Brad Furrness in Fur Squadron

Game Design Document

Daniel Joyce - Games Programming Module

Index

Index

One-Sheet Design Summary

Title: Brad Furrness In Fur Squadron
Genre: Side-Scrolling Shoot em' Up
Target Audience: Men and Women, 12-25
Platforms: Windows PC
Big Idea: Space shoot em' up with multiple modes including two-player cooperative mode.

Unique Selling Points:

- Multiple Game modes such as Story Mode, Arcade Mode and Survival Mode.
- Two Primary weapons along with two special weapons to decimate the Tethidarus.
- A multitude of intelligent enemies along with epic end-stage bosses.
- Team up with Furrlicity in Two-Player Local Co-operative.
- Collect accolades during gameplay to unlock new ship designs

Play Mechanic: The main mechanics of the ship consists of 2 main weapons which you can alternate between. One is more single-target focused, which deals more damage in a thin area, whereas the other is based on using the flexibility and area of effect damage to attack ships which would be harder to destroy with the other weapon.

You can pick up orbs which increase the damage and the effect of the weapons. These orbs resonant with the colour of the weapons, similar to Raiden in which if you pick the orb which resonates the colour of your currently selected weapon, it will increase the power of that weapon. Although, there's a flip-side: If you pick up an orb that does not resonate the same colour as your currently equipped weapon, the weapon will reset it's power back to it's original power.If the player dies, the player will lose all power-upgrades that the weapons of the ship obtained which makes dying highly inconvenient and increases the challenge for the player.
There is also a special weapon which allows the player to damage every enemy on-screen as well as destroy any enemy bullet on screen also. This allows breathing room for the player and is a common mechanic in shoot em up games, to allow the player to have counter-play in very stressful situations.

Game Summary:
Brad Furrness, A hot headed new recruit of the Furmanity Space Defence Force ends up going through a wormhole, which teleports him to an unknown part of the galaxy.There, he encounters a new threat to Furmanity; The Tethidarus, a squid-like race that only knows war. It is up to Brad Furrness to stop this threat from destroying his home planet Fuerth.

Similar Competitive Products: Sine Mora, R-Type, Touhou and Geometry Wars series

Game Design

Summary

Brad Furrness, A hot headed new recruit of the Furmanity Space Defence Force ends up going through a wormhole, which teleports him to an unknown part of the galaxy.There, he encounters a new threat to Furmanity; The Tethidarus, a squid-like race that only knows war. It is up to Brad Furrness to stop this threat from destroying his home planet Fuerth.

Gameplay

The game is a side-scrolling shooter which consists of a player controlling a space ship which is on rails with limited mobility (able to use both the X and Y axis, but never can traverse further than the camera's view). Using the ship's weapons you will seek to destroy any enemies that the player comes into contact with. The game also contains local co-operative which allows a second player to control another ship which has all the same characteristics of the first ship with slight visual modifications so that the players are not confused.

The Players

The player will control a ship (Brad Furrness' ship) and will have the ability to select what the ship will look like during the main menu before entering a mission/mode.

Player 2, will control a different ship which is a direct replica of Player 1's ship.
It is a different colour scheme to differentiate the players so that there are no confusions of who is controlling what ship to allow for visual carity.

They will not be able to interact with each other, for example, their projectiles will not affect each other nor will there be any collision between them, to make sure there is fluidity in their co-operative play. Their only focus is to deal with the enemies that appear on the screen

The Enemies

There are four main enemies in the game which will come into contact with the player as the game progresses the levels.

These consists of:

The Asturroid - It's an asteroid which very slowly goes across the screen. It has no danger to the player other than coming into contact with the player. This allows for a base line 'enemy' which the player can destroy to get used to the game's controls.

The Pshutr - Very basic enemy that will hover around the screen taking shots at the player, it will die in one hit from the player's weapon. It provides the first challenge for the player where the enemy fights back.

The Kamakatzi - This unit does not shoot out projectiles, but it will home in on the player's position, meaning that it's a priority target for the player to kill. This presents a new challenge for the player as it becomes the most dangerous target on the screen.

The Furbadier - This unit is similar to the Asturroid, although when destroyed, shoots out twelve projects, all in different directions.

