# Technical Design Document

## Working Title: Horde

### Horde Team

#### Reece Howe

Contents

[Technical Design Document 1](#_Toc437258673)

[Two Headed Ogre 1](#_Toc437258674)

[The Bee Team 1](#_Toc437258675)

[Introduction 4](#_Toc437258676)

[Key Systems 6](#_Toc437258677)

[Primary Systems 6](#_Toc437258678)

[Secondary Systems 7](#_Toc437258679)

[Tertiary Systems 7](#_Toc437258680)

[System Interaction 8](#_Toc437258681)

[The Game Controller 8](#_Toc437258682)

[The Player Controller 8](#_Toc437258683)

[Weapon Physics (Melee) 8](#_Toc437258684)

[The Projectiles 8](#_Toc437258685)

[The Enemy 8](#_Toc437258686)

[Class Diagrams 9](#_Toc437258687)

[UML 10](#_Toc437258688)

[Enemy Behaviour 10](#_Toc437258689)

[Enemy Animator 11](#_Toc437258690)

[Left Arm Melee Animator 11](#_Toc437258691)

[Right Arm Magic Animator 12](#_Toc437258692)

[Program Flow 13](#_Toc437258693)

[Platform Considerations 14](#_Toc437258694)

[Deployment Platforms 14](#_Toc437258695)

[Imported Assets 14](#_Toc437258696)

[Required Tools – Game Dev 14](#_Toc437258697)

[Required Tool – Project Management 14](#_Toc437258698)

[Organisation and Reference 15](#_Toc437258699)

[Reference Documents: 15](#_Toc437258700)

[Filename Conventions 15](#_Toc437258701)

[Folder Structure 15](#_Toc437258702)

[Art Detail 15](#_Toc437258703)

[Environmental Art 16](#_Toc437258704)

[Animations 16](#_Toc437258705)

[Sound Effects 16](#_Toc437258706)

[Level Design 16](#_Toc437258707)

[Conclusion 17](#_Toc437258708)

## Introduction

The document outlines the technical requirements of this project, as well as the process involved for development of this project, from a technical standpoint.

The Project Pitch

Two Headed Ogre

2 player shared control of a player character.

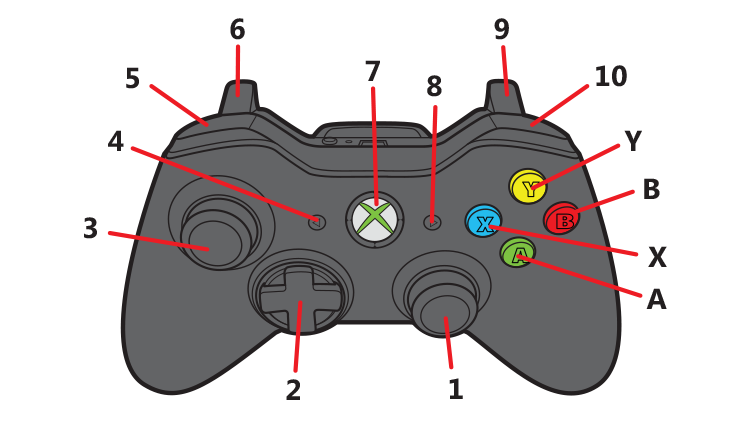
Movement: both players are able to move the character

Combat: melee and magic

Goals: defeat enemies

Story Goals: complete objectives.

### Controller Scheme



|  |  |  |  |
| --- | --- | --- | --- |
| **Control Scheme** | **Description** | **Magic** | **Melee** |
| 1 | left stick | Reticule Movement | Reticule Movement |
| 2 | item selector |  |  |
| 3 | right stick | Character Movement | Character Movement |
| 4 | Option Left | Option Left | Option Left |
| 5 | Left Bumper |  |  |
| 6 | Trigger Left |  |  |
| 7 | Home | Home | Home |
| 8 | Option Right | Option Right | Option Right |
| 9 | Trigger Right |  |  |
| 10 | Right Bumper | Attack Activation | Attack Activation |
| A | Action 1 | earth | Bash |
| B | Action 2 | fire | Swing |
| X | Action 3 | wind | Sweep |
| Y | Action 4 | water | Special |
|  |  |  |  |
|  |  |  |  |
|  |  |  | Implemented |
|  |  |  | Not Implemented |

