Hullbreaker

A poster of a space ship

Description automatically generatedBreak them apart!

# 2. Table of Contents

Contents

[2. Table of Contents 2](#_Toc152771945)

[3. Introduction 4](#_Toc152771946)

[3.1. Concept 4](#_Toc152771947)

[3.2. Feature Set 4](#_Toc152771948)

[3.3. Genre(s) 4](#_Toc152771949)

[3.4. Target Audience 4](#_Toc152771950)

[3.5. Game Flow 4](#_Toc152771951)

[3.6. Look and Feel 4](#_Toc152771952)

[3.7. Scope 4](#_Toc152771953)

[3.7.1 Complexity 4](#_Toc152771954)

[3.7.2 Novelty 4](#_Toc152771955)

[3.7.3 Breadth 4](#_Toc152771956)

[4. Investigation of Project Idea 5](#_Toc152771957)

[4.1 Literature Review 5](#_Toc152771958)

[4.1.1 Procedural Generation REWORD + P-GEN 5](#_Toc152771959)

[4.1.2 Dynamic Difficulty 6](#_Toc152771960)

[4.1.3 Card-Based Combat REDO 7](#_Toc152771961)

[4.2. Feasibility 8](#_Toc152771962)

[4.2.1. Market 8](#_Toc152771963)

[4.2.2. Similar Products 8](#_Toc152771964)

[4.2.3. Unique Selling Points 9](#_Toc152771965)

[4.3. Technical Investigation 9](#_Toc152771966)

[4.3.1. Target hardware for Deployment 9](#_Toc152771967)

[4.3.2. Development hardware and software 10](#_Toc152771968)

[4.3.3. Game Engine 10](#_Toc152771969)

[4.3.4. Scripting Language 10](#_Toc152771970)

[4.3.5. Development procedures 10](#_Toc152771971)

[5. Project Management 10](#_Toc152771972)

[5.1. Project Methodology 10](#_Toc152771973)

[5.2. Detailed Schedule 11](#_Toc152771974)

[5.3. Schedule Management 12](#_Toc152771975)

[5.4. Version Control 12](#_Toc152771976)

[5.5. Risk Analysis 12](#_Toc152771977)

[5.6. Test Plan 12](#_Toc152771978)

[6. Project Design 12](#_Toc152771979)

[6.1. Gameplay 12](#_Toc152771980)

[6.1.1. Game Progression 12](#_Toc152771981)

[6.1.2. Mission/Challenge Structure 13](#_Toc152771982)

[6.1.3. Objectives 13](#_Toc152771983)

[6.1.4. Play Flow 13](#_Toc152771984)

[6.2. Mechanics 13](#_Toc152771985)

[6.3. Screen Flow 13](#_Toc152771986)

[6.4. Game Options 13](#_Toc152771987)

[6.5. Replaying and Saving 13](#_Toc152771988)

[6.6. Story and Narrative 13](#_Toc152771989)

[6.7. Game World 13](#_Toc152771990)

[6.8. Characters 13](#_Toc152771991)

[6.9. Levels 13](#_Toc152771992)

[6.10. Interface 13](#_Toc152771993)

[6.11. Artificial Intelligence 13](#_Toc152771994)

[6.12. Game Art 13](#_Toc152771995)

[7. Appendices 13](#_Toc152771996)

[7.1. Credits 13](#_Toc152771997)

[7.2. References 13](#_Toc152771998)

# 3. Introduction

## 3.1. Concept

## 3.2. Feature Set

## 3.3. Genre(s)

## 3.4. Target Audience

## 3.5. Game Flow

## 3.6. Look and Feel

## 3.7. Scope

### 3.7.1 Complexity

HullBreaker will feature various complex mechanics such as:

* Procedurally generated solar systems which will contain many unique and generated planets and events to keep the game feeling new and to increase replayability.
* Dynamically scaling difficulty that adapts to how far the player has progressed and how well they are doing to give a more challenging and engaging experience.
* A unique upgrade system via the player being able to salvage parts from defeated enemies to add on to their own ship to provide various bonuses such as passive effects, new attacks and actions or additional stats.
* A card-based combat system where the parts on the players ship determines the cards in players combat deck offering many different decisions for players when it comes to ship construction and deck building.

