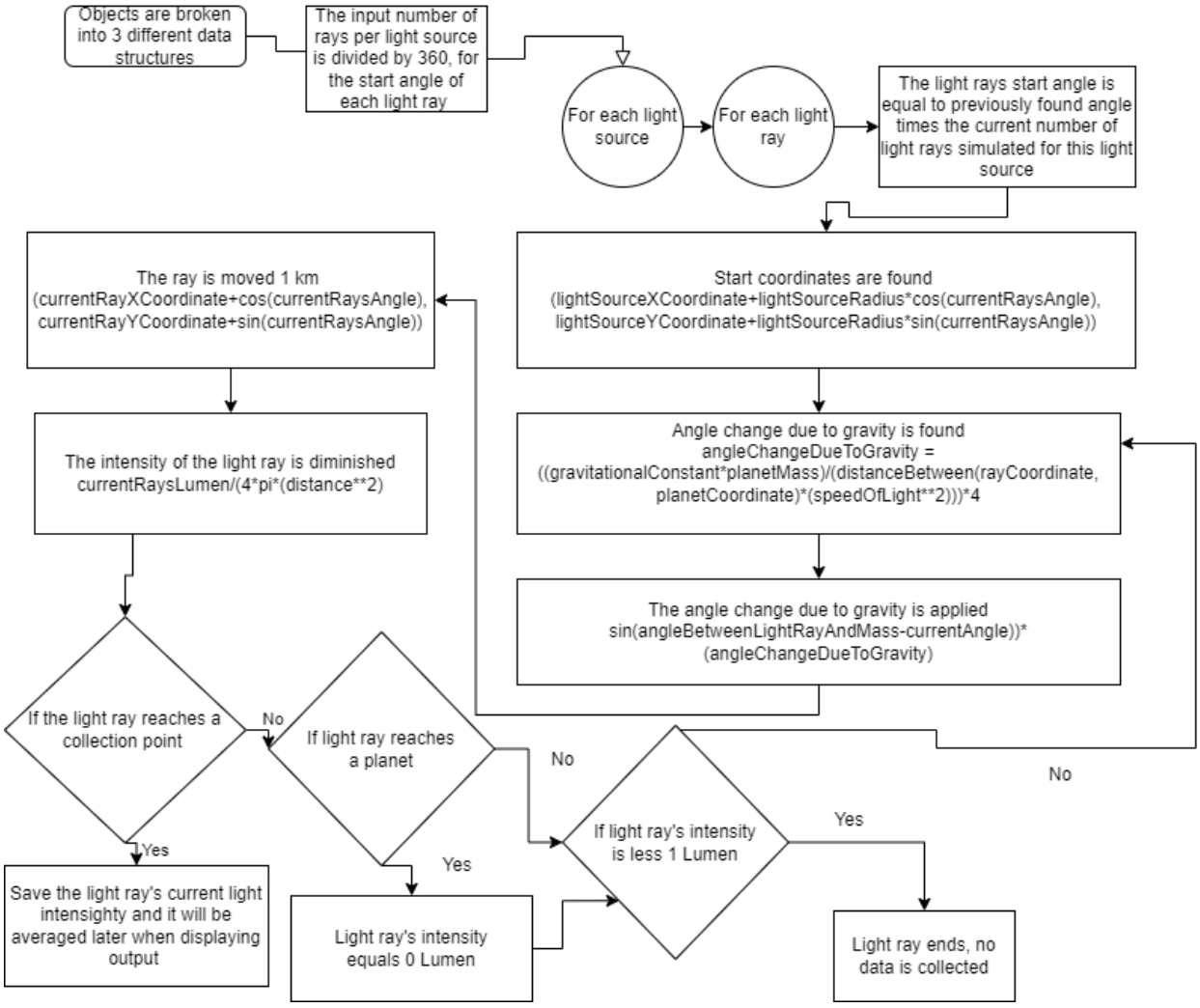

Figure 2.1

2. A. A Short Example Simulation

Input scenario (This is testInput1.json):

-A light source at (100km, 100km), radius of 10 km, intensity of 100000 Lumen, and mass of 100 kg

-A collection point at (150km, 100km) and radius of 5km

-Simulating 4 light rays per source

If the user wanted to simulate 4 light rays, a very small number, the simulation would do 360 degrees/4 which equals 90 degrees between each light ray revolving around the light source. Then the simulation would start simulating light rays.

For light ray 1:

angle = 90*0=0 degrees (light is traveling “East” at this angle)

starting coordinates = (100km+10km*cos(0), 100km+10km*sin(0)) = (110km, 100km)

Then the light begins to travel,

The only object of mass nearby is the light source it came from, so

angleChangeDueToGravity = ((6.6743 × 10-11 m3 kg-1 s-2*100kg)/(1km*(299 792 458 m / s ^2)))*4 = 1.702e-24

sin(0-0))*(1.702e-24) = 0 degrees (the mass is so little it would not significantly change the light ray’s trajectory and without any other “bending” due to time dilation)

new coordinates = (110+cos(0), 100+sin(0)) = (111km, 100km)

Distance is now equal to 1 km, the light rays intensity changes by 100000/(4*pi*(1**2)) = 7957.747 Lumen

This repeats until the light ray eventually reaches the collection point with 6.140 Lumen left

The other light rays have angles of 90, 180, and 270 degrees, all of which would just go off into space and dissipate.

Since only one light ray made it to the collection point, 6.140Lumen/1 = 6.140 Lumen is the final output.

