BINARY STAR MODELING: A COMPUTATIONAL APPROACH

Author:
Daniel Silano

Faculty Sponsor:
R. J. Pfeiffer,
Department of Physics

ABSTRACT

This paper illustrates the equations and computational logic involved in writing *BinaryFactory*, a program I developed in Spring 2011 in collaboration with Dr. R. J. Pfeiffer, professor of physics at The College of New Jersey. This paper outlines computational methods required to design a computer model which can show an animation and generate an accurate light curve of an eclipsing binary star system. The final result is a light curve fit to any star system using *BinaryFactory*. An example is given for the eclipsing binary star system TU Muscae. Good agreement with observational data was obtained using parameters obtained from literature published by others.

INTRODUCTION

This project started as a proposal for a simple animation of two stars orbiting one another in C++. I found that although there was software that generated simple animations of binary star orbits and generated light curves, the commercial software was prohibitively expensive or not very user friendly. As I progressed from solving the orbits to generating the Roche surface to generating a light curve, I learned much about computational physics. There were many trials along the way; this paper aims to explain to the reader how a computational model of binary stars is made, as well as how to avoid pitfalls I encountered while writing *BinaryFactory*.

Binary Factory was written in C++ using the free C++ libraries, OpenGL, GLUT, and GLUI. A basis for writing a model similar to *BinaryFactory* in any language will be presented, with a light curve fit for the eclipsing binary star system TU Muscae in the final secion. Any diagrams not obviously drawn in MSPaint were taken as generated by *BinaryFactory* or MSExcel as a screenshot.

COMPUTING THE ORBIT

The first step in displaying the stars is determining the positions of the stars along an orbit. For simplicity, the stars are assumed to be just point masses orbiting each other. Given an initial position $\boldsymbol{S_{initial}}$, a velocity $\boldsymbol{v}$, and an acceleration $\boldsymbol{a}$, the equation for the position of any given star after a time interval t can be written as follows (bold face means vector):

$$\boldsymbol{S_{final}} = \boldsymbol{a}t^2 + \boldsymbol{v}t + \boldsymbol{S_{initial}} \qquad (2-1)$$

Here, the initial positions of the stars are a function of the mass ratio and are due to the variable location of the center of mass. Every time the user changes the mass ratio, a function is called that determines the new positions. This function initially sets the position of the less massive star at (0.5, 0, 0), and the more massive star at (-0.5, 0, 0). This makes the distance between them always 1, and helps a great deal in reducing computing times later on. Now, the x position of the center of mass is given by:

$$\frac{\sum_{i=1}^{2} m_i x_i}{\sum_{i=1}^{2} m_i} = \frac{(-0.5 * m_1 + 0.5 * m_2)}{(m_1 + m_2)} \qquad (2-2)$$

In this equation, and otherwise in this paper, it is assumed that mass 2 $\leq$ mass 1. Now, the position of each star is adjusted corresponding to the result of the position of the center of mass. The center of mass is always centered at the origin to provide consistency in viewing orbits. In this case, the center of mass will always be ≤ 0 on the x-axis because $m_1 \geq m_2$. Because of this, the center of mass

needs to be moved to the origin so the center of the orbit is always in the middle of the screen. This is accomplished by shifting the initial positions of the stars to their new corrected position. The position of the less massive star should be farther away from the origin, and the more massive star should be closer. Because the center of mass x position is ≤ 0, the new coordinates are given by the equations below (both have x and y coordinate of 0).

$$S_{1x} = -0.5 - CenterOfMass_x \quad (2-3)$$

$$S_{2x} = 0.5 - CenterOfMass_x \quad (2-4)$$

Given the initial position it is now necessary to find the initial velocity of the star.

$$|\boldsymbol{v}_{\boldsymbol{initial(1)}}| = \sqrt{\frac{Gm_2R(1+\epsilon)}{r^2}} \quad (2-5)$$

In equation (2-5), R is equal to the distance of star 1 to the center of mass, r is equal to the distance between stars, and ϵ is a parameter describing the eccentricity of the orbit which ranges from 0 to 1. It is important to note that the velocity reduces to that of uniform circular motion when the eccentricity parameter is equal to 0 (the least eccentric orbit). This equation is applied to both stars, as they both have the same eccentricity. However the mass in the equation and the distance to the center of mass must be changed. Because the stars are assumed to be at periastron here, the only component of the velocity is tangential and in the y-direction. The directions are arbitrary, but here I chose the velocity of the less massive star to be in the positive y-direction, and the more massive star to be in the negative y-direction.

