Shurgeken

Game Design Document

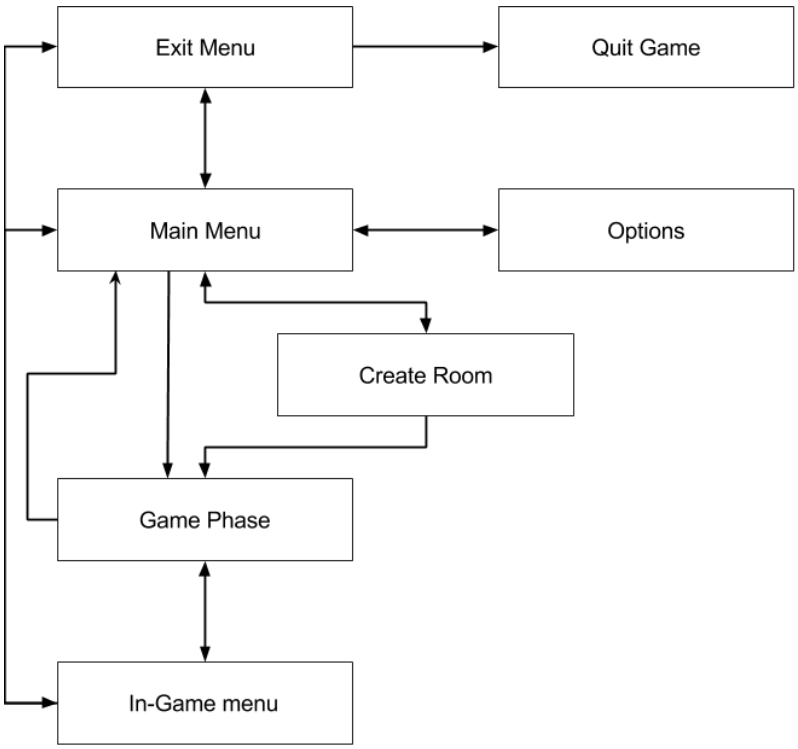
Cherry Frosting Everywhere
Adam Kimble, Chris Leight, Albert Maah, Nicolas Re, Kris Wenger

Introduction

Overview

Programmed in C# on the Unity3D Engine v5.5.0f3 Shurgeken revives the age old third-person classic, capture the flag with a new far east feel. Uses of interactive lighting elements help the player to stay concealed in the shadows to outmaneuver opponents. Or players can take a more direct approach and cross blades with them head on in melee weapon combat. Beware though, death in this game means jail time with the only method of escape being an ally coming to the rescue. Do you have what it takes to overcome your opponents and reign as the top ninja around. Or will you fail and spend all eternity in jail. Available for play in both singleplayer and online co-op multiplayer.

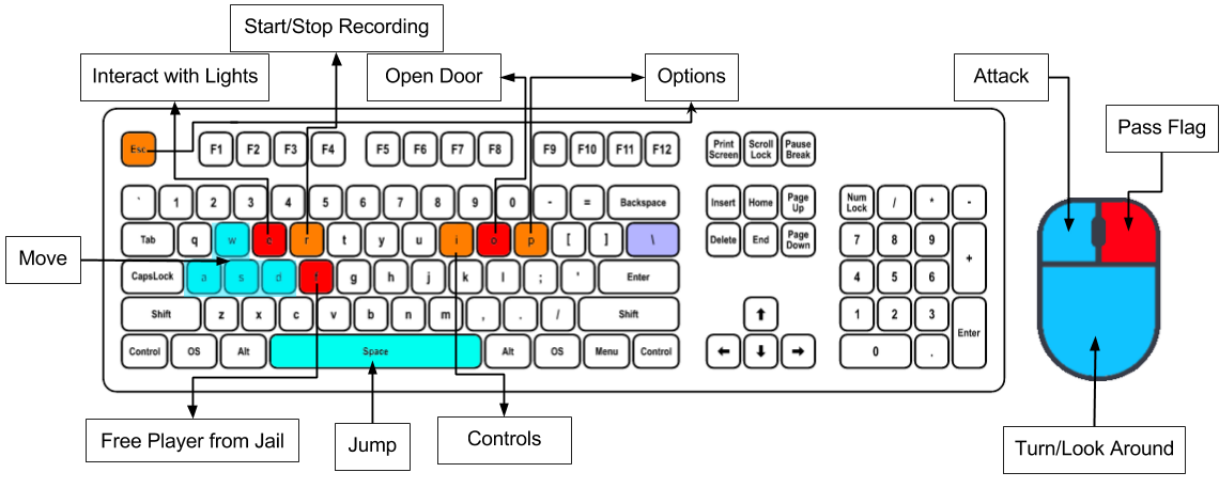
State Diagram



The above diagram shows the game's overall flow through its states. As shown players can freely move between the main menu, and the options and exit menu to configure their game or exit it. Players can then either enter solo play or create/connect to a networked room. Once in a game players then have access to their in-game menu which lets them change settings in game, exit to main menu, or exit the game entirely. Once the game is finished the players are returned to the main menu.

