The Luminous GUI:

Efficiently Performing and Visualizing Stellar Collision Simulations, An Analysis of S0-2's Orbit of Sgr A*

Maxwell Boshaw

Advisor: Dr. James C. Lombardi, Jr.

Second Reader: Dr. Doros Petasis

ALLEGHENY COLLEGE

PHYSICS

Department of Physics

Allegheny College

Meadville, Pennsylvania

April 2021

Abstract

S0-2 is a star that orbits the supermassive blackhole at the center of our galaxy Sagittarius A* (Sgr A*), and S0-2 offers a unique opportunity to run realistic simulations of objects orbiting a black hole. By using the Smoothed Particle Hydrodynamics code *StarSmasher*, we can perform these simulations. However, the use of this code typically requires time and practice to produce worthwhile simulations. It is the intention of this paper to create a more efficient process by which researchers can use this code and receive quality data. To do so, this paper will introduce a Graphical User Interface (GUI) that will take user input as data, run the *StarSmasher* program, and eventually produce visual output in the form of a column density plot of the collision in addition to a light curve produced by a secondary program and using a cutting-edge Luminosity code, *FluxCal*. The code currently accomplishes the task of initializing the *StarSmasher* program using user-input, but still requires further code implementation to successfully automate the entire collision process and data visualization. Further additions could implement *FluxE* and *FluxCal* to gather luminosity data that could broaden the scope of these simulations and allow researchers to accurately simulate or predict realistic systems.

Acknowledgments

I want to thank the following people:

Dr. James C. Lombardi Jr.: For his constant support of my academic pursuits. He allowed me to start work in his lab while I was a first-year and I ended up staying for three extra years. I was allowed to explore, struggle, and fail under the vigilance of someone that wanted to see me succeed. None of what I am doing now or the places I will go next would be possible without his help.

Dr. Adele Poynor: For her advisement and strategies to better manage my academics and stress. She helped me become a better student and showed me ways to be a better person in the process. Without her approval my very first year at Allegheny, I would have not been able to take Physics 120 as a first-year, so it is very much thanks to her that I was able to continue on the path of physics at all.

Dr. David Statman: For his constant check-ins with the material. Even when dealing with some of the hardest conceptual physics I have ever encountered, he always checked for understanding and made sure no one was left behind. The habits I created in his courses - specifically math gym - will no doubt carry into the work I will do in graduate school.

Dr. Doros Petasis: For his constant pushing of myself to be the best physicist I could be. Each time I have met with him, he always sought to push me to my limits. He taught me how to look for novel ways to solve problems and that even when something is hard, there is always a solution even if it isn’t immediately apparent.

Dr. Dan Willey: For his compassion and understanding in the courses I have taken with him. Even when I struggled with the hardest physics course I had ever taken (E&M), he helped me ensure I understood the material and was able to complete my work to a sufficient degree when I doubted it was possible.

Pr. Betsy Sumerfield and Ballroom: For the creative outlet and respite from stress I needed during this extremely tense and stressful time of my senior year. I worried I wouldn't be able to dance again before I graduated because of covid, but I am very glad I had the chance to return. The fond memories I will carry on from Ballroom classes alone is enough to last many years, but the friendships and support I received from Betsy and my classmates, like Josephine Hughes, are things I will carry with me and cherish forever.

My Fiji Brothers: For being a constant of my four years at Allegheny and providing me with numerous opportunities to grow as a learner and as a leader. The bonds I created and strengthened gave me a support system that without which I would have certainly suffered much more during covid and during all my years. The social aspect of the college learning environment is quintessential to growth, and I very much was able to grow into my own because of the brotherhood.

Ryan Mitchell Boshaw: For being the reason I look to the stars in the first place. Without you there is no Physics major Max Boshaw. I will forever be grateful for the impact you have had on my life. Starting with using your telescope and loving Orion's Belt and now applying to work at some of the best universities and observatories in the world so I can have an even better view. Thank you for being my inspiration.

Contents

List of Figures 7

Chapter 1 - Introduction 8

Chapter 2 - Theory 9

2.1 Galactic Center 9

2.1.1 Andrea Ghez 9

2.1.2 Keck Telescope 9

2.1.3 Sagittarius A* 10

2.1.4 S0-2 10

2.2 Black Holes 12

2.2.1 Schwarzschild Radius 12

2.3 Orbital Dynamics 14

2.3.1 Tidal Forces 14

2.3.2 Radial Velocity 15

2.3.3 Gravitational Redshift 17

2.4 Flux 17

2.4.1 Luminosity 18

2.4.2 Light Curves 19

Chapter 3 - Methodology 20

3.1 GUI 20

3.1.1 PyQt Interface 20

3.1.2 JSON Packaging 20

3.2 Smoothed Particle Hydrodynamics 20

3.2.1 Traditional Use of *StarSmasher* 20

3.2.2 Automating *StarSmasher* and Implementing the GUI 21
3.3 Running the GUI 21
3.3.1 Initial Conditions 21
Chapter 4 - Results and Analysis 22
4.1 Automation 22
4.1.1 Stellar Relaxation 22
4.1.2 Stellar Object Collisions 23
Chapter 5 - Conclusion 24
5.1 Future Work 24
Appendix A - Code 25
A.1 starsmasher_gui.py 25
A.2 input_json.py 29
A.3 init_json.py 33
A.4 How to use the GUI 35

List of Figures

Figure I: Image of S0-2 and S0-102's orbits [2]. 10
Figure II: Example ellipse showing the semimajor axis, a, the semiminor axis, b, the star at one focus with mass M, the orbitting planet of mass m, and the relationship the eccenetricity, e, has with a. 11
Figure III: The illustration depicts blue and red-shifted light being emitted by a star from the perspective of a far-away observer. 15
Figure IV: The curve above is a fictional example of what a basic characteristic light curve of a stellar object may look like. 19
Figure V: The current version of the StarSmasher GUI data collection interface 21
Figure VI: Relaxation results as viewed through a program called JSplash that helps visualize StarSmasher data. The top row shows particle plots in a two-dimensional view of the two stars made by a successful test run of the GUI. Data for star 1 (left column) is Mass (in solar masses) = 10, Radius (in Solar radii) = 5.6, and Particles = 5000. Data for star 2 (right column) is Mass (in solar masses) = 5, Radius (in Solar radii) = 3.8, and Particles = 2500. The bottom row shows a log column density plot of the same stars. 23

Chapter 1 - Introduction

The purpose of a simulation is to represent and predict naturally occurring phenomena as closely as possible. Doing so gives us a better and more assured understanding of the universe around us. However, to measure accuracy, real-life data of these natural phenomena are necessary. The Keck Observatory is one such source of taking these real-life measurements and has produced incredible amounts of data surrounding the black hole at the center of our galaxy, Sagittarius A* (Sgr A*), and its surrounding stars, including S0-2 [1]. *StarSmasher* is the simulation companion for this data. This Smoothed Particle Hydrodynamics code is the measuring stick of comparison for the analogous data observed by the Keck Observatory. The code itself is a detailed web of calculations, Fortran code, and gravity calculations written in cuda, but our interest lies in explicitly running the program itself.

A priority endeavor is to make the process of creating and running these simulations as simplified as possible to encourage improvement of the accuracy of the simulations themselves. To achieve this, we will be creating a Graphical User Interface (GUI) for users to enter collision data and automate the file creation process necessary for the simulation to run and produce viable output. We could then use our simplified simulation process to run analysis of the star S0-2's orbit of Sgr A* and compare our results to that of S0-2's actual orbit with assistance from a cutting-edge new luminosity code, FluxCal.

Chapter 2 - Theory

2.1 Galactic Center

Our galactic center refers to the area at the center of our Milky Way Galaxy that is home to our supermassive black hole Sgr A* and its accompanying stars and celestial objects. The observation of this center gives way to years' worth of research and includes data on stars like S0-2 [1].

