The Use of Gamification to Teach the Cosmic Distance Ladder

By
Benjamin Hoober-Burkhardt

A thesis submitted in partial fulfillment for the degree of

Bachelor of Arts

In

Physics and Astronomy

Faculty Advisor: Professor Bryan Penprase

Pomona College

May 2014
Abstract

Gamification is a method of teaching material through the use of game design techniques. These techniques can range anywhere from abstractions of reality to goals and motivation. Initially, research was done into the methods of gamification, exploring game design principles and methods of active learning. Game design principles focused primarily on ways to engage and motivate students to learn and understand the material more efficiently. Active learning exploration involved researching the effectiveness of such a method, as well as the ways in which the topic is implemented. Once these ideas had been solidified, a topic, the cosmic distance ladder, was chosen as the core material behind the creation of a platform game. With the intention of creating a game that would be applicable as supplementary material for an introductory astronomy course, the game explores many of the topics found on the cosmic distance ladder, creating a way for students to actively engage in methods used to find distances throughout the universe.
Acknowledgements

I must first thank Professor Bryan Penprase for his assistance with my project. Not only did he help me brainstorm and create the topic, he assisted with finding resources and research into the topic of gamification. Thank you.

Thank you to Professor Phil Choi for showing me the beauty of the field of astronomy in my freshman year, as well as the advice over the course of my college career. I’d also like to say thank you for guiding me to this topic in the first place. I never would have done this without you first positing the idea.

I cannot thank Bella Waldrop enough. She has been a friend for a long time and when I needed someone to do the artwork for my game, she stepped up and providing me with everything for which I could have ever asked. She did every ounce of the artwork with only two weeks’ notice and still wanted to do more, even when I said everything looked perfect. I cannot possibly thank her enough for the hard work and dedication she put into assisting with the creation of the game.

Lastly, I want to thank my family. They have been there my entire life, even if I wasn’t always cognizant of it, and the support they’ve given me in following my passions will be forever cherished. Thank you, everyone, for everything you’ve done for me.
# Contents

Table of Contents

1 Introduction and Background .......................................................... 1
   1.1 The Storyline ................................................................. 1
   1.2 The Cosmic Distance Ladder ............................................. 1
   1.3 History and Examples of Gamification ................................. 2
   1.4 Pygame ........................................................................... 3

2 Theory ......................................................................................... 4
   2.1 The Cosmic Distance Ladder ............................................. 4
      2.1.1 Trigonometric Parallax .............................................. 4
      2.1.2 Main Sequence Fitting ............................................... 5
      2.1.3 Cepheids ................................................................. 6
      2.1.4 Type Ia Supernovae and Globular Clusters .................. 7
      2.1.5 Hubble’s Law ........................................................... 9
   2.2 Game Design Principles .................................................... 10
      2.2.1 Abstractions of Reality .............................................. 10
      2.2.2 Interactive Story Lines ............................................. 11
      2.2.3 Rules ....................................................................... 12
      2.2.4 Goals ....................................................................... 12
      2.2.5 Rewards ................................................................. 13
      2.2.6 Feedback .................................................................. 13
      2.2.7 Simplicity and Readability ....................................... 14
      2.2.8 Aesthetics ............................................................... 15
   2.3 Active Learning and Gamification ........................................ 16
      3.1 Benefits of Active Learning ............................................. 16
      3.2 Components of Active Learning ..................................... 20

3 Story and Design ............................................................................. 23
   3.1 The Levels ........................................................................... 23
      3.1.1 Level 1 – Introduction ............................................... 23
      3.1.2 Level 2 – Parallax Angles .......................................... 25
      3.1.3 Level 3 – Main Sequence Fitting ............................... 27
      3.1.4 Level 4 – Cepheids ..................................................... 29
      3.1.5 Level 5 – Globular Clusters ...................................... 30
      3.1.6 Level 6 – Hubble’s Law ............................................. 32
3.2 Exploring the Design ........................................................................ 34
  3.2.1 Choosing a Platformer ................................................................. 34
  3.2.2 Cosmic Distance Ladder Activities ............................................. 35
  3.2.3 Understanding the Programming ................................................ 39

4 Future Work .................................................................................................. 49

References ...................................................................................................... 50
1 Introduction and Background

Although it is only a simple word, gamification is slowly gaining popularity in teaching and learning circles. It can be used anywhere, ranging from credit card reward systems to simulations for classrooms, jobs or the military [1]. With the movement to optimize ways of learning, gamification is in a perfect slot to become an important part of teaching for anything. Gamification is using game-based mechanics, aesthetics and game thinking to engage people, motivate action, promote learning, and solve problems [1]. For this thesis, a game will be created using the techniques of game design and gamification to successfully gamify the learning process of an introductory astronomy course, focusing on the core topic of the cosmic distance ladder. The primary goal is to create a game which can be used as a supplemental tool for implementing this important part of introductory astronomy learning.

1.1 The Storyline

There are a number of ways to begin creating such a game, but in this case something akin to the platform game Super Mario has been chosen. The player will take on the role of a futuristic scientist. The world has forgotten astronomy to a major degree, and this scientist has done research into the beginnings of astronomy: the cosmic distance ladder. He explores a number of topics and finally finishes his research, before deciding to go home with the intention of beginning his journey the following day. However, upon arrival the next day, he finds that the lab has been invaded by robots. The scientist runs through the lab, avoiding dangerous spills and killer robots. Just before reaching his office, a portion of the collapsing ceiling hits him and he loses his memory of the cosmic distance ladder. Upon awakening, he enters his office just in time to see all of his work stolen by the villain, who accidentally leaves behind the first page of the research. This is the beginning of his journey to retrieve all the pieces of his work, battling robots on planets and moons across the universe and using each successive piece of his work to travel deeper into the vast fields of space, until finally retrieving each page of his research and finding the answers to life, the universe and everything.

1.2 The Cosmic Distance Ladder

See Section 2.1 for an in-depth explanation of the components of the cosmic distance ladder

A key part of the game will be the underlying topics found in a normal introductory astronomy course, with a specific focus on the re-creation of the cosmic distance ladder. Players will use the various tools found on the cosmic distance ladder to travel between different planets, solar systems and galaxies, eventually journeying to the farthest reaches of the known universe.
Figure 1.1 A diagram depicting the methods and distances of the cosmic distance ladder. Credit: The Worlds of David Darling, Cosmic Distance Ladder

The first step on the scientist’s journey will involve trigonometric parallax, followed by observations of star clusters for main-sequence fitting. After that, work will go into the period-luminosity relation for Cepheid stars and another standard candle, globular clusters. The last step involves using Hubble’s law. Each level of the distance ladder involves the player doing an action that allows them to better understand the tool which they are using. For instance, main sequence fitting involves fitting a cluster to the main sequence, and the Cepheid game involves the brightness of the star. All of these things will be covered in such a way as to promote learning the concepts in a fun, interactive setting, using the principles of game design and gamification.

1.3 History and Examples of Gamification

See Section 2.2 for an explanation of the principles of game design and gamification

Although video games have been growing in popularity for the past three decades, the idea of changing menial tasks that are normally found boring into game-like creations has only recently begun. This is primarily due to the success of the video game industry, and the realization of the potential provided by gamifying different aspects of the learning process. Some of the earliest attempts to gamify learning began with concepts like Oregon Trail and Where in the World is Carmen San Diego?, fairly successful games in their own way, but also all alone at the top of successful educational games [2]. Despite this success, others programs failed due to a lack of the core principles found in game design, namely a lack of a goal and a lack of motivation to complete said goal in a meaningful way. Since then, however, there have been a number of attempts to use video game concepts in real world scenarios outside of video games.
These concepts have resulted in a number of brilliant projects like the piano steps subway entrance and a potential HIV cure. In the case of the piano steps, a subway station noticed that the majority of people chose to use the escalator at a specific stop instead of using the stairs, even when they found that the stairs were almost always empty and the escalator was almost always busy. In an effort to encourage people to exercise and decrease the traffic on the escalator, the stairs were altered so that each one played a different musical note when stepped on. Almost immediately, people began testing out the stairs and the use increased by 66 percent [1]. Another incredible use came from the crowd-sourced game experience *Foldit* which drew over 46,000 people and solved the mystery of how a key protein may help cure HIV. The experience took 10 days for a solution to be found – this was revolutionary as it had been worked on by scientists for over 15 years without an answer [2]. These are, of course, only limited examples. More and more of these incredible instances are popping up all over the world, as people begin to understand the strength of gamification.

1.4 Pygame

Along with the principles of game design and gamification, the program Pygame was chosen as the primary source for designing this game. Pygame is a powerful tool used for creating video games of all kind, from games like *Pac-Man* to others like *Slingshot*. The program is powerful in that it can create a number of different programs with only a basic knowledge of the coding language, Python. It can also handle games that have a number of elements in it, which is good for a platform game like the one that will be created. Along with its strength for creating solid games without an in-depth comprehension of programming, it also has a number of tutorials and support available. If there is ever a problem, it can easily be solved through books or online. All this ties together as the best choice for programming a basic game intended for learning the concepts of the cosmic distance ladder.

![Figure 1.2](image.png)

**Figure 1.2** A version of Pac-man created in Pygame. *Credit: Mac Pan, Matthew Rollings*
2 Theory

For the success of this game, three core ideas must be understood – the cosmic distance ladder, game design principles and active learning/gamification. The cosmic distance ladder is integral for the storyline and for applying astrophysical concepts to the game. Game design principles are necessary for the motivation and engagement of players with the game. Active learning and gamification are used to help explore the ways into which a game can be focused to allow for a player to better understand the material placed before them. These core ideas will be discussed in depth in the following sections.