### 3.7.2 Novelty

HullBreaker will do two things differently from standard card battler roguelikes, feature a unique deck building system based around modular ship building and having an open world navigation map free for the player to traverse fully.

HullBreakers deck building system is one that is completely unique to itself and is a mechanic not seen in other games of its genre. The modular deck building system adds additional layers of depth to the players decision making and deck building strategy.

Open world navigation is also a mechanic not commonly seen in roguelike games. Many roguelikes follow a linear stage progression system, but HullBreaker makes use of a single large scale, procedurally generated map that allows the player to freely explore previously visited routes and conquer all encounters in each zone before progressing.

### 3.7.3 Breadth

HullBreaker will feature three areas the player can traverse in the large world map, The Inner System, The Middle System and The Outer Worlds. Each area will have a variety of encounters the player may come across such as enemy ships, broken vessels, random events, and an area boss which must be defeated before the player can travel to further areas.

HullBreakers deck unique deck constructing mechanic features many ship parts and cards provided by said parts. Ship parts come in a variety of sizes and can provide many different effects when attached to the players ship, these effects include but are not limited to:

* Adding additional cards to the player’s deck.
* Improving cards of a specific type.
* Providing passive in-combat effects (Additional health, Enemy debuffs, etc.)
* Providing out-of-combat effects (Larger map navigation vision, ability to see upcoming encounters, etc.)

The cards that the player will be using during combat also have a multitude of effects including ones such as:

* Dealing damage.
* Blocking damage.
* Healing.
* Buffing and Debuffing.
* And many more.

While navigating the game world the player can also encounter areas such as shops for ship parts, vendors to sell their extra ship parts, random events, enemies, mini bosses and area bosses. This world the player navigates is a large solar system map split up into three concentric ring areas which make up the world of HullBreaker.

When defeating enemies, the player will be given the option to salvage there fallen enemies ship to acquire additional parts for themselves. The quality of these parts varies from enemy to enemy and area to area with mini bosses providing stronger parts and area bosses providing an artifact, strong single time attainable powerups that do not take up a slot in the player’s ship.

# 4. Investigation of Project Idea

## 4.1 Literature Review

### 4.1.1 Procedural Generation REWORD + P-GEN

Procedural Generation, in simple terms, is the creation of data by computers. This data can be theoretically anything but where procedural generation is most used is the creation of content for video games and various media. Procedural generation is most commonly used for the creation of content and assets which possesses random elements or would be tedious to create manually. In the modern-day procedural generation is mostly associated with the generation of maps and terrain in games such as “Minecraft” and “Terraria”. It is also heavily associated with the “roguelike” genre of games, having become a core aspect of the genre over the past four decades with some of the first games of the genre making use of procedural generation for level generation, the most notable of which being “Rogue” the early 1980’s dungeon crawler known for inspiring the genre.

There are many benefits to generating the levels / stages of a game procedurally with some of the most prominent being added replay value, not needing to spend development time on level design and reduced memory usage. The arguably greatest benefit of the above listed is the added replay value. Having a games stage be randomly generated means the player is all but guaranteed to never see the same stage twice. This does wonders for keeping games fresh and when combined with an equally enthralling game play loop can lead to creating a near endless experience for the player.

On the opposite side of this spectrum however lies the downsides of procedural generation with the most notable of them being increased randomness, difficulty adding scripted events and it being more taxing on hardware. All though adding randomness to a game can increase it’s replay value it can also act as great detriment towards the balance of the game. The generation could cause scenarios where the player is unable to progress or even miss important events. A game that demonstrates the rights and wrongs of procedural generation would be “Risk of Rain”.

“Risk of Rain” is a roguelike where the stages the player must progress are procedurally generated, with treasure and the exit being randomly place across the stage. The game is praised for its addictive gameplay and difficulty, but it has also seen its fair share of detraction due to how the generation of its treasure and, most importantly, stage exits can at times be inaccessible. This makes the game somewhat of a balancing nightmare due to its heavy utilization of procedural generation, sometimes causing the player to receive an overwhelming number of items by the end of the first stage or to get close to none on subsequent playthroughs. This proves that too much variance can be a detriment to games that wish to incorporate procedural generation. The developers of “Risk of Rain” did correct this issue in the sequel “Risk of Rain 2” by reducing what areas of stages could be procedurally generated, guaranteeing a minimum number of treasure chest that can spawn on each stage and by limiting the positions the stage exit could spawn on to only several positions on the stage that player is assured to reach.