Game Mechanics

Control Schema



Health/Combat/Jail

Every player spawns into the map with five points of health, which do not regenerate. Players and enemies are equipped with a single melee weapon which can be used to inflict damage on their targets. Players and enemies lose one point of health every time they take damage. If a player loses all five bars of health they will be “knocked out.” They will then respawn in the neutral jail located at the center of the map. Enemies that lose all of their health will also be “knocked out”, until they respawn after a brief delay. In network play, players can be freed from the neutral jail. This is done by having an allied player go to the jail entrance and hold down the F key for a few seconds. Players freed from the neutral jail then respawn back at their base’s spawn point.

Figure 1: A player in the neutral jail trigger a lose state in solo play

Figure 2: Player freeing an ally from the neutral jail triggering the “JAILBREAK” UI

Flag Interaction

The goal of Shurgeken is to capture the enemy’s flag, located within the main building on the opposing side of the map. The flag appears as a glowing blue orb. Guards are also able to spot players carrying the flag more accurately. This is because the flag is itself a light source. Once a player picks up the flag, they lose their ability to fight. Players holding the flag can throw it o a short distance in front of them. A player regains their combat ability once they are no longer holding the flag. If a flag is thrown out of the map or a guard picks up the flag, it respawns in its base. Similarly if the flag is scored it respawns at its base.

Figure 3: Player with the enemy team’s flag

Interactive Lighting

Campfire and torches are scattered throughout the stage, which provide guards a boost to their vision. Players can extinguish and relight these lights by standing close to them and pressing E. Once a light is extinguished, nearby guards will have their field of vision cut. Some lights cannot be disabled, however, such as the light emitted by the flag and the ambient light in the spawn room.

Figure 4: Debug visualization of guard vision

The guards' vision depends on the player's proximity to active lights. The farther the player moves from light sources, the harder it is for the guard to detect the player. Because of this, it's easiest to escape the guards in a darkened area. The white bar in the debug visualization tool displays the guard's maximum vision range, while the red bar displays it's effective vision range due to lighting.

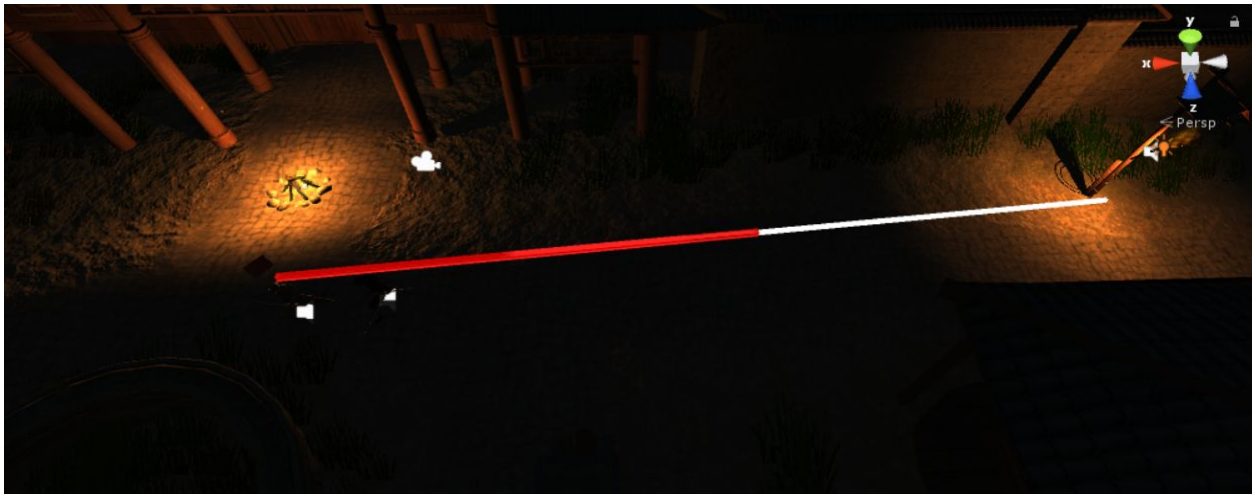


Figure 5: As the player moves away from light sources, the guard's vision range decreases

Sound Design

Sounds for the game are managed over the network to ensure that all player, enemy, and environmental sound events occur at the same time for all clients. Movement and combat noises caused by players may alert guards if the noise level is within their sphere of hearing. Players must be careful when infiltrating the enemy base to do so silently to minimize their chance of detection.