2.1.1 Andrea Ghez

Plenty of researchers deserve credit for the amazing research they do. The work that some scientists dedicate enormous portions of their life studying can make massive waves within the scientific community. As this research is centered on the center of our galaxy, the star S0-2, and the observations surrounding them, it seems only right to recognize one of the pioneers of these observations and Nobel Laureate, Andrea Ghez [2]. Her discovery of the supermassive black hole Sagittarius A* is what won her the Award in 2020. Without her contributions to the research of the galactic center and continued work in the subject, no part of this paper would even be a topic of conversation or an idea for research to engage with.

2.1.2 Keck Telescope

To observe, we must have an observatory. The Keck Telescope in Hawaii is the source of nigh all of the data used in study of our galactic center and plays a critical role in collecting information necessary for understanding what our small corner of the universe contains [2]. This telescope is our primary source and thus will be key in reviewing and comparing our simulation results and data to what we have been able to observe and monitor at our galactic center. The Keck observatory consists of two massive 10-meter mirror telescopes that produce optical and infrared observation data through use of a laser to pierce Earth's atmosphere and lessen any interference it may cause.

The Keck I and Keck II telescopes use adaptive optics systems and a laser guide that allow massive improvements in image quality by a factor of 10 to 20. It also increases the percentage of observable targets in the sky to 70-80 percent as opposed to 1 percent without the adaptive laser. This Sodium laser helps compensate for atmospheric distortions and creates an "artificial star" that is what allows the telescope to have these extra precisions [3].

2.1.3 Sagittarius A*

The supermassive black hole at the center of our galaxy, Sagittarius A*, is a source of great curiosity and thus thorough analysis and observation. This massive object is host to a number of stars and stellar objects swirling around it at tremendous speeds. It is at this heart of our galaxy where we witness relativistic events and attempt to find a deeper understanding of how this massive object works as well as our universe as a whole [1, 2, 4, 5].

2.1.4 S0-2

One of many stars that orbit Sgr A* is S0-2. Despite being one of many stars total, S0-2 is unique in the fact that it has one of the shortest orbits of these stars and in fact is only one of two stars with full phase coverage and thus is one of the most documented stars around Sgr A* [2]. In particular, S0-2 has explicit research on its possible binarity and the interactions it has with relativistic redshift in its orbit [4, 5]. One of the main reasons S0-2 gets studied so much is because it acts as a natural occurring experiment to study the effects of Einstein's theory of general theory of relativity of an object near a supermassive black hole.

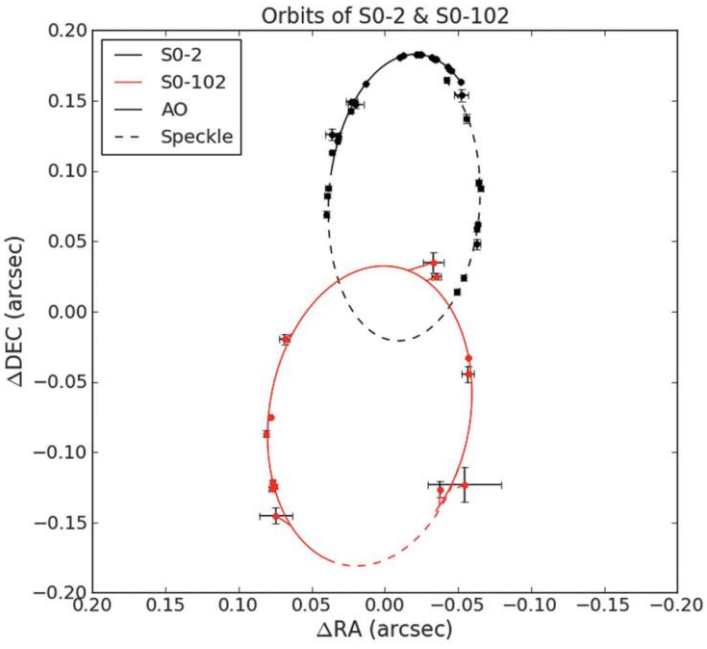

Figure I: Image of S0-2 and S0-102's orbits [2].

With an estimated mass of about 9 times the mass of our Sun, S0-2 is a relatively young massive star orbiting Sgr A* and that fact alone raises questions on how it formed in such a volatile environment [6]. Simulating these well-observed systems help us understand how they work on more nuanced levels and even allow the potential for determining potential initial conditions that could have created the systems we observe.

Shown in Figure IV are the orbits of the two shortest known period stars to orbit Sgr A* according to their right ascension and declination. S0-102's orbital period is actually 11.5 years as opposed to S0-2's 16 years [2, 5]. When we consider the orbital periods of stars, we should look to Kepler's Laws to assist our understanding of how we come about those results. In this case, we want to explicitly use Kepler's Third Law and we will derive it by starting with the integral of the equation for the Second Law over one Period. The variables used are μ for the reduced mass of the system, *L* for angular momentum, *P* for the orbital period that we integrated over, μ for the reduced mass, and *A* for the area. We will end up solving for the period squared in terms of known orbital measurements.

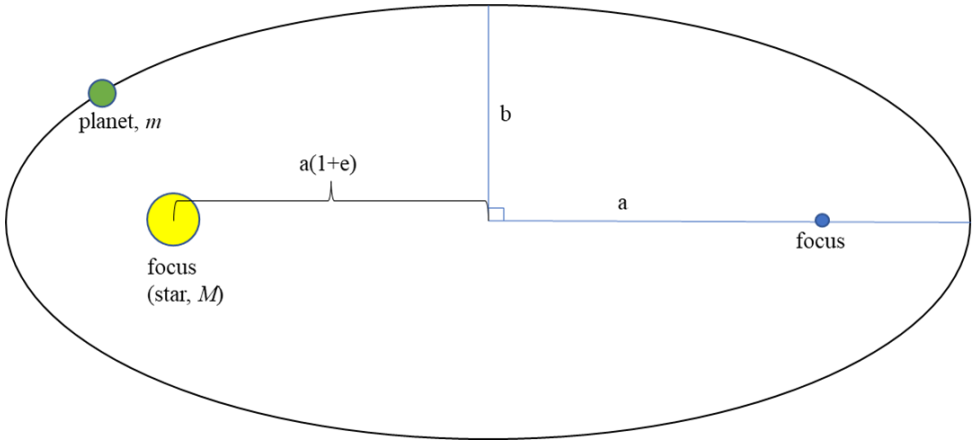

Figure II: Example ellipse showing the semimajor axis, a, *the semiminor axis,* b, *the star at one focus with mass* M, *the orbitting planet of mass* m, *and the relationship the eccenetricity,* e, *has with* a.

Of these orbital measurements we should know how to find is the semimajor axis *a* via $b^2 = a^2(1 - e^2)$, where *e* is eccentricity and *b* is the semiminor axis and the equation is derived through the application of the Pythagorean theorem on an ellipse. In addition, we can express the area, *A*, of this ellipse through $A = \pi ab$. We should also know the expression for total orbital angular momentum $L = \mu\sqrt{GMa(1 - e^2)}$ [7].

$$A = \frac{1}{2}\frac{L}{\mu}P \text{ [7]}$$

$$P^2 = \frac{4\pi^2 a^2 b^2 \mu^2}{L^2}$$

$$P^2 = \frac{4\pi^2}{G(m_1 + m_2)} a^3 \quad (1)$$

In the instance where our objects are highly disparate in mass - as is the case with S0-2 and Sgr A* - this equation becomes simplified.

$$P^2 = \frac{4\pi^2}{GM_{sgr\ A*}} a^3 \qquad for\ m_{S0-2} \ll m_{Sgr\ A*} \quad (2)$$

2.2 Black Holes

Massive objects spread throughout our universe exist and some of them cannot even be directly observed. Black holes are such an object. An astrophysical object that is so inconceivably dense that it does not allow light to escape the region it inhabits and appears to the observer as simply-black. These wonders of nature form from dying stars, typically of about 15 times the mass of the sun or greater. However, there exists a larger category of black hole separate from the so-called solar mass category: supermassive black holes. These range in mass millions of times greater than the mass of our own sun. They typically reside in parts of the universe where galaxies surround them. Our very own Milky Way is one such galaxy and Sgr A* is one such supermassive black hole.