## Key Systems

### Primary Systems

Game Controller

* Input Controller Management (Utilises InControl for Cross Platform Controller Support)
* Player Combat Selection (Melee/Magic)
* Combat Data (‘StatTree’ for Health, Attack Damage, Attack Cooldown Timers) for all combat assets
* Sound Library (searchable list of game sounds which can be called from other scripts by their identifier)

Player Controller

* Shared player movement (Tank controls; with speed and rotation limitations, rigidbody physics)
* Magic Controllers (Crosshair, Animation, projectile instantiation, cooldowns, damage)
* Melee Controllers (Weapon Physics, Animation, cooldowns, damage)
* Health
* Level Management (Restart Level\Game)
* Sound FXs

HUD

* Life
* Crosshair
* Player Movement Indicators

Projectiles

* Projectile Management (Destroy old projectiles)
* Attack Damage
* SoundFX

Enemy

* Player Detection (Sight, and Attack Range, Is Player Alive)
* Movement (Navigation, Rigidbody Physics)
* Movement Transitions (between Nav Mesh Agent and Rigidbody Physics)
* Health
* Attack
* Animation
* Behaviour Management (Idle/Pursue/Attack/Death)
* SoundFX
* Spawning

Structure

* Building Destruction

### Secondary Systems

Game Controller

* Trigger Manager (Chapter, Quest, Objective, and Checkpoint Management)
* Head Rotation (Rotate ogre heads in the HUD so players are aware of what the other player is doing with their movement)

Player Controller

* Ogre State Transitions (Dying, Death, Respawn)
* Dialogue

Enemy

* Off Mesh Links
* Death Transformations
* Enemy Management (Tidy and Limitations)

Structure

* Structure Health
* Tower Shield

Menu Systems

* Panel Changer

### Tertiary Systems

Game Controller

* Environmental Stats
* TagsScript – object identification by name or group
* Editor Mode Manager (disable debug mesh or objects)
* Dialogue Management

HUD

* HUD feedback (damage response, quest objectives, kill count)

Enemy

* Additional Behaviour Management (Guard/ Patrol / Flock / Surround Target / Alarm)

## System Interaction

These systems talk to each to make the game function:

### The Game Controller

The GameController handles Trigger Management, Level Transition, Controller Management, Sound Management, and higher level variables.

It shares information with objectives, player, enemies, enemy spawn objects, buildings, the HUD, and allows for a central location to manage in game Tags.

### The Player Controller

The Player Controller handles player related information; it gets controller information from InControl devices, and uses this to process player movement, animation, camera rotation, projectile instantiation.

It also communicates with the Game Controller for collision identification, combat statistics, and level Management.

### Weapon Physics (Melee)

The weapon physics (club physics) follows the animated move of the players’ melee arm. It communicates with the Player to player for healing effects and attack damage.

It also communicates with the enemies and structures when applying damage effects

