UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Developing an interactive space simulation game featuring procedural content generation

Bachelorarbeit

zur Erlangung des Grades eines Bachelor of Science (B.Sc.)
im Studiengang Computervisualistik

vorgelegt von

Michael Sewell
(sewell@uni-koblenz.de)

Erstgutachter:	Prof. Dr.-Ing. Stefan Müller (Institut für Computervisualistik, AG Computergraphik)
Zweitgutachter:	Dipl.-Inform. Diana Röttger (Institut für Computervisualistik, AG Computergrafik)

Koblenz, im September 2012

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Abstract

Procedural content generation, the generation of video game content using pseudo-random algorithms, is a field of increasing business and academic interest due to its suitability for reducing development time and cost as well as the possibility of creating interesting, unique game spaces. Although many contemporary games feature procedurally generated content, the author perceived a lack of games using this approach to create realistic outer-space game environments, and the feasibility of employing procedural content generations in such a game was examined. Using current scientific models, a real-time astronomical simulation was developed in Python which generates star and planets object in a fictional galaxy procedurally to serve as the game space of a simple 2D space exploration game where the player has to search for intelligent life.

Zusammenfassung

Prozedurale Synthese, das Erzeugen von Computerspielinhalten durch die Verwendung von pseudo-zufälligen Algorithmen, ist ein Themenbereich mit wachsendem Interesse in wirtschaftlichen und akademischen Kreisen, verdankt sowohl durch ihre Eignung zur Senkung von Entwicklungszeit und -kosten als auch durch die Möglichkeit, prozedurale Synthese zur Erzeugung von interessanten und einmaligen Spielwelten einzusetzen. Obwohl viele aktuelle Computerspiele prozedural generierte Inhalte verwenden, gebrauchen nur wenige diese Methoden zur Erzeugung realistischer Weltraum-Spielumgebungen, und die Umsetzbarkeit der Anwendung prozeduraler Synthese zur Entwicklung eines solchen Spiels wurde untersucht. Aktuelle Modelle aus der Forschung wurden verwendet, um eine in Echtzeit laufende astronomische Simulation in Python zu entwickeln, welche Stern- und Planetenobjekte in einer fiktiven Galaxie prozedural erzeugt, die als Spielwelt eines einfachen 2D-Weltraumspiels dient, in welchem der Spieler nach intelligentem Leben suchen muss.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Procedural content generation (PCG), specifically procedural content generation for games (also called PCG-G [Hen+11]) is tentatively defined as "programmatic generation of game content using a random or pseudo-random process that results in an unpredictable range of possible game play spaces." [Dou08c]. Doull [Dou08b; Dou10] gives a thorough and seminal discussion of the problems in defining the scope of the term. There is also a difference between the terms procedural *content* generation and procedural generation: the former is used to refer to techniques that generate elements that affect gameplay, while the latter can also be used to refer to the generation of non-gameplay elements such as lighting, textures and sound effects.

PCG techniques often imitate natural processes to generate plausible game content before and notably during gameplay. They are most often used in the areas of terrain and level generation. Compared to manually-crafted, static content, the pseudo-random nature of PCG algorithms can be used to increase the replay value of a game by offering the player new and unique content every time the game is played.

For a game to be considered procedural, randomness is a necessary, but not sufficient condition [Dou08a]. Procedural content is about exploring a set of rules underlying a seemingly random system. A procedural game "presents a *biased* randomness where discovering the underlying rules is a necessary part of play." [Dou10].

Procedural content generation remains a niche field when compared to other game-related topics such as artificial intelligence or computer graphics. But academic interest in procedural content generation is increasing (shown in Table 1). This trend is likely driven by the increased usage of procedurally generated content in the video game industry, especially for big-budget titles [Hen+11], as more game development studios come to realize that game production costs can be lowered significantly if content can be generated automatically.

Although procedural content generation is most commonly used for outdoor and indoor ("dungeon") maps ("levels") or environments [Hen+11], there are few sci-fi games set in space which feature an algorithmically created space environment based on real astrophysics. In this thesis, the feasibility of developing such a game is examined. A space exploration game called *Starship: Solitude* was developed for this purpose, set in a fictional universe which is generated (on the macroscopic level, i.e. only large-scale celestial objects are simulated) every time the game is played.

Table 1: Number of Google Scholar (scholar.google.com) search results returned for the term "procedural content generation" by year. Retrieved on 2012-07-03. Value for 2012 only up to August 3rd.

Year	Results
2002	0
2003	1
2004	1
2005	3
2006	6
2007	7
2008	20
2009	23
2010	48
2011	93
2012*	51

1.2 Prior and related work

Accrete (1969?), *Starform* (1988) and *StarGen* (1999) are consecutive iterations of scientific star system formation computer simulations based on the astrophysical theory of mass (or core) accretion [DC69]. The original algorithms these programs are based on were developed before the first discovery of exoplanets (planets outside the solar system, Table 2 shows exoplanet discovery dates). Because of this, they are based on the assumption that the planets in our solar system are representative of those in other star systems. The discovery of exoplanets very much unlike the ones in the solar system has since made clear that this is not the case and current scientific models no longer rely on the core accretion model exclusively.

Gadget 1 and Gadget 2 (2001–2005, www.mpa-garching.mpg.de/gadget), their successor *Arepo* (2009, www.mpa-garching.mpg.de/~volker/arepo), *Starburst* (2002-2005, starburst.sourceforge.net) and *Athena* (2000, trac.princeton.edu/Athena) are scientific simulations intended to run on massive supercomputers. They are used, although not exclusively, to simulate the formation of one or more galaxies over billions of years, with a single simulation run possibly taking up to a few months [Ast12].

The usage of procedurally generated universes—or parts thereof—in sci-fi video games began with the seminal 1984 space trading game *Elite*. *Elite* had the distribution of stars in its multiple galaxies and the characteristics of its planets procedurally generated, although star systems followed a very simple model: one star, one planet and one space station per system. Its 1993 successor, *Frontier: Elite II*, improved on the

Table 2: Number of exoplanets discovered by year as of 2012-08-03 [Sch95].

Year	Planets
1989	1
1992	3
1994	1
1995	1
1996	6
1998	7
1999	11
2000	19
2001	13
2002	30
2003	26
2004	31
2005	33
2006	29
2007	61
2008	61
2009	80
2010	114
2011	189
2012*	59

simulation aspect by allowing for multiple planets per systems, moons around planets, multiple-star systems, planetary day and night cycles as well as stars of all spectral classes.

A recent example of a commercially successful game featuring a procedurally generated space environment is the MMORPG *EVE Online*, first released in 2003 but still updated regularly, which features star systems that were generated using a disc accretion model in a fractally generated galaxy [Gua08]. Note that the world in *EVE Online* is static instead of dynamic (or *offline* instead of *online* [Tog+10]), meaning that it is not generated during runtime and kept in memory, as is the case for e.g. *Elite*. Instead, *EVE Online*'s world was generated once during the development phase of the game and then shipped with it, resulting in a finite game space.

Two space exploration games with procedurally generated environments currently in development are *Infiniverse* (`infiniverse-game.com`) and *Infinity: The Quest for Earth* (`infinity-universe.com`).

Space Engine (`en.spaceengine.org`) is a hobbyist space simulation software that lets the user navigate space in 3D, starting from Earth. Real astronomical data is taken into account for the known parts of the

universe, while anything beyond that is generated procedurally.

1.3 Goals and requirements

The aim of this thesis is to develop a simple space exploration game built on top of an astronomical simulation. For both simulation and game aspects, different goals have been defined.

For the simulation aspect, realism is important. Objects should be generated using pseudo-random algorithms that produce results which closely resemble those of the real-life counterparts of those objects. They should be a good fit to current astronomical data or, in the case where no observational data is available (e.g. for the presence of life on other planets) be in line with current scientific consensus.

At the outset of this thesis the feasibility of algorithmically recreating the formation and evolution of an entire universe on the macroscopic scale, requiring simulation on a timescale of billions of years ranging from the Big Bang to the eventual heat death of the universe, was examined. Additionally, since a new, different universe should be created every time a new game is started, the simulation has to run in real time.

During the early research phase, it soon became apparent that the evolution of space is too complex for the author, who is a student of a different field, to teleologically ("bottom-up", Appendix A defines this term) implement in the allotted time frame. In addition, galaxies can contain hundreds of billions of stars. Having this many stars in the game would likely not only cause difficulties in gameplay, but also in data storage and processing. The decision was made to limit the scope of generated space to a section of a single Milky Way-like galaxy that is created using a top-down approach, with stars and planets being the most relevant objects to the game, reducing the total number of generated systems to a manageable number of around 500.

The game need only be simple as its purpose is to serve as a "frontend" to the astronomical simulation running in the background by allowing a player to explore the generated universe in an appealing fashion. This exploration theme should be the main focus of the game, with other gameplay considerations being secondary, in the hope that exploring the results of the simulation should be interesting enough to provide an incentive to play the game. The attributes that define the generated stars and planet should be made visible to the player. In addition, there is another concept inherent to the space exploration theme of the game that should define gameplay: as the player character traverses the stars he covers great distances at great speeds and should experience the effects of relativistic time dilation. This effect should be made visible and understandable to the player.

Although *Starship: Solitude* is not intended to be an *educational*

game per se, to play it efficiently, the player has to understand the relation of a star's properties to the likelihood of it hosting habitable planets. This and the inclusion of time dilation, a concept of special relativity which generally exceeds high school education, suggest the game might have some educational value by teaching these astronomical concepts to the player.

Gameplay should feel realistic to the player. This means objects in the game, in particular the player's starship, should behave as expected if they were real. In the case where excessive realism is detrimental to gameplay, alterations can be made to make the game more enjoyable.

1.4 Approach

As the author is not a student of astronomy, at the outset of this thesis, intensive research was done to obtain the required knowledge about the physical attributes of celestial objects, general relativity and current beliefs on the possibility of life on other planets. After the necessary research was done, the astronomical simulation was developed. It was programmed in the Python programming language due Python's suitability for fast development, the *Pygame* game development package for Python and the author's familiarity with the language. When developing the algorithms for star and planet generation, an effort was made to use the most current research results where possible, as, particulary in the relatively new field of exoplanetology, scientific theories change rapidly and textbooks become outdated within months. The quality of these algorithms was ensured by comparing their results to the most current available astronomical data. Once the simulation had reached a mature stage, the game was built around it.