2.2.1 Schwarzschild Radius

Since direct observations of black holes are quite impossible, instead we gather data from the surrounding area to determine the properties of these objects. In this case, the Schwarzschild radius refers to the radius necessary for an object of a certain mass to become a black hole. It is also a good estimate for the event horizon of a black hole, particularly ones that are non-rotating and uncharged [7, 8]. In a mathematical sense, we can think of achieve an understanding of the Schwarzschild radius through an understanding of gravity and the equation for the escape velocity. Specifically in the instance when the total energy is zero due to the law of conservation of energy.

$$E = \frac{1}{2}mv^2 - \frac{GMm}{R} \quad (3)$$

In this equation, *G* represents Newton's gravitational constant of $6.674 \times 10^{-11} \frac{N*m^2}{kg^2}$, m represents the mass of the object in motion, *M* gives us the mass of the massive object that m is being acted on from, *v* is the velocity of object *m*, *E* is the total energy of the system, and *R* is the radius between *m* and *M*. By first taking the expression for the total energy of an object being acted on by the gravitational potential energy of a massive object and subtracting that from the overall kinetic energy of the object, we start with Equation 2.1. Our next step is to figure out what our escape velocity would be for this generic object. To do this, we must know that the total gravitational potential energy of this object needs to be zero at the 'end' of its travel away from the massive object. We also need to know that the kinetic energy at this distance would also be zero, meaning that the total energy of the system is zero. By the law of conservation of energy this also means that our initial total energy is zero and then we can begin to solve for the escape velocity.

$$0 = \frac{1}{2}mv_e^2 - \frac{GMm}{R}$$

$$\frac{1}{2}mv_e^2 = \frac{GMm}{R}$$

$$v_e^2 = \frac{2GM}{R} \quad (4)$$

Normally, to find the escape velocity you would next take the square root of both sides, but for our purposes we want to find the Schwarzschild radius and therefore will now think of this equation as the radius at which light would no longer be able to escape a black hole if it crossed this threshold. We will replace our v_e^2 term with a c^2 term to represent the speed of light squared, or $(3 \times 10^8 \frac{m}{s})^2$.

$$c^2 = \frac{2GM}{R}$$

$$R = \frac{2GM}{c^2} \quad (5)$$

Solving our new equation gives us our formula to determine the Schwarzschild radius of an object if it were a non-rotating non-magnetic black hole. For context in our understanding of Sgr A*, its Schwarzschild radius is roughly $1.2 \times 10^7 km$ or 0.08 times the distance from Earth to our sun [9]. Compared to the closest approach of S0-2, which is 120 times the distance from Earth to our sun, the Schwarzschild radius is many times closer to Sgr A* [5].

2.3 Orbital Dynamics

Orbital mechanics are a Newtonian concept, but as we start to deal with larger and faster objects, the sense of understanding we have surrounding these objects in orbit shifts to a much more relativistic viewpoint. We need to be able to understand how these more complex interactions affect the systems we are trying to represent so we can better represent them in our simulations.

2.3.1 Tidal Forces

Black holes and stars exert an unbelievable amount of gravity on their surroundings. However, the gravity that nearby stellar objects experience is not constant. The perfect example of this interaction is our very own moon exerting gravity on Earth and causing a bulge in the oceans which creates the tides, hence tidal forces. The interaction is more complex than a simple tug, and it is actually the effect of different parts of an object being pulled by gravity differently depending on their distance to the other object.

$$F_m = G\frac{Mm_1}{r^2} \text{ [7]} \quad (6)$$

This equation represents the force acting on a test object with mass m_1 at a distance of *r* away from the center of mass of the more massive object *M*. If we next consider that the test mass is on Earth and is not in line with the moon, we then have the tidal force interactions to consider. In fact, when we consider these forces, the differential force between this test mass and

another test mass ends up as the derivative of the force with respect to the radius, *dr*, which is the separation of the two masses relative to their centers of mass [7].

$$(\frac{dF_m}{dr})dr = -2G\frac{Mm}{r^3}dr \quad (7)$$

Notice how this tidal force term scales as $\frac{1}{r^3}$ instead if $\frac{1}{r^2}$ as the normal effect of gravity. This tells us that we will have less of a tidal force at far distances and a much higher one at very close distances.

2.3.2 Radial Velocity

Radial velocity refers to the motion of a star along its orbit as seen from the observer's point of view. We can observe a star's radial motion when it is moving wither towards or away from us

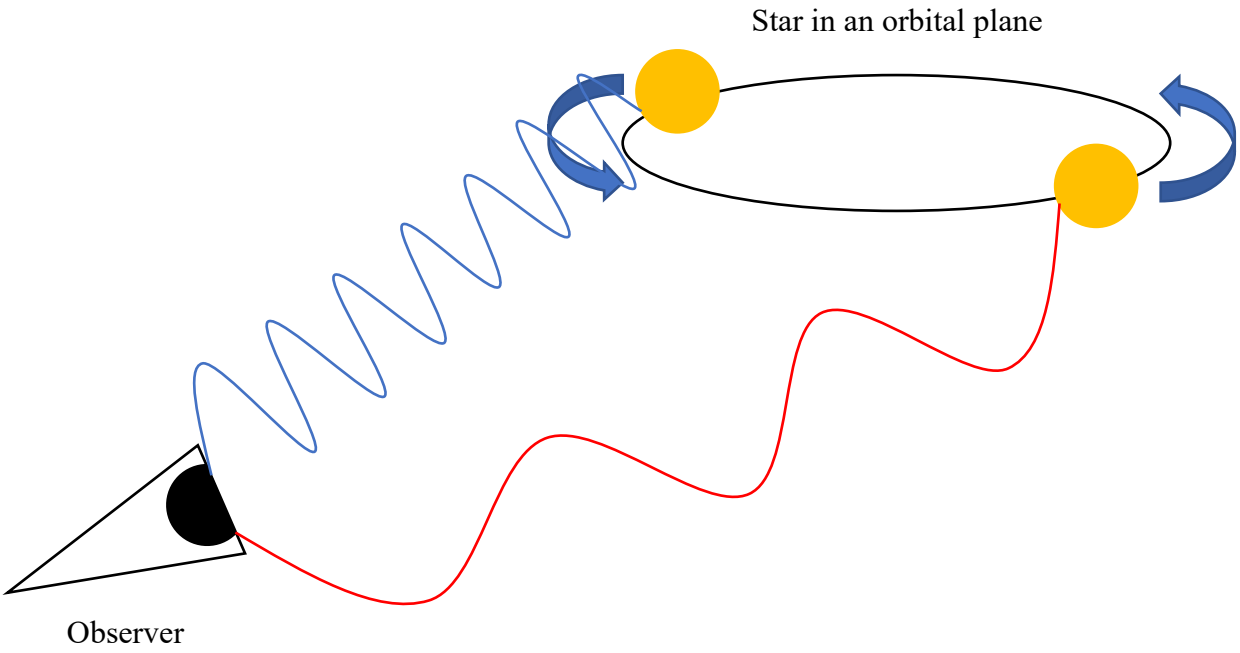

Figure III: The illustration depicts blue and red-shifted light being emitted by a star from the perspective of a far-away observer.

as the observer in its respective plane of motion. When the star is moving toward us, the light it emits gets blue-shifted due to the Doppler effect compressing the waves of light. When a star moves away from us, the light is instead expanded by the Doppler shift and appears red-shifted.