3. The Used Units for the Project

All radi, distances, and coordinates are in kilometers. (0, 0) is the origin in all coordinate systems

Intensity of light is measured in Lumen.

All masses are in kilograms.

4. The Pre-Made Test Inputs

Test inputs 1-3 are to test if light is traveling correctly from a star and if the collection points are working correctly. Test input 4 is to show that light rays are being "bent" by objects with a very large mass present in the environment; the star is on the left of the planet and the collection point is on the right of the planet. The star, planet, and the collection point all have the same radius so unless the light rays change in direction due to the planet's mass causing time dilation, the collection point should not receive any light as all the light rays in its direction will be absorbed by the planet. However, the collection point does receive light rays, meaning that light is changing in direction due to the planet's mass. Test input 5 is just to ensure that multiple objects that vary in order and size are able to be simulated.

5. The Limitations of this Project

The major limitations of this project is that because it is in a two dimensional environment, any real world applications are heavily limited as the real world is in a three dimensional environment. A three dimensional simulation would not be difficult to build using what is already created, but it reaches a level of complexity that would make showing what my understanding of the course is harder and in general be more time consuming.

Also in real solar systems, collection points, light sources, and objects of mass all move, and this simulation does not allow for that. This is because again that would reach a level of

complexity that would make showing what my understanding of the course is harder and again be much more time consuming.

6. An Explanation of why the Equations Selected were Selected

The trigonometry like finding the start coordinates of a light ray and how to move them 1 km at a time were picked out of simplicity; also, I don't know another way to simulate this environment all objects in this simulation are circular.

The general relativity equation to find how much the light trajectory could change due to time dilation was presented in class;

$$\theta_{\text{gr}} = 4\frac{Gm}{rc^2}.$$

Figure 6.1

however, the idea of using a general relativity equation was presented with support from my TA and then further backed by Sanjoy Mahan in a paper titled "Estimating light bending using order-of-magnitude physics". However, to actually apply the result of this calculation required more research.

If the resultant of the calculation in figure 6.1 was 6 degrees (a large change in directory), the following trajectory change in light rays (the arrows in this context) due to objects of large mass (the center circles in this context) was required as seen below in figure 6.2.

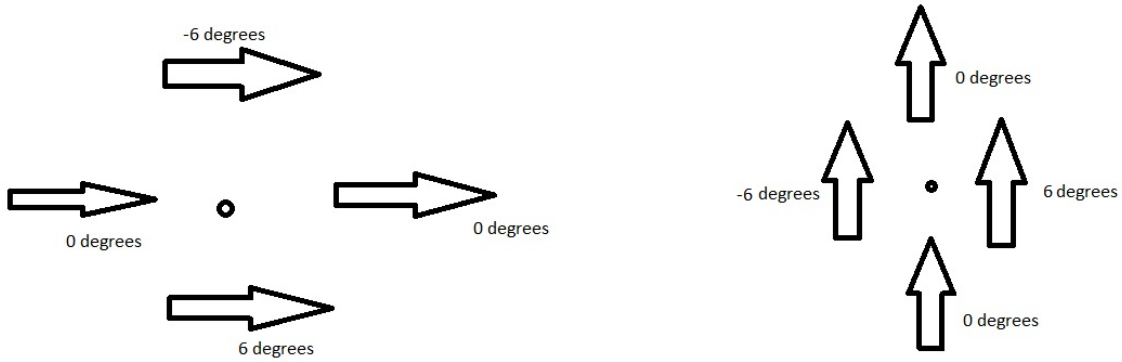

Figure 6.2

This behavior was further backed and the equation to apply this behavior was found in the article “Gravitational Lensing - by Ricky Leon Murphy:” on the website “Astronomy Online". The sin(angleBetweenLightRayAndMass-currentAngle))*(angleChangeDueToGravity) equation required research on how to use python’s built in math library to obtain the angle between two points, which was found in “*Math — Mathematical Functions — Python 3.11.0 Documentation*” and “Python Calculate Angle Between Two Points With Code Examples”.

The final major equation used was the inverse square law.

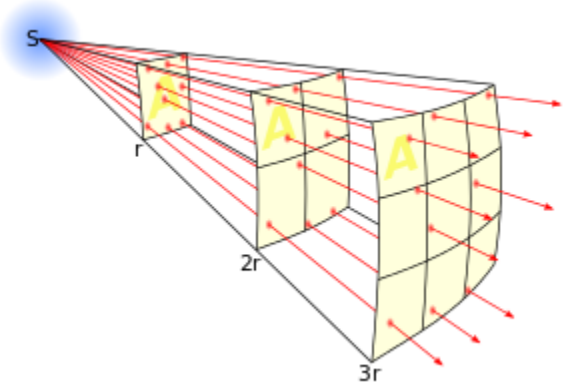

Figure 6.3 (*Inverse Square Law - Energy Education*)

Which was explained and presented in class and the input variables were simple to obtain.

Work Cited

"Astronomy Online." *Copyright Â©2004 - 2006*, astronomyonline.org/Cosmology/GravitationalLensing.asp.

Inverse Square Law - Energy Education. energyeducation.ca/encyclopedia/Inverse_square_law.

Mahajan, Sanjoy. *Estimating Light Bending Using Order-of-magnitude Physics*. Cambridge, 2002.

Math — Mathematical Functions — Python 3.11.0 Documentation. docs.python.org/3/library/math.html.

Python Calculate Angle Between Two Points With Code Examples. www.folkstalk.com/tech/python-calculate-angle-between-two-points-with-code-examples .