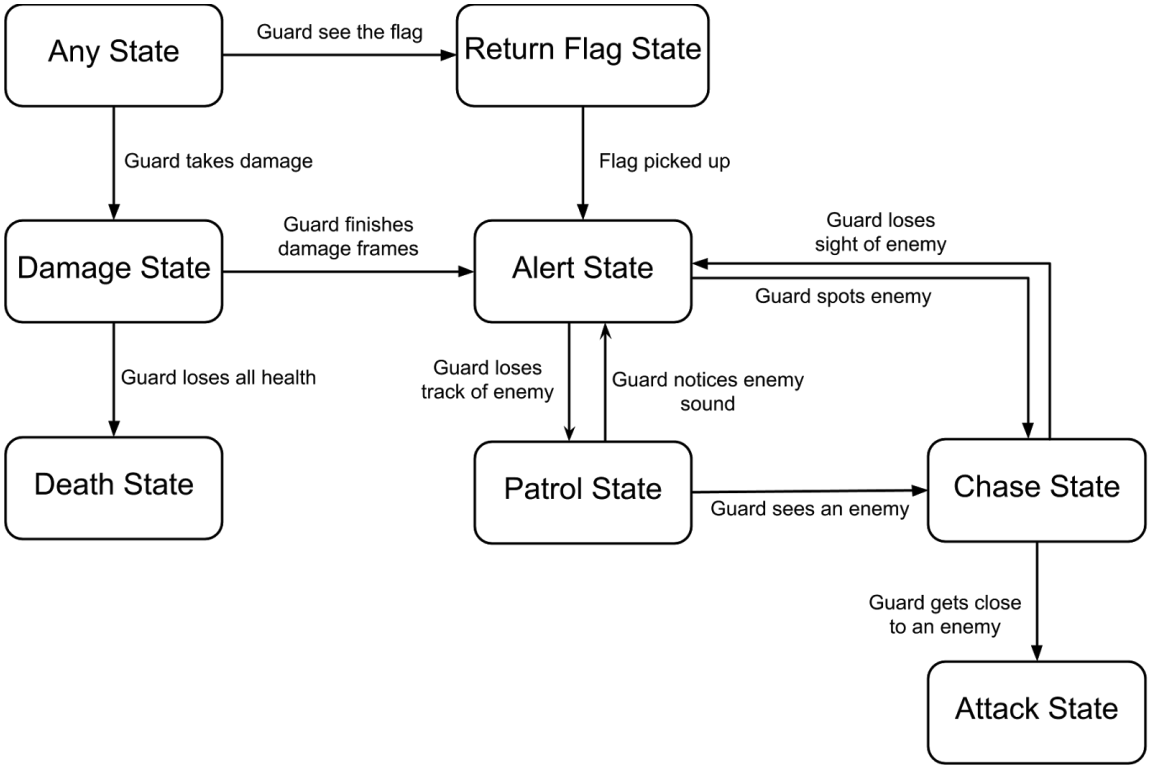
Artificial Intelligence

The game's guards use artificial intelligence to give the players a challenge. The guards' artificial intelligence is broken into three main parts. These parts are: how they are created and interact with the environment, their pathing, and their state pattern.

The first main part of a guard is how they are created. Guards are created with the game's networking in mind. They are created at load time for the level. This allows us to easily scale the number of guards which the player or players are competing against. When they are created guards' values are initialized. These values includes their attack distance, their chase time, and their alert time, to name a few. This also allows guards to be much more modular in their creation. The modular design ensure the guards are easier to pass over the network and can be scaled to meet the game's difficulty settings.

Another major initialization the guards go through when they are created is the assigning of their paths. At load time assigned a patrol path, which contains waypoints.. The path the guard is assigned is randomly decided from a pool of four different paths. Once assigned a path a guard then is assigned random waypoints from that path. The randomization of points occurs once when the path is assigned and then again every three minutes. This is done to add another layer of randomization and replayability to the game. The patrol pathing is designed using an interfaces so guards can quickly and easily switch paths they are currently patrolling as easily as they switch waypoints. The paths are controlled by a path manager who when called returns the waypoints for a certain path which the guards then use to create their random pathing.