2.1 Cosmic Distance Ladder

The following section will use An Introduction to Modern Astrophysics 2nd Edition (Carroll & Ostlie) as a primary reference

2.1.1 Trigonometric Parallax

The simplest method for determining distances to planets within the Solar System or to stars within the Milky Way can be found using trigonometric parallax. Using basic trigonometric functions, distances to various objects can be found using either Earth’s diameter for close objects, or Earth’s orbital diameter for more distant objects.

![Figure 2.1](Image) A representation of how to calculate the distance to stars using the Earth’s orbital diameter and the star’s parallax angle. *Credit: An Introduction to Modern Astrophysics 2nd ed., Carroll & Ostlie*

Given the distance from the Sun to the Earth is one astronomical unit, the distance $d$ can be found with the following equation.

$$d = \frac{1 \ AU}{\tan p} \approx \frac{1}{p} \ AU$$ (2.1)
The equation uses 1 AU as the distance from the Earth to the Sun, $p$ as the parallax angle (usually in seconds) and $d$ as the distance from the Sun to the distant star in astronomical units, as shown in Figure 2.1. Trigonometric parallax runs into problems when the distances become too great as the angles become exceedingly difficult to measure the farther the star is from the Solar System, but the method is useful within about 100 light years.

### 2.1.2 Main Sequence Fitting

Main sequence fitting is another method for determining distances to distant objects within the galaxy but is used for clusters of gravitationally bound stars instead of single stars. The concept of main sequence fitting comes from the assumption that plotting the apparent magnitude instead of the absolute magnitude of a cluster on the main sequence will result in only a vertical shift on the Hertzsprung-Russell diagram because the stars in the cluster are all relatively the same distance and age from the Solar System. Apparent magnitude is a measure of the brightness of the star from Earth and absolute magnitude is a measure of the intrinsic brightness of the star, where brightness is a measure of the luminosity of the star. Luminosity is simply the amount of energy released every second. Apparent magnitude can be thought of as how the brightness changes over distance; a star that is one hundred light years away will appear less bright than a star only one light year away. Absolute magnitude mimics this concept by finding how bright the star would be if it were at a set distance of ten parsecs from the Earth. The vertical shift on the H-R diagram comes from a difference between these two magnitudes, as pictured below, where the red is a cluster with its apparent magnitudes compared to absolute magnitudes.

![Figure 2.2 An example of main-sequence fitting. Credit: Main Sequence Fitting, Ron Markham](image-url)
As can be seen from Figure 2.2, a cluster (red) has been fit onto the main-sequence (black) with a difference in magnitudes of about 15.5. To get to the necessary equation for solving the difference in magnitude, the inverse square law must first be used. The inverse square law is as follows:

\[ F = \frac{L}{4\pi r^2} \]  

(2.2)

\( F \) is the flux measured at a distance \( r \) from a source of luminosity \( L \). Imagine a shell encasing a star with radius \( r \). At that distance, the amount of light passing through the shell would be the luminosity of the star – the farther the light travels from a source, the lower the flux. A ratio between absolute magnitudes can be created because the ratio between the brightness should be directly related to the ratio of the fluxes, since luminosity, the measure of brightness, is directly related to flux as shown in equation 2.2. This results in the following equation.

\[ \frac{F_2}{F_1} = 10^{(m_1-m_2)/5} \]  

(2.3)

\( F \) is the flux and \( m \) is the apparent magnitude of the star. Combining equations 2.2 and 2.3 results in

\[ 100^{(m-M)/5} = \left( \frac{d}{10 \text{ pc}} \right)^2 \]  

(2.4)

Where \( m \) is the apparent magnitude, \( M \) is the absolute magnitude and \( d \) is the distance to the star. This can be rearranged to get the following equation.

\[ d = 10^{(m-M+5)/5} \]  

(2.5)

Simply putting in the values found in Figure 2.2 allows for a calculation of the distance. This method is primarily used for calculations within the Milky Way Galaxy, at ranges of 100 thousand light years.

### 2.1.3 Cepheid Stars

Cepheid stars are pulsating stars that periodically change brightness. Although they can be found in just about any galaxy, they have a constant relation between their period and luminosity making it a standard candle. A standard candle is the simply an object with a known
luminosity that can make use of the inverse square law, which, for Cepheids, is possible because the period of pulsations can be observed to calculate the luminosity.

![Period-Luminosity Relationship](image)

**Figure 2.3** A plot showing the period-luminosity relation for Cepheids. *Credit: European Space Agency*

The consistency of the period allows astronomers to calculate magnitudes of Cepheid stars, as shown in the following equation.

\[
M_v = -3.53 \log_{10} P_d - 2.13 + 2.13(B - V)
\]  

(2.6)

In this instance, \( M \) is the absolute magnitude of the Cepheid in the V band, \( P \) is the pulsation period in days and \( B-V \) is the color index. Once the absolute magnitude has been found, it can be input into equation 2.5 along with the observed value of the apparent magnitude to calculate a distance to the Cepheid. As Cepheids tend to be very bright and can be found in any galaxy, they are commonly used for distances ranging up to 10 million light years, the range to numerous nearby galaxies.

### 2.1.4 Type Ia Supernovae and Globular Clusters

As mentioned above, a standard candle is an object with a known luminosity. As Cepheids are only bright enough to reach nearby galaxies, two other major standard candles, Type Ia supernovae and globular clusters, are used for the more distant galaxies. For Type Ia supernovae, the absolute magnitude of peak brightness falls around -19.5. Given this value, simply observing a supernova can provide part of a light curve, which can be used to determine the peak apparent magnitude of the supernova. Applying equation 2.5 again will allow for a distance to be determined.
Figure 2.4 A light curve for numerous Type Ia supernovae. After correcting for time scales, as brighter supernovae wax and wane more slowly than faint supernovae, the light curves match. Credit: Science at Berkeley Lab

The case is similar for globular clusters, as the brightness of globular clusters in a galaxy can be used to find a globular cluster luminosity function. The distribution of globular clusters is well described by a Gaussian function.

Figure 2.5 A globular cluster luminosity function for M31, Andromeda, in the V band.

Figure 2.5 shows an example of a globular cluster luminosity function, which needs to be fit to a Gaussian function. Once fit, the peak of the function can be compared with the values of the peak for a globular cluster luminosity function for the Milky Way. These two peaks, along with equation 2.5, can be used to successfully estimate distances to galaxies. These two methods can be used to calculate distances of up to 10 billion light years.
2.1.5 Hubble’s Law

In 1925, Edwin Hubble began researching distant galaxies using Cepheids. As he did, he came to the realization that the recessional velocity of a galaxy was directly related to its distance from the Milky Way.

![Figure 2.6 Hubble’s original calculation for the Hubble constant. Credit: An Introduction to Modern Astrophysics 2nd ed., Carroll & Ostlie](image)

From this work, he was able to form an equation. The equation is as follows.

\[ v = H_0 d \]  

(2.7)

For Hubble’s equation, \( v \) is the recessional velocity of a galaxy, \( d \) is the distance to the galaxy and \( H_0 \) is Hubble’s constant. Simply having knowledge of the redshift of the galaxy could now allow Hubble to calculate a distance and is a common strategy for astronomers today. This is an effective strategy for calculating distances to distant galaxies, especially those that cannot be reached using standard candles.
2.2 Game Design Principles

The following section makes use of The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education (Kapp, Karl M.) as a primary reference.

Games require a number of different principles to work. From understanding how to implement goals or define the rules of a game, to having a story line with great aesthetics allows a designer to create something magnificent. This is not always the case for every game made and although a number of these concepts can be followed well, successful implementation will usually allow a game to grab a player’s attention and motivate them to play and complete the game. The list here is not a complete list because there is no truly complete list of game design principles; however, understanding some of these basic ideas will allow for the beginnings of a powerful game that can move and motivate a player.

2.2.1 Abstractions of Reality

Games are not real; this does not mean they cannot feel real to the person playing them and provide meaningful, insightful information to the player. Monopoly and Chess are prime examples of this, as they take the concepts of financial monopolies and wars and put them into a small, manageable space where one can conceptualize the ideas without necessarily requiring the experience of being part of a monopoly or fighting a war. The game Risk is a perfect example of this concept.

![Figure 2.7 The board for the game Risk. Players attempt to gain control of the world by conquering all territories. Credit: Risk, Hasbro Inc.](image-url)
In reality, when a person moves a platoon to a certain location, they might not find out until days or weeks later that moving that platoon resulted in an attack elsewhere. In games, however, players can immediately see the consequences of their actions. In *Risk*, moving troops from Alaska to Central America may result in opponents attacking the now weakened Alaska or building up their forces in Venezuela to counter attack. Games allow players to take these abstractions of reality and apply them in a way that can convey information from which players can learn.

2.2.2 Interactive Story Lines

An interactive storyline is an exceptional way to get a player involved in a game. The player should be able to feel themselves integrated into the game, as the story lets them experience what the character experiences. A novel must engage the reader, just as a game must engage a player. The player should feel drawn to the character and have their perceptions adjusted to more closely align to those of the character. The character should develop physically or emotionally, or the journey the character goes through should define them in some way. A clear example of this comes from the game *Uncharted 2: Among Thieves*, where the main character, Nathan Drake, goes on a quest to find the lost city of Shangri-La. Along the way he develops relationships with his fellow adventurers, while he slowly finds out about the person he is and the person he wants to be. The story line allows the player to directly experience what Drake goes through and sympathize with how he feels both physically and emotionally. As he struggles to climb through the snowstorm after being shot, the player can directly feel the anger, betrayal and depression he feels because of a story line that perfectly engages its players.