### The Projectiles

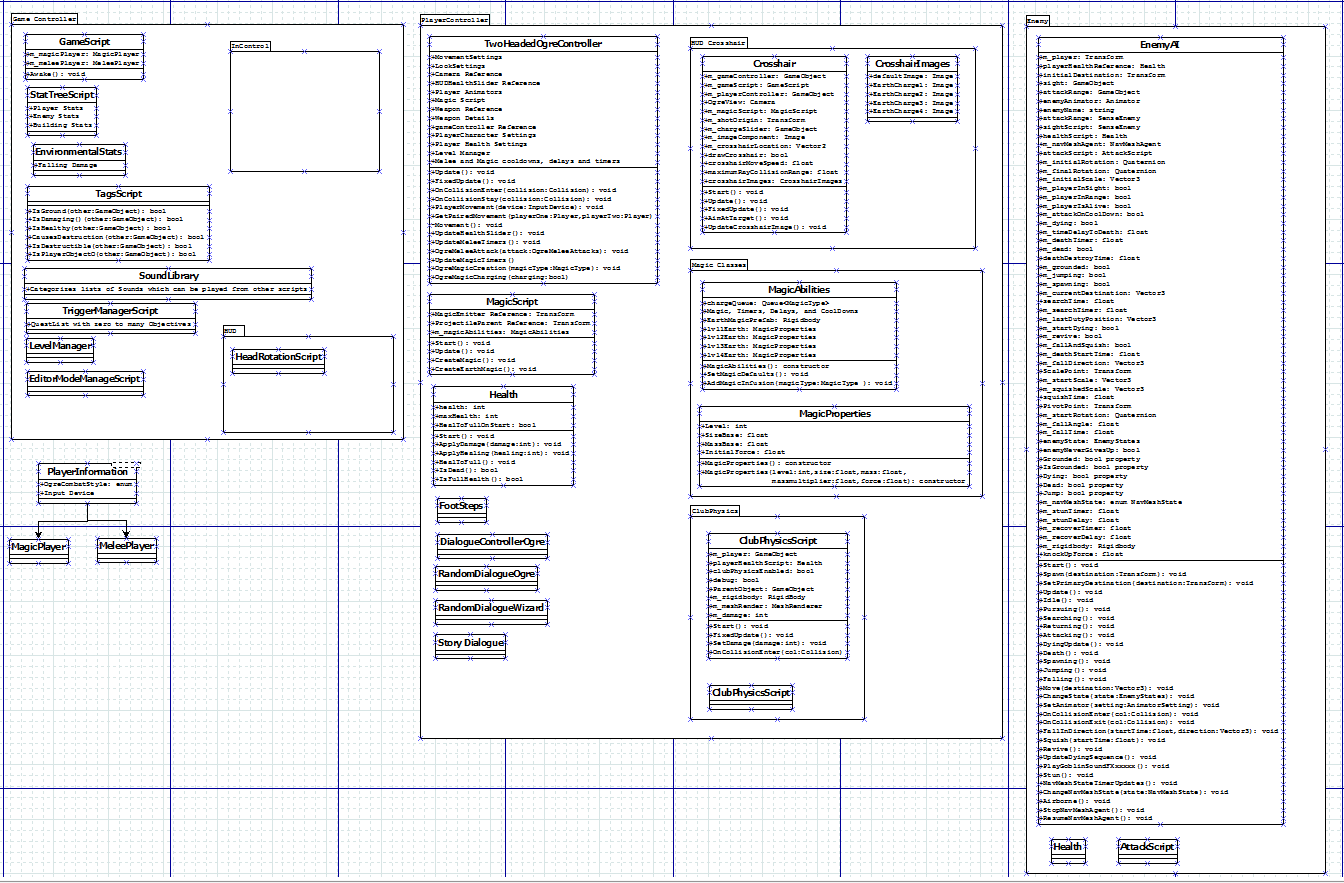
The Projectile gets its statistics from the Player Controller uses this information to deal different levels of damage to the enemy characters and structures.

### The Enemy

The Enemy gets details from the Player Controller in regards to location and living status.