$$\boldsymbol{v}_{\boldsymbol{initial(1)}} = \begin{pmatrix} 0 \\ -|\boldsymbol{v}_{\boldsymbol{initial(1)}}| \\ 0 \end{pmatrix}, \quad \boldsymbol{v}_{\boldsymbol{initial(2)}} = \begin{pmatrix} 0 \\ |\boldsymbol{v}_{\boldsymbol{initial(2)}}| \\ 0 \end{pmatrix} \quad (2-6)$$

To find the acceleration, the equation to be used is the same from Newton's theory of gravitation.

$$|\boldsymbol{a}_1| = \frac{F}{m_1} = \frac{Gm_2}{r^2} \quad (2-7)$$

In (2-7), r is equal to the distance between the stars at a given point. Finding a unit vector $\widehat{\boldsymbol{u}_1}$ from the least massive star (1) to the more massive star (2) is found using (2-8) below.

$$\widehat{\boldsymbol{u}_1} = \frac{\boldsymbol{S}_2 - \boldsymbol{S}_1}{|\boldsymbol{S}_2 - \boldsymbol{S}_1|} \quad (2-8)$$

$$\boldsymbol{a}_1 = |\boldsymbol{a}_1| * \widehat{\boldsymbol{u}_1} \quad (2-9)$$

Equation (2-9) can be expressed for star 2 by switching the masses in (2-7) and the indices in (2-8) and using the same process as above. However, in this case the acceleration and velocity are changing with time/position. For this reason, it is necessary to shrink the time interval as small as possible and iterate this process in order to obtain an orbit that is close to theoretically perfect. So after iterating once and obtaining some new position after a small time interval, the equations for the new velocity for the less massive star (1) is as follows.

$$\boldsymbol{v}_1 = \boldsymbol{a}_{1(current\ frame)} * t + \boldsymbol{v}_{1(previous\ frame)} \quad (2-10)$$

However, the $\boldsymbol{a}_1$ here is NOT the acceleration from the previous frame. Before the stars are repositioned one must first use equation (2-7) to find the acceleration, then (2-10) to find the new velocity, and lastly (2-1) to find the new position. Equation (2-10) can also be generalized for star 2, just switch the indices of star 1 and star 2. The speed of the program can be altered by choosing how many iterations of this positioning sequence are computed before each frame is displayed to the screen (less iterations between each displayed frame results in a slower animation).

GENERATING THE STAR SURFACES

The first assumption to be made in this model of binary star systems is that both stars are in hydrostatic equilibrium. When this is true, the internal gas pressure of a star balances its own gravity and results in a spherical shape. However, when another star orbits very closely to the given star, an additional gravitational force acts to pull matter on the near side resulting in what is known as a Roche surface.

In order to solve this problem, we must invoke Newton's Second Law and use an altered coordinate system. In this system, the star for which we want to find the shape is placed at the origin with the positive x-axis going through the center of the second star. The goal is to find the radius of the star centered about the origin at different values of theta and phi, the angles of a spherical coordinate system. Figure 1 illustrates the gravitational forces and pressure force on a point on the surface of a star.