*Figure 2.8* Nathan Drake struggles through a snowstorm after he is shot and his train crashes down the side of a mountain. *Credit: Uncharted 2: Among Thieves, Naughty Dog*
2.2.3 Rules

Rules provide a player with the boundaries of the game. They can stretch from extremely constricting rules to fairly minimal, but every game requires them to function. Rules primarily define how a player can interact with their environment, from achieving a goal, to scoring points, to understanding how many people can join them in their game. These rules are important as they allow the player to understand the things they are capable of doing and the things they cannot. When a player goes into a game they must determine how high they can jump or how quickly they can run or whether or not they have abilities to shoot fireballs or fly. The rules define the environment in which the player must play and allow designers to apply concepts of the game.

2.2.4 Goals

Goals are an integral part of not just games, but life as well. People need goals to have motivation and a drive to succeed – games require people to feel the exact same way. This does not necessarily have to be a complex goal, as complex goals can be frustrating or difficult to visualize. Take the simple game of Tetris; players have a goal of “do not let the blocks reach the top of the screen”. This breaks down into a number of minor goals, from correctly identifying ways to clear blocks and finding ways to fit blocks together to prevent this from happening. The main goal, however, is simple and easy to understand; a goal that is able to motivate the player to action and provide incentive and feedback for continued play. Without a goal the entire game becomes somewhat meaningless. Without a driving goal, what motivation is there to complete a task?

![Figure 2.9](image)

*Figure 2.9* A player fails to achieve their goal in Tetris. *Credit: Tetris Friends*
2.2.5 Rewards

Reward systems are consistently used in society in just about every meaningful way. Credit cards have reward systems for using their cards more than others. A kindergarten student might gain a star on the wall for good behavior. An airline might give passengers points to get their free flight after several trips. These are all examples of rewards systems that give the player, whether the player is a consumer, kindergartner or passenger, bonuses for doing something that the credit card company, teacher or airline might like. In games this is no different. Completing a quest will result in bonus experience or extra gold or powerful weapons; all things that draw the player in more. Without a reward, the player would lack an incentive to continue attempting to reach the goal. If they receive nothing for completing a task, what would be the point of doing the task in the first place? Rewards are simply another way to motivate a player and have them keep playing.

Figure 2.10 Trophies earned by completing certain quests in Playstation 3 games. Credit: Playstation 3, Sony

2.2.6 Feedback

Feedback is an integral part of a game because it allows players to learn from their mistakes immediately and comprehend how their actions change the course of a game. No matter what happens in a game, a player receives feedback. While playing a third-person shooting game the player is made aware of the weapons they possess, the ammunition available for their weapons, the amount of health they currently have, the time left in the match, the number of kills on their team, the number of kills for the opposing team, the number of kills and deaths the player has, the number of points they have scored over the match and potentially even more. This is an absurd amount of feedback, but it is no different than wandering into a new destination; a game simply seeks to take the amount of feedback received from daily life and incorporate it into a game. This can be extremely powerful for learning purposes for this reason; imagine someone who has never before played a third-person shooter and, while running around a map, sees a blinking red item on the ground. They move towards it to figure out what it is, then, with little warning, they explode. Obviously, to the veteran player, the blinking red item on the ground was
a land mine, but the new player had no idea what it was. However, because of how the game has delivered feedback to the new player, they now know that walking on the little blinking red items on the ground is a bad idea because it kills them. This feedback is instantaneous and shines as a wonderful example of how video games allow players to learn. A player can now plan ways around the mines in the future or even learn to use the weapon to their advantage.

Figure 2.11 Land mines litter the ground in front of a player. If the player walks over a mine they die, providing the player with immediate feedback. Credit: Warhawk, Incognito Entertainment

2.2.7 Simplicity and Readability

Simplicity of the different aspects is core for any game. A player that looks at a game and finds that they are unable to comprehend their next, or even first, move will immediately become frustrated and potentially leave the game for good. When they look at the interface they need to understand the capabilities of their character and what they are allowed to do and what they are not allowed to do. If, for instance, a character was able to shoot fireballs to easily defeat a boss, but the game never touched on the fact that the character can shoot fireballs, it would leave the player frustrated and hopeless as they no longer have any way to complete the task at hand. There should be an inherent visibility to what the player is doing and the player should never have to become frustrated with an aspect of the game, whether it is from poor interfacing, bad explanations or simply boring exercises. That is not to say a game cannot be successful if the game is difficult – challenges are important – but something being overly challenging with little to no reward for the difficulty of the challenge leads to players rage quitting. Readability is imperative for a game, and a player can only have as much fun from the game as their knowledge of what they can do allows them.
2.2.8 Aesthetics

Aesthetics are a core component of what makes a game work. Although they are certainly not the most important part of a game, they still enhance the users experience and allow someone to become embroiled in the game. A game like Chess has its own aesthetics – the design of the king and queen, rook and knight, bishop and pawns. Not only are these pieces different, but they are distinguished from the opponents through different colors. On top of this, designers can put their own artistic talent into the pieces, whether the pieces may use a certain type of wood or metal or plastic, or take the appearance of characters from a movie or book. Designers can create something that allows players to see beauty in their work from a visual perspective. Not only can a designer put their own flavor of work into a piece, they can also help the player become integrated with the game to a greater degree. A dark, quiet mining ship in outer space is the perfect place to have a game focused on scaring players from the terror of the ferocious alien enemies, which was the setting and design of the game *Dead Space*. An engineer was sent to fix a broken mining ship only to discover a vicious, deadly alien species had been unleashed on the ship. The game is filled with dark rooms, eerie sounds and terrifying monsters appearing from unlikely locations. This helps draw the player directly into the game as they can feel the terror Isaac feels as he attempts to escape from the ship and rescue his fellow crew members. Aesthetics, while not the defining aspect of a game, are a core part of what can make a game go from good to great.

![Figure 2.12 Isaac picks up his first weapon after running from his first encounter with the alien species. Designers create a world filled with messages scrawled in blood, darkness, shadows and eerie sounds.](Credit: Dead Space, EA Games)
2.3 Active Learning and Gamification

Active learning and gamification are traits actively tied together in a movement to increase activity for students as they learn different subjects. The methods have a number of benefits which will be discussed in the following section, which will then be followed by work down in an attempt to use active learning. The goal is not to show that gamification is the end all be all for learning, but that it is an extremely powerful tool that can be used to great effect for learning experiences.

2.3.1 Benefits of Active Learning and Gamification

Active learning and gamification are tied together in an attempt to strengthen areas of learning that tend to be hard to understand or follow [4]. Although active learning will never be the answer to all problems, it has been shown to be extremely effective in a number of areas for a number of reasons [5]. A few of the major benefits can be derived from research done exploring the effectiveness of active learning and games-based learning.

The most important question that must be asked, immediately, is whether or not gamification and active learning show a measurable increase in learning. Many researchers have addressed this question, with important work looking into whether or not the outcomes were positive. A large portion of this research involved meta-analyses which will be discussed below. In one such analysis, researchers found that computer-based games showed a positive impact on learning as compared to conventional learning. Of the sixty-four studies this group looked into, they found that the results were positive 53.1% of the time, with mixed results 26.6% of the time and no difference 18.8% of the time. Only 1.6% of studies explored seemed to favor conventional over games-based learning [6]. Another paper came to the conclusions that 56% of the time little to no difference was found, 32% of the time games and simulations were favorable, 7% favored games but had questionable controls and 5% favored conventional methods [7]. Another paper, focused on the use of gamification for business practices, found a significant increase in every study they explored [4]. Not every case will be perfect for learning, as evidenced by the slightly mixed results above, but it is common that games-based learning will have a net positive effect on learners.

Beyond meta-analyses, there has also been research done more specifically into introductory courses for different fields, namely biology, physics, physiology and mathematics. Below, two of these studies will be reviewed, one with a focus on an introductory biology course and another on an introductory physics course. Both studies sought to determine whether or not active learning would alter the scores of their students. The biology course focused on the scores for each successive year it applied this new method, with the first year used as a control to gauge performance. The physics course used data for many different colleges over the course of one semester, comparing the results of courses taught with applied engagement to courses taught more traditionally.
For the biology course, students saw a general increase in proficiency over the course of two years. The scores seen in Figure 2.8 were taken during the final exam for the class. The test covered two major topics, logistic growth and life-history evolution as one topic and island biogeography the other. The research did not necessarily show a direct increase in every year from 2006 to 2008, but it did show an overall upward trend, with students scoring on average 91% on the final exam in 2008, as opposed to the average of 86% in 2006 [8].

Figure 2.13 A graph of test scores for an introductory biology course. Credit: [8]

Beyond the work done in the biology course, another study was undertaken at MIT in an attempt to explore how active learning might benefit students in an introductory mechanics course. For this course, the researchers focused on using the value of gain, which is defined as

\[ G = \frac{S_f - S_i}{100 - S_i} \]  \hspace{1cm} (2.8)

\( S_f \) is the score on the exam at the end of the course, while \( S_i \) is the score on the same exam before the course. This value measures the increase in total score divided by the highest possible increase in score, with a higher gain correlating to a better final score. A perfect score on the final exam returns a value of 1 as the maximum, while a value less than 0 means the student performed worse on the final exam. For the MIT researchers, they found that for interactive engagement courses the gain came out to an average of 0.48±0.14sd, while only seeing a gain half as effective in traditional courses with a value of 0.23±0.04sd. This shows that students that took the interactive courses performed nearly twice as well as those who did not; nearly half the points missed the first time were gained back in the interactive course, while only one quarter were gained back in the traditional course [9].

Groups also examined how it altered higher-order thinking. One researcher came to the conclusion that computer-based games generally increased higher-order thinking, allowing for a
quicker understanding for tasks such as planning and reasoning [6]. This was reviewed in the following excerpt.