Doppler shift should be defined more thoroughly for the purposes of this explanation of radial velocity. In conceptual terms, the Doppler shift is any compression or expansion of wave motion through a medium due to motion of either the observer or source [7]. Mathematically, we can represent this shift as a ratio of the wavelengths compared to the speeds. Any change in the wavelength increases the difference in perceived speed of the source. More specifically, we can represent this relationship as the following equation.

$$\frac{\lambda_{observed} - \lambda_{rest}}{\lambda_{rest}} = \frac{\Delta\lambda}{\lambda_{rest}} = \frac{v_r}{v_s} \text{ [7]} \quad (8)$$

However, this equation is only representative of sound waves. Light has a slightly different Doppler shift equation due to the relativistic properties that light speed is accompanied by. When we consider what is necessary to find the Doppler shift for light, we assume the velocity of the object relative to the observer is on the order of relativistic speeds near the speed of light, *c*. Accounting for the reference frame of the observer, the difference between perceived and actual velocity of the event is in the time that an event takes place. This time dilation as it is called, can be expressed using a variation of the equation for proper length.

$$L' = \frac{L}{\sqrt{1 - u^2/c^2}} \text{ [7]} \quad (9)$$

In this equation, L' is the rest length, L is the moving length, and u is the velocity relative to the observer. This is all to show that reference frames of objects that move fast enough will appear to an observer to be a different length than they actually are. In addition, if u is a relatively low speed, say 100 kilometers per hour (roughly 62 miles per hour), then u^2/c^2 is negligible and we end up observing the same length of the moving object as if it was at rest. Using this relationship, we can now define the relativistic Doppler shift by applying the relationship of time being equal to one divided by the frequency.

$$\nu_{obs} = \frac{\nu_{rest}\sqrt{1 - u^2/c^2}}{1 + \left(\frac{u}{c}\right)cos\theta} = \frac{\nu_{rest}\sqrt{1 - u^2/c^2}}{1 + \frac{v_r}{c}} \text{ [7]} \quad (10)$$

The radial velocity is given by the term in the denominator, specifically we see the relationship of $v_r = ucos\theta$ for our radial velocity [7]. The simple explanation for the necessity of these extra terms is that as events - in this case the incident light arriving to an observer - occur in space, they also occur within time. Essentially, because we are dealing with two

directional components, we end up using the relation $cos\theta = (\frac{adjacent}{hypotenuse})$ to describe the term that ends up being the radial velocity.

2.3.3 Gravitational Redshift

The massive gravity of black holes heavily affects the objects around them. The effect of these black holes is so great, in fact, that even light itself is entirely changed when interacting with a black hole. The light emitted from a star such as S0-2 in the nearby area of Sgr A* is no exception to these effects [5]. This redshift is separate from the one caused by orbital motion of a star. In this case of redshift, we have gravitational impact from the black hole. This interaction was actually determined to be caused by a combination of special relativistic and gravitational redshift by Do *et al*. [5]. What this means is that the gravity of the black hole is so strong that it creates the same effect as a radial velocity induced redshift- except many times stronger due to the sheer magnitude of the gravitational force. Gravitational redshift is expressed using the frequencies of the light at the initial position and at infinity. We use this in tandem with an integration from that same initial position to infinity and the expression for Newtonian gravity $g = \frac{GM}{r^2}$. Since we are also considering specifically light beams, we also will need to include a $\frac{1}{c^2}$ term to help us describe the parameters necessary. After this integral, which can be seen in more detail in *An Introduction to Modern Astrophysics*, we are left with the following result to express the gravitational redshift. We will represent the initial frequency as ν_0 and the frequency at infinity as ν_∞. The integral takes place from an initial distance, r_0, to infinity.

$$\frac{\nu_\infty}{\nu_0} = (1 - \frac{2GM}{r_0 c^2})^{1/2} \text{ [7]} \qquad 11$$

2.4 Flux

A new bleeding-edge code written by Roger Hatfull, PhD., *FluxCal* is able to take output files from a collision simulation program such as *StarSmasher* and turn those files into a light profile of the stellar objects. *FluxCal* takes basis from *FluxE*, another program which visualizes the temperature and flux values of stellar mergers from simulations that would be produced by *StarSmasher* [10].

An important term to be familiar with when working with these programs is optical depth. Optical depth is the average depth that you can see into something, in this case a star. It is a term formed around the mean free path of a photon. A single mean free path is the average path length that a photon can enter into a star before deflecting. The optical depth, τ, is directly related to the opacity, κ. This means that when something has a higher opacity and lacks transparency, that object has a smaller optical depth and the average mean free path is smaller. This relationship also includes the density of the object, ρ.

$$d\tau_\lambda = (-\kappa_\lambda \rho)\mathrm{ds} \quad (12)$$

With this equation, we see that the optical depth of a photon of a given wavelength, λ , along a path, *ds*, is defined by the density of the material the photon is travelling into and the opacity of that object for that same given wavelength.

2.4.1 Luminosity

The measure of the total amount of energy emitted from a stellar object over time is called the luminosity. This is different from the apparent brightness we may see from an object and is purely a measure of radiated energy from an object such as a star. It may also be referred to as the intrinsic brightness of an object. In the case of this research, we are mainly concerned with how data from the Keck Observatory and subsequent research compares to what luminosity data we may gather from *FluxCal* using our *StarSmasher* output.

$$L = 4\pi R^2 \sigma T^4 \quad (13)$$

$$F = \sigma T^4 \quad (14)$$

These two equations give us a mathematical representation for the luminosity and the flux respectively. Based on the previous explanation and now these equations, we can break down exactly how flux and luminosity are related. Our luminosity function has a $4\pi R^2$ term in addition to the flux term itself. This makes sense because flux is a term that is given by the luminosity per unit area. Therefore, the luminosity term is the flux - which we can also refer to based on its units of Watts per square meter - over an entire surface. In terms of units, luminosity is the number of Watts that an object produces.