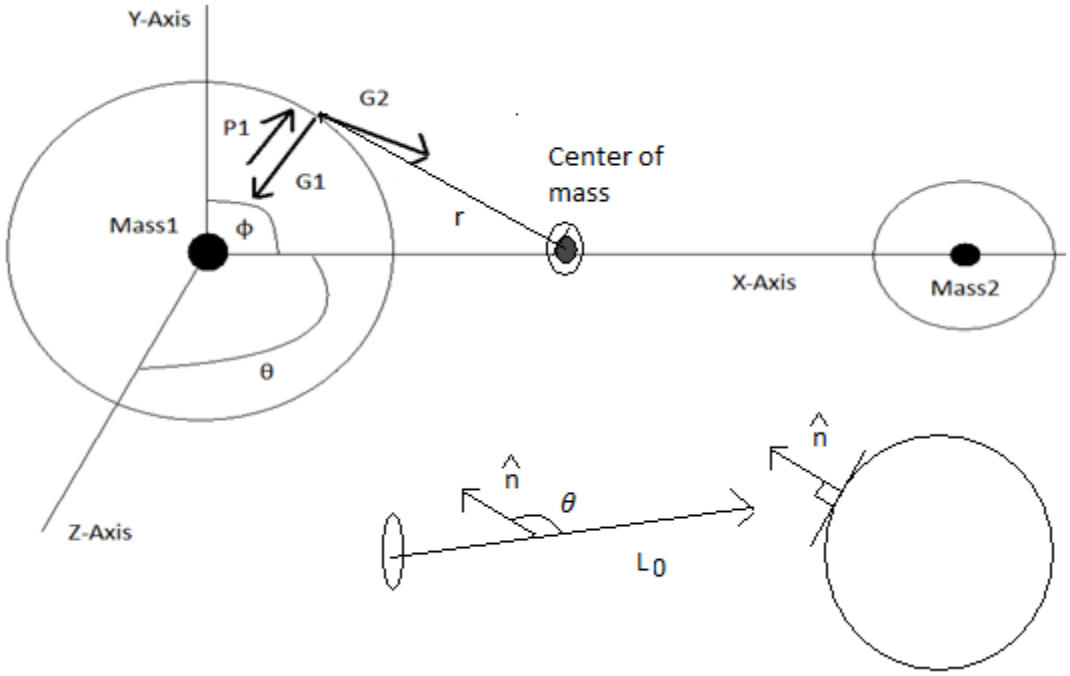

Figure 1 – Upper panel: coordinate system used to find the radius of the points on the Roche surface. Lower Panel: the normal vector to a point on a stellar surface from a line of sight to an observer

The equations below form a theoretical basis for finding the radius of the target star as a function of the spherical angles, theta and phi.

$$\boldsymbol{F}_{\boldsymbol{gravity}} + \boldsymbol{F}_{\boldsymbol{pressure}} = -m\omega^2\boldsymbol{r} \qquad (3-1)$$

In (3-1), the vector $\boldsymbol{r}$ is a vector relative to the center of mass and rotational axis (the stars rotate around the z-axis) that points towards a mass element on the star surface. We can define the pressure force, $\boldsymbol{F}_{\boldsymbol{p}} = -(V * \nabla P)$, where V is the volume of the star and ∇P is the gradient of the pressure. Now, we can define a potential Ψ, where $\nabla P = -\rho\nabla\Psi$ and ρ is the density of a star. The resulting force vector for the pressure is $\boldsymbol{F}_{\boldsymbol{pressure}} = V * \rho * \nabla\Psi = \ m\nabla\Psi$.

$$\boldsymbol{F}_{\boldsymbol{gravity}} + m\omega^2\boldsymbol{r} = -m\nabla\Psi \qquad (3-2)$$

From Kepler's 3rd Law we know that $\omega^2 = \frac{GM}{R^3} = \frac{G(m_1+m_2)}{R^3}$, where R is the distance between the mass element and the center of the other star. So, the resulting function of the potential is in equation (3-3) below.

$$\boldsymbol{F}_{\boldsymbol{gravity}} + \frac{Gm(m_1+m_2)}{R^3}\boldsymbol{r} = -m\nabla\Psi \qquad (3-3)$$

Once we have found the force and acceleration as a function of the potential, we can assume that the potential is the same at all points on the surface. This results in an equipotential surface. An analogy of this is that having a surface with an equal potential means that if the potential was electric in nature, any charge placed on the surface would not move. Here, if a small mass was placed at any point on the surface, it would not move. Solving for the forces in terms of components and converting to a spherical coordinate system leads to the equation below which was obtained from a paper by D. Bruton (2004):

$$\frac{1}{\tilde{r}_1} = \frac{1}{\tilde{r}_{pole(1)}} + q_1\left(\frac{1}{\sqrt{\tilde{r}_{pole(1)}^2}}\right) - q_1\left(\frac{1}{\sqrt{\tilde{r}_1^2 - 2\tilde{r}_1\lambda + 1}} - \tilde{r}_1\lambda\right) - \frac{1}{2}(1+q_1)(1-\upsilon^2)\tilde{r}_1^2 \quad (3-4)$$

Here, $\tilde{r}_1 = \frac{radius}{distance\ between\ stars}$, $\tilde{r}_{pole(1)} = \frac{radius\ of\ point\ touching \pm y-axis}{distance\ between\ stars}$, $q_1 = \frac{m_2}{m_1}$ (the mass ratio), $\lambda = \sin\theta\cos\phi$, and $\upsilon = \cos\phi$.

It is clear that this is not a function explicitly in terms of $\tilde{r}_1$, which is what is desired. However, this function may be iterated to find a value for f $\tilde{r}_1$. It has been shown that the convergence occurs after roughly 25 iterations. This process is done by using an initial value for $\tilde{r}_1$ on the right side that is the same as the pole radius ($\tilde{r}_{pole(1)}$). The user should input a pole radius, as this radius is invariant under this process and will remain constant throughout the orbit. After iterating once and finding a value for $\tilde{r}_1$, the value obtained can then be used again on the right side of the equation and iterated to find a second value for $\tilde{r}_1$. This process can be repeated multiple times as a program goes in steps of θ and ϕ to find the radius of any point on the surface.