“One benefit is making learners become engaged by the material, thus invoking a state of ‘mindfulness’ in which learners employ effortful and meta-cognitively guided processes. Learning in a mindful way results in knowledge that is considered meaningful and useful, as compared to the inert knowledge that results from decontextualized learning strategies. With simulated visualization, authentic problem solving, and instant feedback, computer games afford a realistic framework for experimentation and situated understanding, hence can act as rich primers for active learning” [6]

This increased understanding and ability to process things actively ties directly into the belief of the effectiveness of gamification for tasks that involve aspects of higher-order thinking. Another researcher came to similar conclusions that learners had better declarative knowledge, procedural knowledge and retention rates of the material. The study also found that those that worked with simulations and games were more confident in their understanding, believing they had truly mastered the material in question [10].

**Figure 2.14** A chart showing the scores of two monkeys after four tests. The last test had no hints available. *Credit: [11]*

Beyond this basic research in meta-analyses, another group of researchers looked into how active learning might benefit rhesus macaque monkeys. The goal of this project involved providing two subjects with five-item lists of photographs to which the subjects had to respond to in order, regardless of location. The two subjects were put through for different tests, in which
the availability of hints was altered. The first test always provided hints, the second allowed for request of a hint, the third allowed hint requests fifty percent of the time and the final one did not allow for any hints. The researchers found that allowing the two subjects to use hints overall hurt their performance when asked to order the items correctly without any hints, while also seeing a distinct trend in the frequency of hints provided with the ability of the subjects to correctly process the order of the images. The researchers concluded from this that active learning facilitates non-verbal serial memory in primates, more or less arguing that active learning will provide greater understanding than passive learning [11].

Another meta-analysis involved the work of a team of researchers at the University of Central Florida. This team explored a few different aspects of computer gaming and interactive simulations as a method for learning. Much like other researchers, they found that learning was certainly increased, although they focused more on the positive effects on cognitive gains and attitude. Over the course of their research they found that higher cognitive gains were observed for simulations and games, but games tended to be slightly less reliable. Beyond the cognitive gains they also saw a distinct increase in the motivation of the learners. When given a chance to interact with the work put in front of them, the researchers found that people had a greater desire to learn the material as compared to traditional methods [12]. Other research groups concluded that instructional computer games facilitated motivation across different learner groups and learning situations [6].

This can be seen from the work done by research taken during an introductory physiology course. Students were given access to two separate courses, one which included active learning and another which focused on the more traditional lecture. Students were found to generally have more self-efficacy, which measures the student’s expectation of success. Students also generally seemed to have a better attitude toward classes for the active learning course, while students in the traditional course found lectures boring. The researcher also makes note that on the attitude questionnaire at the end of the year students who had taken the active learning course were more inclined to believe that physiology was manageable, while students from the traditional courses more commonly believed physiology was too difficult for them. Over 27% of the students taking the traditional course believed physiology was too hard for them, while only about 15% believed the same for the active learning course [13].

![Figure 2.15](image.png)

<table>
<thead>
<tr>
<th>Item</th>
<th>Scale: 1 = Strongly Disagree ... 5 = Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.20 Human physiology is too hard for me.</td>
<td>38.6</td>
</tr>
<tr>
<td>20.20 Human physiology is too hard for me.</td>
<td>18.8</td>
</tr>
</tbody>
</table>

*Figure 2.15* The values are percentages of people who answered in that way. The active learning group is shown first while the traditional learning group is shown second. *Credit: [13]*

On top of all of the overwhelming research showing that gamification can be effective in a variety of situations and has definitely improved the ability of students to learn, some
researchers decided to delve into an exploration of whether or not there was a dependence on age or gender. The results obtained from this research debunked this fairly profusely. There is no difference in learning ability gained from gamification and active learning for people of different age, gender or race. Any and all people can benefit from active learning [12].

2.3.2 Components of Active Learning

Active learning courses are not all successful. Some fail to successfully engage their students, resulting in passivity and decreased knowledge gains [14]. Others have split focus that do not directly engage the material and can mislead a learner about the information the simulation is supposed to provide. Games and simulations may not provide enough feedback. Some might be too difficult to understand, forcing players to have to spend a large portion of time learning the mechanics of the game instead of learning the presented material [5]. All of these problems hinder the ability of a student to learn actively, but can all be solved with focus on the necessary applications. One such necessity is the targeting of specific content with precisely defined objectives. It must be used in the way it was intended to be used – trying to take a simulation and using it for a different goal will not work. It also must be tied directly to the object in mind [7]. A game about problem solving in physics will be less effective than a game about problem solving in computer science when a student is attempting to learn about problem solving for a computer science course. Another necessity is that the games should be directly embedded into the instructional program, instead of as a stand-alone product. As long as the game is integrated successfully with feedback and assistance being directly available, the learner should have increased gains. This helps establish why things are happening in the game and how these events tie to the goals of the learning experience [5]. As long as the player can be directly involved in a program that provides feedback and assistance where necessary, along with using something for that which is was created, the player should see increased knowledge absorption.

There are many methods for courses that have wished to implement aspects of active learning and many have been successful. In this instance, the important methods that were implemented will be discussed and analyzed for their effectiveness in how they apply to the theories behind active learning.

![Table 2](image-url)

**Table 2** Basic active-learning lecture (modified)

<table>
<thead>
<tr>
<th>Activity</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engagement activity</td>
<td>5 minutes</td>
</tr>
<tr>
<td>Lecture segment</td>
<td>15-20 minutes</td>
</tr>
<tr>
<td>Activity</td>
<td>5-10 minutes</td>
</tr>
<tr>
<td>Lecture segment</td>
<td>15-20 minutes</td>
</tr>
<tr>
<td>Activity</td>
<td>5-10 minutes</td>
</tr>
<tr>
<td>Closure/evaluation</td>
<td>10 minutes</td>
</tr>
</tbody>
</table>

**Figure 2.16** Shows the basic active learning lecture. *Credit: [13]*
The most common theme throughout active learning circles is reorganizing the lecture style. The general concept is to initially draw students in, and then follow up with a lecture for a small portion of the class. After the lecture, students are provided with a problem of some kind which they are asked to solve in groups of anywhere from 3-5 people. After the students have solved the problem, groups have leaders randomly selected to present the findings of their group. After receiving the answers, the professor can go over the answers, explaining the faults found in incorrect answers and how to reach the correct conclusions for the problem [13].

The lecture can begin in any number of ways. A common goal is to begin the class with a description of the goals for the day. This might involve solidifying an understanding of some concept in physics, such as force diagrams and how to draw them. The lecture also tends to be centered on one topic, but can be incorporated into one major overarching theme. This can help tie together different concepts and build bridges between subjects that may, at first, seem different. This helps fill the role of specific content being taught to help the students focus on one primary thing [8]. While going over the lecture, the professor can actively engage the students by encouraging questions and providing ways to ask students questions. This can come in the form of wireless remotes or “clickers” as they are normally called, where the teacher can record the responses of the student to make sure topics are understood as the material is slowly synthesized [8].

After the lecture, students break into groups usually created at the beginning of the semester. These groups tend to consist of anywhere from three to five people, with the goal being active engagement by the students on a topic that directly pertains to the lecture. The problem should not take more than 5 minutes to complete and the professor will normally wade through the room, observing the students’ work and providing input and assistance when necessary [15]. This process allows the students more direct contact with the professor, while, at the same time, allowing the students to get to know each other better and feel less like another in a crowd. The entire process is more intimate and personalized, usually allowing the student to achieve a higher mastery of the topic at hand [16]. The general sense of the professor during this time is that the instructor becomes a watcher and listener [17]. Sometimes included at this time are 1 minute outlines/essays where the student is asked to quickly design a general concept diagram to make sure the major topics of the lesson are being covered. This allows the professor to make sure the student is correctly obtaining the right knowledge and is more aware of how topics fit together. For instance, energy might be broken into branches of kinetic and potential, which might then be broken into common equations used to calculate those values [15].

After the students have overviewed the question in their groups, leaders of each group will be randomly selected to present the work completed. The professor can record this information and then go over the correct way to complete the problem and touch on problems the students encountered while trying to complete the question. This provides students with immediate feedback while they are still processing the concept; problem sets can fail to function well in this regard due to students never getting feedback that can be processed and incorporated.
directly into the topic. The immediate feedback is a core part of the active learning cycle and is one of the most key parts of the group learning process [8].

While this is the overall most common way the process is completed, many professors have input their own flavors into the mix. One professor who worked in a large lecture hall provided students with name tags and wireless microphones. The name tags sought to increase the interaction between students, increasing personalization and letting students more easily work together. The microphones also allowed the students to communicate not only with the professor but also with the other students, ensuring that everyone could clearly hear the questions. In addition to the name tags and microphones, the professor also implemented an email system whereby the students could ask questions personally and privately; students had no desire to stand out in a crowd of over one hundred people and email allowed questions to be asked in a method the gave students privacy to voice concerns over the subject [16].

Other innovative professors held weekly quizzes that would allow keep the students up to pace for the class. This allowed the students to be continuously dedicating a small amount of time every week to the subject, instead of cramming all of the information in at once. Along with the weekly quizzes, vocabulary words were added to each lecture as a way to target key words the students should know and hopefully allow the students to better understand the topics they were reviewing [8]. Some professors also implemented weekly reading reflections, designed to help the student reflect on the material with a small overview of the topic. Along with this, course reviews were held throughout the semester in an attempt to help students who found their study habits failing or who were having a difficult time working with the active learning style [15].