There is one current boss in the game at the end of the first level:

MechaRhomboteuthis - This enemy is the first boss of the game which has a very large healthbar and will shoot out many projectiles at the player. Overall it has a very basic attack pattern which the player can easily predict as it is the first boss of the game, and should be somewhat predictable to allow the player to understand attack patterns for later bosses in the game.

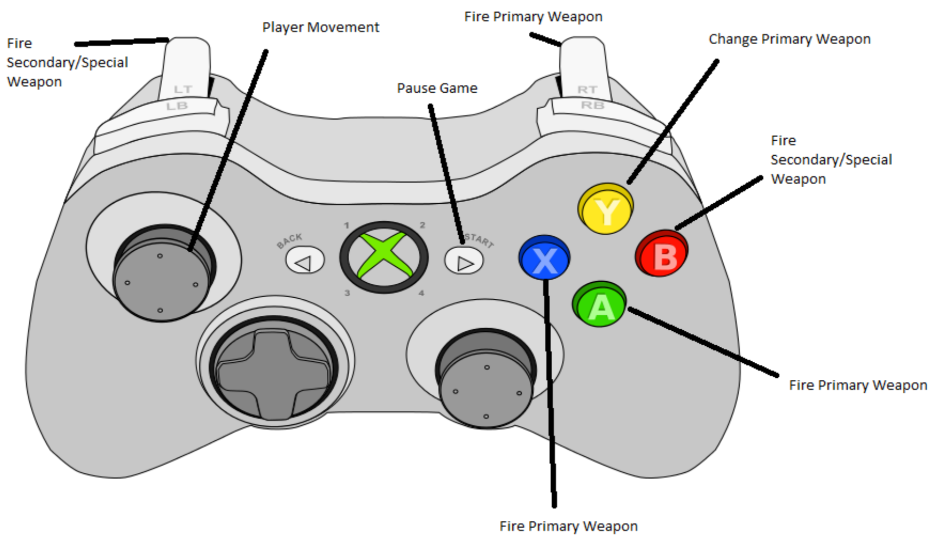
Technical

Screens

1. Title Screen
   a. Start Game (Game Mode Select)
   b. How To Play
   c. Settings
   d. Leaderboards
   e. Credits
2. Game Mode Select
   a. Story Mode
   b. Arcade Mode
   c. Mission Select
   d. Ship Select
3. Game
   a. Next Level
4. End Credits

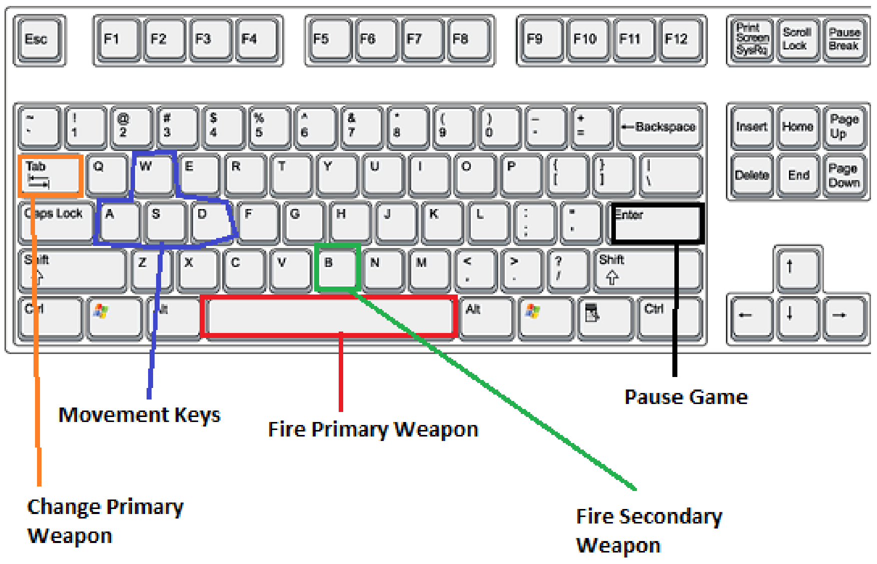
Controls

Controls will be navigational buttons for moving the ship in various directions as well as action buttons such as primary weapon fire, secondary weapon fire and the button for switching primary weapons.
Based on the Xbox 360's Gamepad, there is a standard button layout and binding, which is as follows:



All controls will be able to be custom mapped pending on the preference of the player. They can edit any of the controller binding settings in the settings menu so that they can get the optimal experience with their controller.

