



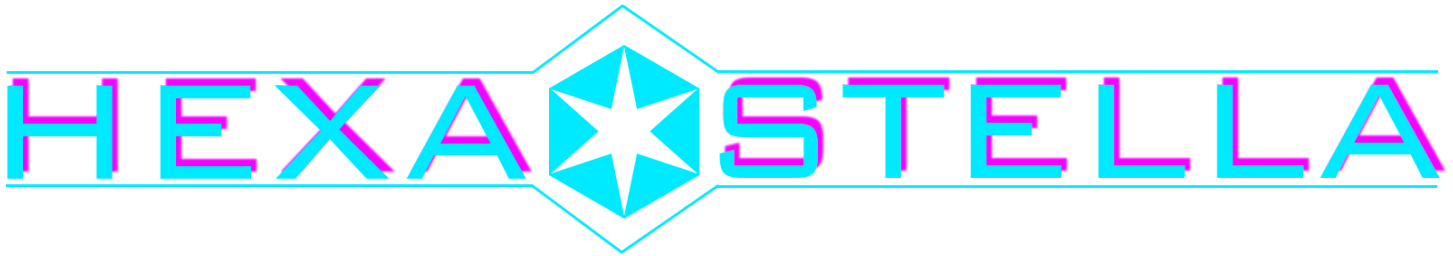
Raincafe

Hexastella Technical Design Document

Welcome to the Hexastella TDD by RainCafe

Prepared for: Vancouver Film School, Technical pre-production, Term 3 Prepared by: Hexastella PG09, GD45.

In this document you will find the details set out to develop Hexastella, the the technical requirements and details to develop Hexastella and required material and information needed.





1.TABLE OF CONTENTS

2.Startup Idea: **3**

3.Game overview **4**

4.Requirements and mockups **5**

5.Technical implementation **6**

6.Technical Diagrams **7**

7.Milestone planning **8**

8. Naming and conventions **9**

2. STARTUP IDEA

When we set out to create a game we wanted to create a game that was not only fun to play but was also engaging and carried out our core mission as a team which is to create a beautiful game that is easy and for anyone to pick up a controller and start playing Hexastella. Our game focus is meant to create a rich and engaging experience that when one plays our game will understand the core aspects and be able to take away a great experience. Our team has a high focus on developing a game in which we will love as well and something that speaks from our hearts and a game that we will be proud to present on industry night. This document is meant to highlight the technical aspect of developing our game but also a document that should be referred back to when needed.

Hexastella is a 3d, third person, single player hack and slash game, which focuses on an elaborate boss fight. In a quest to defeat an unknown powerful alien, a humanoid robot enters an alien spaceship and fight the monster. The game features a quick pace fight between a player and a powerful AI. The game has a glow in the dark neon art style, and space-punk vibe.



3. GAME OVERVIEW

Hexastella is a 3d, third person, single player hack and slash game, which focuses on an elaborate boss fight. In a quest to defeat an unknown powerful alien, a humanoid robot enters an alien spaceship and fight the monster. The game features a quick pace fight between a player and a powerful AI. The game has a glow in the dark neon art style, and space-punk vibe.

Core mechanics:

The player:

Conventional movements such as running around and jump

Dodge roll to right and left

Flash forward that inflicts damage

Range laser attack

Hexagon Interaction

The Alien Monster:

Movement

Midrange mass projectile

Long range electroshock

Roll around

Wave area attack Hexagon Interaction

3. GAME OVERVIEW CONTINUED

Unique Feature

Hexagon interaction:

The floor of the spaceship is tiled with eight pairs of hexagon; each pair has a unique mechanics that are usable by the player and the AI alike. To create a broad range of dynamics and increase the playability, the player can interact with the environment by activating the hexagons, and should be mindful to protect themselves from the AI, as it can use the environment as well.

4. REQUIREMENTS & MOCKUPS

Requirements

Core game requirements for the player:


- The ability to jump
- The ability to walk and navigate around the platform
- The ability to perform a dodge roll
- The ability to perform a dash attack
- The ability to perform a shield operation from the boss
- The ability to shoot a laser at the boss
- The ability to interact with hexagons in the game environment

Core game requirements for the Boss:

- Core AI features that will allow the boss to attack the player and learn over time
- The ability to interact with hexagons in the game environment
- The ability to perform a roll attack that will attack the player at certain times.
- The ability to perform a jump wave attack in which when the boss jumps a wave will be created in a specific radius around the boss and if the player is in that radius they will die.
- The ability to perform a projectile attack in which the boss will shoot projectiles at the
- The ability to spawn in places around the map
- The ability to perform a tentacle attack.
- The ability to walk and navigate around the game scene
- The ability to gain new abilities when the boss steps on certain platforms.