The final main part of the guards is what gives them their intelligence, their state pattern. The state pattern was designed to give players a challenge while also making the guards as unintelligent as possible. Each guard has an enemy state pattern class which controls its states. They then have access to their patrol, alert, chase, attacking, damage, death, and flag pick up classes. Each class is implemented off the enemy state interface. Which guarantees the guard transitions from any of its states. The state pattern evolved to look something like this:



The patrol state is the guard's default state, in which the guard patrols between its randomly chosen waypoints. If the guard is alerted to a player, either through the player entering its sound range, or through environmental factors, it transitions into its alert state. In the alert state or patrolling state, if the guard sees a player they transition to their chase state. In the chase state, if the AI enters its attack range it stops and attacks the player. If the player escapes the enemy's attack range it goes to alert state or chase state depending on if the enemy can still see the player. In any state, if the AI sees the flag it prioritizes picking up the flag over any other state. The guards are able to take damage and die. If the guards die they will respawn after a few seconds, and will be assigned a new patrol path which they will then follow.

Networking

The main purpose of networking in the game is to allow players to cooperate towards achieving the goal of capturing the blue flag. The system used for this game is UDP as it does not require port forwarding, which is inhibitive to players who are not technologically inclined. There are two methods for transmitting data over the network, data streams and RPCs. There needs to be a balance of data streams and RPC's. This is because data streams are useful, but require a large amount of bandwidth. RPC's are able to update information on other clients' games, but do not continually relate the updates presented.

One of the two methods used in to network this game is data streaming. Data streaming is used to relay continually updated information, such as AI and player positioning and states. As the AI wanders the map and encounters players, and the AI's state updates. As the player moves around the map, the other player's game instances need to be updated with that player's information. Player animation codes are also transmitted over the network to allow other players to see what each other are doing. Whether they are attacking or just running around. As light synchronization is important to the game, light's status is transmitted continuously over the network to ensure that lights are in the proper state. If we were to not transmit the state over the network, a player may change a light's status, and the next player to join may see the light in the opposite state.

The other form of network transmission used in the game is remote procedure calls (RPCs). These are for single time actions which change the game state. An example of an RPC is dealing damage to another player or an AI, as this does not happen frequently. The game uses RPC's for dealing damage, light interaction, and throwing the flag, The AI used in the game is controlled by the master client, therefore when interacting with each player, the AI must use an RPC in order to deal damage or receive damage.

User Interface Design

The user interface or UI is used to provide players with the proper tools to navigate through the game and provide information about the current state of their game. A main menu screen greets the player at the start to allow them to choose between network and solo play.

Figure 6: Main Menu Screen

A settings menu allows players to change resolution, screen style, game volume, brightness and contrast. Settings are saved between game and menu and are able to be changed on the fly. They also allow the player to select the difficulty level of their games and invert their y-axis.

Figure 7: Settings Screen accessed from Main Menu

In the network play menu the player has a choice of joining an existing lobby or creating their own lobby. These lobbies will be PvE and have the difficulty that was specified in the host's settings and will allow other players to join in from the rooms menu.