2 Game objects

There are three essential types of objects in the game: stars, planets and the player's ship. Each has a number of attributes and methods, given below. Unless otherwise stated, these formulas were translated verbatim to code.

2.1 Star objects

Stars in the *Starship: Solitude* have the following attributes generated and assigned during their creation at the beginning of the game.

2.1.1 Color Index

In astronomy, the color of a star is given by its *color index*. Color index of an object is given by the observed difference in visual magnitude of an object, using two color filters. The most common filters (and those used in the game) are a blue filter and a visible light filter, giving the *B–V color index*. An in-game star's color index and absolute magnitude values are "generated" by selecting a random star from the *HYG* database, a list of the properties of around 120,000 real stars [Nas11], extracting that star's color index and absolute magnitude (explained below) values and altering them slightly by up to ±5% for additional variety. This way, not only is the color-magnitude relation kept, but the used values are realistic. This is important because the color index and absolute magnitude are the "base" properties for a star from which all other star attributes in *Starship: Solitude* are derived, either directly or indirectly. The game resorts to picking known color index and absolute magnitude value pairs from a list because the two are related in such a way that it cannot be expressed as a simple function, as it is done for the other star attributes below. This relation is best shown by the Hertzsprung-Russell diagram (show in Figure 1), which shows that stars fall into certain bands or regions of B–V and visual magnitude value pairs.

2.1.2 Absolute Magnitude

Absolute magnitude measures the intrinsic brightness of an object at a specific distance (10 parsecs, or 32.6 light years, for stars). The value is given on a logarithmic, unitless scale, with lower numbers indicating brighter stars. The Sun has an absolute magnitude value of 4.8, for example. Stars in the game are assigned their absolute magnitude value from the color index–absolute magnitude value pairs taken from the HYG database, as explained in the description for the color index above.

2.1.3 Luminosity

Luminosity is a linear measurement of brightness, defined as the total amount of energy emitted from a star in watts W. The luminosity L in solar luminosities $L_\odot$ is calculated from the star's absolute visual magnitude $Mv_\star$ using the formula

$$\frac{L}{L_\odot} = 10^{\frac{Mv_\star - Mv_\odot}{-2.5}}$$

where $Mv_\odot = 4.83$ is the sun's absolute magnitude [Kar+07]. Even though absolute magnitude and luminosity are both measures of brightness and one can be calculated from the other, some of the formulas below require one or the other, so values are provided for both.

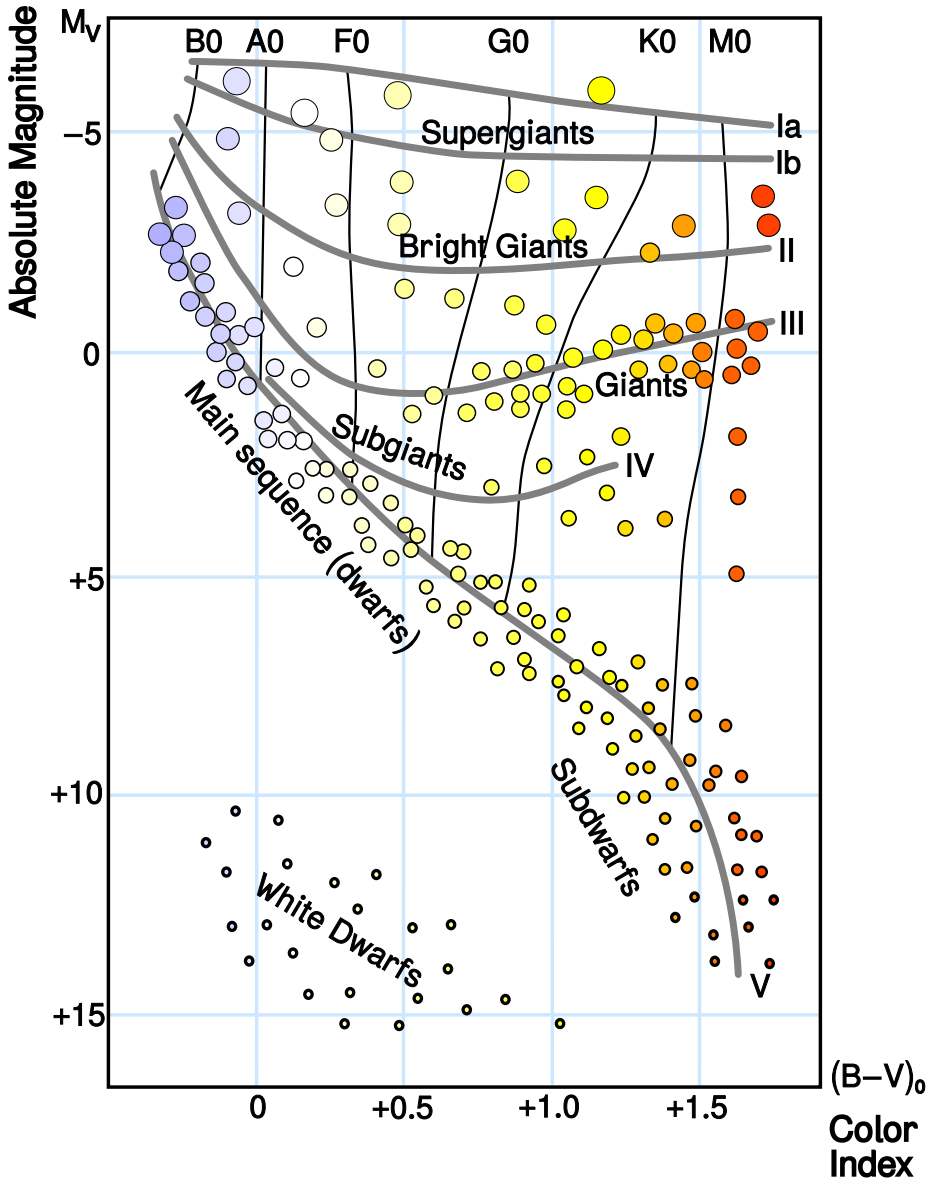


Figure 1: The Hertzsprung-Russell diagram plots magnitude against the color of stars. The different stages of stellar evolution are visible in the shape of clumps or bands of stars across the image.

2.1.4 Metallicity

Metallicity describes the elemental composition of any celestial object, but is most commonly used for stars and written as [Fe/H]. *Metal* in an astronomical context refers to all elements other than Hydrogen H or Helium He, instead of the usual meaning implying a certain electrical conductivity and chemical bonding. If a star has a high metal content, it not only implies that the star is younger (since unlike Hydrogen and Helium, the heavier elements were not produced by the Big Bang, but later by nucleosynthesis in stars or supernovae) but there also has been found a relation between the [Fe/H] content of a star and the number of planets in that star system, with a higher [Fe/H] value implying more planets in the star system. However, this correlation may not extend to giant stars, to stars of intermediate metallicity, to M dwarfs or to the occurrence of low-mass planets [Per11; Mar+05].

In the game, this relation between [Fe/H] content of the star and number of planets in the system is implemented by first generating the star's planets and then deriving the star's metallicity from the number of planets. It is important for the player to be able to see the relation between star metallicity and number of planets, as [Fe/H] content is shown when selecting a star and more planets around a star indicate a higher likelihood that some of those planets are habitable.

The in-game formula was made specifically for the game and is based on the distribution of metallicities of stars with observed exoplanets. With n being the number of planets, $\mu = 0.032\,574\,620\,1$ and $\sigma = 0.270\,866\,369\,6$ are the average and standard deviation of the star metallicities taken from a list of known exoplanets [Sch95] and $\mathcal{N}(\mu, \sigma^2)$ being the normal distribution:

$$X \sim \mathcal{N}(\mu, \sigma^2)$$
$$[\text{Fe/H}] = X - \log 2.6 + \log(n+1)$$

2.1.5 Temperature

A star's effective temperature is calculated using the formula given in [San93], where temperature is defined as a function of metallicity and B-V color index.

$$\log T = \{0.0049([\text{Fe/H}]) - 0.288\}(B-V)_{mag} - 0.002([\text{Fe/H}]) + 3.941$$

2.1.6 Coordinates

Internally, the game uses 2D floating point coordinates to place star and ship objects on the screen. A star's coordinates on the 2D grid are generated randomly on its creation, checking to make sure that stars

aren't too close to each other (at least 4 light years apart). In reality, stars don't form randomly or in isolation. Instead, they form within giant molecular dust and gas clouds and build so-called star clusters which can have thousands of members [Cen12].

2.1.7 Mass

In astronomy, there is a well-known relation between the luminosity of a star and its mass, called the *mass–luminosity relation*. Originally, an in-game star's mass M was calculated using the mass–luminosity relationships given by Kutner and Salaris and Cassisi [Kut03; SC06], who state that luminosity is related to power by a power of α:

$$L/L_\odot = (M/M_\odot)^\alpha$$

But the value of α changes across the range of stellar masses. With $M_\odot$ being the sun's mass:

$$\begin{aligned}
&\alpha = 1.8 \text{ for } M < 0.3M_\odot \\
&\alpha = 4.0 \text{ for } 0.3M_\odot \leq M < 3M_\odot \\
&\alpha = 2.8 \text{ for } 3M_\odot \leq M < 20M_\odot \\
&\alpha = 1 \text{ for } M \geq 20M_\odot
\end{aligned}$$

However, this function is unsatisfactory because it is discontinuous and gives poor results for very luminous stars as it makes them too massive. Instead, regression was performed on a set of known mass and luminosity values for binary stars (stars in multiple-star systems are the only stars for which mass can be accurately measured) given in [TAG10]. The power function obtained by regression was perceived to give very luminous stars not enough mass, and a corrective linear function was added to give the combined function

$$M/M_\odot = 0.967L^{0.255} + 5.19 \times 10^{-5}L - 0.0670$$

2.1.8 Radius

A star's radius R is calculated using the luminosity L and the effective temperature T and equals [Kar+07, page 104]:

$$R = \sqrt{\frac{L}{4\pi\sigma T^4}}$$

2.1.9 Name

Of the hundreds of billions of stars in the Milky Way, only those who appear brightest to the naked eye have historical, "real" names like *Arcturus* or *Alpha Centauri*. Most others are referred to by the star catalogue that they appear in and the number they are assigned in that catalogue. For example, *HD 1461* is the name of star 1461 in the Henry Draper star catalogue. Despite this convention, *Elite* and the *Master of Orion* series of games are examples of space games that feature randomly-generated fictional systems that assign random "real" names to all their systems. *Starship: Solitude* also generates its system names randomly, using a Markov Chain algorithm that uses 850 Basic English (http://ogden.basic-english.org/) words as its dictionary. The game rejects names already in use, giving each system a unique name.