Although the methods may vary, the general concept holds true for all instructors attempting to implement active learning into the classroom. Each concept can be directly derived from games, particularly the ability for students to actively engage in the material and having feedback provided directly to them immediately. Combining active learning and game mechanics allow for a powerful approach to teaching and, although not perfect for all situations, extremely effective for many, ranging from hard sciences like Physics or Biology to softer sciences like Sociology or Psychology. All that matters is the implementation of the mechanics and active learning will do the rest.
3 Story and Design

Over the course of the next chapter the design of the game will be discussed. The primary goal was the implementation of the principles of both game design and active learning in an attempt to create a supplementary game for an introductory astronomy course, with a specific focus on the cosmic distance ladder. The game is a platform game, a type of game where the player traverses a small level, avoiding enemies and deadly hazards through the use of platforms, and the player works through these levels until they reach some final destination. The final destination in this instance is an observatory, where the player explores one of the aspects of the cosmic distance ladder. Upon completion, they will move to the next level. The following sections will explore this in more detail.

3.1 The Levels

The following section focuses on the story line of the game. The major events are covered, as well as minor descriptions of the levels. Each level includes at least one screenshot of the world that is explored and at least one screenshot of the cosmic distance ladder related activity.

3.1.1 Level 1 – Introduction

The game begins with an introduction to the story of a great scientist and his adventure.

Figure 3.1 A screenshot of the title page of the game, Cosmic Horizons.

The player is a character in a world where science has fallen by the wayside. Society no longer has the curiosity so prevalent in today’s world, with scientists hardly exploring anything beyond the world around them. However, occasionally, a scientist will appear that has a great drive to
explore the world outside their world. This is the character the player controls. The scientist has been doing research into calculating distances to astronomical objects and has just completed his research. He intends to take this information to explore the universe but arrives at his lab the day of his departure to find his lab has been invaded by robotic forces.

Figure 3.2 A screenshot of the beginning of the first level, the Lab.

The robots have caused a great amount of mayhem, destroying the lab. The scientist makes his way past the robots and other hazards in the lab. Just before reaching his office where his research is stored, he is injured by falling debris, causing him to lose his memory of his research. When he awakens, he runs past his office to encounter the villain controlling the robots. The villain tells the scientist he is taking the research and intends to travel to the ends of the universe, finding the answer to life, the universe and everything. The villain then jumps into his ship and flies away, leaving the scientist behind.

Figure 3.3 A screenshot of the first encounter with the villain.
However, one piece of research is left behind as the villain escapes, and the scientist retrieves it. Although he has forgotten how to use his research, his pages allow him to still use his knowledge. On his research he learns the location of a forgotten Earth observatory where he can find data for calculating distances with parallax angles, as well as the equations for calculating distances.

![Parallax Angles Diagram](image)

**Figure 3.4** The first page of the scientist’s research, detailing parallax angles.

### 3.1.2 Level 2 – Parallax Angles

The scientist travels to the location of the observatory, finding himself a little ways away from his destination. However, he again finds the area littered with robots, again, as well as a floor of jungle spikes, a trap placed there long ago. The scientist must leap across platforms, avoiding the dangerous flying robots and the deadly spikes beneath him.
After finding his way past the spikes and robots, he reaches the observatory where he can begin using his research. In the observatory he finds long forgotten research involving values of parallax angles. The scientist is then given a screen where he must draw the lines needed to determine a distance using a parallax angle.

The scientist is provided with the equation on the research page and must use it and the value of the parallax angle to input a distance. The scientist cannot leave the observatory unless the value is correct. After inputting the correct value, the player will leave the observatory and travel a small distance just past the observatory to find the second page of the scientist’s research, left behind by the villain again after the villain finished his work in the jungle observatory.
This page contains information on main sequence fitting, along with an observatory in deep space, which can now be traveled to because of the distance calculations performed earlier. The scientist quickly jumps in his space ship and travels to the next world.

### 3.1.3 Level 3 – Main Sequence Fitting

The scientist lands on a volcanic moon after using the knowledge from the parallax calculations to get a distance. He is again forced to make their way through the maze of enemies and pools of lava, until the player reaches the observatory on the volcanic moon.
This time the scientist is given a display of the main sequence. The research page provides details about the main sequence, going into stellar evolution. Once the scientist feels confident about his understanding of the main sequence, he is asked to use the data found in the observatory to fit a cluster to the main sequence and find a distance value using the equation for distance based on magnitudes.

The scientist takes this new distance value and leaves the observatory, where he finds the next page of his research. The page contains information about Cepheids and a location which can be found using the distance determined in the previous observatory.

**Figure 3.9** A screenshot of the astronomy task involving main sequence fitting.

**Figure 3.10** The third page of the scientist’s research, detailing Cepheid stars.
3.1.4 Level 4 – Cepheids

This time the scientist lands on an ice moon that functions just like the previous levels. He must make his way past the robots using his platform jumping skills, making sure to avoid the patches of ice. Patches of ice cause him to lose control and slide in whatever direction he was moving, so he has to be careful not to find himself sliding into a deadly robot. The scientist makes his way through these hazards until finally reaching the observatory.

![Figure 3.11 A screenshot of the fourth level, the Ice Moon.](image)

This observatory contains information about Cepheid stars. The scientist looks at one nearby Cepheid and must click rapidly to watch the Cepheid brighten. Once the Cepheid has brightened enough and a light curve has been created, he gains access to the average magnitude and period of the Cepheid’s pulsations.

![Figure 3.12 A screenshot of the astronomy task involving Cepheid distance calculations.](image)
Using the values found with the light curve, as well as the information contained within his research page on Cepheids, the scientist finds he is able to calculate a distance using the given information and the period-luminosity relationship. After leaving the observatory, he finds another page left behind by the villain, this one detailing the location of a recently exploded supernova and the way to calculate distances using globular clusters.

![Figure 3.13](image)

**Figure 3.13** The fourth page of the scientist’s research, exploring globular cluster luminosity functions.

### 3.1.5 Level 5 – Globular Clusters

The scientist lands on a moon outside of a supernova that has recently exploded. He must make his way past the robot horde once again, this time making sure to watch his step as meteors fall all along his path. With deft movements and stellar agility, the scientist is able to make it to the observatory.
This time the scientist is asked to look at the data provided by the observatory and determine the location of a number of globular clusters based on the full-width half-max (FWHM) of the sources. Sources that have the correct FWHM can be clicked on, and the data will be recorded. After the scientist finds 10 sources, he creates a histogram.

The histogram shows data based on the chosen clusters, creating a fitting a Gaussian to the plot and determining an average peak. The scientist then uses the peak along with the given data on the Milky Way’s globular cluster luminosity function to determine a distance. After leaving the observatory, he finds the final page of research and uses it to travel to a black hole.
3.1.6 Level 6 – Hubble’s Law

The scientist travels to a position just outside of the event horizon of a black hole. The level is littered with hundreds of robots and platforms and the player must use every trick in the book to make their way through the level. It is obvious the villain must still be here by the number of robot present in the level, so the scientist hurries as quickly as he can to catch up to the villain. Barely able to make it past the robot army, the scientist struggles into the observatory.

Figure 3.17 A screenshot of the sixth and final level, the Black Hole.
Once the scientist reaches the final observatory, he finds a small simulation of the big bang. A small point jumps around on the screen until it explodes, filling up the whole screen. Directly after that, three galaxies are shown moving away from the Milky Way, and the velocity of one of the galaxies is provided. The scientist uses this knowledge of the movement, Hubble’s law and the velocity to find the distance, cementing the understanding of the final part of his research.

![Figure 3.18 A screenshot of the astronomy task involving Hubble’s law.](image)

The scientist makes his way out of the observatory, only to finally encounter the villain. He finds the villain distraught because, despite traveling to the ends of the universe, he had not found the answers to life, the universe and everything. The scientist consoles him and point out that there are still a great many other things to explore in the universe and perhaps, together, they might be able to find the answer together. The villain agrees to work with the scientist and the two hop in their space ships, traveling off into the universe.

![Figure 3.19 A screenshot of the last part of the final level.](image)
3.2 Exploring the Design

The following section focuses on the creative choices used throughout the game explaining why certain choices were made as well as going in-depth on the programming of the game.

3.2.1 Choosing a Platformer

The very first decision I had to make was the kind of game I wanted to make. There are plenty of different genres in the gaming industry – role playing games (rpg’s), shooting games (shooters), platform games (platformers), racing games, strategic games (real time strategy/rts or turn based strategy/tbs) – and each has a different use for attempting to achieve a certain goal. In my case I knew my primary goal was the construction of something useful for either physics or astronomy, eventually settling on the decision to create a story focused on the cosmic distance ladder and essentially re-creating the path of the great scientists of the world as they slowly but surely found ways to measure larger and larger distances. This is not exactly how the game ended up, but it did follow the general structure. I wanted to create separate levels the player could walk through, each focusing on a different aspect of the cosmic distance ladder. This meant a few things: one, I had to choose a type of game where the player could explore a world, two, I had to choose a game where levels were useful and, three, I had to choose a game focused on linear progression. Now a few of these tie into different genres; the first, for instance, works great for rpg’s, but the a game focused on the evolution of the character (which is what an rpg entails) focuses much less on the content trying to be conveyed. I could go for more of a strategic puzzle game, but that would eliminate the ability to actually explore worlds, which was something I wanted to implement. Luckily, however, there was one genre that pretty easily implemented everything I wanted to achieve – the platformer.