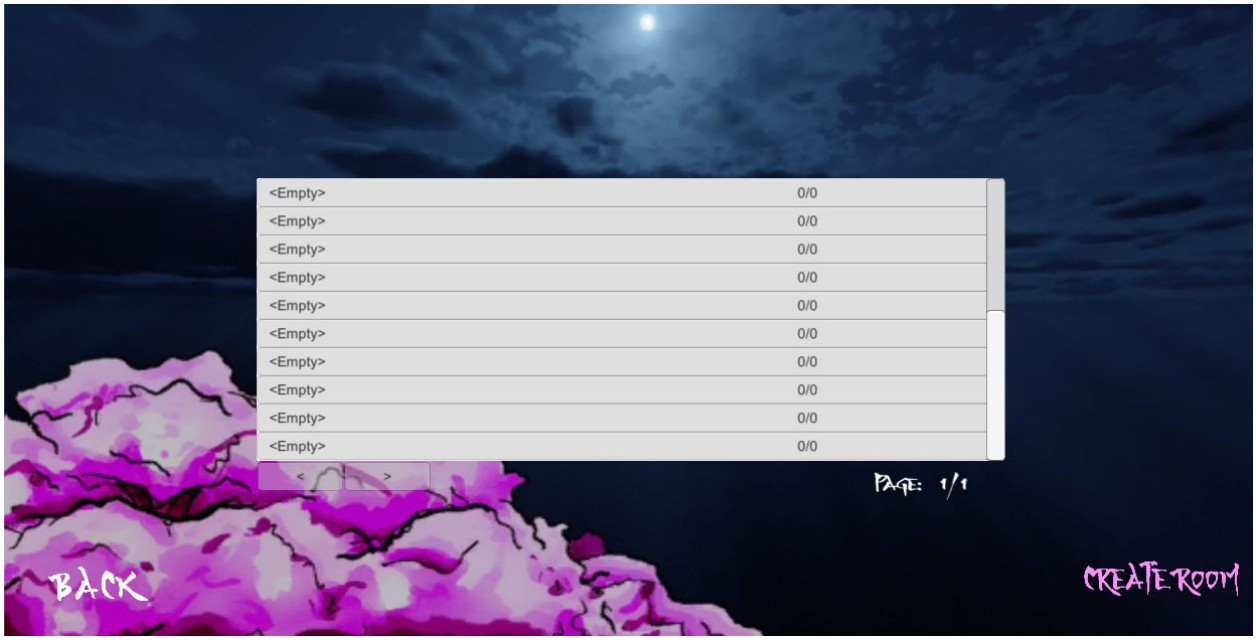


Figure 8: Lobby Manager Screen

Once in game the UI provides the player with the ability to adjust their settings like in the main menu using the options menu. Pressing the hotkey ‘P’ or ‘Escape’ will open the in game options menu.