2.4.2 Light Curves

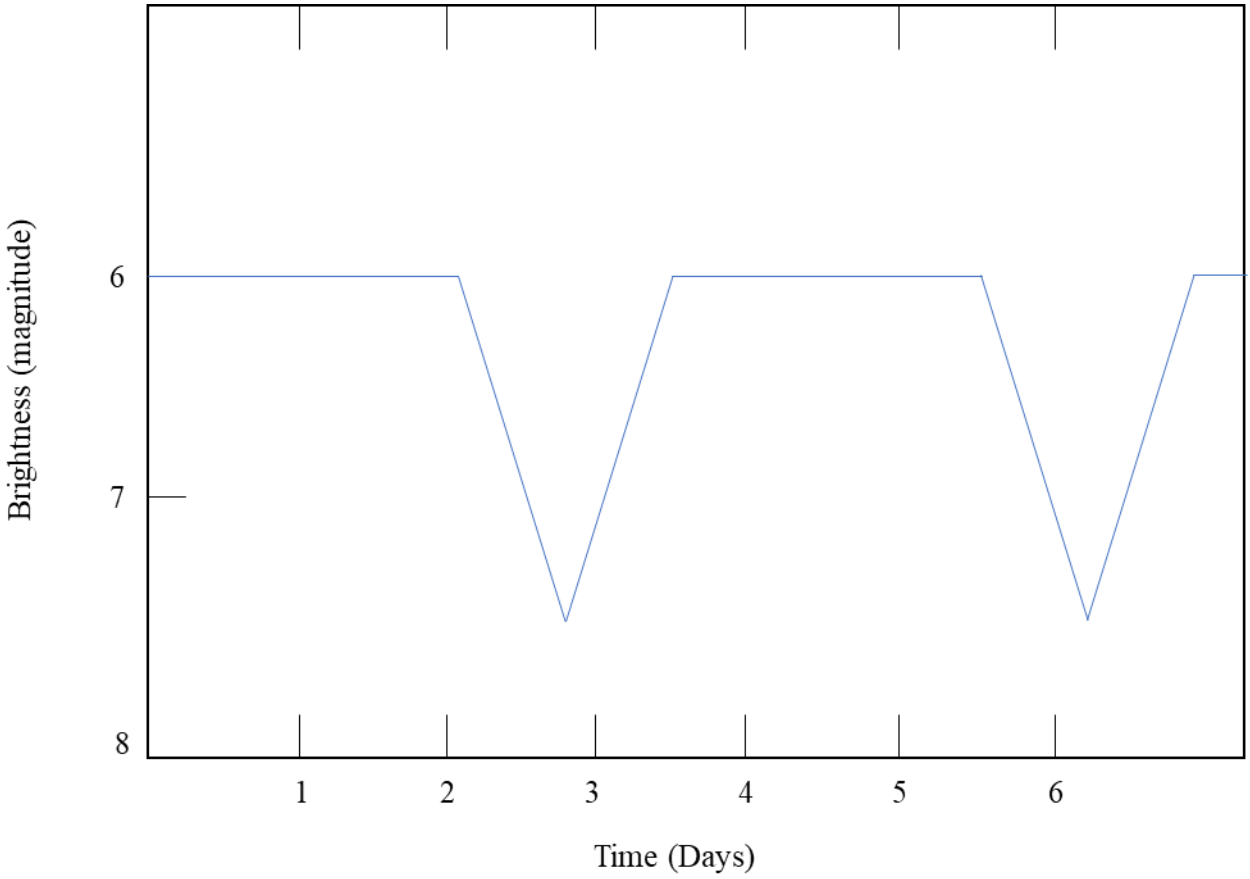

Figure IV: The curve above is a fictional example of what a basic characteristic light curve of a stellar object may look like.

Light curves are visual representations of what the observed light intensity from an object is over time. We can use them to determine how nearby objects interact with each other. In Figure 2.2, we see what an exaggerated light curve would look like. In this figure the brightness decreases by nearly two magnitudes over a period of roughly a day on a regular interval of a few days. This could be attributed to another object passing in front of the six-magnitude object being observed. We can use data from a light curve like this one to understand what the objects we are observing look like.

Chapter 3 - Methodology

3.1 GUI

A Graphical User Interface (GUI) is an interface that uses icons or graphics to allow a user to interact with a program or machine. GUIs allow for easier accessibility to users and provide quicker ways to perform basic and advanced actions on an operating system.

3.1.1 PyQt Interface

Python is a multi-purpose programming language and within it is a toolkit called PyQt. This toolkit is implemented as a plug-in for Python and allows the programmer to create a Graphical User Interface.

3.1.2 JSON Packaging

An abbreviation for JavaScript Object Notation, JSON is a way to easily store and transport data between files. In Appendix A.2 and A.3, the use of JSONs allows for data to be easily stored and printed into text files for use in our *StarSmasher* code.

3.2 Smoothed Particle Hydrodynamics

A method of simulation for fluids, Smoothed Particle Hydrodynamics (SPH) is performed through creation of particles or “blobs of fluid” and using these to simulate larger scale stellar objects. The particles interact with nearby particles and give each other accelerations which in turn change the velocity as well as the position [8].

3.2.1 Traditional Use of *StarSmasher*

StarSmasher is itself an SPH program and is used to take stellar profiles and collide these created stellar objects and collide them with one another. Additions to the original code have been

performed to make it more accurate under specific conditions, such as accounting for electron degeneracy pressure in white dwarfs [8]. It has also been used to interact with other programs to simulate different types of astronomical activity, in particular, the examination of runaway collision objects and globular clusters, visualization of temperature in stellar mergers, and visualization of luminosity of stellar objects [11, 10].

3.2.2 Automating *StarSmasher* and Implementing the GUI

Using JSON and the PyQt GUI creation toolkit, we are able to create a succinct and easy to use user interface that collects collision and stellar profile data for use in *StarSmasher*. The user inputs data into the interface, using JSON this data is then packaged neatly into a format that can be printed into files that can be read by *StarSmasher*, then the code runs the *StarSmasher* program on the newly created files.

3.3 Running the GUI

3.3.1 Initial Conditions

The use of the GUI streamlines the initial condition file creation process and allows the user to focus on what data they wish to manipulate rather than the process of creating the files themselves.

As the code stands currently, Figure III is what is produced by the program and shows all the data necessary for a collision between two stars. There is also already built-in functionality for the collision of a star and a black hole as shown by the 'Object 2 type:' combo box.

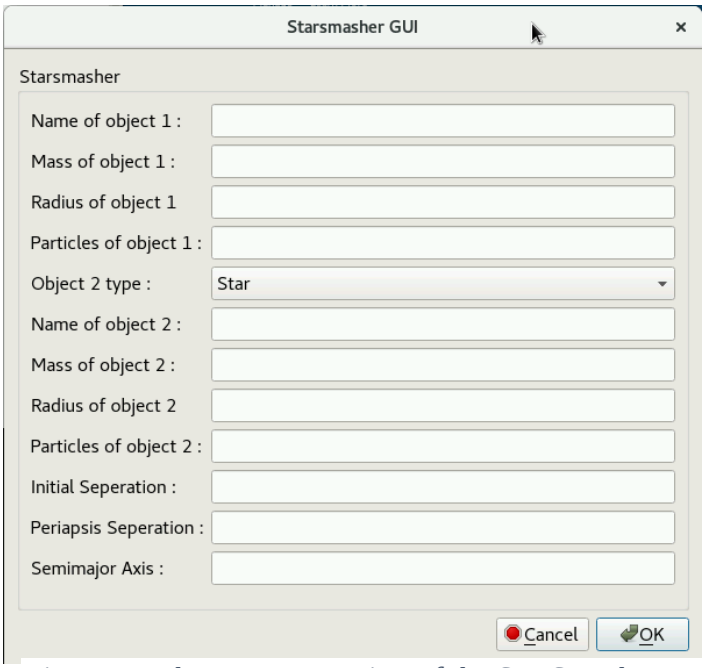

Figure V: The current version of the StarSmasher GUI data collection interface

Chapter 4 - Results and Analysis

4.1 Automation

The process of automating data collection is successful and succinct using the newly made GUI. Following this new data collection and file creation process, we are able to perform our necessary simulations with ease and focus almost entirely on the visual representation and analysis of our results. This data can be piped directly into *FluxCal* from *StarSmasher* and provide us with novel results which we can use as comparison for the real-world analog of the events.

4.1.1 Stellar Relaxation

All files necessary to create stellar profiles were successfully created by the code and organized according to the specifications of the user. The utility and flexibility of the JSONs to be able to create multiple diverse files in customizable and user-friendly locations was an absolute success. The original *StarSmasher* tutorial requires ample time and experience with the terminal environment to successfully create the profiles necessary to even begin to run relaxations. The GUI has successfully been implemented to the point where the file creation process for these stars can now be done nearly instantaneously, whereas the previous completion time to create the stellar profiles and begin relaxations could be anywhere from 10 to 30 minutes depending on the user's level of expertise.