Platform games, at the most basic instance, are simply games focused on a player making their way through a (usually 2d) level, jumping on platforms to reach some target at the end of the level. Other things can usually be implemented into this design as well, such as enemies or dangerous hazards, as well as simple interactions with villains and other things. In my case, I implemented robots that the player had to avoid, tying them into the story line as minions of the villain. I also implemented different hazards based on the type of level; the lab got toxic spills, the jungle had deadly spikes, the volcanic moon had pools of lava, the ice moon has patches of slippery ice and the supernova had a meteor shower. All of these things contribute to the game, making it more difficult but also integrating the player with the general aesthetics and feel of the game. The linearity of platformers was extremely desirable as well. It meant I could easily put together a small level that would directly lead to a destination, in this case observatories, which would allow the player to try out a new activity. These two things, I think, work out quite well together and allow the player to have fun with the game, going through a period of level exploration and enjoyment into a period of critical thinking and simulation work.
3.2.2 Cosmic Distance Ladder Activities

Before I began creating the game, I had to choose a topic on which to focus. As I wanted to create something useful for an introductory science course, I considered different things found within the standard introductory course, until finally settling on the cosmic distance ladder. A ladder provided a simple, visual progression on which the game could be based. As rung of the distance ladder was passed, so too would the player progress through the levels. This was a simple, effective topic on which the game could now be based.

Since the topic was the distance ladder, I began choosing the things on which I felt were necessary to strongly represent the basics of the ladder. Trigonometric parallax was an obvious decision as a beginning spot, as it incorporated minimal astronomical knowledge and could easily be understood with a solid understanding of geometry. Although not the first step of the distance ladder, it seemed the most relatable to a general populace and a good starting position with which the game could begin. After trigonometric parallax, I went with main sequence fitting. The main sequence is a core part of understanding stellar evolution and using this method allowed for not only increased understanding of the meaning of the main sequence, but also another way to progress on the distance ladder. Next came Cepheids, a step up from main sequence fitting. Cepheids are bright, pulsating stars which simply required understanding of the period-luminosity relationship, something rather straightforward – the longer the period of the star, the higher the luminosity. The next step was slightly more difficult, as I had to make a choice between type Ia supernovae or the globular cluster luminosity function. Although I recognize the importance of the supernovae, it felt almost felt like any designed simulation mini-game would result in something akin to the Cepheid project and so I chose, instead, to work with the globular cluster luminosity function. The final level incorporated the highest rung of the ladder, Hubble’s law. Hubble’s law incorporates a number of astrophysical tools and seemed appropriate as the final level for the game.

Each level had its own astronomy related task, dealing with a separate rung of the ladder. Once the introductory level was completed, the player would travel through each level to find an observatory where the task would begin. The first task involved trigonometric parallax, focusing on the geometry of the set up. The player is asked to connect lines between the three major points: the Sun, the Earth and the target star. As they connect the lines they are given labels to understand the why those values are chosen. For instance, the Earth-Sun line provides the player with the astronomical unit (AU), allowing the player to see the base of the triangle and how it relates to the Earth and Sun. These lines were intended to help the player see through a more visual process what actually happens when someone needs to find a distance using the parallax angle.
def parallax(start, end, line, our_distance):
    font = pygame.font.Font(None, 30)
    finished = False

    # Sun
    sunx = 140
    suny = 200
    sunrad = 20
    pygame.draw.circle(screen, White, [sunx, suny], sunrad, 0)

    # Moon
    orbitrad = 100
    pygame.draw.circle(screen, White, [sunx, suny], orbitrad, 1)

    # Earth
    earthx = 140
    earthy = 500
    earthrad = 8
    pygame.draw.circle(screen, White, [earthx, earthy], earthrad, 0)

    # Our Star
    starx = 450
    stary = 200
    starrad = 15
    star(450, 200)

    # Background Stars
    star(550, 100), star(550, 150), star(550, 200), star(550, 250), star(850, 300)

Figure 3.20 The first part of the code used to run the parallax function. This part draws the initial objects.

After trigonometric parallax, the player moves on to main sequence fitting. Main sequence fitting involves the evolution of stars and how they fit to the main sequence. The player is given a small plot of a cluster and told the stars are all believed to be the same distance and age from solar system. Once provided with the cluster, they are given a graph of the main sequence and asked to fit the cluster on to the main sequence.

def main_sequence(fit_loc, our_distance):
    font = pygame.font.Font(None, 30)
    fitx = fit_loc[0]
    fity = fit_loc[1]
    distance_calc = False
    finished = False

    if fitx >= 272 and fitx <= 280 \ 
    and fity >= 10 and fity <= 15:
        distance_calc = True
    else:
        screen.blit(the_fit, (fit_loc))
        time_to_fit = font.render('Drag the cluster on right', 1, White)
        time_to_fit2 = font.render('to fit the main sequence', 1, White)
        screen.blit(time_to_fit, (110, 250))
        screen.blit(time_to_fit2, (110, 270))

Figure 3.21 Part of the code for fitting the cluster to the main sequence. This part determines whether or not the cluster has been fit to the main sequence.

As they move along the main sequence, they can hopefully visualize the path of the evolution of the stars in the cluster and see how the unevolved stars still fit on the main sequence. Once they have successfully fit the cluster to the main sequence, the player is shown the apparent
magnitudes of the fitted cluster compared to the absolute magnitudes of the stars on the main sequence. The player can then use these values and the given equation to find the correct answer of the distance. Again, the player is asked to visualize the process of actually fitting the cluster, almost like it’s a puzzle piece, in an attempt to help better understand the way to use main sequence fitting.

Cepheids were next in line. As Cepheids focus on using the period-luminosity relationship, players are asked to click rapidly to brighten a Cepheid star and as they click the star slowly brightens until a graph of the entire period is displayed.

```python
def cepheids(i, rectx, our_distance, brightness, click):
    font = pygame.font.Font(None, 30)
    screen.blit(cepheid_graph, (200, 75))
    cover = pygame.Rect(rectx,130,300,170)
    pygame.draw.rect(screen, (0,0,0), cover, 0)
    finished = False
    if i > 255 and brightness == False:
        period = font.render('Use the period and apparent magnitude'\
                            ' to find the distance', 1, White)
        distance_value = font.render('d (nearest kpc) = '+our_distance,1,White)
        magnitude = font.render('21.38',1,White)
        zero = font.render('0',1,White)
        days = font.render('3',1,White)
        screen.blit(period, (10,10))
        screen.blit(distance_value,(350,40))
        screen.blit(magnitude,(270,205))
        screen.blit(zero,(279,336))
        screen.blit(days,(553,336))
        pygame.draw.line(screen, White, [270,201], [550,201], 1)
    elif brightness == False:
        time_to_click = font.render('Click rapidly to make the '\
                                    'cepheid brighter',1,White)
        screen.blit(time_to_click, (10,10))
```

**Figure 3.22** One part of the Cepheid code. This part asks the player to find the distance using the provided values for period and magnitude.

Once the player has the period of the star in front of them, they are also given access to the observationally determined apparent magnitude, and then asked to first find the absolute magnitude of the Cepheid using the period. Once they have both the absolute and apparent magnitudes, they must again find the distance. The hope for this activity was the visualization of the light curve created by the pulsating star and understanding how the period of the light curve relates to the luminosity of the Cepheid.

The fifth level involved the globular cluster luminosity function (GCLF). In this case the player has to look through a field of stars and find the correct location of the globular clusters.
def gclf(gclf_coords, gclf_fwhm, gc_counter, used, click, our_distance):
    font = pygame.font.Font(None, 30)
    coord = pygame.mouse.get_pos()
    fwhm = 0
    finished = False

    if gc_counter < 10:
        gclf_border = pygame.Rect(50, 50, 500, 300)
        pygame.draw.rect(screen, (255, 255, 255), gclf_border, 5)
        screen.blit(globular_clusters, (50, 50))

        for x in range(len(gclf_coords)):
            if gclf_coords[x][0] + 1 >= coord[0] >= gclf_coords[x][0] - 1 and 
                gclf_coords[x][1] + 1 >= coord[1] >= gclf_coords[x][1] - 1:
                fwhm = gclf_fwhm[x]
                fwhm_number = font.render('FWHM: ' + str(fwhm), 1, White)
                screen.blit(fwhm_number, (340, 360))
                if click == True and fwhm >= 6.7 and coord not in used:
                    gc_counter+=1
                    used=used+[coord]
                    counter_text = font.render('Found Clusters: ' + str(gc_counter), 1, White)
                    screen.blit(counter_text, (130, 360))
                    directions = font.render('Hover over stars to view the FWHM', 1, White)
                    screen.blit(directions, (150, 20))

Figure 3.23 A part of the GCLF code. This part asks the player to find 10 sources with a FWHM of 6.7 or greater.

After finding 10, they move on to a histogram of the ten collected clusters and use them to create the globular clusters luminosity function. I felt that this represented how to create the function because the function is simply based on the most common apparent magnitudes found within the clusters. Since globular clusters are standard candles, the player can compare the absolute magnitude, found using the peak of the GCLF for the Milky Way, with the apparent magnitude of the globular clusters found in the distant galaxy, again using the GCLF. The player compares these two values and can pop out another distance.

The sixth and final level involved Hubble’s law. This was not something explicitly easy to illustrate, so I decided to go for something fairly straightforward and basic. In this case, I began with a simple point jumping about the screen until it exploded, filling the screen with its expansion. This immediately transferred into the creation of four galaxies, three of them moving away from the first, the first being the Milky Way. This was supposed to illustrate the expansion of the universe and show how galaxies are moving away from the Milky Way with some recessional velocity. The player can then use a given velocity and Hubble’s law to find the final distance, solidifying their understanding of each of the rungs of the cosmic distance ladder.
def hubble(bigbangstats,our_distance):
    font = pygame.font.Font(None, 30)
    finished = False
    bangx = bigbangstats[0]
    bangy = bigbangstats[1]
    bangrad = bigbangstats[2]
    bangcounter = bigbangstats[3]
    galaxyx = bigbangstats[4]
    galaxyy = bigbangstats[5]
    galaxycounter = bigbangstats[6]

    if bangcounter < 500:
        bangcounter+=1
        shift = random.randrange(-1,2)
        bangx+=shift
        shift = random.randrange(-1,2)
        bangy+=shift
    elif bangcounter >= 500 and 500 > bangrad:
        bangrad+=2
        pygame.draw.circle(screen,White, [bangx,bangy], bangrad, 0)

Figure 3.24 A part of the Hubble code. This part creates a small circle that will randomly jump around the screen for a small period until it explodes.