To do this for the second star, the positions must be adjusted so now the second star is at the origin and the first is along the positive x-axis. There is a second value for the pole radius of the second star ($\tilde{r}_{pole(2)}$) that should also be inputted by the user for the same reasons as for star 1. The only value that must be changed is the mass ratio q_2 for which $q_2 = \frac{m_1}{m_2}$. This is an important change and will greatly affect the outcome of this algorithm if the mass ratio that is being used is not correct.

It is also recommended that the coordinates of these points as well as the coordinates of the centers of each star on the x-y plane (the orbiting plane) are stored in a list or array to be used later.

DISPLAYING THE STARS

The first step in displaying the star is determining which part of the surface is actually visible to an observer. In order to do this, it is necessary to find a vector that is normal to the star surface at a given point. Considering that the star is represented by points (one can use a sinusoidal distribution for the number of latitude points to approximate an equal area representation for each point), one must consider all points that have an angle of $\measuredangle$ 90 degrees to be out of view from a potential observer (see fig. 4a).

There are generally two different methods of accomplishing this goal. The first is to assume that the object is far enough away that the lines of sight connected from an observer to any point on the star form parallel lines. In this case, a vector from the line of sight to the center of the star is considered to be the same for any point on the star. An alternative approach is to place the coordinates of an observer considerably far away, and to actually calculate a vector to each point on the star. Either approach will have the same result, as long as the observer is placed sufficiently far away in the second method.

The method for finding the angle between a vector normal to a point where $\hat{\boldsymbol{n}}$ is the normal vector and $\boldsymbol{L_0}$ is a vector pointing from the observer to a point (line of sight) should be familiar to most, but is shown below in equation (4-1).

$$\theta = \cos^{-1}\left(\frac{\hat{\boldsymbol{n}} \cdot \boldsymbol{L_0}}{\|\hat{\boldsymbol{n}}\| \|\boldsymbol{L_0}\|}\right) \qquad (4-1)$$

However, the vector normal to the surface of a roche lobe is not equivalent to the normal to the surface of a spherical object. In the case of modeling binary stars, the x,y, and z components of the pressure provide the normal vector to the surface. These can be found in Kallrath & Milone (2009) and are tabulated below. It is also assumed that the star configurations are that of what is shown in Figure 1, where the target star should always be at the origin and the other star along the positive x-axis.

$$n_x = -\left(\frac{-x}{r^3} + \frac{q(d-x)}{\tilde{r}^3} + (q+1)x - \frac{q}{d^2}\right) \qquad (4-2)$$

$$n_y = -\left(-y\left[\frac{1}{r^3} + \frac{q}{\tilde{r}^3} - (q+1)\right]\right) \qquad (4-3)$$

$$n_z = -\left(-z\left[\frac{1}{r^3} + \frac{q}{\tilde{r}^3}\right]\right) \qquad (4-4)$$

In these equations, $\hat{\boldsymbol{n}} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$, d is the distance between star centers, $r = \sqrt{x^2 + y^2 + z^2}$, $\tilde{r} = \sqrt{(d-x)^2 + y^2 + z^2}$, and (x, y, z) are the coordinates of the target point within the coordinate system of Figure 1. These equations all have the extra negative sign to make sure that the normal vector components all point outwards from the star surface.

Now that the points that are in front to the observer are the only points being considered to be displayed, the next step is to eliminate points that are being eclipsed. One simple method is to determine which star is in front, and then cross-check all of the points in the star that is behind to see whether they are behind any points from the front star. A very convenient way of accomplishing this is to draw a vector from the observer to a point on the star that is being eclipsed, as well as a vector to a test point on the star that is in front, and convert them both to unit vectors of length 1. Once this is done, one can see whether the x, y, and z components of the unit vector from the eclipsed star point are within a certain range of the x, y, and z components of the unit vector to the star in front. This process needs to be repeated many times, checking each point on the star in back with each point on the star in front. One can adjust range of values $\pm$ x, $\pm$ y, and $\pm$ z that are allowed to eliminate a bigger or smaller area of points in the star that is being eclipsed.