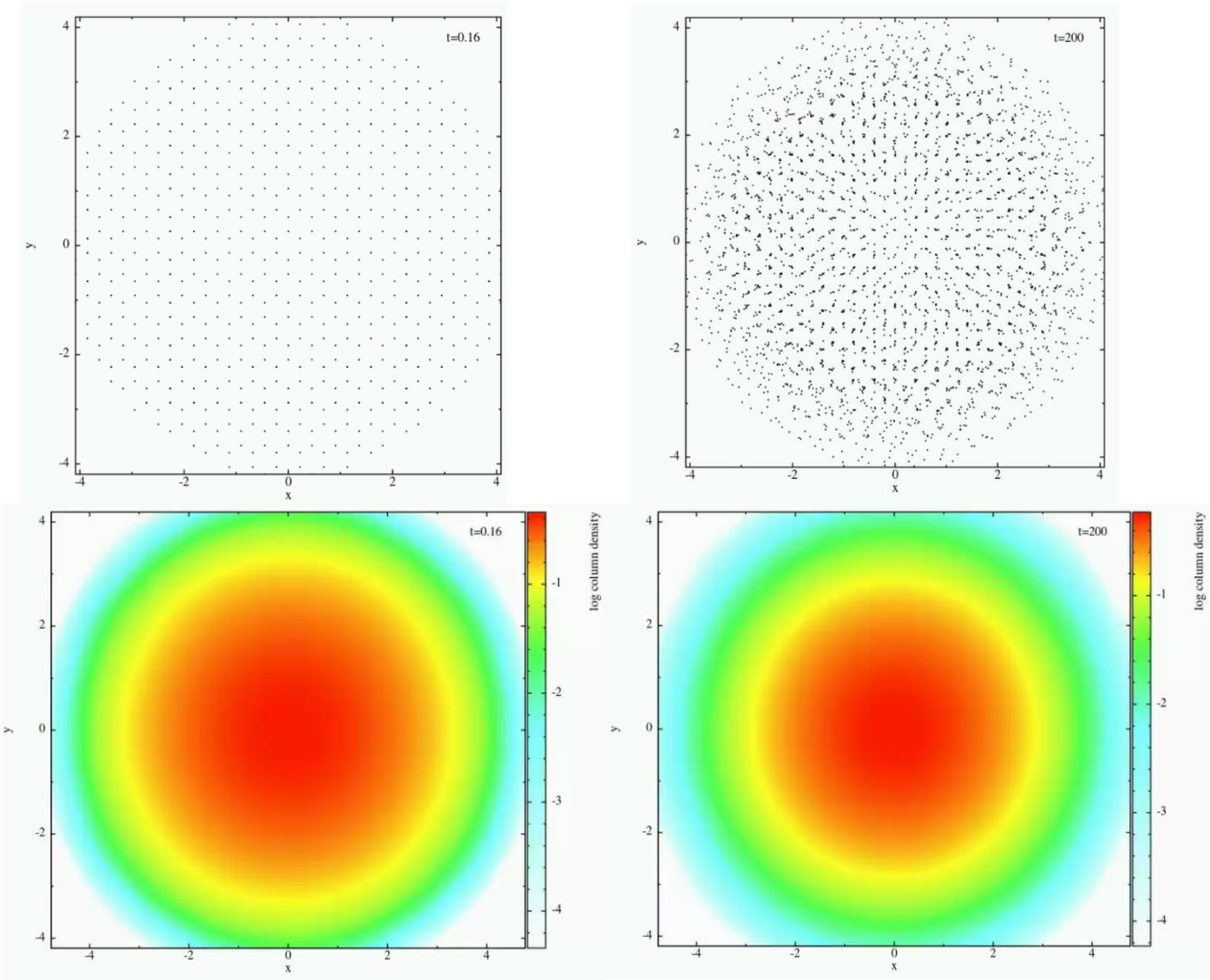

Figure VI: Relaxation results as viewed through a program called JSplash *that helps visualize* StarSmasher *data. The top row shows particle plots in a two-dimensional view of the two stars made by a successful test run of the GUI. Data for star 1 (left column) is Mass (in solar masses) = 10, Radius (in Solar radii) = 5.6, and Particles = 5000. Data for star 2 (right column) is Mass (in solar masses) = 5, Radius (in Solar radii) = 3.8, and Particles = 2500. The bottom row shows a log column density plot of the same stars.*

4.1.2 Stellar Object Collisions

Unfortunately, the code does not yet have processes dedicated to actually performing the collisions with the data provided. However, despite this setback, the code has templates and functionality designed specifically with the creation of the necessary collision folder in mind. The structure and data necessary to be able to run the collisions automatically exists within the classes containing the JSONs for the sph.input and sph.init files.

Chapter 5 - Conclusion

5.1 Future Work

Following the semi-successful creation of a user interface and an automated file creation process, the next steps in this line of research are to run test collisions to ensure the user interface can actually write proper input and produce necessary output. Once the practicality of the GUI is confirmed, it will be beneficial to perform explicit runs on the star S0-2 and Sgr A*. Data from these simulations can then be turned into input for the *FluxCal* code and subsequently be compared to the Keck Observatory data and the light curves of this star's orbit. This automation program can easily be expanded to more quickly and accurately stop collisions if they are not conserving energy. Automating the collision simulation in *StarSmasher* requires only an additional application of the JSONs designated for the sph.input and sph.init files. The data necessary for collisions is already collected by the GUI at the initial stage of the program.

Further continuation possibilities of this research range anywhere from adding additional functionality to simply testing the existing functions to continuing the overarching goal of this research, which is to accurately represent stellar systems like Sgr A* and S0-2. The uses for this code grow as further testing is done to add functionality, accuracy, and ease of use. It is entirely plausible that this code could even be transformed into a mobile application that runs low resolution collisions at fairly high accuracy. The simple concept of making a complex topic easier to access for potential researchers opens the doors to higher scientific understandings of the general population as well as the capability to have more research done quickly and efficiently.