2.1.10 RGB Color

A star's visual spectrum color is given by its temperature. Each star in *Starship: Solitude* is colored according to its temperature, based on a color lookup table that provides blackbody[1] RGB values at different temperatures [Cha01]. The table provides values for temperature values in steps of 100K, and the game interpolates between those values when assigning star color.

2.1.11 Habitable Zone

The habitable zone (HZ) of a star is defined as the range between minimum and maximum distance from that star a planet could contain liquid water. Since water is considered a prerequisite for life, it is thought that planets without liquid water would not naturally develop life.

As the star itself evolves and grows hotter, the habitable zone moves outwards and so there is also the concept of a *continuously habitable zone*, which is the range of orbital distances over which liquid water can exist continuously for long enough for life to evolve [Per11, page 238]. This view of a habitable zone does not consider the existence of alternative biochemistry or subsurface deposits of liquid water on planets too cold to sustain liquid surface water. Since pressure and solutes can change the freezing and boiling points of water, life could exist outside even outside the habitable zone. Life has been observed in liquids between −20 °C and 121 °C [Bil+06].

[1] A black body is a hypothetical physical body that absorbs all incoming electromagnetic radiation and is also an ideal thermal emitter, meaning it emits energy evenly across all frequencies based on its temperature. In astronomy, black bodies serve as close approximations of stars.

Table 3: The number of planets around a host star in Starship: Solitude *is based on a geometric distribution. This table shows the probability that there are at least n planets around a star. Systems with more than 15 planets are possible, but very unlikely.*

n	$P(X >= n)$
0	1
1	0.615
2	0.379
3	0.233
4	0.143
5	0.088
6	0.054
7	0.033
8	0.021
9	0.013
10	0.008
11	0.005
12	0.003
13	0.002
14	0.001
15	0.001

The HZ is calculated as a function of the luminosity $L/L_\odot$ and the effective temperature of the star T in K [Tor12d]. With r_i denoting the inner (closer to the star) boundary of the HZ and r_o the outer boundary:

$$r_i = \left(0.720 - 2.76 \times 10^{-5}(T - T_\odot) - 3.81 \times 10^{-9}(T - T_\odot)^2\right)\sqrt{L/L_\odot}$$
$$r_o = \left(1.77 - 1.38 \times 10^{-4}(T - T_\odot) - 1.43 \times 10^{-9}(T - T_\odot)^2\right)\sqrt{L/L_\odot}$$

2.1.12 Number of planets

In *Starship: Solitude*, the number of planets around a star is generated solely based on the assumption that on average, every star of the Milky Way hosts $1.6^{+0.72}_{-0.89}$ planets [Cas+12], leaving out other factors such as spectral type and mass of the star. The algorithm in *Starship: Solitude* for determining the number of planets is based on the assumption that not all star systems have planets and that the probability of a system having 0, 1, ..., n planets follows a geometric distribution $¶(X = n) = p(1-p)^n$ with $p = (1+1.6)^{-1}$, which gives an average planet number of $\mu = 1.6$ and a standard deviation of $\sigma = 2.04$. Table 3 shows the resulting probability that a star has a given number of planets.

2.2 Planet objects

Planets in *Starship: Solitude* have a large number of attributes generated, more than those for stars. Some of the rules for generating these attributes have been taken straight from the academic literature, while others were created using the values for the solar system planets as a baseline. The values used for the attributes of our solar system planets can be found in Table 8.

2.2.1 Mass

When the game creates a planet, first a random mass m from a list of real exoplanet masses [Sch95] is chosen and randomly altered by ±10% for variety.

But there is a problem with this approach, namely observational bias: large and massive exoplanets are easier to detect than smaller ones, and so most confirmed exoplanets fall into the former category. However, research indicates that smaller planets are likely to outnumber giant ones [Cas+12; LMP09]. To account for this, m is modified by a quadratic function created specifically for *Starship: Solitude*. The quadratic function is based on three value–value mappings:

The smallest known exoplanet mass ($0.002\,10\,\mathrm{M_J}$) is mapped to half the mass of Mercury ($8.70 \times 10^{-5}\,\mathrm{M_J}$). The average exoplanet mass ($2.90\,\mathrm{M_J}$) is mapped to the average mass of planets in the solar system ($0.176\,\mathrm{M_J}$). The assumption here is that planets in other star systems would, on average, have the same mass as those in the solar system (a claim which cannot currently be proven). The largest exoplanet mass ($31\,\mathrm{M_J}$) is mapped to 1.10 times itself, assuming the most massive discovered exoplanet is near the maximum mass a planet can have.

Taking these value pairs as x, y values of a function, we can perform quadratic regression on them using a least squares polynomial fit. The resulting polynomial gives the new planet mass M as

$$M = 0.0370m^2 - 0.0469m + 0.000\,185$$

Planets below $0.500\,\mathrm{M_\oplus}$ are unlikely to be able to maintain a life-supporting atmosphere due to their low gravity and lack of plate tectonics, and planets above $10\,\mathrm{M_\oplus}$ can attract a H-He atmosphere and become gas giants [Per11, page 285]. This is reflected in the habitability metrics defined further below.

2.2.2 Density

In the solar system, the more massive planets are generally also less dense. There is little data about the densities of exoplanets available,

so we assume that this relation also exists elsewhere. Modeling the mass–density relation of the solar system planets as a power function, we get

$$d = 1.06 \times 10^9 M^{-0.222}$$

which is an adequate, but not perfect fit around the data with a coefficient of determination $R^2 = 0.754$. The resulting d is modified randomly between ±35% (the usual value of ±10% would not allow for some mass/density values of the solar system planets).

2.2.3 Radius

The equatorial radius R of a planet is calculated using the formula for the mass of a sphere. M is the mass in kg and d is the density of the planet.

$$R = \sqrt[3]{\frac{3M}{4\pi d}}$$

2.2.4 Semi-major axis

The semi-major axis a of an ellipse is the largest radius from the center of the ellipse to its edge. It is used in astronomy when describing the orbit of a celestial object.

a can be derived by the mass and orbital period of an object. In our case, only the mass is known, and the weak relation between semi-major axis and mass is used to calculate the semi-major axis using a power function, doing regression on the known values for exoplanet masses and semi-major axes. The resulting power function is

$$a = 0.453 M^{0.494}$$

which is only a loose fit to the values with the coefficient of determination $R^2 = 0.204$. However, there is an observational bias in the known semi-major axes of exoplanets because exoplanet detection by radial velocity is more likely to find planets orbiting close to its star. As a remedy, the average $a_{avg} = 0.603\,\mathrm{AU}$, as measured by taking the average of the masses from the exoplanet list after running them through the function, is brought to the level of the solar system average $a_{\odot avg} = 8.45\,\mathrm{AU}$ by multiplying each a by $\frac{a_{\odot avg}}{a_{avg}} = 14.0184$, the underlying assumption being that the solar system is a good model for other planetary systems. Finally, the value is randomly altered by ±10% for added variety.

There are no measures in the game to make sure that the resulting orbits are actually stable or that even if they were stable, objects wouldn't collide with each other.

Table 4: Exoplanet Mass Classification (EMC) [Tor11a].

Planet Type	Mass in $M_\oplus$
Asteroidean	0 to 0.00001
Mercurian	0.00001 to 0.1
Subterran	0.1 to 0.5
Terran	0.5 to 2
Superterran	2 to 10
Neptunian	10 to 50
Jovian	50 to 5000

2.2.5 Orbital period

The orbital period of a planet around its star is calculated according to Kepler's Third Law:

$$T = 2\pi\sqrt{\frac{a^3}{G(M+M_\star)}}$$

Where a is the semi-major axis, M is the mass of the planet and $M_\star$ is the mass of the star.

2.2.6 Mass class

Starship: Solitude uses various classification schemes proposed by the Planetary Habitability Laboratory at the University of Puerto Rico at Arecibo (PHL). One of these is the classification by mass, called the Exoplanets Mass Classification (EMC) [Tor11a]. It suggests seven labels to use to describe exoplanets by mass. Table 4 shows these definitions.

2.2.7 Albedo

Albedo is a measure of the reflectiveness of a surface. A surface with an albedo value of 1 is a "white" surface (a perfect reflector), while a surface with an albedo value of 0 reflects nothing. Since there is little data available for the albedos of exoplanets, the albedos of solar system planets are used as a reference. Their albedo values are distributed with an average value $\mu = .3375$ and a standard deviation $\sigma = 0.1696$ and these values are used as the parameters of a normal deviation when generating the albedo of planets in the game. Albedo values A are only possible between 0 and 1 and if the generated albedo is out of this range, it is generated again until it fits into the range.

2.2.8 Effective temperature

The effective or equilibrium temperature of a planet is the temperature a planet would have if its star were its only heat source. This means that interior heating effects of the planet, such as heating from greenhouse gases, is ignored [Per11]:

$$T_e = T\left(\frac{R_\star}{2a}\right)^{\frac{1}{2}}(1-A)^{\frac{1}{4}}$$

Here, T is the stellar effective temperature, a is the planet's semi-major axis and A is the albedo.

2.2.9 Surface temperature

Instead of simulating a greenhouse gas effect or other forms of internal heating, we calculate the surface temperature by scaling the effective temperature by a factor f of the planet's albedo. This is done because there exists a (weak) relation between albedo and the difference in surface and effective temperature, at least for the solar system planets. To obtain a value for f, we first calculate

$$x_i = \frac{T_{s_i} - T_{e_i}}{A_i T_{e_i}}$$

for every planet of the solar system. Then we retrieve the geometric mean of these values to get f:

$$f = \left(\prod_{i=1}^{n} a_i\right)^{1/n}$$

which gives us $f = 0.517$. Then, we can get the surface temperature in K of a planet in the game by using its effective temperature and albedo values:

$$T_s = T_e + A T_e f$$

2.2.10 T-PHC

The Thermal Planetary Habitability Classification (T-PHC) is one of the exoplanet habitability classifications proposed by the PHL. It categorizes potentially habitable planets into six groups based on their surface temperature [Tor11b]. Table 5 shows which temperature ranges belong to which group.