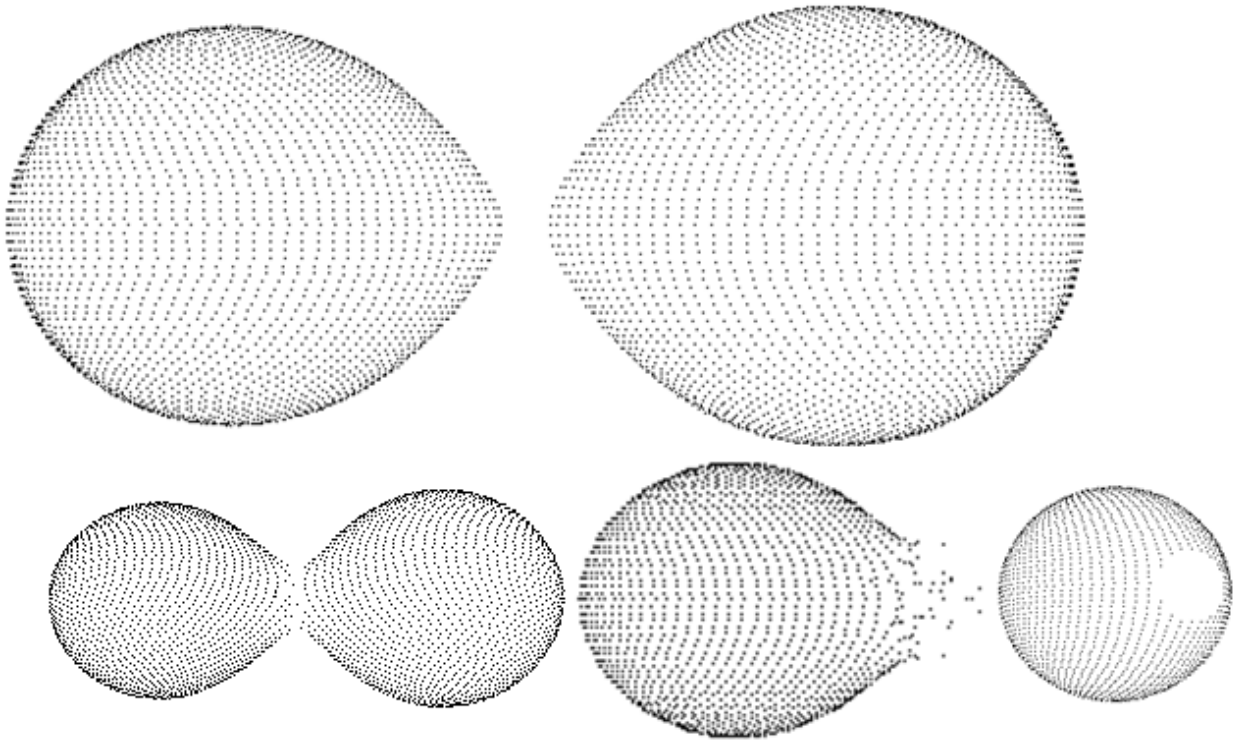

Figure 2 – From left to right starting from the top: Two stars with Roche surfaces, two stars with overfilling lobes and a contact cylinder between them, a star with an overfilled lobe, and a star with a hole

The last consideration in displaying the stars is how to deal with situations where the Roche lobe of a star is overfilled (see Figure 2). This happens when the radius of the star becomes too large and overextends the lobe. In this case, the solution presented in equation (3-4) breaks down.

This presents an interesting problem when dealing with very closely orbiting binary star systems where the lobes of the individual stars are slightly overfilled but form a contact cylinder between them. In this case, the stars share a communal surface between them, and the total surface area of the system decreases.

Figure 2 shows two stars sharing a contact cylinder. There is a slight gap in the stars because their lobes have overfilled and the density of points decreases rapidly at the end of the lobe. One important aspect of this contact cylinder is that it has no points inside of it. A simple way of accomplishing this effect is to have a test during the generation of the radius of a point that results in any point over a certain radius not being displayed. This maximum radius should be customizable to fit the sizes of the stars given, and the total of the maximum radii for the stars should not exceed their separation. This will result in a 'hole' in the star as in Figure 2, but when two stars with this effect are joined together, their holes are filled by the contact cylinder.