This style of procedural generation with limits works wonders by still allowing the game to be varied while not causing situations in which the game is unbeatable.

HullBreaker will use this limited procedural generation to create the navigation system for the stages of the game as it will help to create the addictive gameplay loop desired for the game and assist in retaining control of game balance.

### 4.1.2 Dynamic Difficulty

Dynamic Difficulty is the process of adjusting the various behaviours and stats in a game based on the performance of the player. These adjustments can be as simple as increasing enemy health and speed to addition of completely new enemies and scenarios depending on how well the player is performing. The players performance can be judge off a number of different statistics and many games have taken various different approaches to this. Some examples statistics that can be indicative of how challenged the player can be by the game include:

* How much the health the player has lost.
* The players average damage dealt to enemies.
* The time it takes the player to overcome and encounter.
* The number of upgrades or power-ups a player possesses.

Etc.

One or more of these above statistics can be allocated a score and then various enemy or world properties can be increased, or even potentially decreased, proportionally to that score. This is excellent way of adding additional challenge to a game and even potentially works to let new players ease into the game if enemies become slightly weaker if they do not perform well.

This way of measuring difficulty how ever can cause issues if not enough data is taken in to determine how well a player is doing. An example of this would be the dynamic difficulty in racing game “Mario Cart”. The further a player is to first place in “Mario Cart” the lower quality of items they will receive, with first place only being able to obtain two types of items. This form of single facetted difficulty scaling encourages a style of play where the player may choose to intentionally slow down to increase the quality of the items they receive. This can be in a positive and negative light as it is a form of strategic play, but it is also abusing the games dynamic difficulty to give the player an advantage they otherwise would not have.

4.1.3 Card-Based Combat REDO  
In the past decade, games, and especially roguelikes, have been utilizing new and different combat systems. Some very notable ones being “The Binding of Isaac's” top-down shooter style of combat and the game “Peglin” with its combat system being very reminiscent of the puzzle game known as “Peggle”. The most popular and interesting form of the combat system in these new age roguelikes would have to be the card game-based system. The most popular example of which, and the main inspiration for HullBreaker, being “Slay the Spire”. In this game, the player is tasked with progressing through a tower to ultimately defeat the boss that lies at  
the top. The combat in “Slay the Spire” consists of the player entering a fight and drawing cards from their deck. These cards have a multitude of different effects varying from dealing damage, healing, gaining shield, adding temporary buffs, debuffing enemies, etc. “Slay the Spire” was the first game of its genre to create a combat system like this and is now regarded as a founder of the “Card-Based  
Roguelike” genre. The combat system itself has many nuances and quirks. It creates many interesting decisions in gameplay and presents many interesting choices. A player can add many strong cards to their deck and hope to see them during combat, or they may choose to reduce their deck size to make seeing certain cards more consistent. It gives great replay incentive and encourages players to keep playing to see if they can create a powerful card combination or find interesting new cards.

HullBreaker combat system is also a card-based one but with a twist when it comes to how the player will build their deck. From the start of the game, the player will select a “Commander” who will start with a certain ship. These ships are made of modular parts that, when editing the ship, can be detached or re-attached. Depending on what ship parts are currently attached, cards will be added to the player's combat deck, and these are the cards with which the player will fight enemies. When the player defeats an enemy, they will be prompted to salvage a part of the enemy's ship to attach to their own, adding more cards to their deck. This will be a fresh new spin on the recently popular genre and will also present its own nuances in design and execution.

## 4.2. Feasibility

### 4.2.1. Market

RESEARCH MARKET WITH RESPECTS TO “StS” AND “RoR”

### 4.2.2. Similar Products

The roguelike game genre is very diverse and boasts a large amount of very popular games. The games which this project draws most inspiration from is Risk of Rain (Hopoo Games, 2013) and Slay The Spire (MegaCrit, 2017).