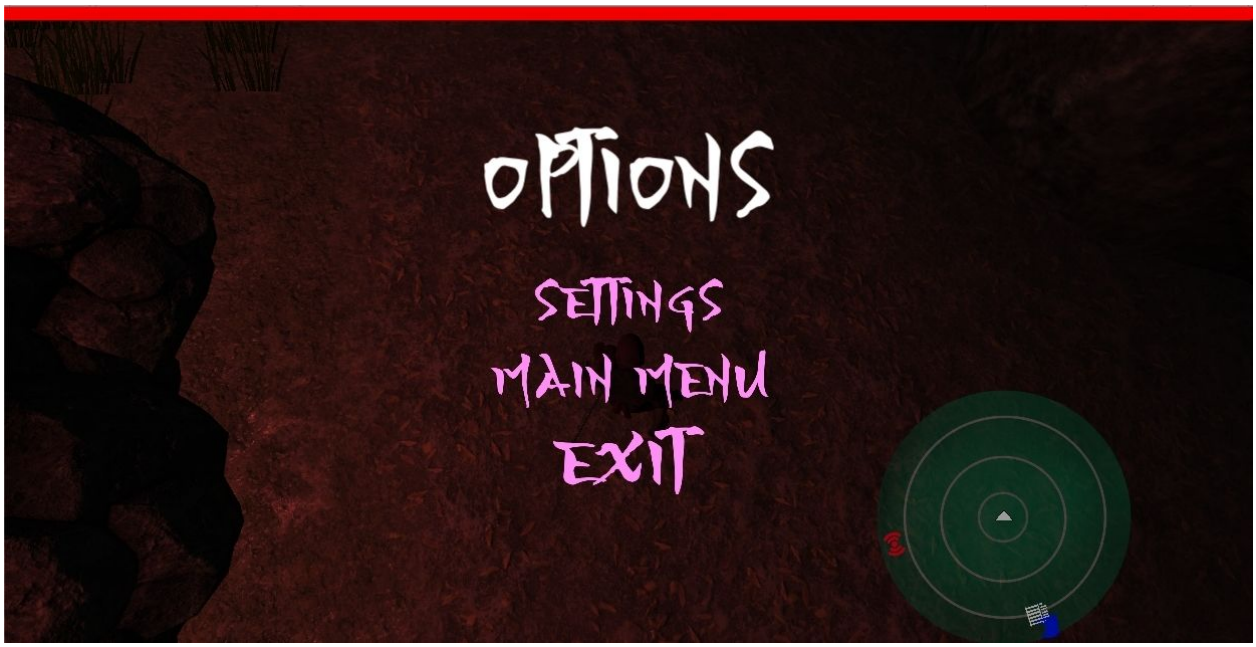


Figure 9: InGame Options Menu

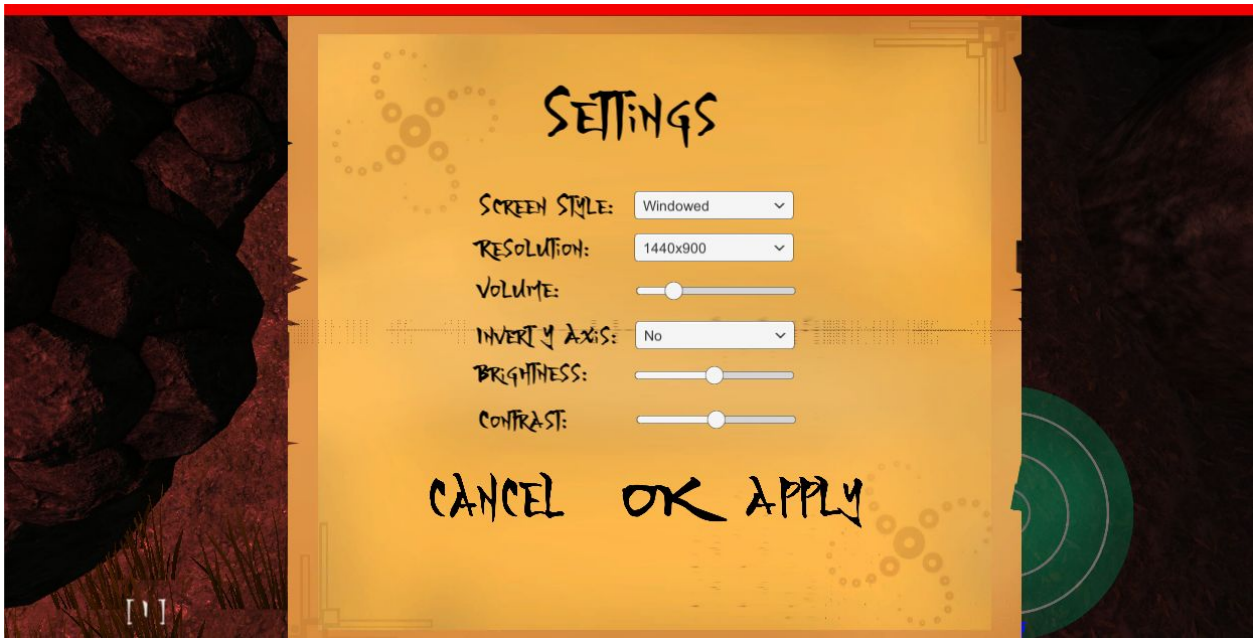


Figure 10: Settings Screen accessed from InGame Options Menu

They can also exit back to the main menu or exit the game completely. This menu does not naturally appear when the game loads as it obstructs player vision. There is also an indicator at the start for the player to press ‘I’ and bring up the controls screen.