Using a keyboard is also another input method as it is, after all a PC game. The mapping of the buttons can be changed at any time, although here are the defaults (USA Layout):



Mechanics

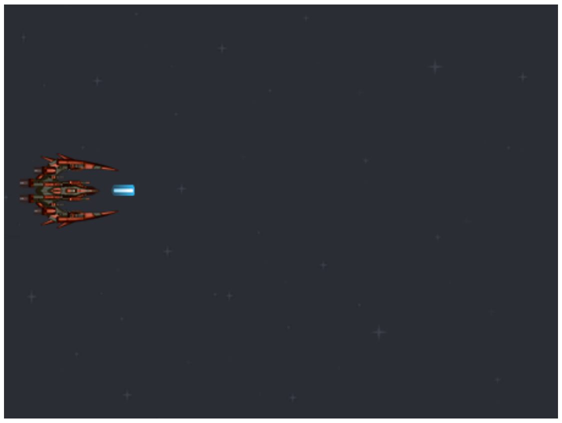
The primary mechanic is the weapons and how they are utilised. The main interest is that they are both very different in terms to what they bring to the gameplay.

There are two main weapons which have a corresponding colour: Red and Blue.
These two weapons are very different in how they behave. Red is more area of effect based, as it will cover more ground but do less damage than Blue.

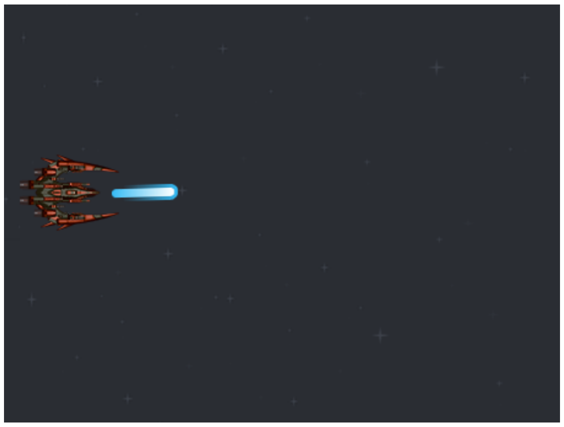
Power ups will help make these weapons stronger. Although the interesting mechanic is that the power-up orb will rotate colours, aka red and blue. This means that if the player is to pick up the orb when it matches with the current weapon equipped, the orb will increase the effectiveness of that weapon. The flip-side to this power-up is that if the player picks up the orb when it does not match the colour of the player's currently equipped weapon then the orb will reset the weapon's power back to it's original state. This will punish the player as it brings in a decision making factor as well as an extra layer of difficulty.

Some examples of the power-difference:

Blue power 1 (base-level):

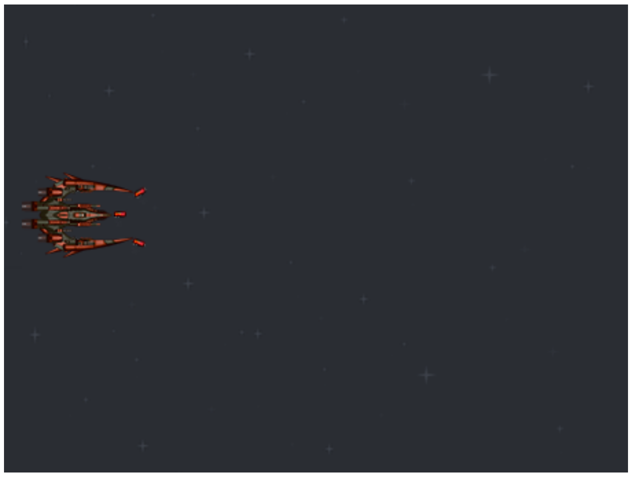


Blue power 3 (After 2 orb pickups)