Slay The Spire is a rogue-like deck-building game developed by MegaCrit. Released in 2017, the game combines elements of the roguelike genre and traditional deck building games. The unique aspect of Slay The Spire lies in its deck-building mechanics, allowing players to strategically choose and upgrade their cards during each playthrough. With a variety of character classes, relics, and constantly evolving challenges, the game offers a highly replayable experience, challenging players to devise new strategies and adapt to the ever-shifting dynamics of the game.



Figure - Slay the Spire Gameplay

Risk of Rain, developed by Hopoo Games and released in 2013, is a multiplayer rogue-like platformer that marries intense action with relentless difficulty. Set on a mysterious alien planet, players navigate procedurally generated levels while battling hordes of hostile creatures. What sets Risk of Rain apart is its time-based difficulty progression – as time elapses, the game becomes progressively harder, demanding swift and strategic gameplay. With a diverse array of playable characters, each possessing unique abilities and skills, players must cooperate to survive the increasing challenges and uncover the secrets of the alien world. The game was also praised for its simplistic pixel art style being able to give tons of expression and personality to its characters with small sized pixelated graphics.



Figure - Risk of Rain Gameplay

Both Slay The Spire and Risk of Rain employ perma-death mechanics, adding a challenging dimension to each playthrough as players face permanent consequences for failure. Additionally, both games feature a gradual progression system where players unlock items and upgrades over time. This dual approach, combining risk with persistent rewards, fosters replayability and strategic depth. Players must adapt and refine their strategies in the face of perma-death consequences while enjoying the satisfaction of unlocking new elements, ensuring each playthrough remains engaging and distinctive. These two mechanisms are core features also present in HullBreaker.

### 4.2.3. Unique Selling Points

TO BE ADDED WHEN FURTHER FLESHED OUT

## 4.3. Technical Investigation

### 4.3.1. Target hardware for Deployment

HullBreaker will be available on PCs running windows 8 or higher:

Operating System: Windows 8+

Processor: Intel Core i5+

Memory: 2GB RAM

Graphics: Nvidia GTX 460 or better

Storage: …

### 4.3.2. Development hardware and software

HullBreaker will be developed on a PC with the following specifications:

Operating System: Windows 11

Processor: AMD Ryzen 5 5600G

Memory: 16GB RAM

Graphics: Nvidia GeForce RTX 3060

### 4.3.3. Game Engine

HullBreaker is made using the Unity game engine. Unity, developed by Unity Technologies, is a versatile game engine for creating 2D and 3D games across platforms. Known for its user-friendly interface and support for C# scripting, Unity is popular among beginner and advanced developers. Unity was chosen as the game engine for HullBreaker as it has many features and various plugins that make development more streamlined and easier, especially for a one-person development team.

### 4.3.4. Scripting Language

HullBreakers scripting language of choice is C#. C# (C sharp) is a modern, object-oriented programming language developed by Microsoft, renowned for its simplicity and efficiency in building applications across platforms. As an object-oriented language, C# promotes modular code organization, enhancing developers' ability to manage and scale projects effectively. C# was chosen as it is the language that is most understood by the developer, it is directly supported by Unity, the games engine and it is relatively easy to write and understand.

### 4.3.5. Development procedures

Research

# 5. Project Management

## 5.1. Project Methodology

The chosen methodology that HullBreaker uses for its development is Scrum.

In the realm of game development, where iterative progress and adaptability are crucial, Scrum has gained widespread adoption for its effectiveness in managing complex projects. The Scrum framework is applied to game development by organizing the process into sprints, typically lasting two to four weeks. During each sprint, the development team collaborates to deliver a potentially shippable increment of the game.

Adopting Scrum in game development offers several benefits. It allows developers to respond to changing requirements and market dynamics efficiently. The iterative nature of Scrum aligns well with the creative and iterative nature of game design, enabling developers to incorporate player feedback and make adjustments throughout the development cycle. This agile approach contributes to more predictable development timelines and higher-quality games as issues are identified and addressed early in the process.