Table 5: Thermal Planetary Habitability Classification (T-PHC) [Tor11b].

Surface temp.	Class name	Short name
−100 °C to −50 °C	hypopsychroplanet	hP
−50 °C to 0 °C	psychroplanet	P
0 °C to 50 °C	mesoplanet, Earth-like	M
50 °C to 100 °C	thermoplanet	T
100 °C to 150 °C	hyperthermoplanet	hT

Table 6: How HZC values are resolved in Starship: Solitude. *Adapted from [Tor12c].*

HZC value	HZC class name
$-1 < x$	iron
$-1 \leq x < 0$	rocky-iron
$0 \leq x < 1$	rocky-water
$1 \leq x < 2$	water-gas
$2 \leq x$	gas

2.2.11 Thermal zone

In addition to the T-PHC, the PHL categorizes exoplanets into three temperature categories which, unlike the T-PHC, can also be given for non-habitable exoplanets. These three categories are *warm* for planets in the habitable zone and *hot* and *cold* for planets closer or farther away from their star, respectively.

2.2.12 HZC

The Habitable Zone Composition (HZC) is one of the habitability metrics used by the PHL. It assigns a score based on the planet's composition by looking at how gassy it is. The formula given in [Tor12c] is:

$$r_i = 2.52 \times 10^{-0.209 + \frac{1}{3}\log\frac{M}{5.85} - 0.0804\left(\frac{M}{5.8}\right)^{0.394}}$$

$$r_o = 4.43 \times 10^{-0.209 + \frac{1}{3}\log\frac{M}{5.52} - 0.0807\left(\frac{M}{5.52}\right)^{0.375}}$$

$$\mathrm{HZC} = \frac{2R - r_o(M) - r_i(M)}{r_o(M) - r_i(M)}$$

with M and R being the mass and radius of the planet in earth units. The values are then resolved and assigned according to Table 6.

2.2.13 HZA

The Habitable Zone Atmosphere (HZA) is another exoplanet habitability metric for exoplanets proposed by the PHL. It assigns a score based on the ability of a planet to potentially hold an atmosphere [Tor12b]. The formula is:

$$v_{e\mathrm{H}} = \sqrt{0.02 T_e}$$
$$v_{e\mathrm{N}} = \sqrt{\frac{0.02 T_e}{14}}$$
$$\mathrm{HZA} = \frac{2\sqrt{M/R} - v_{e\mathrm{H}} - v_{e\mathrm{N}}}{v_{e\mathrm{H}} - v_{e\mathrm{N}}}$$

where M, R and T_e are the mass and radius of the planet in Earth units and the effective temperature in K. HZA values below -1 indicate no atmosphere, a value between -1 and 1 can sustain a $CO_2 - H_2O - N_2$ or metal-rich atmospheres of terrestrial planets and a value above 1 indicates the H – He atmosphere of gas giants.

2.2.14 Escape velocity

The speed needed to escape from the gravitational field of a planet is

$$v_e = \sqrt{\frac{2GM}{R}}$$

where M and R are the mass and radius of the planet and G is the gravitational constant [Kar+07, page 123].

2.2.15 Name

If a planet hosts intelligent life, it is assigned a random "real" name to indicate that the natives have named their planet, generated by the same Markov chain algorithm used for the naming of stars. Otherwise, planet naming attempts to follow astronomical convention: Usually, the first exoplanet discovered in a system is assigned the letter *b*, the second *c* and so on. Since all planets of a system in *Starship: Solitude* are "discovered" simultaneously, the planet closest to its star is given the letter *b*, second closest *c* and so on. For example, the third planet in the *Vox* system would be called *Vox d*.

2.2.16 HZD

The Habitable Zone Distance (HZD) is another habitability metric proposed by the PHL. It assigns a score based on the exoplanet's position

within its star's habitable zone. This is done by calculating the distance of the planet from the midpoint of the habitable zone and normalizing this value to half the width of the habitable zone [Tor12d]. The formula is

$$\mathrm{HZD} = \frac{2a - r_o - r_i}{r_o - r_i}$$

where a is the semi-major axis of the planet and r_o and r_i are the outer and inner borders of the habitable zone, respectively. A value between -1 and 1 means the planet is within the habitable zone.

2.2.17 ESI

The Earth Similarity Index (ESI) determines how Earth-like a planet is by comparing a planet's radius, density, escape velocity and surface temperature to Earth's reference values [Sch+11]. With R, d, v_e being the planet's radius, density and escape velocity in Earth units and T_s being the planet's surface temperature in K, ESI is calculated as a weighted geometric mean of the individual property similarities: [Tor12a]

$$\begin{aligned}
\mathrm{ESI}_r &= \left(1 - \left|\frac{R-1}{R+1}\right|\right)^{1.75} \\
\mathrm{ESI}_d &= \left(1 - \left|\frac{d-1}{d+1}\right|\right)^{0.935} \\
\mathrm{ESI}_v &= \left(1 - \left|\frac{v_e-1}{v_e+1}\right|\right)^{1.43} \\
\mathrm{ESI}_T &= \left(1 - \left|\frac{T_s-288}{T_s+288}\right|\right)^{0.179} \\
\mathrm{ESI} &= \left(\mathrm{ESI}_r \times \mathrm{ESI}_d \times \mathrm{ESI}_v \times \mathrm{ESI}_T\right)^{1/4}
\end{aligned}$$

Torres [Tor12a] gives a more thorough explanation of the ESI.

2.2.18 Image

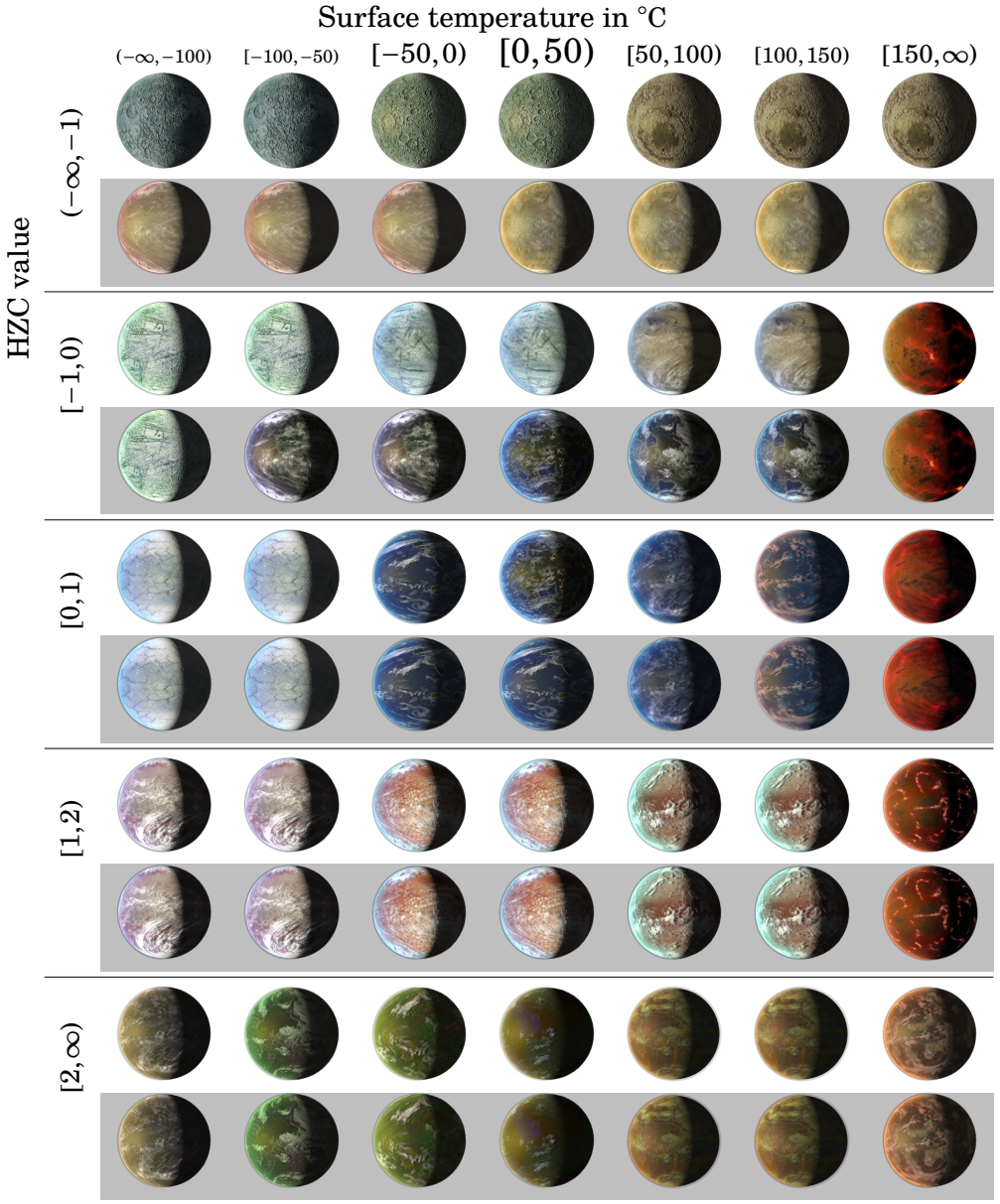
Each planet in *Starship: Solitude* is represent visually by an image assigned to it based on the planet's attributes, as explained in Table 7.

2.2.19 Habitability and life

For a planet to allow for life as we know it[2] to evolve, certain conditions must be met. Most importantly, it must allow for the presence of liquid water.

[2] In theory, life on other worlds could follow alternative biochemistries, as speculated by both science fiction and science. However, only carbon–water-based life was considered for this thesis.

Table 7: Planets in Starship: Solitude *are displayed using one of these images. Which image is used for a given planet depends on the planet's HZC value (pairs of rows in the table below), temperature value (columns) as well as the planet's atmosphere class (if the planet's atmosphere is metals-rich, the image is chosen from the rows with a gray background, otherwise from the rows with a white background). The images used for the planets are merely an artist's representation and are unlikely to be accurate depictions of the surface of such planets. This mapping was done by using the author's best guess on which image fits which attributes and aiming for an even distribution of images across attribute categories.*



A planet's temperature is determined by the distance to its star, and planets too close to their star will evaporate any water, while planets too far away will only have water in a frozen state. How far away a planet needs to be to support water depends on the star's size and temperature.