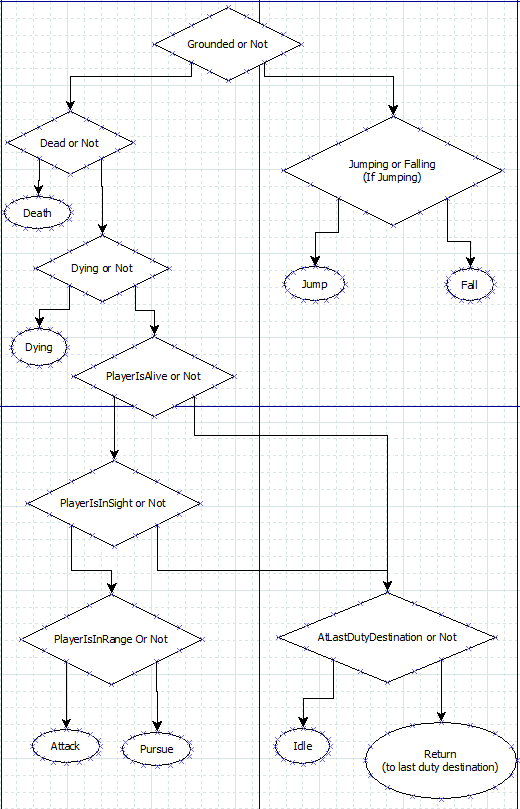
## Class Diagrams

See Attached for better detail

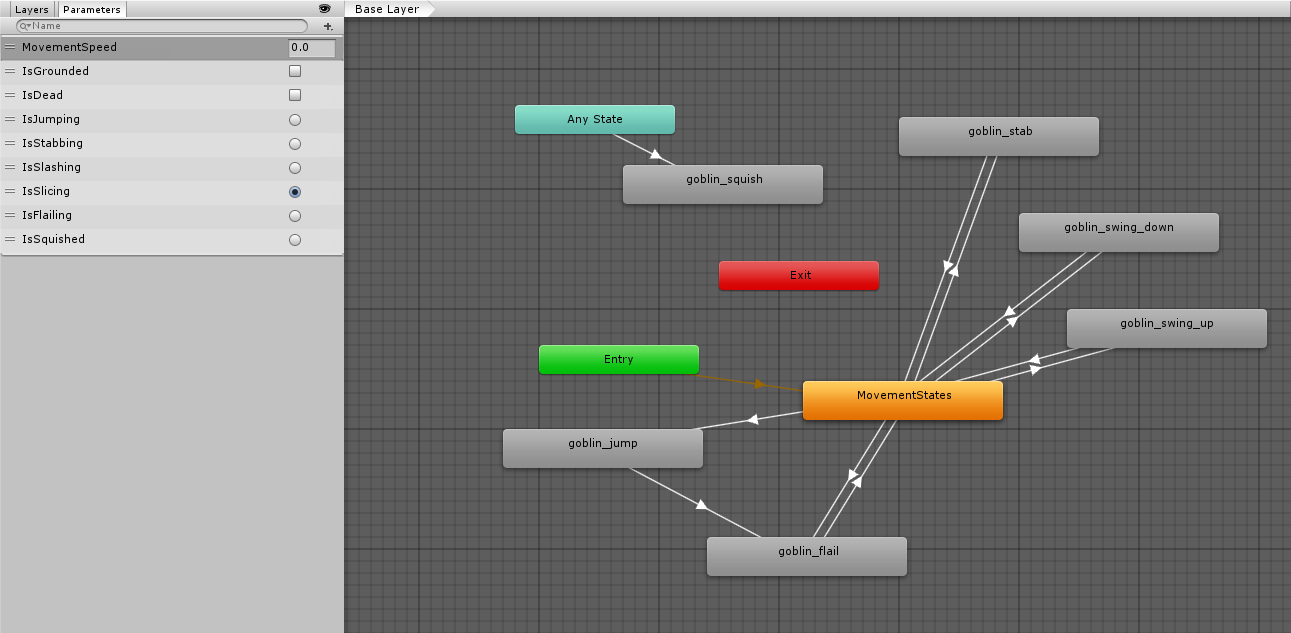


## UML

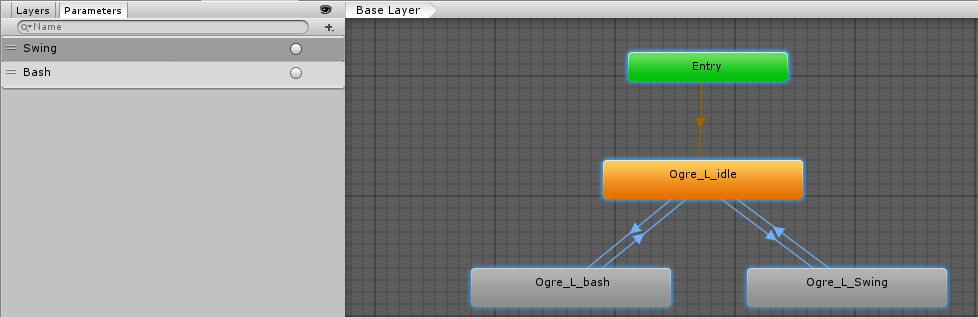
### Enemy Behaviour



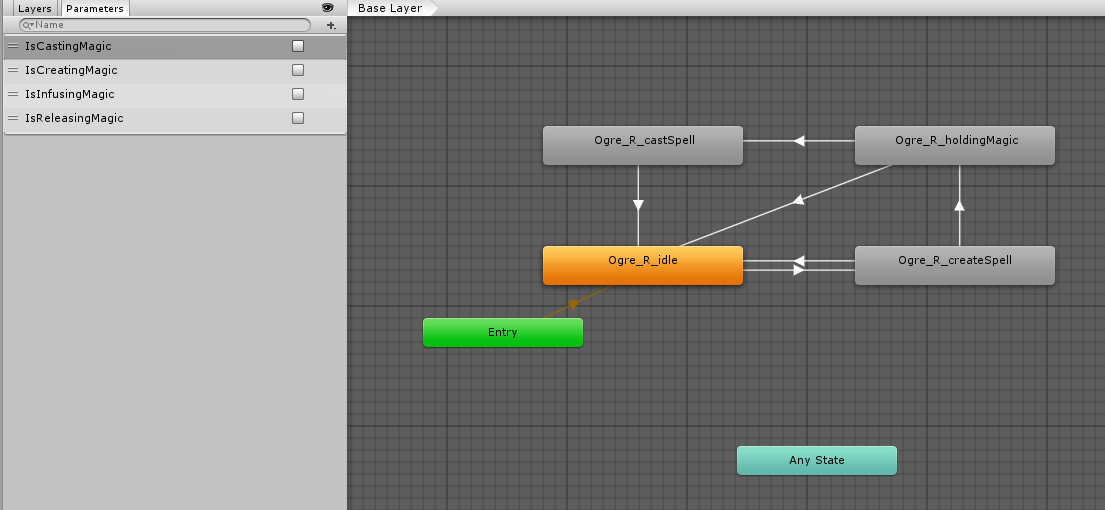
### Enemy Animator



### Left Arm Melee Animator

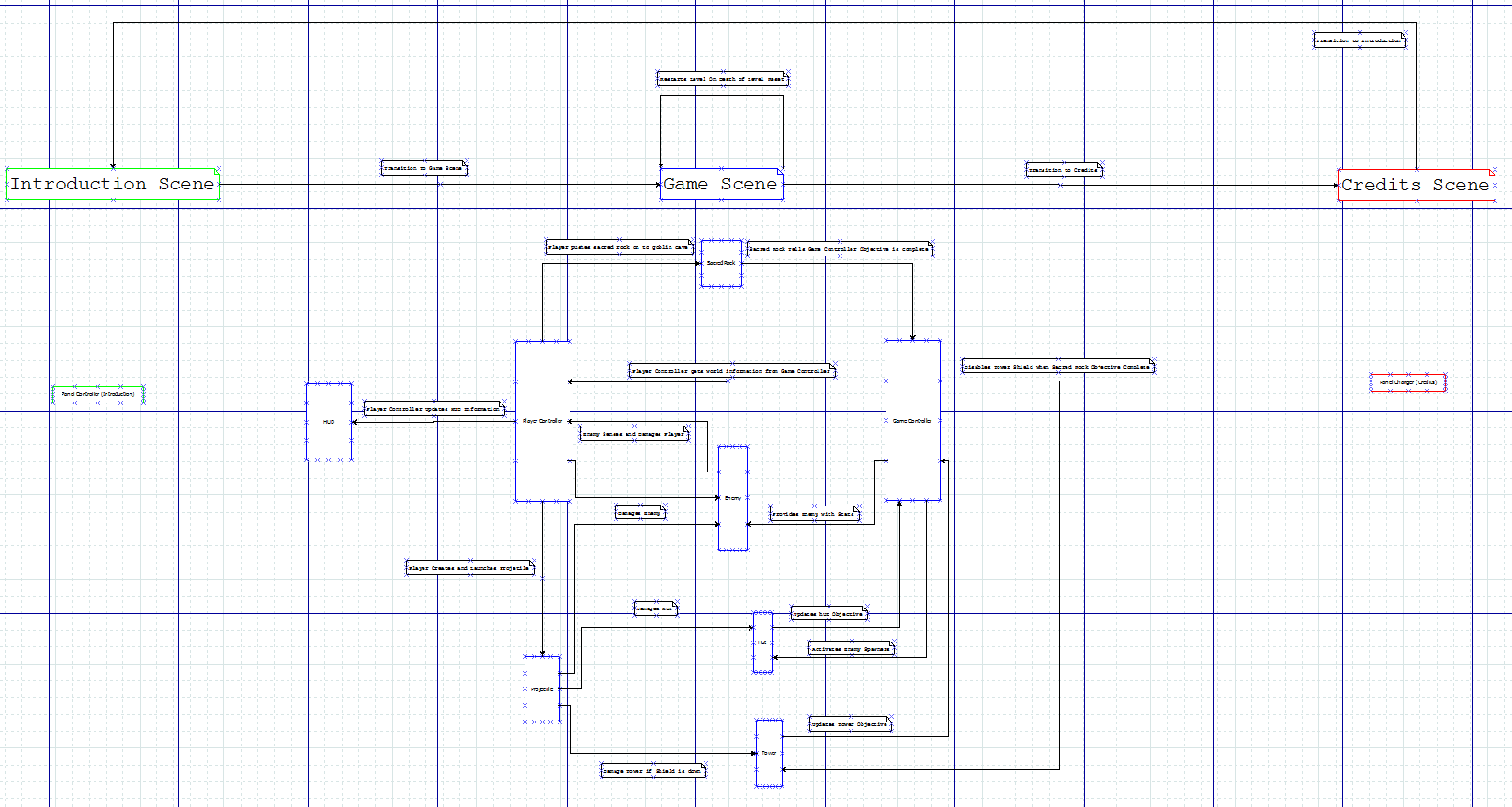


### Right Arm Magic Animator



## Program Flow

See attached for better detail (Flow/Activity Diagram)



## Platform Considerations

### Deployment Platforms

PC (Any Windows 7 machine) (with controllers)

Xbox One

Sony Playstation

16x9 screen ratio

### Imported Assets

InControl for cross-platform controller support

### Required Tools – Game Dev

Microsoft Visual Studio 2015

Unity3D

### Required Tool – Project Management

GitHub and Source Tree – version control

Trello – task management

Facebook – team communication

Google Docs – class diagrams

Dia – UML flow