Scrum was chosen as it allows for easier breakdowns of complex tasks into smaller manageable portions. Having smaller lists of tasks that are to be tackled at the start of each sprint makes the development of HullBreaker much easier. Here is a breakdown of the components of scrum that are and are not used in this project:

|  |  |  |
| --- | --- | --- |
| Scrum Component | Used? | Reason |
| Assigned Roles | No | This is a solo project there is no need for assigned roles |
| Daily Stand-ups | No | This is a solo project there is no need for daily stand-ups |
| Next Sprint Plannings | Yes | Each sprints work is planned at the end of the previous sprints |
| Sprint Reviews | Yes | There is a review at the end of each sprint |
| Sprints | Yes | Every two weeks |
| Product Backlog | Yes | Used to assign tasks at the beginning of each sprint |
| Sprint Backlog | Yes | Used to carry over work from previous sprints |
| Scrum Board | Yes | Trello used as a scrum board to manage sprints |

## 5.2. Detailed Schedule

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sprint # | End Date | Duration (Weeks) | Deliverables | Progress | DevLogs |
| 1 | 5/10/2023 | 2 | * GDD 4.1 * Prototype started. * GDD 3.7 * Prototype Assets | Complete | [Devlog](https://www.youtube.com/watch?v=9d9cGrX6shk&list=PLN2RTZi90m-QxoqgEXHS_zz7dc4ub7ubA&index=1) |
| 2 | 19/10/2023 | 3 | * Map Generation Prototype * Grid System Prototype * GDD Section 4 progress | Complete | [Devlog](https://www.youtube.com/watch?v=GQZyDtT5vS8&list=PLN2RTZi90m-QxoqgEXHS_zz7dc4ub7ubA&index=2) |
| 3 | 9/11/2023 | 2 | * Scriptable Objects for Cards and Ships * GDD Section 4 | Complete | [Devlog](https://www.youtube.com/watch?v=e74DjNpcYDw&list=PLN2RTZi90m-QxoqgEXHS_zz7dc4ub7ubA&index=3) |
| 4 | 23/11/2023 | 2 | * GDD Section 5 * Further Development on Scriptable Objects | Complete | [Devlog](https://www.youtube.com/watch?v=EIM7ckBKA_Q&list=PLN2RTZi90m-QxoqgEXHS_zz7dc4ub7ubA&index=4) |
| 5 | 7/12/2023 | 2 |  |  | Devlog |
| 6 |  |  |  |  | Devlog |

## 5.3. Schedule Management

Trello is used to manage a scrum board and is updated during each sprint to reflect the work being done and the work completed.

This Trello board can be via the following link here: <https://trello.com/b/ipX7ZOWC/fyp-anthony-okeeffe-hullbreaker>

## 5.4. Version Control

HullBreakers version control is managed using GitHub.

GitHub allows for easy linking with Unity projects and also has many useful features such as branching, which can be used for testing various version of the game and also functions to provide a backup for the games files in case it would become corrupted or somehow deleted.

The link to the development repo can be found here:

<https://github.com/GuyGoose/FYP_Material>

## 5.5. Risk Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Risk Likelihood | Risk Impact | How to Avoid? |
| Project scope to large | High | High | Prioritizing core mechanics and cutting back on any unnecessary features |
| Asset creation to time consuming | Medium-High | Low | Will hopefully be fully hand made but if time becomes an issue can be outsourced |
| Certain features take longer than others | Medium | Medium | Planning out the best implementation of these features so they can easily integrate with little modification down the line |
| Sickness/Absence | Low | Medium | Keeping ahead on work so I can allow some breaks if needed |
| File corruption | Low | HIGH | Frequent commits to GitHub and saving local backups at crucial points |

## 5.6. Test Plan

# 6. Project Design

## 6.1. Gameplay

### 6.1.1. Game Progression

In HullBreaker the player progresses by exploring the current system they are in, defeating the system boss to progress to the next. Each system can take anywhere from 10 to 20 minutes to locate the boss without taking exploration time into the equation. Due to the open world navigation system of the game the player could choose to fully explore a system before progressing to the next to maximise their strength, however doing this will cause the games dynamic difficulty to ramp up enemy strength to account for the players increased power. As the defeat more enemies in a system an invisible stat called “Notoriety” will increase, enemy stats such as damage, health and even number of enemies will increase proportionally with this score. Even if the player defeats a system boss with a low notoriety score it will be set to a flat amount upon entering the next system.