The chance for a planet in *Starship: Solitude* to allow for life to evolve (called chance of life or COL) is calculated by taking four habitability metrics into account: HZD, HZC, HZA and ESI. Since for each of these indicators, a value of 0 indicates most suitable for life[3], the combined chance of life can be calculated by taking the sum of the absolute values of the habitability metrics. Then, a value of 0 would indicate a maximally habitable and a value above 1 would indicate a non-habitable planet. This simple formula was originally used for the game, but proved too strict: very few planets would have the right attributes to score a summed habitability value of less than 1. Since a very low rate of habitable planets in the game would surely be frustrating to the player, simulation accuracy was sacrificed for gameplay and instead of the sum, the average of these values is used as the chance of life on a planet.

The result is that a lot more planets are considered habitable. To determine if a planet actually has life (which is different from just being able to support it), the COL, which ranges from 0 (no life possible) to 1 (definitely has life), is compared to a random value $x = \text{rand}(0,1)$. If $x \leq \text{COL}$, that planet has life.

If the planet is determined to have life, we have to decide what type of life it hosts. Types of life and the probability for that type to evolve is based on a few statements from an article on the PHL website:

> We expect that extraterrestrial life in exoplanets, if any, is most probably microbial life. [...] More complex life such as animals and vegetation is a second possibility. Intelligent life should be much rarer. [...] If we randomly put on stars around us 1,000 Earth replicas but at different evolutionary stages, we will find about 130 without life, 740 with only microbial life, 130 with plants and animals, and only one of those with intelligent life. [Tor12f]

Based on this, the game randomly determines the type of life with a 1 in $871 = 130 + 740 + 130$ chance of intelligent life, a 130 in 871 chance of plant and animal life and a 740 in 871 chance of only microbial life. This result is that planets with intelligent life are generated only very rarely.

[3] Except for ESI, for which habitability peaks at 1. Since ESI values only range from 0 to 1, this is easily remedied by simply using the ESI value minus 1, which gives a peak habitability at 0.

2.3 Moving at relativistic speeds

In *Starship: Solitude*, the time it takes for the player's ship to travel between stars has to be calculated relativistically, as the ship covers great distances at great speed. It propels itself forward at a constant acceleration equal to Earth's gravity g. This might not seem like much, but since the ship is accelerating continuously, after about a year it reaches speeds of $> 0.7c$, where relativistic time dilation (given by the *Lorentz factor*, defined further below) becomes noticeable.

The game keeps track of the two relevant time frames: *proper time*, as would be measured by a clock on the ship (and that the player character experiences), and the time that passes from a non-accelerating observer's point of view. While the ship is moving, the time passed and the distance travelled are continuously updated and shown to the player in the upper right corner of the screen. For this to be possible, the game has to keep track of both the distance from the point of origin (the system the ship previously halted at) to the target system, called the *total distance* d_Σ, and the distance between the ship and its target at any point in time, the *current distance* d. With these distance values, the observer time and proper time passed, the current velocity can be calculated for any point along the route. It is important to note that since the ship has to stop at the target (instead of speeding past), it has to invert the direction of its acceleration at the halfway point.

Most of the following formulas are adapted from [GK06; MTW73].

The total time t_Σ of the trip to the target destination is given by

$$t_\Sigma = 2\sqrt{\left(\frac{0.5 d_\Sigma}{c}\right)^2 + \frac{2 d_\Sigma}{a}}$$

where d_t is the total distance to the target in m, c is the speed of light and a is the acceleration.

The velocity of an object moving relativistically at constant acceleration a after t seconds is given by

$$\mathrm{v}(t) = \frac{at}{\sqrt{1 + \left(\frac{at}{c}\right)^2}}$$

However, the ship is supposed to stop at its destination and must start decelerating at the halfway point, so the formula becomes

$$v = \mathrm{v}(\min(t, 0.5 t_\Sigma)) - (\mathrm{v}(0.5 t_\Sigma) - \mathrm{v}(\min(0.5 t_\Sigma, t_\Sigma - t)))$$

The distance in m an object has moved after t seconds at constant acceleration a is given by

$$\mathrm{d}(t) = \frac{c^2}{a}\left(\sqrt{1 + \frac{at}{c}^2} - 1\right)$$

But for an object halting at its destination, this becomes

$$d = \mathrm{d}(\min(t, 0.5t_\Sigma)) + \mathrm{d}(0.5t_\Sigma) - \mathrm{d}(\min(0.5t_\Sigma, t_\Sigma - t))$$

The proper time for an object moving relativistically at constant acceleration a after d meters is given by

$$\mathrm{T}(d) = \left(\frac{c}{a}\right)\cosh^{-1}\left(\frac{ad}{c^2} + 1\right)$$

$\cosh^{-1}$ is the inverse hyperbolic cosine function. Again, since the ship will stop at its destination and must start decelerating at the halfway point, this formula becomes

$$T = \min(\mathrm{T}(d), \mathrm{T}(0.5d_\Sigma)) + \mathrm{T}(0.5d_\Sigma) - \mathrm{T}(\min(0.5d_\Sigma, d_\Sigma - d))$$

In *Starship: Solitude*, time only progresses when the ship is moving. It was decided that when the ship does move, it should be the proper time T that changes at a constant rate instead of the observer time t. This means that as the ship accelerates, Δt between game frames should increase, but ΔT stay the same. This is achieved by incrementing the in-game time by a multiple of the current time dilation, as given by the Lorentz factor [UBB05]:

$$\gamma = \frac{1}{\sqrt{1 - \left(\frac{v}{c}\right)^2}}$$

The visual effect this creates is that the distance the ship moves between each frame does not scale linearly with its velocity, as it would if Δt were constant. Since γ only increases noticeably at near-light speeds (shown in Figure 2), the distance the ship has moved on the screen follows the curve shown in Figure 3.

Taking these calculations into account, the in-game function for moving the player's ship towards its target is called every game frame while the ship is moving and does the following:

1. Determine the current Lorentz factor γ at the ship's current velocity v.

2. Increase the ship time t by $10 \times 10^5 \gamma$ seconds. The factor 10×10^5 was chosen because it moves the ship sprite along the screen at a pace that was felt by the author to be neither too fast nor too slow.

3. Because t has now changed, recalculate v using t and the total time the ship will be moving.

4. From the total distance d_Σ from the ship's former resting point to the target destination (passed as an argument to the movement function) and t, calculate the distance d the ship has moved so far.

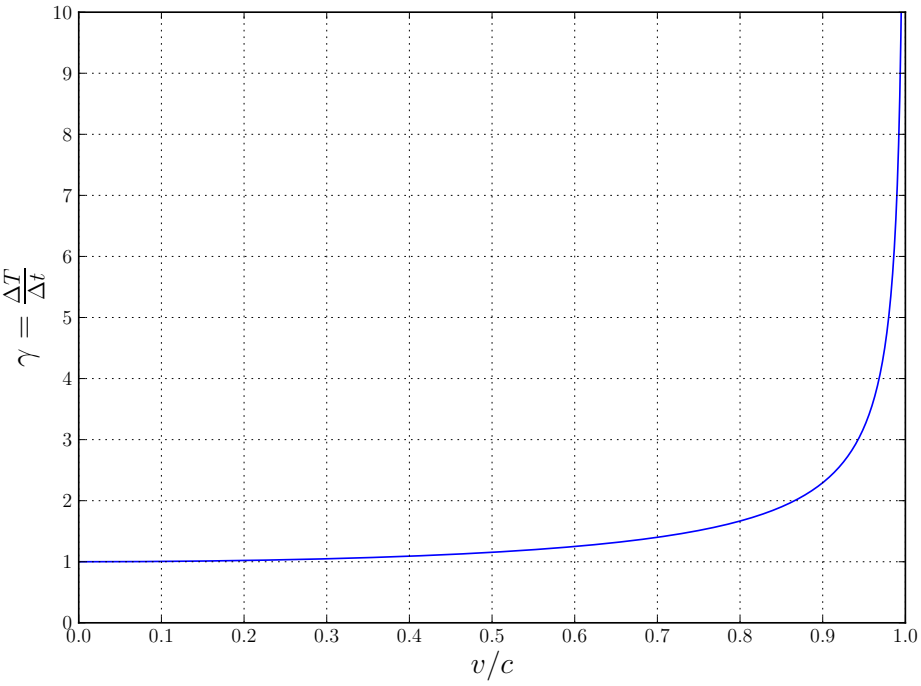


Figure 2: Plotting the Lorentz factor shows the steep increase in time dilation at near-light speeds.

5. Determine T from d and d_Σ.

6. Alternate between animation frames of the ship. Ship sprites have two frames, one with thrusters on and one with thrusters off. If the ship has reached its destination and is standing still, use the frame with thrusters off.

7. Determine the heading of the ship by calculating the vector from the ship to the destination star.

8. Using the ship's heading, rotate the ship sprite towards its destination.

9. If the ship is past the halfway point, reverse the ship sprite to show that the ship is now decelerating and pointing its thrusters "backwards".

10. Move the ship to the appropriate point on the line between the ship's former resting point (the origin) and the target destination, using the fraction d/d_Σ.