### 3.2.3 Understanding the Programming

Programming the game involved a number of aspects. First and foremost was the creation of the character and allowing them to move about on the screen. After that was completed, I moved on to creating background and entire levels in which the character could explore, as well as adding in a ‘floor’ on which the character could stand, platforms which the character could use to get to other parts of the level and gravity, allowing the player to constantly fall whenever they fell off a platform or needed to jump to get somewhere. Once the character’s ability to maneuver in the environment was complete, hazards and robots had to be implemented. This involved collision functions that would allow the game to tell if a player and a robot overlapped each other. As the character only has one life, even one collision resulted in the player’s death, forcing them to restart, also another thing needing implementation. The levels then required beginning and endings, defined once the character reached a certain point, allowing them to pass to the next level. In this case, levels were created using the exact same functions, except they required slightly different variables and constants, defined at the beginning of each and every new level. Another test required addition of level events. Events included introductions, explaining the character’s thoughts, working in an observatory and interactions with the villain. Each of these things will be discussed below.

The first thing that has to be understood is the basic structure of the game loop. The game loop is an infinite while loop – a loop that allows the game to essentially run everything contained within it, before running the entire thing all over again for eternity. This allows the
player to consistently update the screen – a character’s movement can be seen because when the player presses a key to move it and the character’s pixel coordinate is altered slightly, the game will then overwrite the previous copy of the character with a new background (although the background image is still the same) and create a whole new character at the new pixel coordinate.

```python
import pygame, sys
from pygame.locals import *

pygame.init()
# The screen where the game is drawn
screen = pygame.display.set_mode((400, 300))
pygame.display.set_caption('Cosmic Horizons')

# The image placed on the background
background = pygame.image.load('the_background.jpg')

# The infinite game loop
while True:

    # The event loop, handles mouse clicks and key presses
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

    # Draws the background image on the screen
    screen.blit(background, (0, 0))

    # Updates the screen
    pygame.display.update()

Figure 3.25 The code necessary to run Pygame. The program makes use of an infinite loop, while True. Each time loop runs through, it will redraw the background then update the screen.

It can be thought of more simply as infinite layers of sheets, over and over. The first sheet goes down and the game draws everything on the sheet, but once it reaches the end of the code it grabs a new sheet and overwrites everything on the new sheet, placing the new sheet directly on top of the old sheet. This is actually an incredibly useful tool and quite brilliant for the construction of a game. Now let’s move on to understanding the aspects that I actually programmed for my game.

I want to first talk about pixel coordinates and character movement. Unlike normal coordinate systems, computer programming flips the y-axis, meaning down is positive and up is negative. So, for instance, if I were to shift a character from a coordinate location of (0,0) to (100,100), it would move right 100 pixels and down 100 pixels. Now this is not a hard thing to implement – simply assigning a key can allow the player to easily shift a character from one spot to another.
Figure 3.26 The code controlling character movement

If, for instance, you want the character to move right, you could set the right arrow key to change whatever the current pixel coordinate of the character is (let’s say 50) to increase by 5. This means each time the right arrow key is pressed, the character will increase by a flat 5 pixels (meaning our 50 will go to 55, 60, etc.). This applies to each key, meaning we can then create a way to move our character all over the screen.

The difficulty with this, however, is the jump. The jump requires rapid movement that shouldn’t all be accomplished in one go. If I were to, for instance, decrease the character’s y-coordinate from 200 to 100, it would look very odd. The character would almost seem to warp from one spot to the next when jumping, while I wanted a much more smooth transition. This meant I needed to have a conditional set up just for jumping. If a player were to press the jump key, they would start decreasing their y-coordinate until they had traveled a certain distance. Once they had traveled this distance, the jump would end.
Figure 3.27 The code detailing the jump function. Allows the character to initially move quickly, but slows down as the character moves higher, just like a real jump.

```python
#Jump function
if jump_dis < 2 and jump_active == True:
    char_locy-=20
    jump_dis+=JUMP_COUNTER
    FPS_CLOCK.tick(10000)
elif jump_dis < 4 and jump_active == True:
    char_locy-=16
    jump_dis+=JUMP_COUNTER
    FPS_CLOCK.tick(10000)
elif jump_dis < 8 and jump_active == True:
    char_locy-=12
    jump_dis+=JUMP_COUNTER
    FPS_CLOCK.tick(10000)
elif jump_dis < 20 and jump_active == True:
    char_locy-=8
    jump_dis+=JUMP_COUNTER
    FPS_CLOCK.tick(10000)
#Stops jump after 20 ticks
elif jump_dis >= 20:
    jump_dis = 0
    jump_active = False
```

After implementing this, however, I found that the character could double or triple jump just by mashing the jump key – to fix this I input a command where if the character were jumping it couldn’t jump again. I actually had to fix this later, but I’ll talk about it after discussing gravity in just a little bit.

After the character was able to move all around the screen, the background, platforms and gravity all had to be implemented. Obviously we don’t want the character running off the sides of the screen, so we actually have to shift the background – and everything with it – to keep the character centered on the screen. Once the character was sitting at an x-coordinate of 280 (the center of the screen for the character in my game), the background actually began shifting in the opposite direction, as seen in the code provided in Figure 3.26. Instead of moving the character, the background would decrease its x coordinate. For instance, if the character was supposed to move 5 pixels to the right, the entire background would shift 5 pixels to the left instead, providing the player with the appearance of movement. This also meant I needed to shift the pixel coordinates of all images, except the character. A platform location, for instance, would have to also decrease by 5 pixels, along with robot locations or hazard locations. This took a little work but did come together in the end.

The background required a little finesse, but was quickly followed up with the construction of gravity. Obviously a game is not much fun when the character simply floats around and creating the illusion of the character actually being in the lab or the jungle allows the player to feel more integrated with the game. To implement gravity simply involved making the character always fall down (aka increase the pixel coordinates of the character) every time the
loop was completed. This gave the appearance that the character was falling. Of course, this led to the immediate problem of the character disappearing off the screen, something that had to be fixed. By implementing a floor, the character was able to actually stand without falling into a bottomless pit. The floor was set towards the bottom of the screen, at around 360 pixels. The character, then, was not allowed to have a y-coordinate of less than 280 pixels (as it was 80 pixels tall). So what could now happen is the character could use a jump and shift upward slowly, before being dragged back to the ground by gravity.