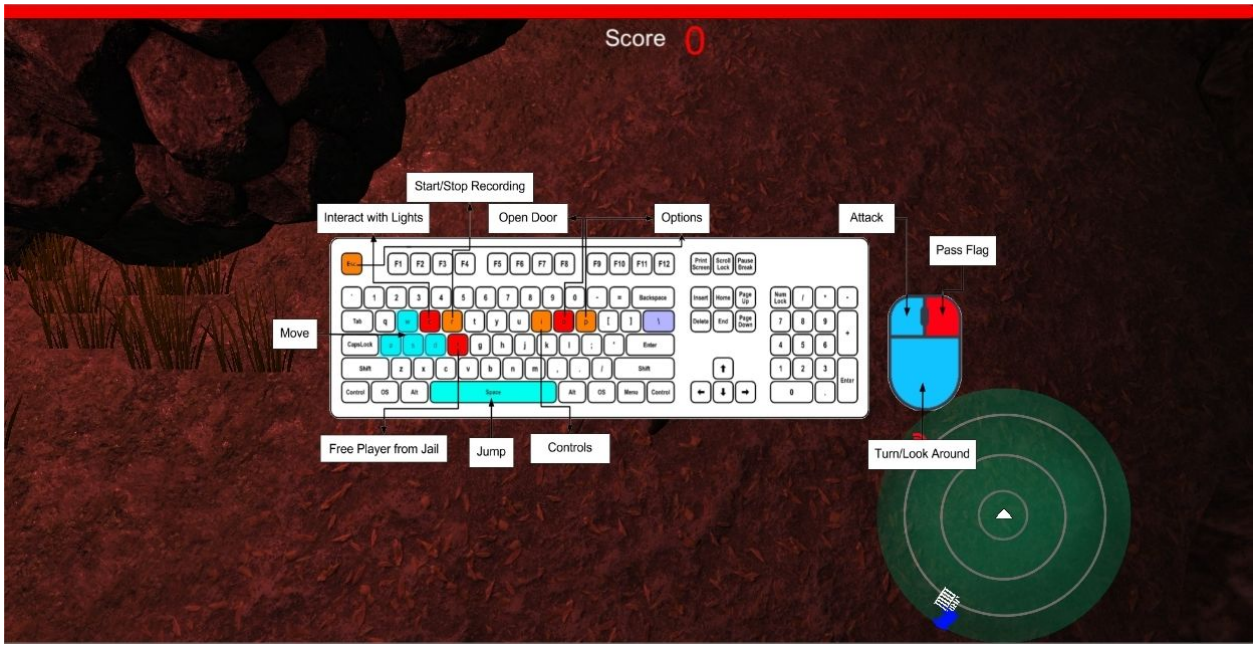


Figure 11: Controls Screen accessed by pressing ‘i’ on the keyboard

The UI also contains a health bar, minimap and score. The score indicates the total number of flags captured by the team and the health bar indicates how much health the player has. The minimap is a radar and only shows nearby enemies and allies. In addition, it provides the player with a general direction and location of their home base (in red), the enemy base (in blue), the enemy flag (also blue) and the jail (in white).

Figure 12: Closeup of the Minimap located in the bottom right of the screen

Between the start of the game and the main menu screen is a splash screen animation of our team logo. The splash screen comes directly after the unity logo and loads into the main menu screen automatically.

Figure 13: Loading Splash Screen at the start of the game

Between the main menu and the game is an additional loading screen to tell the player that the game is in the process of loading. Depending on how the player's computer handles the load of the game the Shuriken in the middle of the screen may be spinning.

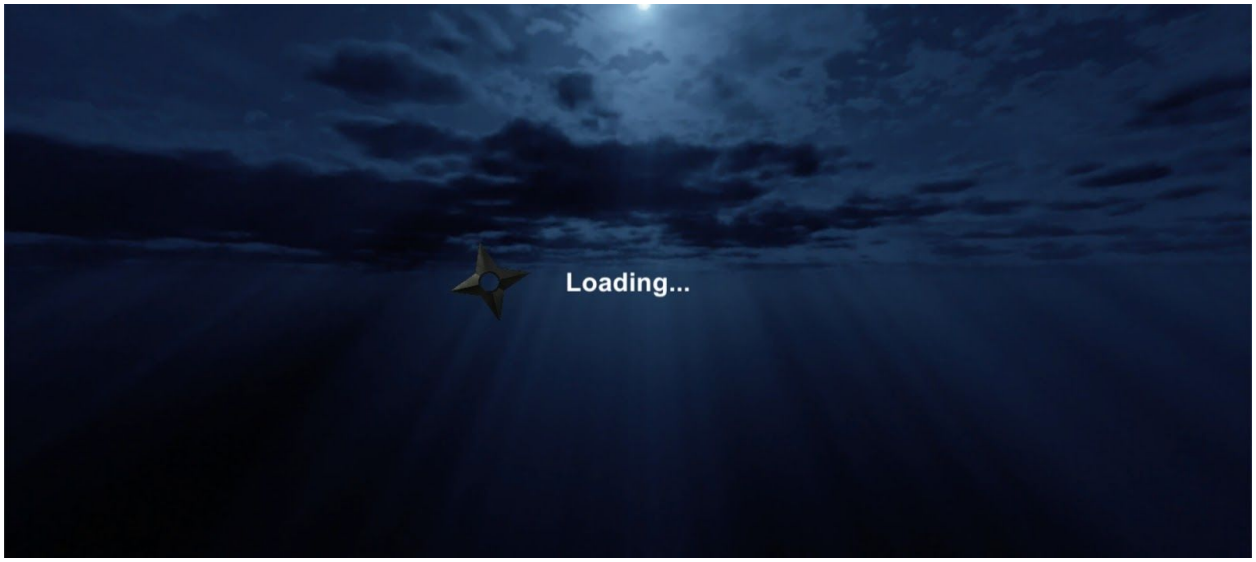


Figure 14: Load screen between main menu and game

Level Design

The level was designed with the traditional three lane system in mind to allow players a variety of paths to the objective, while still maintaining designated focus areas of combat. Clutter and interactive lighting were added to give players cover and strategic diversity in their approach to capturing the object. A neutral jail is located in the center of the map which houses players that die. Invisible triggers were placed on the outer limits of the map to prevent players and flags from leaving the intended play area. If a flag leaves the play area it will respawn at its appropriate base. If a player leaves the play area they will respawn in the neutral jail.