### 6.1.2. Mission/Challenge Structure

The mission of the characters in HullBreaker varies between them but each character’s overall goal is to defeat the 3 system bosses.

These 3 bosses are chosen at random upon beginning a run and each system has a pool of 3 bosses that can be occupying them, making for a total of 9 unique boss encounters.

### 6.1.3. Objectives

The objective of HullBreaker is to defeat all three area bosses and take revenge on the HullBreakers Inc. executives.

### 6.1.4. Play Flow

The flow of the game and how fast it is played is largely dictated by the player and there play style. The player can choose to take as little encounters as possible and speed through the games content or explore each encounter and event for a longer game play experience. The slower more methodical playstyle may result in players gaining more power then initially intended but HullBreakers dynamic difficulty should counter act this issue.

## 6.2. Mechanics

### 6.2.1. Navigation

HullBreakers navigation system is made to be open and expansive. The player will traverse the game world via a “Universe Map”, this map consists of three concentric rings containing encounter various planets or locations. These rings are our systems. These systems are named as such:

* The Inner System / “ALT NAME”
  + The Players starting point.
  + Contains the easiest encounters and events.
  + Has a central planet named “Hull-Haven”, a shop whose contents will update as the player progresses. (Each time a boss is defeated)
  + Contains the first boss.
* The Middle System / “The Planes”
  + The System accessed after defeating the first boss.
  + More difficult encounters and more diverse events.
  + More likely to contain planets with shops or ship bays.
  + Contains the second boss.
* The Outer System / “The Beyond”
  + The System accessed after defeating the second boss.
  + Contains the most difficult encounters and events can prove more hazardous.
  + Less likely to contain planets with shops or ship bays but will contain higher value upgrades and ships.
  + Contains the third and final boss.

These systems all contain several “Destination points”. These points make up the spaces which the players fleet traverse. They can be empty and simply serve as a connection for more points or they could contain an encounter, event, shop, ship bay, etc. The player navigates these points via clicking a point adjacent to the one they reside in and choosing to travel there. This form of world map and navigation takes heavy inspiration from Stellaris (Paradox, 2016), a similarly sci-fi space themed game.