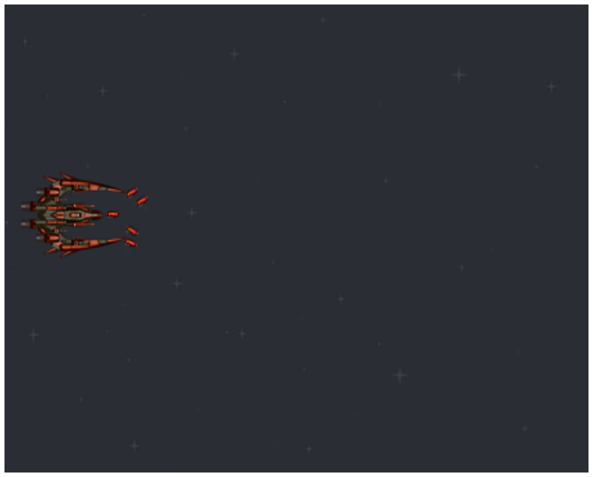


You can see from here, there's a significant increase visually. This will also be reflected in the damage aspect.

Red Power 1 (Base-level):



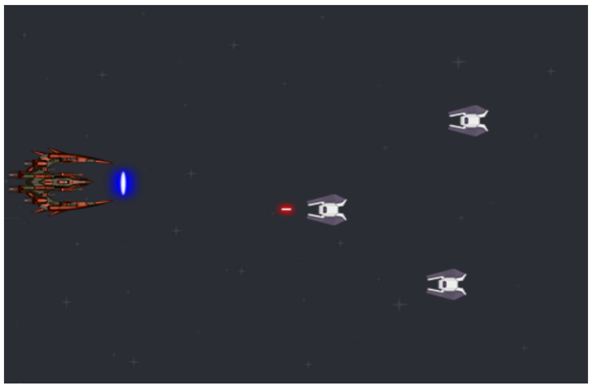
Red Power 3 (Base-level)



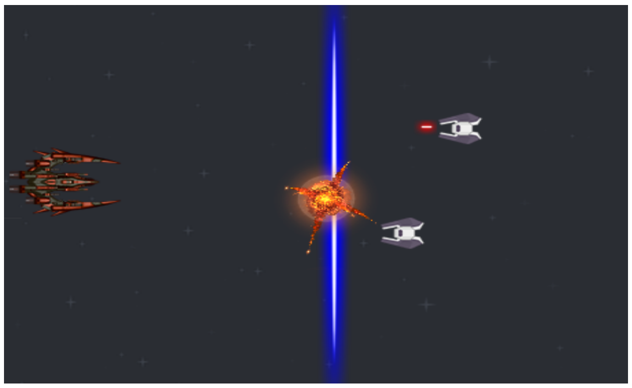
As you can see, this weapon has more of a spread, the more power-ups the player picks up. This allows for more damage spread out over a wider area allowing the player to apply damage from different positions on the screen.

The player will need to learn how to effectively use each weapon in correct situations where it may be more beneficial to use one weapon over the other. Another mechanic which is commonplace in many shooters, is the “bomb” mechanic, or in this game, more appropriately, the line laser.

The line laser is the player’s special weapon which the player can only hold two charges of, which allows the player to damage every enemy on-screen as well as destroy any enemy projectile which could possibly harm the player. This is used in the most stressful situations where the player could be under a lot of danger from the enemies on the screen. It allows for breathing room as it will in most cases destroy every enemy on screen unless the enemy is a boss or tougher than the standard enemies.



The Start of the laser fire



The laser destroying in a line what’s on screen

Hardware

The current platform that is targeted for release is Windows. The main reason for publishing to a PC platform is because it allows for a large demographic to be reached as well as allowing for a multitude of different input devices. The PC market has surged in the past few years as a prime platform for many game companies to release their product on due to the large success of it's many distribution platforms such as Steam.

In terms of inputs, it is best that both keyboard and Xbox 360 gamepad are chosen as the two primary forms of input. These are both very different and provide different things for the player, although it mainly comes down to preference in a game such as this. Both input methods are very popular on PC, so it only makes sense to include these two.

The exposure a game on Windows can get due to the wide variety of platforms that a developer can deploy to is very healthy for the game.
For example, Steam, Desura, GoG, Humble Bundle and also Bundle Stars are all platforms in which to distribute PC games, each one has it's own advantage.