Appendix A - Code

A.1 starsmasher_gui.py

```
from PyQt5.QtWidgets import *
import sys
import os
import json
from input_json import StarSmasherInputFile
from init_json import StarSmasherInitFile
import subprocess
import shutil


class Window(QDialog):
    """ initialization of basic GUI and visual settings of interface """
    def __init__(self, parent=None):
        super(QDialog, self).__init__(parent)
        app.setStyleSheet("* {font-size: 12pt;}")
        self.setWindowTitle("Starsmasher GUI")
        self.setGeometry(200, 200, 600, 800)
        self.guiLayout = QGroupBox("Starsmasher")
        self.objectTwoComboBox = QComboBox()
        self.objectTwoComboBox.addItems(["Star", "Black Hole"])
        self.objectOneName = QLineEdit()
        self.objectTwoName = QLineEdit()
        self.objectOneMass = QLineEdit()
        self.objectTwoMass = QLineEdit()
        self.objectOneRadius = QLineEdit()
        self.objectTwoRadius = QLineEdit()
        self.objectInitialSeperation = QLineEdit()
        self.objectPeriapsisSeperation = QLineEdit()
        self.objectSemimajorAxis = QLineEdit()
        self.objectOneParticles = QLineEdit()
        self.objectTwoParticles = QLineEdit()
        self.createForm()
        self.buttonBox = QDialogButtonBox(
            QDialogButtonBox.Ok | QDialogButtonBox.Cancel)
        self.buttonBox.accepted.connect(self.getInfo)
        self.buttonBox.rejected.connect(self.reject)
        mainLayout = QVBoxLayout()
        mainLayout.addWidget(self.guiLayout)
        mainLayout.addWidget(self.buttonBox)
```

```
        self.setLayout(mainLayout)

    def getInfo(self):
        """Turn all data into strings
        Note that Particles should be an integer, and Mass and Radius should be
        floats, but for simplicity it is easier to have them be strings so they
will print properly"""
        str_objOneN = str(self.objectOneName.text())
        str_objOneM = str(self.objectOneMass.text())
        str_objOneR = str(self.objectOneRadius.text())
        str_objOneP = str(self.objectOneParticles.text())
        str_objTwoN = str(self.objectTwoName.text())
        str_objTwoM = str(self.objectTwoMass.text())
        str_objTwoR = str(self.objectTwoRadius.text())
        str_objTwoP = str(self.objectTwoParticles.text())
        #print(self.objectOneName.text())

        """This method prompts the user to make folders to store
        the stellar profiles, init, and input files in"""
        file_dialog = QFileDialog()
        path1 = file_dialog.getExistingDirectory()
        print(path1)

        str_comboBox = str(self.objectTwoComboBox.currentText())
        if str_comboBox == "Star":
            file_dialog = QFileDialog()
            path2 = file_dialog.getExistingDirectory()
            print(path2)

        """This portion of the method is where the JSON is implemented.
        You can also use a manual list and formatting of the objects created by
the
        GUI data collection."""
        with open(path1 + '/sph.input', 'w') as f:
            obj1 = StarSmasherInputFile(str_objOneN, str_objOneM, str_objOneR,
str_objOneP)
            str_obj1 = str(obj1)
            f.write(str_obj1)
            # shutil.copy2 command relies on the template folder existing where
the GUI is being run from
            # and containing the command_for_running executable.
            shutil.copy2('./template/command_for_running', path1)
            shutil.copy2('./template/test_gpu_sph', path1)
            print('star1 worked')
            with open(path1 + '/sph.init', 'w') as f:
```

```
                obj3 = StarSmasherInitFile("sph.init")
                str_obj3 = str(obj3)
                f.write(str_obj3)

        if str_comboBox == "Star":
            with open(path2 + '/sph.input', 'w') as f:
                obj2 = StarSmasherInputFile(str_objTwoN, str_objTwoM,
str_objTwoR, str_objTwoP)
                str_obj2 = str(obj2)
                f.write(str_obj2)
                shutil.copy2('./template/command_for_running', path2)
                shutil.copy2('./template/test_gpu_sph', path2)
                print('star2 worked')
                with open(path2 + '/sph.init', 'w') as f:
                    obj4 = StarSmasherInitFile("sph.init")
                    str_obj4 = str(obj4)
                    f.write(str_obj4)

        self.close

        '''Command to run relaxations.
        WARNING: DO NOT ALLOW USER INPUT FOR subprocess.run, POSES SIGNIFICANT
SECURITY THREAT IF ALLOWED.'''
        #subprocess.run("cd", shell=True)
        subprocess.run("cd; cd "+path1+"; ./command_for_running", shell=True)
        #subprocess.run("pwd", shell=True)
        #print(path1)
        #print(path2)
        #subprocess.run("./command_for_running", shell=True)
        #subprocess.run("cd", shell=True)
        subprocess.run("cd; cd "+path2+"; ./command_for_running", shell=True)
        #subprocess.run("./command_for_running", shell=True)

        """The next part of the implementation process is to use the
self.objectInitialSeperation,
        self.objectPeriapsisSeperation, and self.objectSemimajorAxis data in a
3rd and final JSON in order to create the
        sph.input file for the collision. Unfortunately this is currently out of
the scope of this project."""


        """This commented out portion can be used to create manual text input
into the
        sph.input and sph.init files. You can also simply edit the JSON and what
it
```

```
        prints to keep the code less cluttered."""
        #with open(path + '/sph.input', 'w') as f:
        #    f.writelines('&Test\n'+'JSON or just manual list\n'+'&end')
        #    print("this step worked")
        #with open(path + '/sph.init', 'w') as f:
        #    f.writelines('&Test\n'+'JSON or just manual list\n'+'&end')
        #    print("step 2 worked")
        #self.close()

    # Labels for data entries
    def createForm(self):
        """This method adds all widgets to the form"""
        layout = QFormLayout()
        layout.addRow(QLabel("Name of object 1 :"), self.objectOneName)
        layout.addRow(QLabel("Mass of object 1 :"), self.objectOneMass)
        layout.addRow(QLabel("Radius of object 1"), self.objectOneRadius)
        layout.addRow(QLabel("Particles of object 1 :"),
                      self.objectOneParticles)
        layout.addRow(QLabel("Object 2 type :"), self.objectTwoComboBox)
        layout.addRow(QLabel("Name of object 2 :"), self.objectTwoName)
        layout.addRow(QLabel("Mass of object 2 :"), self.objectTwoMass)
        layout.addRow(QLabel("Radius of object 2"), self.objectTwoRadius)
        layout.addRow(QLabel("Particles of object 2 :"),
                      self.objectTwoParticles)
        layout.addRow(QLabel("Initial Seperation :"),
                      self.objectInitialSeperation)
        layout.addRow(QLabel("Periapsis Seperation :"),
                      self.objectPeriapsisSeperation)
        layout.addRow(QLabel("Semimajor Axis :"), self.objectSemimajorAxis)
        self.guiLayout.setLayout(layout)

if __name__ == '__main__':
    """This method is necessary to open the GUI window and actually run the
program"""
    app = QApplication(sys.argv)
    window = Window()
    window.show()
    sys.exit(app.exec_())
```

A.2 input_json.py

```
import os

class StarSmasherInputFile:

    def __init__(self, f, m, r, n):

        # Members
        self.__filename = f
        self.__mass_of_obj = m
        self.__radius_of_obj = r
        self.__particles_of_obj = n
        '''comments in this function show the other variables that an additional
        JSON will store and wrtie to create the collision sph.init file.'''
        #self.__init_sep = i
        #self.__peri_sep = p
        #self.__semi_axis = s

        self.__json_rep = self.__export_to_JSON()

        # Print Assistance: allows data to be printed
        self.__lookup_table = {
        " STARMASS": "Mass of Object 1",
        " STARRADIUS": "Radius of Object 1",
        " n": "Particle Count of Object 1",
        }
        #" init0": "Initial Speration",
        #" peri": "Periapsis Seperation",
        #" semi": "Semimajor Axis",

    # sph.input file
    def __repr__(self):
        to_file_str = " &INPUT"
        to_file_str += "\n TF=200, ! desired final time to stop simulation"
```

```
        to_file_str += "\n DTOUT=1, ! how often an out*.sph files should be
dumped"
        to_file_str += "\n NNOPT=65, ! controls neighbor number.  Leave it at 22
to get almost 40 neighbors."
        to_file_str += "\n NRELAX=1, ! Relaxation flag.  0=dynamical calculation,
1=relaxation of single star, 2=relaxation of binary in corotating frame with
centrifugal force, 3=calculation in corotating frame with centrifugal and
Coriolis forces"
        to_file_str += "\n hfloor=0.015d0,"
        to_file_str += "\n EQUALMASS=0d0, ! particle mass is proportional to
rho^(1-EQUALMASS), so EQUALMASS=1 has equal mass particles and EQUALMASS=0 is for
constant number density."
        to_file_str += "\n NINTVAR=2, ! 1=integrate entropic variable A,
2=integrate internal energy u"
        to_file_str += "\n ngravprocs=1, ! The number of gravity processors (must
be <= min(nprocs,ngravprocsmax))"
        to_file_str += "\n NEOS = 0, ! 0 = polytope, 1 = ideal + rad, 2 = .eos"
        to_file_str += "\n! The Courant numbers cn1, cn2, cn3, and cn4 are for
SPH particles:"
        to_file_str += "\n! dt_sph=1/(1/dt1 + 1/dt2 + 1/dt3 + 1/dt4)"
        to_file_str += "\n cn1=0.2d0,   ! dt1=cn1*h/v_signal"
        to_file_str += "\n cn2=1.d30, ! dt2=cn2*(h/|a-a_smoothed|)^0.5"
        to_file_str += "\n cn3=0.03d0, ! dt3=cn3*u/|du/dt|"
        to_file_str += "\n cn4=1.d30,  ! dt4=cn4*v_signal/|a-a_smoothed|"
        to_file_str += "\n! The Courant numbers cn5, cn6, and cn7 are for a
particle i that is a compact object (co):"
        to_file_str += "\n! dt_co=1/(1/dt5 + 1/dt6)"
        to_file_str += "\n cn5=0.05d0,! dt5=cn5*r_ij/v_ij (minimized over all
other particles j)"
        to_file_str += "\n cn6=0.05d0, ! dt6=cn6*(r_ij/a_ij)^.5 (minimized over
all other particles j)"
        to_file_str += "\n cn7=1.d0,   !
r_ij=(x_ij^2+y_ij^2+z_ij^2+cn7*h_i^2)^.5"
        to_file_str += "\n! The final timestep dt is the minimum of dt_sph and
dt_co for all particles i"
```

```
        to_file_str += "\n nkernel=2"
        to_file_str += "\n TRELOFF=1000"
        for entry in self.__json_rep:
            to_file_str += f"\n{entry}={self.__json_rep[entry]},"
        to_file_str += "\n &END"
        return to_file_str

    def __export_to_JSON(self):
        return {
        " STARMASS": self.__mass_of_obj,
        " STARRADIUS": self.__radius_of_obj,
        " n": self.__particles_of_obj,
        }

    def get_JSON(self):
        return self.__json_rep

    def get_lookup(self):
        return self.__lookup_table

if __name__ == "__main__":
    """The following are entries within sph files. These act as default
    values unless the user overwites them or changes them for a given simulation.
"""
    paramlist = {
    "E1": {"entry": "sph.input",
    "m": 2,
    "r": 3,
    "n": 1000,
    }
    }

    for entry_section in paramlist:
        entry = paramlist[entry_section]
        obj = StarSmasherInputFile(
```

```
        entry["entry"],
        entry["m"],
        entry["r"],
        entry["n"],
        )

        with open(os.getcwd() + f"/{entry['entry']}", "w") as f:
            f.write(repr(obj))

    """This object declaration shows how the JSON prints text to the terminal
    so the user can view what the input was versus the text within the sph
files"""
    # obj = StarSmasherInputFile("Blyat", 0.2, 0.5, 1)
    # json_rep = obj.get_JSON()
    # lt = obj.get_lookup()
    # print("Print Rep:\n")
    # for entry in json_rep:
    #     print(f"{lt[entry]}: {json_rep[entry]}")
    # print("\nText Rep:\n")
    # print(obj)

    '''defunct testing'''
    # print(f"\nEZ Print:\n{obj}\n")
    # print(type(repr(obj)))
    # print(type(obj))
    # with open(os.getcwd() + "/exf.txt", "w") as f:
    #     f.write(repr(obj))
    # params = StarSmasherParams(0.2, 0.5)
    # new_obj = V2StarSmasherInputFile("Test", params)
    # defunct testing
    # to_file_str = "&input"
    # for entry in json_rep:
    #     to_file_str += f"\n{entry}={json_rep[entry]},"
    # to_file_str += "\n&end"
    # print(to_file_str)
```