GENERATING A LIGHT CURVE

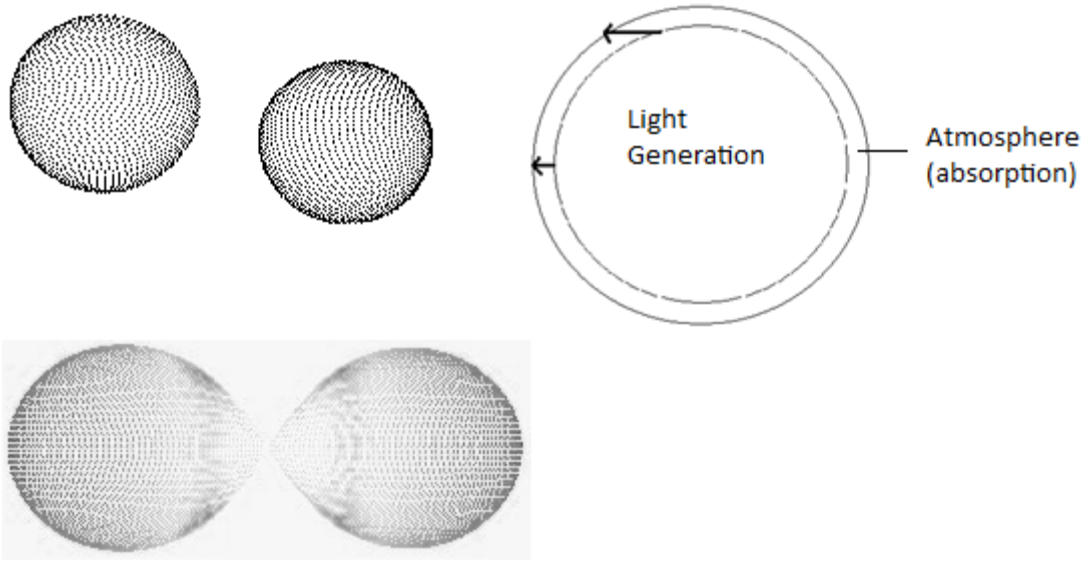

Figure 3 – From left to right: a: Limb darkening effects, b: an explanatory diagram, and c: the reflection effect shown on the bottom

The ultimate goal of any model of a binary star system is to generate a light curve as seen above. Observational data from binary star systems in a given wavelength range can be fitted using a light curve and can provide insightful information about the two orbiting stars. Because stars in this computational model are represented by points, the focus of this section will be determining how much light is generated from each point, and how much light is generated in total by the system at a given time.

The phase along the x-axis of Figure 5 is just a number representing the stage of orbit that the stars are in. A phase of 0 always corresponds to the dimmer star eclipsing the brighter star (the deeper eclipse on the light curve). The phase is just the current time in the orbit (0 starting at the primary eclipse) divided by the total amount of time for one revolution. It is often necessary with eccentric orbits to compute this time quickly, before displaying the stars, using the equations from part 2 by checking for the time when the stars complete a full orbit.

The first step in the light curve generation process is finding out how much light from a given star at a surface temperature T is represented by each point on the star surface. Assuming the star radiates as a black body, Planck's Law can be used to find the black body intensity of a star in a given wavelength range. The equation below is from online Astrophysics notes from R.J. Pfeiffer.

$$B = \sum_{\lambda=\lambda_1}^{\lambda_2} \frac{2hc^2}{\lambda^5} \frac{1}{e^{hc/\lambda kT} - 1} \Delta\lambda \qquad (5-1)$$

In equation (5-1), h is planck's constant, c is the speed of light, λ is the wavelength, and k is boltzmann's constant. Care should be taken when selecting a proper $\Delta\lambda$ that minimizes computing time while preserving accuracy akin to that of an integral. It can be said that the total flux of the star incident on a detector is $F_{\Delta\lambda} = \pi \frac{R^2}{r^2} B$. Here, R is the radius of the star, and r is the distance of the star to an observer. Because the stars are assumed to be far enough away that their orbital separation does not significantly increase their distance to an observer, the distance dependence can be ignored. Also because each star is assumed to have the same number of points, any dependence on the points can be removed so the final equation for the amount of light coming from the star is $F_{\Delta\lambda} = \pi R^2 B$. To reduce the number of calculations, each point can initially be assumed to have this flux. This can be done because the total amount of flux is going to be normalized later on.

Now that there is a number representing the light generated from each point in the star, it is necessary to include various atmospheric and geometric effects that affect the amount of light coming from points at different positions on each star. The flux coming from each point will be referred to as L. The first effect is the cosine law.

$$L_{final} = L_{initial} * \cos\gamma \qquad (5-2)$$

In equation (5-2), γ is the angle between the surface normal at the given point and the line of sight to the observer from the point (opposite direction of L_0 in Figure 1). This law arises from the fact that when an element of surface (represented by a point) is tilted, the amount of surface area seen from an observer is decreased, so the amount of light from that area is decreased by a function of the tilt angle.

The second effect that must be modeled is the limb darkening effect. This is shown in Figure 3, where the outer ring of the star, called the limb, is darkened by increased atmospheric absorption around the edges. This results because light coming from the outer parts of the star must pass through more stellar atmosphere than points in the viewable middle of the star surface. Figure 3 shows a primitive diagram describing this effect.