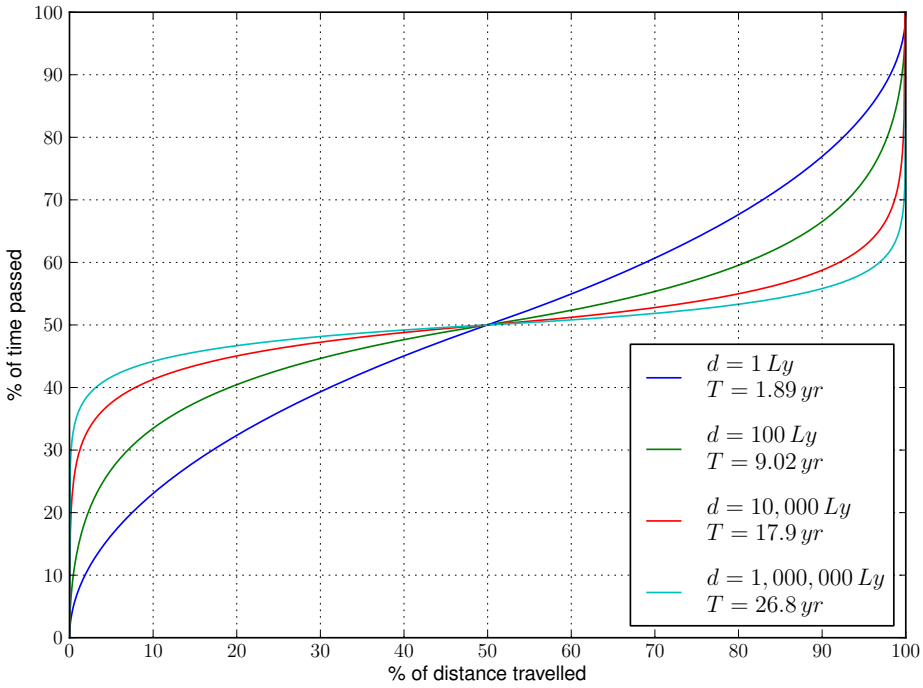


Figure 3: This is a plot of the distance covered by an object accelerating at 1 g *to proper time passed T, braking halfway to stop at the destination. The legend shows the maximum d and T values for each line, but in the graph the values have been normalized to show that the graph grows flatter near the halfway point. This is because the object is spending more of its time at near-light speeds, where the increased Lorentz factor means the object experiences a relativistic time dilation.*

3 Game concepts and interface

Starship: Solitude is written in Python, using the video game-related functions provided by the *Pygame* (`www.pygame.org`) package. The theme of *Starship: Solitude* is space exploration in a procedurally generated galaxy. There is no combat or interaction with other characters. The game either ends in a loss when the player runs out of time or in a victory when the player finds intelligent life. Any decisions the player makes during game play relate to which star system he should travel to next.

The player's avatar is a starship, a spacecraft designed for interstellar travel. This ship behaves like a "relativistic rocket", capable of sustaining a 1g acceleration, converging towards light speed. No such spacecraft propulsion technology is known to science and no explanation as to the inner workings of the ship is given in the game.

The game has an upper time limit: the lifespan of the player character.

It is necessary to have a loss condition to make the player weigh his decisions carefully, lest he lose the game. Even though travel in the game is sub-luminal (slower than the speed of light) and the distance between stars is measured in light-years, because the player's ship moves at speeds approaching c the player character experiences time dilation, giving him enough time to visit more than a few star systems before death of old age sets in.

The interface is simple and essentially consists of two different view, or gameplay, modes: in *galaxy view*, the player is shown a scrollable view of the game's "galaxy", the generated star objects randomly distributed on the game grid. The player can hover over a star to see its attributes and click on it to select it, which will draw a line from the ship to the star and show distance and travel time to that star. If the player clicks on the selected star, the ship begins travelling towards that star and the game view centers on the ship. Figure 4 shows the appearance of the galaxy view. If the player clicks on the star the player's ship is currently at, the game changes to *system view*.

As the name suggests, the system view displays the list of objects in that star system. The star itself is shown partially in large on the left side of the screen, whereas the planets are distributed horizontally along the screen. The player can hover over the star or planets to see their attributes. Additionally, a "top-down" view of the planetary orbits is shown in the bottom left on the screen which can not be interacted with. The rings of the orbits are colored according to the planet's chance-of-life value: the orbit of an uninhabitable planet is drawn in white, and habitable planets are colored from red to yellow to green depending on their chance-of-life value. The orbit view is simplified, as it does not take into account orbital eccentricity or inclination as these have no effect on gameplay. Figure 5 shows the system view. The system view was inspired by and is similar to that used in the 1996 strategy game *Star Control 3*.

To win the game, the player must find intelligent life. Instead of randomly picking a star to visit, the player should consider which star is likely to host planets with life using the attributes of that star shown when the star is selected in the system view. To know which combination of star attributes are likely to lead to habitable planets in that star system, the player has to learn these relations through experience and repeated playing of the game. Attributes like [Fe/H] and mass of a star are strongly tied to the habitability of its planets, but other attributes are also minor factors. Additionally, the distance of the player's ship to a star is important, as having to move an increased distance means sacrificing some of the remaining time in the game.

The game is played almost entirely with the mouse, save for some option toggles that require the keyboard.

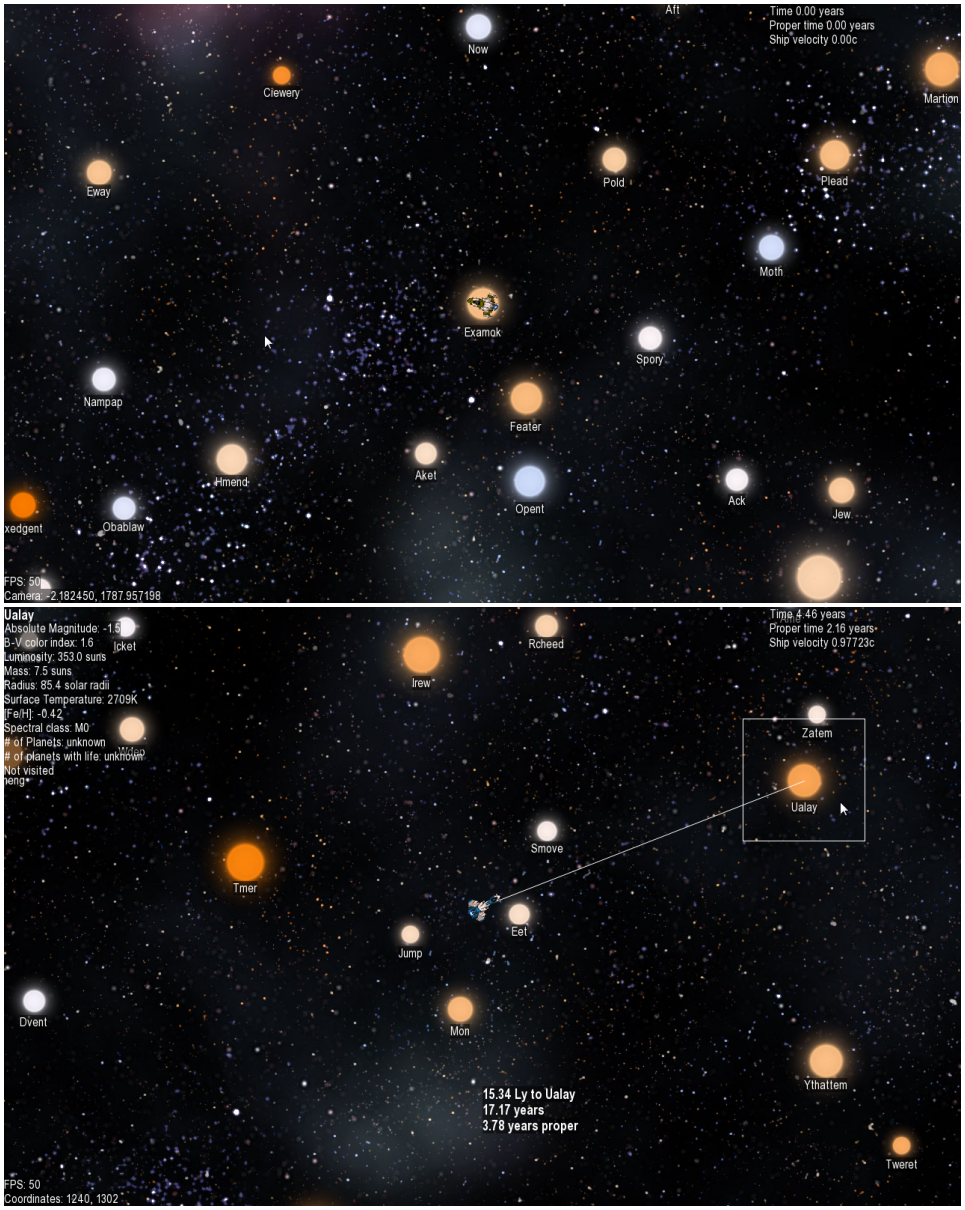


Figure 4: The two screen captures above show the galaxy view mode of Starship: Solitude*, in which the player moves from star to star. Stars are colored according to their temperature and each star has a unique name. In the top left, properties of the selected star that are relevant to gameplay are shown. As the player character does not originate from Earth, giving data in solar system units might seem odd, but it helps gameplay by giving the player known reference values. In the top right, the passed time since the beginning of the game and the current velocity of the player's ship is shown. Distance in both space and time between the player's ship and the selected star are shown in the center bottom. These values are updated in real time as the ship approaches its destination.*