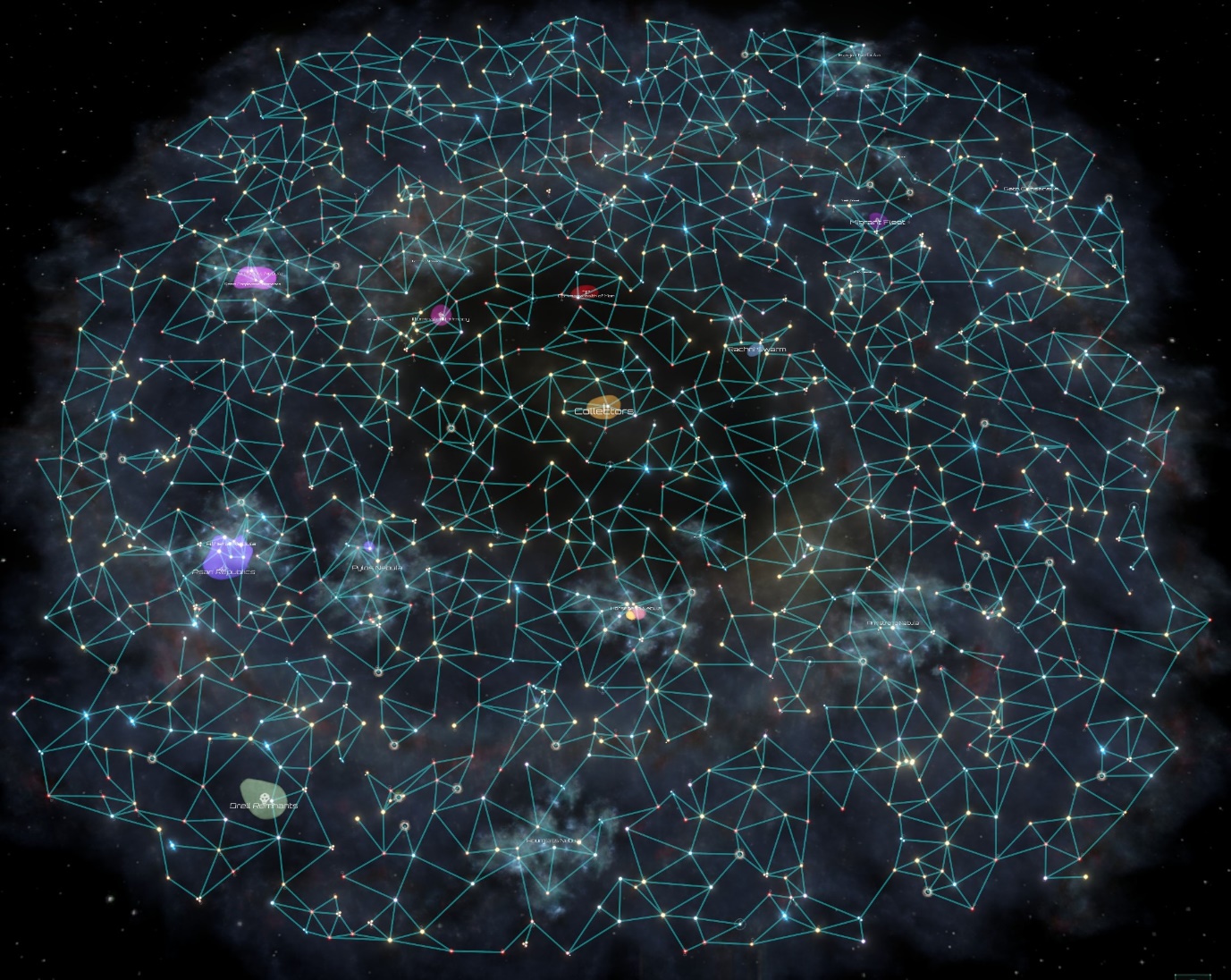


Figure - The default map type in stellaris

### 6.2.2. Encounters and Events

When traversing the system map the player will find various encounter and event areas.

Encounters are when the player locates another fleet of ships on a destination point. If an adjacent point contains a hostile encounter, it will be displayed with a RED exclamation mark. If the encounter can be avoided via alternative means (e.g. Paying off the enemies, dialogue options, etc.) Then it will be displayed with a YELLOW exclamation mark. If the encounter point is friendly, it will be displayed with a GREEN exclamation mark. What determines if these encounter points are hostile or not is the players relations with various factions in the game.

Events can occur somewhat rarely on destination points. These are indicated on adjacent destination points with a question mark. Events vary wildly can contain any combination of free upgrades, an encounter, dialogue that can yield upgrades or encounters or both, etc. Examples of some events include:

* Asteroids – The player encounters an asteroid field:
  + Avoid? – Skip event, nothing gained.
  + Explore? –
    - 50% chance: Take X amount of damage.
    - 30% chance: Find an upgrade.
    - 15% chance: Nothing happens.
    - 5% chance: Unique encounter. (Elite Enemy)
* Enforcer Checkpoint – The player encounters an enforcer checkpoint.
  + If positive with enforcer faction – Free passage.
  + If neutral with enforcer faction – Pay a fee.
    - Cannot pay fee? – Encounter.
  + If negative with enforcer faction – Encounter.
* And many more…

### 6.2.3. Factions

In the world which HullBreaker takes place there are many factions of people and creatures. Some destination points are populated with people form these factions and how they interact with you will be dependant on your relations with their faction. If a point is occupied by a faction an icon showing their insignia will be visible on the map. Each starting character has their own amount of reputation with these factions, but the player can also gain and lose favour with any of them depending on there actions within a run.

The factions present in the game are as follows:

* Enforcers – The

## 6.3. Screen Flow

## 6.4. Game Options

## 6.5. Replaying and Saving

## 6.6. Story and Narrative

## 6.7. Game World

## 6.8. Characters

## 6.9. Levels

## 6.10. Interface

## 6.11. Artificial Intelligence

## 6.12. Game Art

# 7. Appendices

## 7.1. Credits

## 7.2. References