Steam provides a safe, reliable platform which a developer can develop and publish their game on. They also have Steam Greenlight which allows developers to gain publicity while developing their game and to gain early funding when they get 'greenlighted' (Where the community votes for the product to be put on the Steam marketplace). Once it is 'greenlighted' the game can then enter 'Early Access' which allows people to buy into pre-release versions of the game to help fund development as well as try the game as it's being developed, it can be very beneficial to a developer as the customers can give feedback based on what they have experienced.

Software

The game is developed in Unity which allows for fast game development and very quick prototyping. Unity is leading 2D development in terms of game engines and is very popular, especially with the release of Unity5 which is in contention with Unreal Engine.

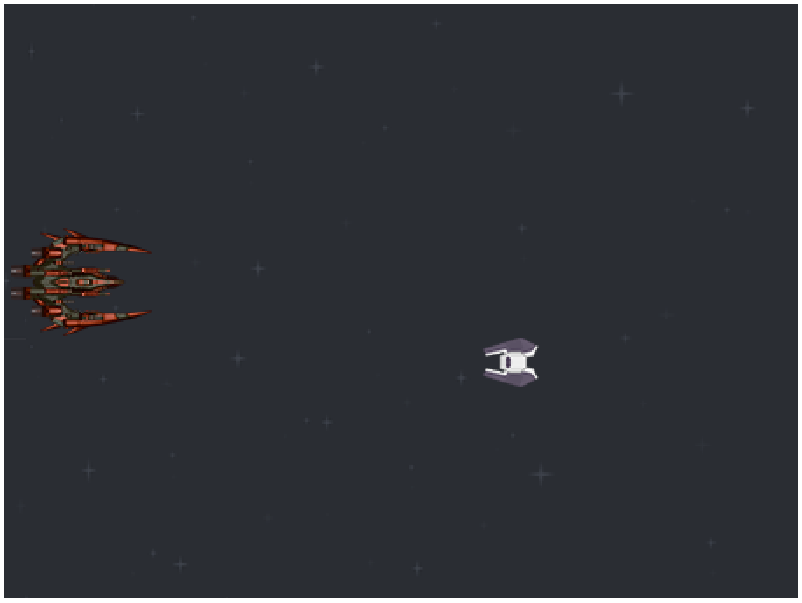
Unity5 has a vast amount of different development techniques for 2D games to be made with precision and out-of-the-box physics engine which can be applied to any object.

The focus of development is making the game, not making an engine for the game, hence developing in Unity allows for the developers to focus primarily on making the game rather than worrying about different graphic/sound/physics problems which can take a long time to set up instead of working on the game.

Level Design & GUI/HUD

The level design is very basic, as most side-scrolling shooters are. The levels will consist of a parallax scrolling background. These may have different themes to represent progression to new levels as the player will traverse to different areas along his quest to save planet Feurth.

The first level, set in space has a very open and generic look to it, as it can be seen from the example:



It's clear that the level is in space, although it's not completely pitch black because the game needs to have an array of colours which would stand out too much on pitch black. Going for a more blue tone allows for more visual clarity of other objects.

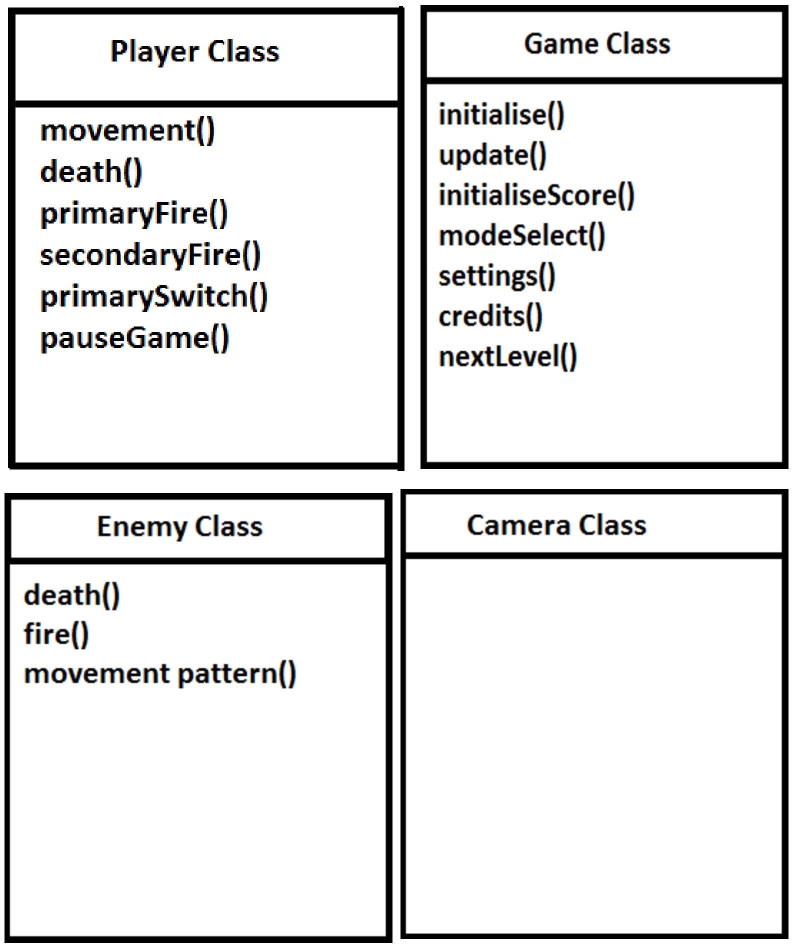
The other levels are the exact same but with a different backdrop.