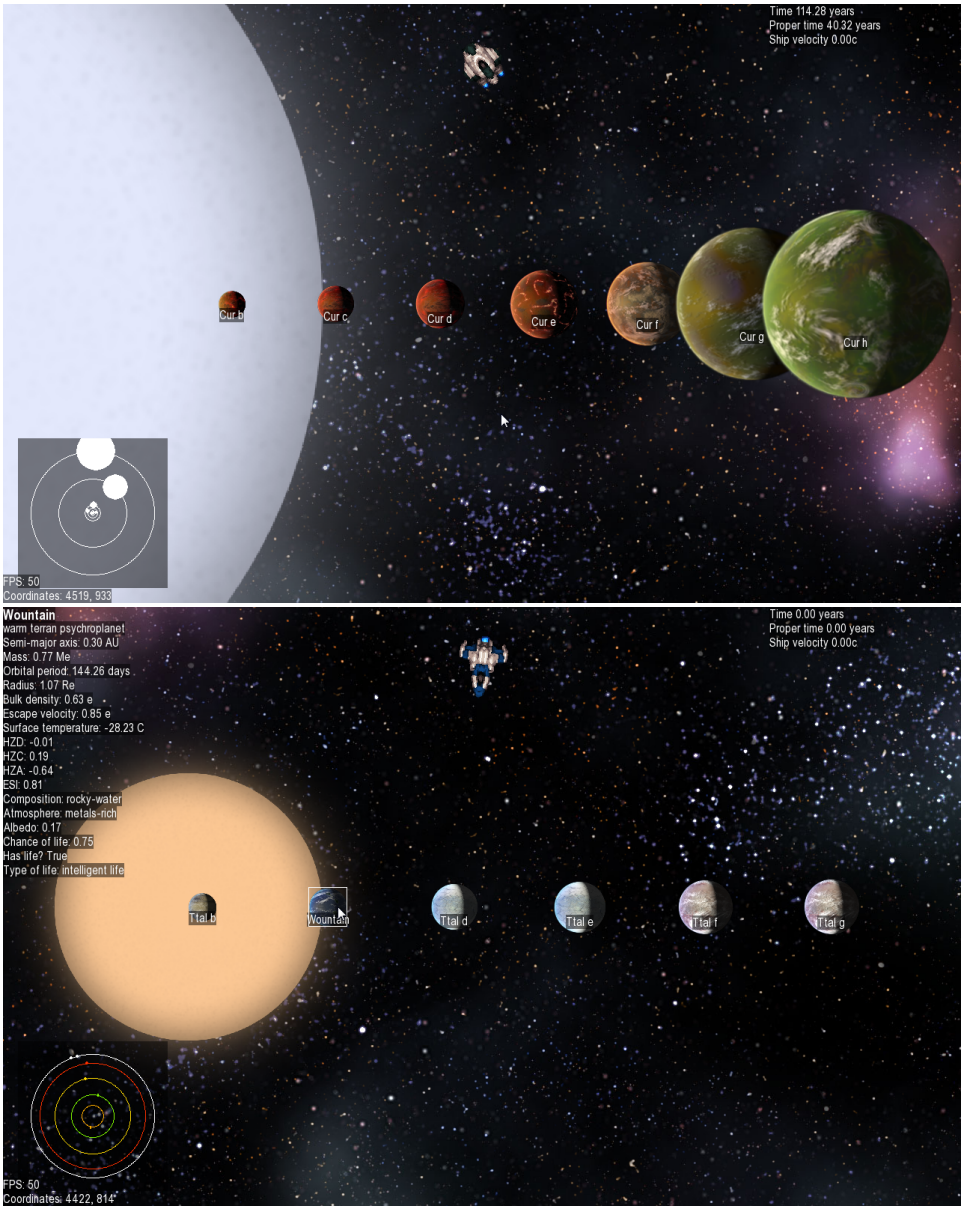


Figure 5: Two screen captures showing the game's system mode, showing the orbital bodies of the star system the player is currently at. The star is drawn at the left edge of the screen, while the planets are centered vertically and distributed horizontally. The display sizes of the star and planets are scaled according to their radii. Visible in the bottom left is a small window showing a top-down view of the orbits in the system. Again, the size of the planets are scaled according to their radii, while the color of the orbit is determined by the planets' chance-of-life attribute. A white orbit means that planet is inhabitable. Habitable planets are colored red to yellow to green with increasing probability of life.

4 Results and evaluation

A simple game with a space exploration theme was developed which uses pseudo-random procedural content algorithms to generate a new game space each time a new game is started. Generation of 500 stars with 800 planets takes only a few seconds on the author's computer. The result has already been shown in Figure 4 and Figure 5. Star and planetary attributes are clearly shown to the player, as well properties that relate to interstellar travel, such as speed, distance and time. Although the game was developed to serve as an accessible interface to the simulation running behind the scenes, it was preferable for the game to also have merit as a *game*, i.e. to be entertaining or educational. Although no user testing has been performed, it is the opinion of the author that *Starship: Solitude* in its current state does not offer a sustained, enjoyable experience. Where the educational aspect is concerned, there might be some value in showing the relations between the properties of a star and the properties of the planets that orbit it, the relation between planetary attributes and planetary habitability, and the scarcity of intelligent life in the universe in general. However, the total educational value is likely negligible compared to that of other hobbyist astronomy software or a textbook on the topic.

The properties of star and planet objects are generated by procedural algorithms that follow the relationships between these properties observed in real celestial objects. Where possible, formulas were used from textbooks or journal articles, otherwise they were created for the game by e.g. performing regression on a set of observed property value pairs. In the latter case, these formulas can be said to be as "accurate" as their coefficient of determination of the function obtained by regression compared to the data set the regression was performed on. In general, while the game is able to create star systems similar to those found in the Milky Way, not all systems generated would be possible in reality as some important considerations are glossed over, such as the age or gravitational interaction of objects. Most generated star systems, however, are plausible and the application can be considered a "real" simulation, although vastly simplified when compared to those used in contemporary science.

5 Conclusions

The primary goals defined at the outset of this thesis, namely the design and implementation of procedural content generation algorithms for stars and planets and the design and implementation of a game based around the simulated universe, were fulfilled. A player can experience interesting

astronomical concepts through the game. During development however, it became clear that a game is not simply enjoyable because it is realistic, and it is doubtful whether a casual player would be interested in the game for more than the few minutes it takes to learn it in its current state.

The feasibility of generating an outer space game environment using real-time procedural content generation was examined by creating an example game where star and planet objects are generated through simple pseudo-random algorithms. Although not nearly as accurate as scientific star system formation simulations, the results are sufficiently plausible to be used in a game that aims to have some degree of realism in its outer space setting.

6 Ideas for future development

Any game with a futuristic space setting can select features and concepts from a near-limitless taken from both science and science fiction. What follows is a list of some of the ideas that were considered for *Starship: Solitude*, but were not implemented in the final release, mostly due to time constraints.

- Originally, the game was going to take place over a much larger time frame of thousands or millions of years or more. The idea behind this was that it would have been interesting to see stars, planets and civilizations change with time. Over macroscopic time scales, life could evolve or vanish, planets and stars could age and new ones form. Currently, a game takes place over a few hundred years at most, not nearly long enough for these changes to be visible. For this feature to be in the game, a larger galaxy and the evolution of these objects would have to be simulated.

- The current algorithm for planet generation does not check if the resulting orbital system is actually possible or if planets would collide or collapse into the sun. An algorithm that balances gravitational interaction to create stable orbits could be created.

- Although the orbital eccentricity is an important factor in planetary habitability as temperatures can vary drastically with increasing eccentricity, it is not accounted for in the game and all planet temperature values are assumed to be averages. It would be preferable if eccentricity was also taken into account as a planetary habitability metric.

- There is no observational data about extrasolar moons, of which we can assume there to be many, and as such they are currently

not implemented. Even if a planet isn't habitable, one of its moons might be, particularly in the case of larger planets.

- Binary (trinary, ...) star systems are common, with about a third of star systems being multiple [Lad06]. It is not known if habitable zones around binary and trinary stars are possible [Bil+06, page 788]. However, there is a lack of statistical data concerning the occurrence of planets around multiple stars and their dynamical stability and the existing data for single star systems cannot simply be transferred to multiple stars. So for the development of *Starship: Solitude*, systems with multiple stars were left out of the game.

- Currently, all objects are held in memory and the game map is limited in all directions. It is desirable to have an infinitely scrollable map, but the way objects are handled would have to be changed significantly to allow for this. The PCG algorithms would have to be changed from a stochastic (random) to a deterministic (based on a seed) generation to allow for persistent objects to be loaded in and out of memory.

- *Starship: Solitude* currently only has a minimal plot ("You are in a spaceship and have to search for intelligent life"). The telling of an interesting story would add to the enjoyment of the game.

- Celestial objects other than stars and planets could be featured in the game. Black holes, nebulas, supernovae, asteroids, asteroid belts, etc.

- A common feature in games are *power-ups* that enhance the player character's abilities. In *Starship: Solitude*, the player could find advanced technologies within the ruins of an extinct civilization that would serve as upgrades for his ship which could increase such attributes as the ship's acceleration or increase the remaining game time.

- When a game ends, the player's performance could be evaluated and put on a high score list. The score could be determined by such factors as how soon the player found intelligent life, how many "lesser" life forms he found on the way and other factors.

- Procedurally generated textures for stars and planets would add more variety to the game and would fit in with its general PCG theme.

- Right now, finding life in *Starship: Solitude* is very dependent on luck. It is desirable to give the player some way of knowing in advance which stars and planets host life. We can look at the SETI

methods currently used on Earth for inspiration: the player could look for radio, microwave or optical trans- or emissions that would indicate advanced intelligent life. Spectroscopic analysis of planet atmospheres could detect biosignatures such as O_2 (Oxygen is very reactive and the Earth's supply would quickly be absorbed by its surface if it wasn't for plants and microbial life continuously producing more) or Ozone (O_3) [Per11, page 288] or other habitability indicators such as H_2O.

- The number and type of planets around a star are currently only revealed once the player's ship has arrived in the system. The player could be able to use current astronomical methods to detect far-away planets.
- The issue of the player's ship's fuel supply is currently conveniently ignored as the type of propulsion of the ship is never explained. The game could be made more interesting by adding a resource management aspect that would have the player be concerned about his fuel supply and would need him to restock it occasionally somehow.

References

[Ast12] Harvard-Smithsonian Center for Astrophysics. *Recreating a Slice of the Universe*. Press release. 2012-09-15. URL: http://www.cfa.harvard.edu/news/2012/pr201223.html (visited on 2012-08-23).

[Bil+06] L. Billings et al. "The Astrobiology Primer: an outline of general knowledge—Version 1, 2006". In: *ICES Journal of Marine Science* 68.2 (2006), pp. 341–348.

[Cas+12] A. Cassan et al. "One or more bound planets per Milky Way star from microlensing observations". In: *Nature* 481.7380 (2012-01-12), pp. 167–169. ISSN: 0028-0836. DOI: 10.1038/nature10684. (Visited on 2012-08-06).

[Cen12] Chandra X-ray Center. *Young Stars and Star Clusters*. Harvard-Smithsonian Center for Astrophysics. 2012-08-10. URL: http://chandra.harvard.edu/xray_sources/young_stars.html (visited on 2012-08-21).

[Cha01] Mitchell Charity. *Blackbody color datafile*. 2001-06-22. URL: http://www.vendian.org/mncharity/dir3/blackbody/ (visited on 2012-08-07).

[DC69] S.H. Dole and Rand Corporation. *Formation of planetary systems by aggregation: A computer simulation*. Rand Corporation, 1969.

[Dou08a] Andrew Doull. *Minesweeper vs. Solitaire*. 2008-05-19. URL: http://roguelikedeveloper.blogspot.de/2008/05/minesweeper-vs-solitaire.html (visited on 2012-09-03).