There are several equations governing limb darkening laws. There is no single accepter equation, but the one that was used in *BinaryFactory* is below (Kallrath and Milone, 2009).

$$L_f = L_i[1 - x(1-\mu) - y\mu\ln\mu] \qquad (5-3)$$

In equation (5-3), x and y are parameters ranging from 0 to 1 governing the intensity of the limb darkening effect, and μ is the cosine of the angle from (5-2).

The third effect is the reflection effect. The term reflection is actually a misnomer. What actually happens with this effect is that when two stars are orbiting extremely closely, they mutually heat each other on their near sides. The result is that the surfaces on the near sides of each star are significantly brighter than the other parts of the star.

This effect is fairly challenging to model. The actual theory behind this effect is fairly complicated and will not be presented here. In *BinaryFactory*, the effect is modeled using a gradient obtained by manipulating the ratio between the distance between stars and the radius of a given point. Because this effect was not generated using accepted theory from a peer reviewed source, it will not be presented here, although it does seem to result in the desired effects to the light curve.

The total amount of light in any given frame is the sum of light coming from every point after the addition of these effects. The last step in this process is the normalization of the light. A simple way of doing this is to store the flux from all visible points in an array. Before displaying the light curve, the program should find the phase with the highest amount of flux, and divide the flux from each phase by that number. This will normalize the flux to a value of 1, which is common when comparing light curves.

The light curve is generated by plotting the phase with corresponding normalized amount of light.

RESULTS

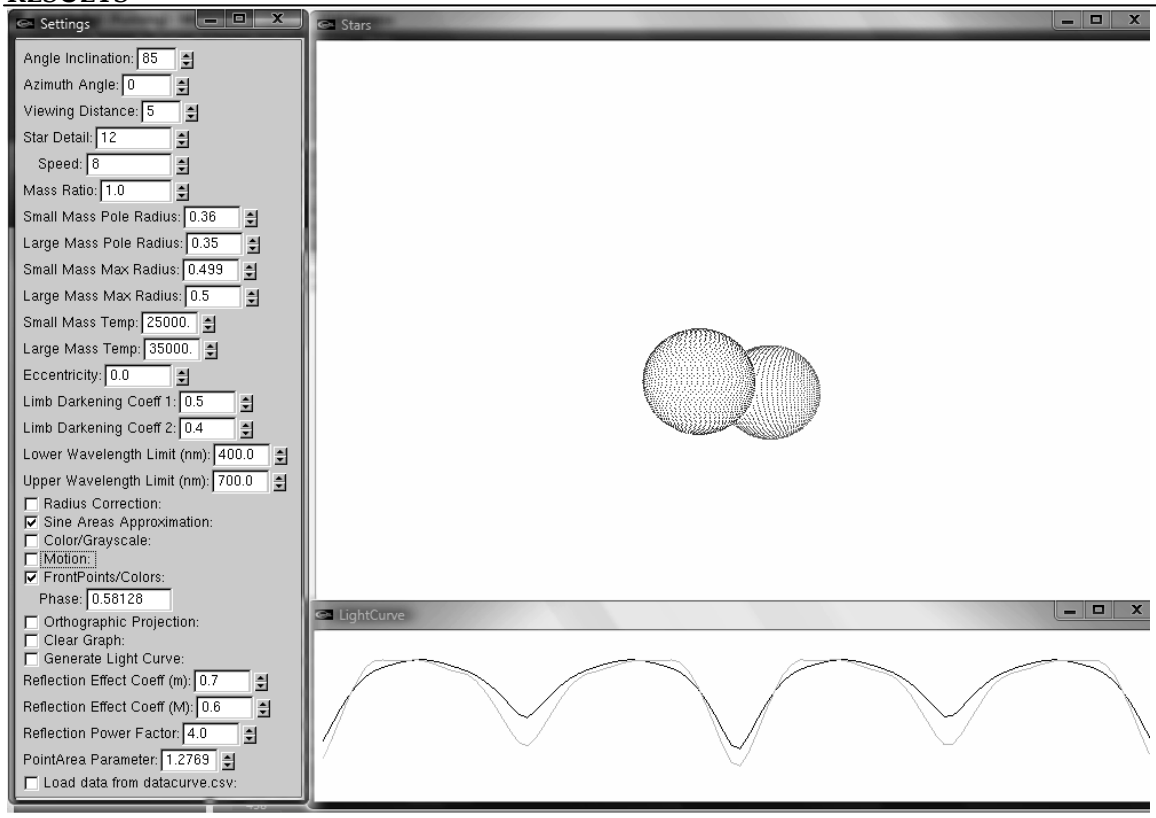

Figure 4 – An interactive screen image of BinaryFactory

The result of this paper was to show the methods involved in creating a model of an eclipsing binary star system. Any model should be user friendly and allow for the change of any of the parameters discussed in this paper. There are many ways to construct a model in any programming language. *BinaryFactory* was constructed with an options menu containing the parameters, a screen showing a real-time animation of the stars, and a screen displaying the generated light curve along. This is shown in Figure 4.