GUI/HUD

The GUI/HUD is very basic. It has a score at the top left which will keep track of how many points the player scores in that level. The HUD will also keep track of lives, the currently equipped weapon and the amount of lasers the player currently has.

Development

Main Classes



Derived Classes

1. Player
    a. Player1
    b. Player2
2. Enemy
    a. Enemy
    b. EnemyAsturroid
    c. EnemyPShutr
    d. EnemyKamakatzi
    e. EnemyFurbadier
    f. EnemyBossMechRombo
3. BaseObject
    a. ObjectProjectileEnemy
    b. ObjectProjectileRed
    c. ObjectProjectileBlue
4. CloneEnemy
5. Camera

Collision Detection

There is collision detection currently present within the game which plays a huge role in the genre as well as in this game due to the nature of having to shoot the enemy ships and to avoid the enemies' projectiles.

The collision is mostly the interaction between projectiles and the ships that are on the screen. For example, if a player is hit by an enemy projectile, it will be a result of a projectile colliding with the ship.

The colliders are mostly 2D box colliders as explained in Unity tutorials as the most basic and efficient type of colliders to use as they are the most inexpensive collider to use on a processor. There are a few instances where a circle collider will be used for when the enemies launch projectiles in the shape of circles.

Enemy Artificial Intelligence

The AI in the game is very basic, each enemy has it's own designated path it follows then will make it's way off the screen to then despawn. The only exception is the boss which stays completely static and will fire off in a predictable pattern which will be on a rotation and the Kamakatzi enemy which is scripted to fly straight to the user's position via a script that updates on every frame to get the user's position to recalculate the path it will take.

Design Patterns

The Command Pattern -This allows the players to custom bind their own keys to the input, this way they can manually decide which buttons dictate which actions the player can take allowing for a wide variety of different layouts for players to use. This is done via making each input it's own subclass. The input handler will store a pointer to a command for each button which will then have it's own designation based on what the player/defaults are set to.

The Prototype Pattern - This pattern allows for spawning a large number of objects while allowing for minimal usage of the CPU and to stop memory leaks in the code. Basically, using object pooling you are able to have an efficient amount of entities spawn and despawn. An abstract class is used to clone and despawn any entity on the screen, this way it's both object orientated and it allows for optimised calls.

Object Pooling Pattern - Due to the nature of shoot em' up games. There are usually a large amount of different objects on the screen at the same time, especially projectiles launched by the player.
To reduce the load on the device that is playing the game, the game has "Object Pooling" which allows the game to run at a higher and more stable framerate to allow for smoother, more fluid gameplay without any slow-downs in any critical moments where the game may have a lot of things on the screen at one time.

Graphics & Audio

Graphics Overview

1. Players
    a. Ship for player one and two
        i. Thruster/engine animation
        ii. Blue Weapon Fire levels 1/2/3/4/5
        iii. Red Weapon Fire levels 1/2/3/4/5
        iv. Secondary Weapon Fire
        v. Death animation
2. Enemies
    a. Enemy
        i. Projectiles
        ii. Death Animation/explosion

3. Ambient
    a. Backgrounds

The visual aesthetics of the game are very basic. There are no proper animations and the textures are completely static. This gives a basic look to the game but also allows visual clarity as the game screen will not be cluttered with lots of unnecessary visuals.