[Dou08b] Andrew Doull. *The Death of the Level Designer: Procedural Content Generation in Games*. 2008-01-14/2008-01-28. URL: http://roguelikedeveloper.blogspot.de/2008/01/death-of-level-designer-procedural.html (visited on 2012-08-06).

[Dou08c] Andrew Doull. *What PCG is*. Retrieved on 2012-08-03. 2008-05. URL: http://pcg.wikidot.com/what-pcg-is (visited on 2012-08-06).

[Dou10] Andrew Doull. *Proceduralism*. 2010-05-31/2011-08-25. URL: http://roguelikedeveloper.blogspot.de/2010/05/proceduralism-part-one-revision.html (visited on 2012-08-06).

[GK06] Philip Gibbs and Don Koks. *The Relativistic Rocket*. 2006. URL: http://math.ucr.edu/home/baez/physics/Relativity/SR/rocket.html (visited on 2012-08-29).

[Gua08] H.F. Guajónsson. “The server technology of EVE Online: How to cope with 300,000 players on one server”. In: *Proc. Austin GDC*. 2008. URL: http://gdcvault.com/play/109/The-Server-Technology-of-EVE.

[Hen+11] M. Hendrikx et al. “Procedural Content Generation for Games: A Survey”. In: *ACM Transactions on Multimedia Computing, Communications and Applications* (2011).

[Kar+07] H. Karttunen et al. *Fundamental Astronomy*. Springer-Verlag Berlin Heidelberg, 2007. ISBN: 9783540341444.

[Kut03] M.L. Kutner. *Astronomy: A Physical Perspective*. Cambridge University Press, 2003. ISBN: 9780521529273.

[Lad06] C.J. Lada. “Stellar multiplicity and the initial mass function: most stars are single”. In: *The Astrophysical Journal Letters* 640 (2006), p. L63.

[LMP09] J.I. Lunine, B. Macintosh, and S. Peale. “The detection and characterization of exoplanets”. In: *Phys. Today* 62.5 (2009), pp. 46–51.

[Mar+05] G. Marcy et al. “Observed Properties of Exoplanets: Masses, Orbits, and Metallicities”. In: *Progress of Theoretical Physics Supplement* 158 (2005), pp. 24–42. DOI: 10.1143/PTPS.158.24. eprint: arXiv:astro-ph/0505003.

[MP00] D. Macri and K. Pallister. *Procedural 3D content generation*. 2000. URL: http://goo.gl/YRYdv.

[MTW73] C.W. Misner, K.S. Thorne, and J.A. Wheeler. *Gravitation*. Physics Series pt. 1. W. H. Freeman, 1973. ISBN: 9780716703341.

[Nas11] David Nash. *HYG Database*. 2011-10. URL: http://www.astronexus.com/node/34 (visited on 2012-08-03).

[Pat10] Amit Patel. *Teleological vs. ontogenetic map generation*. 2010. URL: http://simblob.blogspot.de/2010/06/teleological-vs-ontogenetic-map.html (visited on 2012-08-06).

[Per11] M. Perryman. *The Exoplanet Handbook*. Cambridge University Press, 2011. ISBN: 9780521765596.

[San93] A. Sandage. “Temperature, mass, and luminosity of RR Lyrae stars as functions of metallicity at the blue fundamental edge. II”. In: *The Astronomical Journal* 106 (1993-08), pp. 703–718. DOI: 10.1086/116676.

[SC06] M. Salaris and S. Cassisi. *Evolution of Stars and Stellar Populations*. John Wiley & Sons, 2006. ISBN: 9780470092200.

[Sch+11] D. Schulze-Makuch et al. “A Two-Tiered Approach to Assessing the Habitability of Exoplanets”. In: *Astrobiology* 11 (10 2011-12-20). URL: http://online.liebertpub.com/doi/abs/10.1089/ast.2010.0592 (visited on 2012-08-11).

[Sch95] Jean Schneider. *The Extrasolar Planets Encyclopaedia*. 1995-02. URL: http://exoplanet.eu/ (visited on 2012-08-03).

[TAG10] G. Torres, J. Andersen, and A. Giménez. “Accurate masses and radii of normal stars: modern results and applications”. In: *Astronomy and Astrophysics Review* 18.1 (2010), pp. 67–126.

[Tog+10] J. Togelius et al. “Search-based procedural content generation”. In: *Applications of Evolutionary Computation* (2010), pp. 141–150.

[Tor11a] Abel Méndez Torres. *A Mass Classification for both Solar and Extrasolar Planets*. 2011-08-16. URL: http://phl.upr.edu/library/notes/amassclassificationforbothsolarandextrasolarplanets (visited on 2012-08-11).

[Tor11b] Abel Méndez Torres. *A Thermal Planetary Habitability Classification for Exoplanets*. 2011-08-09. URL: http://phl.upr.edu/library/notes/athermalplanetaryhabitabilityclassificationforexoplanets (visited on 2012-08-11).

[Tor12a] Abel Méndez Torres. *Earth Similarity Index (ESI)*. 2012. URL: http://phl.upr.edu/projects/earth-similarity-index-esi (visited on 2012-08-11).

[Tor12b] Abel Méndez Torres. *Habitable Zone Atmosphere (HZA): A habitability metric for exoplanets*. Planetary Habitability Laboratory @ University of Puerto Rico at Arecibo. 2012-06-30. URL: http://phl.upr.edu/library/notes/habitablezoneatmospherehzaahabitabilitymetricforexoplanets (visited on 2012-08-11).

[Tor12c] Abel Méndez Torres. *Habitable Zone Composition (HZC): A habitability metric for exoplanets*. 2012-07-01. URL: http://phl.upr.edu/library/notes/habitablezonecompositionhzcahabitabilitymetricforexoplanets (visited on 2012-08-11).

[Tor12d] Abel Méndez Torres. *Habitable Zone Distance (HZD): A habitability metric for exoplanets*. 2012-07-30. URL: http://phl.upr.edu/library/notes/habitablezonesdistancehzdahabitabilitymetricforexoplanets (visited on 2012-08-11).

[Tor12e] Abel Méndez Torres. *HEC: Data of Potential Habitable Exoplanets and Exomoons*. 2012-08-04. URL: http://phl.upr.edu/projects/habitable-exoplanets-catalog/data (visited on 2012-08-15).

[Tor12f] Abel Méndez Torres. *HEC: Introduction to Habitable Worlds. Potential Habitable Worlds*. 2012. URL: http://phl.upr.edu/projects/habitable-exoplanets-catalog/introduction (visited on 2012-08-11).

[UBB05] A. Unsöld, B. Baschek, and W.D. Brewer. *The New Cosmos: An Introduction to Astronomy and Astrophysics*. Springer, 2005. ISBN: 9783540678779.

[Wes07] Mick West. *Teleological vs. Ontogenetic*. 2007-01. URL: http://cowboyprogramming.com/2007/01/02/teleological-vs-ontogenetic/ (visited on 2012-08-06).

Appendix A Taking an ontogenetic (top-down) approach to galactic formation and evolution

When attempting to algorithmically recreate natural processes, developers using PCG techniques have used exotic terms to describe two distinct approaches: *teleological* and *ontogenetic*, originally terms used in metaphysics and evolutionary biology, respectively. These terms seem to have first been used in a PCG context in an article for Intel in 2000 [MP00], in which they are defined this way:

> To understand the difference between the teleological and ontogenetic approaches, imagine the modeling of clouds. The teleological approach would model the properties of water, its evaporation, temperature of the environment, and so on, to try and produce the desired end result from the bottom up. The ontogenetic approach would be to observe properties of the end result (such as the small wispy bits change more frequently than the larger bits, the shape changes more as the wind picks up, low-lying cloud tends to be darker, and so on). [T]he ontogenetic approach [...] is more suitable for most real-time applications. [MP00]

Not all game developers find these terms appropriate [Wes07; Pat10] and suggest using "bottom up" instead of teleological and "top down" instead of ontogenetic. Terminology aside, knowing the distinction of these two different approaches to simulating natural processes is important.

For this thesis, the accurate simulation of the formation and evolution of the game's galaxy—the teleological approach—was mostly omitted for the sake of both development and processing time. Instead, most attributes of stars and planets are generated top-down or "as-is", without taking into account the evolutionary processed behind them.

Appendix B Additional tables

Table 8: Astronomical values of the solar system planets used in some of the calculations made for Starship: Solitude. *Values taken from wolfram|alpha.*

Planet	m/kg	r/km	d/g cm^{-3}	v_e/m s^{-1}	a/AU	T_{surf}/K	T_{eff}(K)
Mercury	3.302×10^{23}	2440	5.43	4250	0.39	452	434
Venus	4.869×10^{24}	6052	5.24	10 360	0.72	733	230
Earth	5.972×10^{24}	6368	5.515	11 180	1.00	287	254
Mars	6.419×10^{23}	3386	3.94	5020	1.52	226	212
Jupiter	1.899×10^{27}	69 173	1.33	59 540	5.20	165	124
Saturn	5.685×10^{26}	57 316	0.7	35 490	9.54	134	95
Uranus	8.663×10^{25}	25 266	1.3	21 290	19.19	62	59
Neptune	1.028×10^{26}	24 553	1.76	23 710	30.07	64	60

Table 9: Planetary habitability metric values for the solar system planets. Taken from the PHL website [Tor12e].

Planet	ESI	HZD	HZC	HZA
Mercury	0.596	−1.46	−0.52	−1.37
Venus	0.444	−0.93	−0.28	−0.70
Earth	1	−0.50	−0.31	−0.52
Mars	0.697	0.33	−0.13	−1.12
Jupiter	0.292	6.12	7.13	8.12
Saturn	0.246	12.95	6.43	5.14
Uranus	0.187	28.15	2.71	3.11
Neptune	0.184	45.28	2.31	4.23

Appendix C Image attributions

- Figure 1: Image by Wikimedia Commons user Rursus, licensed under the CC-BY-SA-3.0 (creativecommons.org/licenses/by-sa/3.0) license.
- Table 7 and Figure 5: Planet images by Mathias Koehler, optisch-edel.de, licensed under the CC-BY-SA 3.0 license.
- Figures 4 and 5: Original starship sprites by Carl Olsson, http://opengameart.org/content/shmup-ships, licensed under the CC-BY-SA 3.0 license.