## Organisation and Reference

Here is a recommended list of organizational information.

### Reference Documents:

Unity Learn:

<http://unity3d.com/learn>

Unity Best Practices:

<http://devmag.org.za/2012/07/12/50-tips-for-working-with-unity-best-practices/>

GitHub:

<https://github.com/NotProGamer/TwoHeadedOgre>

### Filename Conventions

All Source code files will be capitalized with meaningful names (eg. Health.cs)

Game Assets to be named using Title case much like source code

AIE Coding standards to be used in source code creation

### Folder Structure

All Imported Assets to be left in the root directory.

All Team created assets to be placed in subdirectory with similar structure.

Example:

*Assets folder*

* *Project*
  + *Scripts*
  + *Prefabs*
  + *Etc.*
* *Standard Assets*
  + *Etc.*

### Art Detail

Scale to be determined and adhered to in order to directly import assets in to the game.

### Environmental Art

Environmental Assets

Trees

Tower

Huts

Pillar

Sign Grass

### Animations

Animations will be required:

Left Arm

Right Arm

Enemy Goblin (At least one with options for additional if time)

### Sound Effects

Sound Effect will be required:

Projectile Collisions

Club Collisions

Footsteps

Explosions/Destruction

Enemy Attacks

Magic Casting

Dialogue

Environment

### Level Design

Assets to be designed created by our designers Rory and Brandon.

### Unity Specific

**Tags**

Player

Zombie

ZombieLure

GameController

**Tag Qualifiers**

bool IsZombie(GameObject other) // check if a game object has one of the zombie tags

**Layers**

Ground

## Conclusion

*Prototype Complete. \*Drops Mic\**

Thanks to **Academy of Interactive Entertainment** for helping our team get this prototype finished.

Special Thanks go to our lectures …. for all their help and support during the making of this prototype.

Super Special Thanks to ***Rory, Brandon, Adrianna, Aanikka, Joanna, Zac***