The game goes for a very sci-fi looking aesthetic to go with the theme of being put into space where everything is very alien. The use of asteroids and alien ships give the player the full immersion of being in a different dimension where things are not what they expect.

There are visuals for each feedback effects to reinforce a change within the game. For example, if the player is to fire a projectile, then the projectile will emit from the player and will launch a corresponding sound. This reinforces that there's been a change to the situation allowing for the player to understand through the visual change of the game.

The biggest influences for this style of game is mostly from Ikaruga and R-Type which are primarily set in space and follow the same sort of visual design as this.

Audio Overview

1. Effects/Feedback Sounds
    a. RedWeaponFire
    b. BlueWeaponFire
    c. EnemyProjectilelaunch
    d. EnemyDeath
    e. PlayerDeath
2. Music
    a. Main Title Music
    b. Credits Music
    c. In-game Music
        i. Level 1 Music

Each effect in the game has an appropriate sound to give feedback to the player that there has been a change in situation. For example when an enemy dies, there will be a small explosion sound made to notify the player that the enemy they have shot has died. This is further backed up by a visual of the enemy being replaced by an explosion sprite.

The music featured within the game is very futuristic, mostly with electronic to further enhance the feeling of a futuristic setting in a space/planetary environment.

Using electronic music like the tracks featured in the game allow for a feeling of fast-paced action further compliments the game's action and pace.

For the audio, the main inspiration came from Tron due to it's futuristic themes as well Raiden which is somewhat between electronic and retro.

Time Management Plan

Overview

I spent approximately 4 days a week working on the game in spare time. The first 3 weeks are the past and for the next 5 weeks are for the future planning of how the game should turn out.

Week 1

Day 1 - Learned how to use the basics of Unity5 along with how to instantiate objects onto the game screen.
Day 2 - Learned input scripting for Unity using C# inside Monodevelop; Unity’s in-built script editor.
Day 3 - Learned about Object Pooling and how it relates to Shoot em’ ups
Day 4 - Used the day to read up on famous shooters such as Raiden and Ikaruga

Week 2

Day 1 - Implemented the player and the basic enemy the Pshutr
Day 2 - Implemented collision detection for player and Pshutr along with spawning the projectiles.
Day 3 - Took the day to learn about how to get the game published on the Steam platform
Day 4 - Wasted most of the day trying to implement Object Pooling and watchin Unity tutorials

Week 3

Day 1 - Implemented small scripts for the AI behavior of the enemies such as pre-determined paths.
Day 2 - spent finding appropriate royalty free sounds for the game
Day 3 - spent implementing sounds
Day 4 - spent implementing sounds

Week 4 Plans

For week 4, I plan to have fully implemented all sounds and the basic player and basic enemy AI such as the pre-determined paths it will take while on the screen as well as the intervals between launching projectiles at the player.

Week 5 & 6 Plans

I will spend week 5 implementing all other enemy types as well as creating the scenario for the boss of the first level. This will be challenging due to the nature of getting the AI to be fluid and be accurate to what I wish to happen in the game.

Week 7 Plans

Implementing the other modes such as Arcade and Co-operative play which will be challenging due to the nature of having two players rather than one and how that'll balance out.

Week 8 Plans

This is the final week which will purely be just finalising the game and polishing it to an acceptable standard for release.

References

1. Raiden : http://en.wikipedia.org/wiki/Raiden_%28video_game%29
2. R-Type: http://en.wikipedia.org/wiki/R-Type
3. Ikaruga: http://en.wikipedia.org/wiki/Ikaruga
4. Gamasutra Arcade Design: http://gamasutra.com/blogs/JoaoBrant/20150202/235543/Magenta_Arcades_Level_Design.php
5. Xbox 360: http://www.xbox.com/en-GB/xbox-360
6. Royalty Free Music: http://incompetech.com/music/royalty-free/faq.html
7. Tron The Movie: http://incompetech.com/music/royalty-free/faq.html
8. Sprite assets: http://opengameart.org/