A.3 init_json.py

```
import os

"""This class allows a preset sph.init file to be created for relaxations of the
stars.
It produces a single output but can be manipulated very easily to create a
sph.init file
for collisions """
class StarSmasherInitFile:

    def __init__(self, f):

        # Members
        self.__filename = f
        self.__json_rep = self.__export_to_JSON()
        self.__lookup_table = {}

    def __repr__(self):
        to_file_str = " &INITT"
        to_file_str += "\n INAME='1es'"
        to_file_str += "\n &END"
        return to_file_str

    def __export_to_JSON(self):
        return {}

    def get_JSON(self):
        return self.__json_rep

    def get_lookup(self):
        return self.__lookup_table

if __name__ == "__main__":
    """The following are entries within sph files. These act as default
```

```
    values unless the user overwites them or changes them for a given simulation.
"""
    paramlist = {}

    for entry_section in paramlist:
        entry = paramlist[entry_section]
        obj = StarSmasherInitFile(
        entry["entry"]
        )

        with open(os.getcwd() + f"/{entry['entry']}", "w") as f:
            f.write(repr(obj))


    """This object declaration shows how the JSON prints text to the terminal
    so the user can view what the input was versus the text within the sph
files"""

    # obj = StarSmasherInitFile("Blyat")

    # json_rep = obj.get_JSON()
    # lt = obj.get_lookup()

    # print("Print Rep:\n")

    # for entry in json_rep:
    #     print(f"{lt[entry]}: {json_rep[entry]}")

    # print("\nText Rep:\n")

    # print(obj)
```

A.4 How to use the GUI

To start the GUI, simply copy the three python files listed in appendices one thru three and the template folder into your preferred working folder or location on your device. Once all the program files are set, simply type ‘python3 starsmasher_gui.py’ or ‘python starsmasher_gui.py’ depending on if you are working from a version of python other than 3.X.X or working within the 3.X.X release of python, respectively, into the command line from wherever you have placed the files. Next, make sure to enter the appropriate data for your collision and then click ‘ok’ on the GUI terminal once you have finished entering the data. You will then be prompted to pick folders in which to place your created objects. If you have a second star, then you will be prompted a second time. Once you get confirmation from the terminal that your objects have been created - denoted by a phrase akin to ‘star1 worked’ - you may exit out of the GUI and go observe your data in the respective folder locations using ‘jsplash out*.sph’.

Works Cited

[1] F. Eisenhauer, R. Genzel, T. Alexander, R. Abuter, T. Paumard, T. Ott, A. Gilbert, S. Gillessen, M. Horrobin, S. Trippe, H. Bonnet, C. Dumas, N. Hubin, A. Kaufer, M. Kissler-Patig, G. Monnet, S. Ströbele, T. Szeifert, A. Eckart, R. Schödel and S. Zucker, "SINFONI in the Galactic Center: Young Stars and Infrared Flares in the Central Light-Month*," *The Astrophyscial Journal,* 2005.

[2] L. Meyer, A. M. Ghez, R. Schödel, S. Yelda, A. Boehle, J. R. Lu, T. Do, M. R. Morris, E. E. Becklin and K. Matthews, "The Shortest-Known–Period Star Orbiting Our Galaxy's Supermassive Black Hole," *Science,* pp. 84-87, 2012.

[3] I. Consulting, "Keck I and Keck II telescopes," 2022. [Online]. Available: https://keckobservatory.org/about/telescopes-instrumentation/.

[4] T. D. A. H. A. G. S. N. G. W. S. S. S. C. A. K. G. J. R. L. a. K. M. Devin S. Chu, "Investigating the Binarity of S0-2: Implications for Its Origins and Robustness as a Probe of the Laws of Gravity around a Supermassive Black Hole," *The Astrophysical Journal,* 2018.

[5] T. Do, A. Hees, A. Ghez, G. D. Martinez, D. S. Chu, S. Jia, S. Sakai, J. R. Lu, A. K. Gautam, K. Kosmo O'Neil, E. E. Becklin, M. R. Morris, K. Matthews, S. Nishiyama, R. Campbell, Z. Chen, A. Ciurlo, A. Dehghanfar, E. Gallego-Cano, W. E. Kerzendorf, J. E. Lyke, S. Naoz, H. Saida, R. Schödel, M. Takahashi, Y. Takamori, G. Witzel and P. Wizinowich, "Relativistic redshift of the star S0-2 orbiting the Galactic Center supermassive black hole," *Science,* pp. 664-668, 2019.

[6] S. G. F. M. F. E. P. M. P. O. E. G. J. D. I. W. T. O. S. F. M. B. A. J.-R. R. G. M. Habibi, "Twelve years of spectroscopic monitoring in the Galactic Center: the closest look at S-stars near the black hole," *The Astrophysical Journal,* 2017.

[7] B. W. &. O. D. A. Carroll, An Introduction to Modern Astrophysics, New York City: Cambridge University Press, 2017.

[8] M. Lachat, *Realistic Representation of Middle-mass White Dwarfs and Collisions with Black Holes,* 2020.

[9] S. N. S. M. Kastytis Zubovas, "Sgr A* flares: tidal disruption of asteroids and planets?," *Monthly Notices of the Royal Astronomical Society,* vol. 421, no. 2, pp. 1315-1324, 2012.

[10] J. E. Wenskovitch, J. C. Lombardi and R. W. Hatfull, "FluxE: exploring flux in astrophysical simulations," *Association for Computing Machinery,* November 2016.

[11] D. Mukherjee, *FORMATION OF RUNAWAY COLLISION OBJECTS Or, How I Learnt To Stop Worrying and Love my Comp,* 2019.