**Subjugation   
Technical Design Document**

**By Pádraig Crotty**

**Table of Contents**

|  |  |
| --- | --- |
| Content | Page |
| Intoduction | **3** |
| Scenes | **4** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**ROUGH INDEX**

**-----Scenes**

\_Preload

Main menu

Lobby

Selection Scene

**-----Singletons**

Managers -> Game, Inputs, Sound, Network, Settings etc

Mirror -> Maybe?? Might be broken apart and split between but a general overview should still be ok

**----Characters**

Character Composition

Weapons and Cameras

Abilities and Passives and lists of all

Minion Composition

**----Map**

Win condition – Currency - Respawns

Buildings

Map generation

Base Minions

Control Points

**-----Sound and Effects**

Sound

Effects

**-----UI and menus**

Hud + tab

Pause menu

Targetable icons and objects

**Introduction**

The purpose of this document is to be a guide to follow when creating all parts of Subjugation, it will contain information on the purpose of scripts and detail what functions and variables will do within them. Also included will be information on the designs of scenes, UI elements, Prefabs and special effects and how the players/minions will interact with them.

**Scenes**

**\_Preload**

The preload contains a persistent game object called \_app, \_app has all the managers of the game attached to it for easy access. It is created in this scene and will remain until the game is closed

**GameManager** -> The core script of the game. Is used when starting games, quitting and exiting the application, communicates with player game object to create LODS -> May need to done somewhere else

**\_SceneManager** -> A script that changes the active scene in the game used at all scene transitions.

**NeworkManager** -> Mirrors network manager, holds information on transport layer, spawn able prefabs and players.

**SoundManager** -> Holds info on all sounds used in the game and is used to play them

**SettingsManager** -> Player settings for key bindings, Mouse sensitivity (Maybe screen size (and maybe then game quality such as LOD distance, but the game shouldn’t be too hard to run anyway so this may not be necessary))

**StatusManager** -> Does status effect ticks updating all status controllers.

**PlayerManager** -> Manages player spawning, TAB pane and player construction/deconstruction

**MinionManager** -> Manages minion object pool, fills after entering game with selectable characters

**ObjectPool** -> Similarly to the MinionManager it pools objects except it pools all objects that are not players and minions or map objects, so all projectiles and effects are the main objects stored and used by this script. Pool is filled when entering game scene and can be expanded if needed, clears after game ends

**Main menu**

Contains UI for accessing settings, quitting the game and joining lobbies.

**Lobby**

This scene is used for connecting one player to another, players choose a name and join though the network to one another, maybe later adding an actual server for matchmaking might be good. But for now, localhost and Hamachi will do.

**Selection**

The selection scene is used by players to select their conquerors and minions

**Game**

**Player**

Players will initial be created in the selection scene for selecting conquerors and minions, then after transitioning to the game scene the player game object will be modified into becoming the conqueror they have chosen. After being modified into a conqueror the players will communicate with the server to start the match.

**Conquerors**

Conquerors will have several components and objects attached to them at creation which will then be broken off into separate objects. A conquerors scripts will consist of a PlayerController, Weapon (Ranged or Melee), Passive (0,1,2,3,4,5), a Health and a MinionCommander script.

Conquerors move by using a nav mesh agent that the players inputs will set the destination for

In order for different scripts with similar effects to function properly such as Weapon Ranged and Weapon Melee the PlayerController and Health will use Unity Action Events. Scripts like passives will subscribe to these events such as weaponFired or damaged.

**CameraFollower** -> Detaches from player on construction, controls a camera that smoothly follows the player and can rotate

**PlayerController** -> Uses inputs to set the destination of the nav target, also uses Inputs script for Unity Action Events such as weaponFired, Ab1, Ab2, Ab3 etc.

**MinionCommander** -> (Can possibly be merged into player controller) Uses Inputs to broadcast commands to selected follower minion types.

**Weapon (Melee/Ranged)** -> Scripts that subscribes to a PlayerController, each weapon script will inherit from a Weapon script that has values for Damaged, Attack Time – How long an attack lasts, Attack speed – Time between attacks, particle effects to play. Any character with particularly special properties can also have specially designated extra Passive scripts called XtraEffects

*Example a Player1 fires a ranged projectile at an opponent, PlayerController -> Inputs -> PlayerController Event -> Weapon Ranged Instantiate proj on server and client-> Projectile travels -> collides with target -> Player2 Health takes X damage on server-> Health on damaged -> Passive\_1 “Dome Shield” activates.*

**ConquerorHealth** -> Inheritance from Health, contains info on player health, regeneration, on damaged event, and a connection to respawn script for player death/respawning. Communicates with UI to display information

**Passive** -> Unique scripts inherited from nothing, passives will vary greatly from character to character

**Abilities** -> Unique scripts inherited from nothing, abilities vary from character to character each with their own effects on players and minions, the cooldown of an ability is managed by the StatusController which in turn is manager by the StatusManager on the server

**Team** -> Used to identify team of a player/minions, tells all other scripts that use teams for damaging and targeting what team they are on

**StatusController**-> Communicates with the StatusManager on the server, has controller over all other conqueror scripts and components. Used to enforce status effects and cooldowns from the server

**RespawnController** -> Used on player death, communicates with PlayerManager for respawning

**MousePointer** -> Turns the players weapons and characters head toward the mouse in the game world, uses clamping to limit rotations