```python
#Gravity animation
if platform_hit[0] == True:
    #Shift Character up top of ground/platform
    #Doesn't move after it sits on top
    char_locy -= platform_hit[1]
    pygame.display.update()
    FPSCLOCK.tick(100)
elif robot_hit == False and hazard_hit == False:
    #GRAVITY when not standing on something and not dead
    char_locy += GRAVITY
    pygame.display.update()
    FPSCLOCK.tick(100)
```

Figure 3.28 The code used for gravity. If the character is not dead or standing on something, they will fall towards the floor.

As I mentioned above, I altered the code to make it so the player couldn’t jump again until the jump ended. This wouldn’t work with gravity; however, because it meant that at the apex of the jump, the player could then spam the jump button and jump again immediately, essentially negating gravity. This wasn’t going to work, so I had to recode this to make it so the player could only jump if they were not in the air.

Successful implementation of gravity and the floor were perfect, so the last thing I needed was implementation of platforms. Platforms required that the player be, first, between the edges of the platform, and, two, above it. With gravity at work, the player could fall down until the character’s y-coordinate matched up with the platform’s y-coordinate, and, once the character was standing on the platform, stopped gravity from activating. This then allowed the character to jump again, potentially to another platform, or fall off the platform and put the player back at the mercy of gravity.
# Determines whether character is standing on an object

def standing(plat_loc, char_stats, backgroundx, GRAVITY, BOTTOM):
    collided = False
    shift = 0
    locations = plat_loc
    char_x = char_stats[0]
    char_y = char_stats[1]
    char_w = char_stats[2]
    char_h = char_stats[3]
    for x in range(len(plat_loc)):
        if char_x-backgroundx+char_w > plat_loc[x][0] -
        and char_x-backgroundx < plat_loc[x][0]+plat_loc[x][2] -
        and char_y+char_h < plat_loc[x][1]+GRAVITY -
        and char_y+char_h >= plat_loc[x][1]:
            collided = True
        if char_y+char_h != plat_loc[x][1]:
            shift = 1
    if char_y+char_h >= BOTTOM:
        collided = True
    if char_y+char_h != BOTTOM:
        shift = 1
    return [collided, shift]

**Figure 3.29** The code that determines whether or not the character is standing on something. This is used to determine whether or not gravity should be in effect.

Character movement was all around completed at this point. The character could run along the ground, jump to platforms and basically maneuver just about anywhere they might want to go, as long as there were enough platforms to which to jump and gravity didn’t pull them down too quickly. The next step, after creating this entire world in which the player could explore, involved making them actually fear their environment. This meant successful implementation of robots and other hazards that could kill the player, if they were to run into one of these dangers.

Robots are messy. They have to run around doing scary things, making the player attempt to avoid them. If they are stationary, they are too easy to avoid; forcing the player to actively decide when to attempt to make their way past the robots. Robot animation was rather straightforward, no different than implementing gravity or other things within the loop, but the difficulty involved finding out whether or not the main character and the robots were touching. If they were touching, the player dies and restarts from the beginning of the level. The general idea behind the coding for this is that the rightmost x-coordinate of the character cannot be greater than the leftmost x-coordinate of the robot and vice-versa. This also holds true for the highest and lowest y-coordinates.
def got_hit(robot_loc, char_stats, backgroundx):
    collided = False
    char_x = char_stats[0]
    char_y = char_stats[1]
    char_w = char_stats[2]
    char_h = char_stats[3]
    for x in range(len(robot_loc)):
        if char_x-backgroundx < robot_loc[x][0]+robot_loc[x][2] \
            and char_x+char_w-backgroundx > robot_loc[x][0] \
            and char_y < robot_loc[x][1]+robot_loc[x][3] \
            and char_y+char_h > robot_loc[x][1]:
            collided = True
    return collided

Figure 3.30 The code that determines whether or not a character has collided with a robot or a hazard.

After the robots came the different hazards, dependent on the level. Each one was fairly similar, but had its own flair. The first level involved toxic spills, stationary objects usually used to simply block off a specific route that the player could otherwise make their way past with ease. The second level implemented spikes, dangerous logs that would impale the player if they were to fall on them. They covered the entire floor of the jungle level, forcing the player to spend the jungle only on platforms. The third level was not much different from the first, but instead of toxic spills the player had to avoid pools of burning hot lava. The fourth level again had the small stationary hazard, a patch of ice, but ice is not innately dangerous. To remedy this, I implemented a code where, instead of dying upon touching the ice, they would instead slide along the ice in whichever direction they were facing. This could very easily result in their death at the hands of a robot if they were not careful. The final level containing hazards was the fifth level, the supernova. This involved meteors raining from the sky. The meteors moved fairly quickly and would spawn at certain positions, making the player have to not only avoid the robots flying through the air and ground, they would also have to be wary of a giant meteor hurtling at them from space. The hazards used the same collision code as the robots, luckily cutting some of the work down.

The penultimate construction involved level events. There were certain points on the level where the character would do something the player could not control and the game would have to be paused momentarily. This involved putting in a conditional where the controls for character movement would lock up and prevent the player from using them anymore, as long as the event continued. After figuring out the pause, I began adding in the actual events. There are two major events, entering observatories and talking to the villain. To do this, I input conditional where if the player were to pass a certain point, they would activate these events. In the case of the observatories, once the player reached an observatory they would activate the event, allowing them to attempt to complete the task. This task would stay on the screen until it was completed, sending the character back outside the observatory to complete the rest of the level.
#Enters observatory
if char_lockx-backgroundx > OBS_X and completed_task == False \
and level_counter > 0:
    pause = True
    at_obs = True
if astro_task == False:
    enter_obs = font.render('Click to enter the observatory',1,White)
    screen.blit(enter_obs, (150,180))
if researcher == True and astro_task == True:
    researchPage(level_counter)
if researcher == False:
    screen.fill((0,0,0))

**Figure 3.31** The first part of the code that controls entering the observatory. Once the character passes OBS_X, the x-coordinate of the observatory, they will activate the astronomy task until they complete it.

The other major event involved talking to the villain at the end of the first level and sixth level. In the first, the player travels until he reaches the villain, and then the game pauses letting the villain say what he wants to say. Once the villain walks off the screen after talking to the scientist, the player is again given back control of the character. The coding was again fairly straightforward as it just involved setting an x-coordinate location equal to the activation of the event.

```python
#Talks to villain
if level_counter == 0 or level_counter == 5:
    screen.blit(villain, (villain_x+backgroundx, villain_y))

#Level 1 Villain talk
if level_counter == 0 and char_lockx-backgroundx > 3400 \n    and talk_done == False:
    if convo_end == False:
        current_line = villainConvo(click,level_counter, 
                               current_line,convon_convo_end)[0]
        convo_end = villainConvo(click,level_counter, 
                               current_line,convon_convo_end)[1]
    pause = True
    isTalking = True
    click = False
    if convo_end == True and villain_x < 4200:
        villain_x+=5
    elif convo_end == True and villain_x >= 4200:
        pause = False
        isTalking = False
        talk_done = True
        current_line = 1
        convo_end = False
```

**Figure 3.32** Part of the code controlling the villain event. The game will output a line every time the mouse is clicked, until all the lines are used up and convo_end is True.
One last thing had to be defined for level events: the end of the level. Once the player reached the final part of the level, the scientist would walk off the screen and the next level would begin. Activating the event was again just a matter of reaching a certain x-coordinate, but everything on the level had to be redefined. The background, platforms, observatories and locations of everything had to be reset to the new level. To do this, I create a list of lists, with the first list holding each of the level, and each of the levels holding all the information necessary to create the level. Once the end of the level was passed, the level counter would increase by one, shifting the information in the list to the following level. All of this information would then be created on the screen using the exact same processes used for creation of the first level. This was easily replicable for each level, with the only downside being the absurd number of variables required to keep track of each thing.

```python
# End of level
elif char_locx-backgroundx > level_end:
    level_counter+=1
    if level_counter <=5:
        current_level = leveler[level_counter]
        char_locx = CHAR_LOCX_START
        char_locy = CHAR_LOCY_START
        backgrounder = backgrounderX_START
        backgroundy = backgroundyX_START
        researcher = True
        completed_task = False
        talk_done = False
        robot_move = 0
        the_number = ''
```

**Figure 3.33** The code that resets the level. Once the character passes a certain point, increases level counter by 1, resets the character back to the start, resets the background back to the start, reactivates research pages and re-defines the completed task and talking events. Also resets the robots back to initial positions and resets the distance value back to an empty string.

The final task involved setting up a title screen and the end credits. The start of the game simply involved putting in a conditional where the game would display the title page of the game until the player clicked on the screen. After clicking, it would send the player into the first level, where they could play the entire game.

```python
if start_intro == True:
    the_intro()
    pause = True

if level_counter == 6:
    credit()
```

**Figure 3.34** Part of the code used to start the game and end the game.
Once they had played through the entire game, the level counter would increase to six (the seventh value as coding starts with zero), and the credits would pop up. From there, the player could click the screen, resetting the level counter back to zero and setting the title page conditional back to true. This meant the player could then go right back into the game, as the game has made an entire loop of the full code.

These were a few of the hurdles I worked through when designing the code. I believe the most difficult stuff involved, number one, keeping track of what conditionals did what and making sure I didn’t accidentally activate a conditional through the use of another function, and, number two, making the character able to jump and move either left or right. It was difficult to keep track of the conditionals partially because my code was messier than it probably should have been, but also because there were just thousands of lines of code. Keeping track of upwards of 50+ conditionals in 1500 lines of code is difficult in and of itself. The jump was something I was never quite able to solve. When a player presses space and the right key, for instance, they begin jumping to the right. However, if they release the space key the right key stops working, until it gets released and re-clicked or the space bar is pressed again. This was the only part of my code I could not debug.

The only major things I would change involve better planning of the code so it is less messy, as well as better ways to tell if a character has collided with a robot or hazard. The biggest flaw with the first was the difficulty in reading the code and keeping track of what everything does, while the second was problematic because if a robot were contained within a 80x80 pixel box, but the actual robot might be thinner as some points, it could appear like the character and robot were not actually touching even if the game believed they were. Solving these two things (as well as the jump problem) would have made the game a better experience all around.
4 Future Work

There are a number of areas that can be explored further. The primary things worth delving into further involve iteration and reworking of the game to create a more effective experience for the players, attempting to test the capabilities of the game in the context of helping teach the cosmic distance ladder, trying different areas for teaching purposes, varying anywhere from electronic circuits to stellar evolution, and increasing distribution of the game.

The game is rather simplistic and basic. This leads to questioning all of the design principles for legitimacy and viability in the course of the game should be attempted. How effective does the story provide a goal for the player? What about the story line? Does it create a powerful abstraction of reality where the player can test out aspects of the cosmic distance ladder? These topics can be explored to a greater degree as the game is analyzed and perfected.

This is not the only way the game can be perfected. The primary purpose of the game was for use as supplementary material in an introductory astronomy course. The best way to see how effective it would be, then, is to actually use it in an introductory course, allowing students to actually attempt to learn parts of the cosmic distance ladder to a greater degree. This would, of course, not replace the actually teaching of the cosmic distance ladder; although the game goes through some of the methods and touches on when to use them, there is more to the distance ladder than just these simple things and actually gaining a stronger understanding through a more in-depth exploration of the topic is important. However, the game can definitely be tested to see how effectively a student can understand the cosmic distance ladder with the integration of the game into the course’s material.

The final truly interesting method worth exploring is changing the material used in the creation of the game. Although this game focused on the cosmic distance ladder, there is nothing preventing other games from being created for other topics. The topics could range from kinematics or electronics or things like momentum all the way into astronomy, exploring stellar and galaxy evolution or even more basic things like orbits. There is no shortage of topics worth exploring and, if the game is an effective method for assisting a student in learning the material, there is no reason other games could not be created.

Outside of furthering the game, the single most important thing is distribution. The game was created in Pygame, a program which most people do not have immediate access, which leads to the necessity of transferring the game to an easier medium where potential players can actually play the game. Transforming the game from a Python program into an app, perhaps, or an easily accessible game on the internet, would allow for incredible distribution of the game and allow anyone to use the game to further their understanding of the cosmic distance ladder.

None of these ideas are too far-fetched in the context of the game and exploring any of them further would be an incredible test to truly see the effectiveness of gamification on the learning experience. Anything that can assist in the transfer of information is a wonderful, beautiful thing, and if gamification is the future of learning, then furthering this research could be explored whole-heartedly.
References