Audio requirements:

- Attack sound
- Hit sound
- Swipe/slash attack sound
- Projectile shoot sound
- Jump sound
- Laser sound
- Platform sound
- Background space sound FX
- Metal sound when the boss or player hits walls or objects

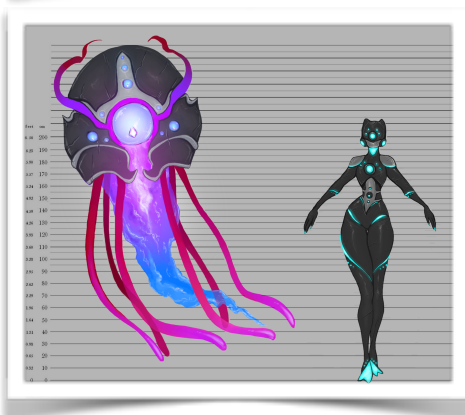
- 
- Death sound when the player dies
 - Death sound when the boss dies
 - Clash sound if the player clashes with the boss or vice versa
 - Sound for in game buttons
 - Main soundtrack for the main menu
 - Block sound
 - Wave attack sound FX
 - Hexagon activation sound
 - Robot voice sounds

Art Requirements:

- Art for the main player
- Art for the boss
- Background art
- Ground tiles art
- Animation for the main character
- Animation for the boss
- Particles FX for the boss
- Particles FX for the main character
- Skill bar (Cool down bar)
- health bar for boss
- health bar for player
- Buttons Main menu
- Buttons for option menu
- Pause screen design and buttons
- Dedicated background/UI for the main screen
- Animations for the main screen
- Texture for main character
- Texture for boss
- Texture for environment
- Texture for background
- Texture for objects in game
- Hit and take damage UI
- Cool down timer
- Successful block HUD
- Ability graphics

Mockups

Everything that will not be included in the app



5. TECHNICAL IMPLEMENTATION

Hardware requirements:

- Microsoft Windows Computer capable of running professional software for design and development
- Apple Macintosh Computer capable of running professional software for design and development
- Xbox controllers
- Keyboards
- Mice and or trackpads
- Software modelling tablets
- Monitors capable of displaying at 1080P HD Graphics

Software requirements:

- Adobe Photoshop
- Adobe Illustrator
- Maya
- Unity
- Sketch
- Github (Git)
- Github desktop
- Perforce
- P4V for perforce
- Slack
- Google Drive
- WhatsApp (for messaging)
- Atom
- MonoDevelop
- Trello (task managing software)
- Wwise
- Calendar

Naming conventions:

In order to keep things organized and fluent throughout the development of Hexastella and have defined naming conventions for folder structures, unity organization structure and variable names as well as comment moderation, control and structure.

Folder Structure:

For folder structure within the development lifecycle if Hexastella we be using the naming conventions of files as defined as: **Hexastella_NameOfFile_RainCafe** which will be contained in folder name such as: **FolderName_Hexastella**. By following these simple yet efficient naming conventions for folder structure we will be able to easily find any file fast and efficiently.

Unity Organization:

Unity can get messy if not organized every time assets are added to the project. In Unity all assets, scripts and files should be placed in the corresponding folders which the file fits into. The folders in which we will have in unity are: **Assets, Temporary Assets, Art, Scripts, Prefabs, Scenes, Sprites, Images, Menu Design**. Note more folders may be added as the development lifecycle continues when a folder is deemed necessary.

Variable Names:


Naming variables in code is important in order to create a efficient code environment. Each variable name should be named making sure that the second word in the variable if there is one is capitalized and if there is only one word in the name to make it capital like this: **public float sinkSpeed** or **public float Speed**.

Special Names:

Since there is multiple spelling for specific words that are different in the American and British English system, we want to make sure we stick to specific names that can not be mixed up and cause frustration in the development of our game. **Color** should always be spelled like this: **Color** and not to be spelled like this: **Colour**. **Favorite** should always be spelled like this: **Favorite** and not like this: **Favourite**. As such all code should be written in American English and not Canadian or British English as this is the industry standard and conflicts can arise if written in Canadian or British English.

Comment Moderation, Control and Structure:

Comments make it easy for anyone on the team to understand what code does what. That is why it is important to use comments frequently and wise. Comments should be added at least every five (5) lines in order to make the code maintainable and reusable. If possible be sure to comment the code every 3 lines unless the code clearly explains what action it performing. Comments should be added above each line and not to be written underneath a function as this could get confusing. Following code structure **spaces** should be used instead of tabs when adding space in code. This is because all spaces are the same no matter what computer or



operating system you are using. Where as tabs can leave different spaces and indentation on different computers and operating systems which could cause code layout structure problems when viewing the file on a Mac or a PC.



6. TECHNICAL DIAGRAMS



7. MILESTONE PLANNING