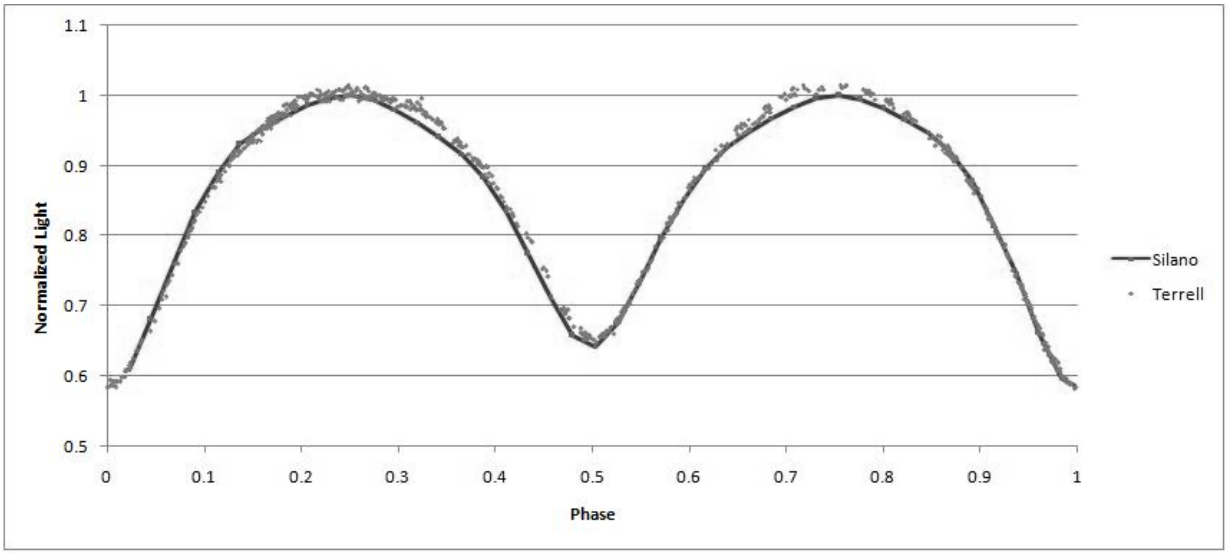

Figure 5 –BinaryFactory light curve fit to the observational data of Terrell (2003) for TU Muscae

Figure 5 shows a light curve generated for TU Muscae in the u bandpass using the mass ratio, surface temperature, inclination, star sizes, and observational data from Terrell et al (2003). The light curve generated is slightly under the observational data, but the overall fit is very good. Fine tuning of the reflection effect as well as tweaking the limb darkening effect would likely result in a 100% fit of the TU Muscae data from *BinaryFactory*. Figure 5 shows how the methods in this paper can be used to construct a model of an eclipsing binary star system that is extremely accurate and generates light curves that closely match observations.

ACKNOWLEDGEMENTS

I would like to thank Professor R. J. Pfeiffer, professor of physics at TCNJ, for his help with this project. Without his assistance and guidance, this project would never have been completed. His class notes in astrophysics provided a reference for many of the equations used in this paper.

I would also like to thank Professor T. Darkhosh, adjunct professor of physics at TCNJ, for his help in solving the orbits. He was vital in the early parts of this project with helping me solve the orbits using classical mechanics.

I would also like to thank Professor D. Terrell, section manager and astrophysicist at Southwest Research Institute's Boulder office. Professor Terrell was kind enough to let me use a figure published in his paper cited below and to use his data to create Figure 5.

REFERENCES

Bruton, D. 2004, SFA Observatory, Roche Lobes and the Morphologies of Close Binary Stars, http://observe.phy.sfasu.edu/downloads/StarLight/Word/RocheLobes.pdf.

Terrel, D., Munari, U., Zwitter, & T. Nelson, R. 2003, AJ, 126, 2988.

Kallrath, J., & Milone, E.F. 2009, Eclipsing Binary Stars: Modeling and Analysis (New York: Springer).