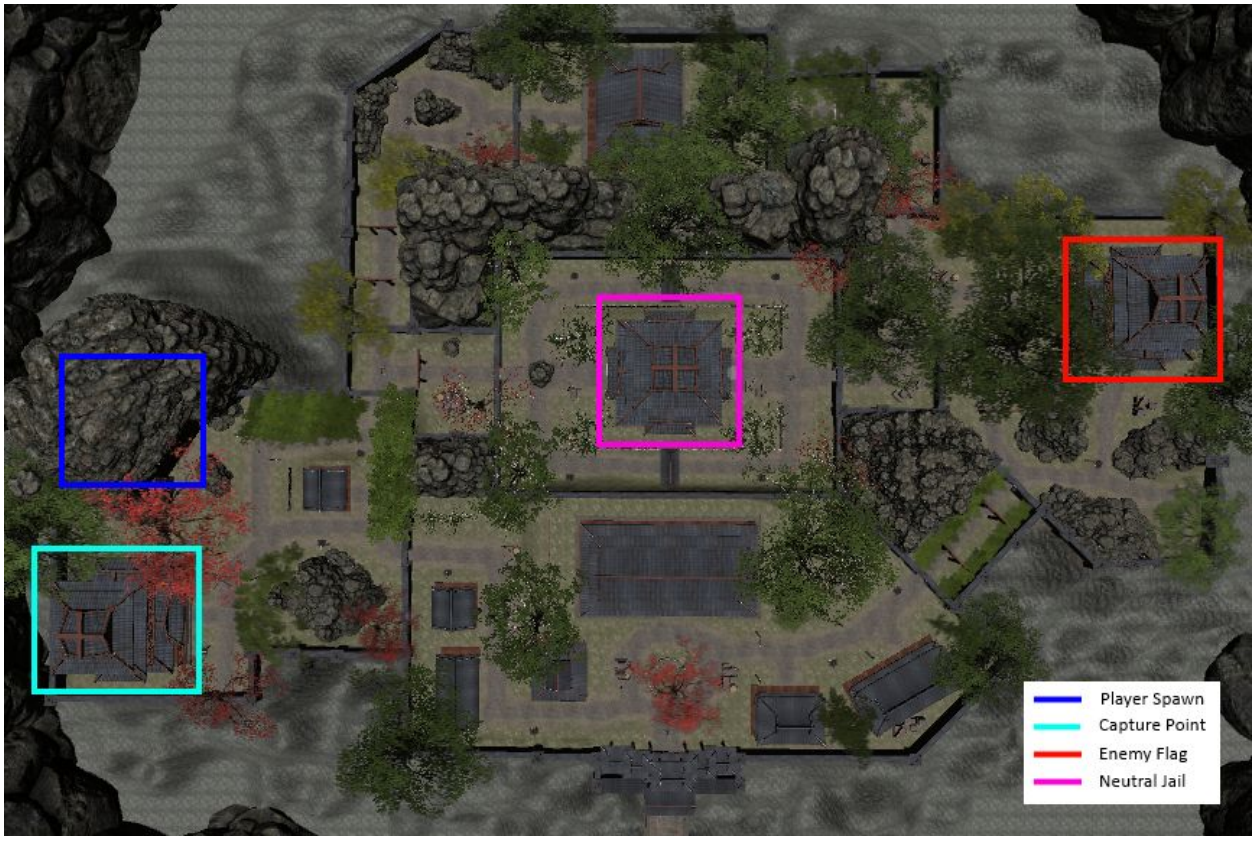


Figure 15: Overview of the map and its key points of interest

Bibliography

- Engine
    - Unity3D v5.5.0f3
- Assets
    - Moons and Night Skydome
        - https://www.assetstore.unity3d.com/en/#!/content/48514
    - Campfire
        - https://www.assetstore.unity3d.com/en/#!/content/45038
    - Simple Torch
        - https://www.assetstore.unity3d.com/en/#!/content/7275
    - Photon Unity Networking
        - https://www.assetstore.unity3d.com/en/#!/content/1786
    - RPG Character Mecanim Animation Pack Free
        - https://www.assetstore.unity3d.com/en/#!/content/65284
    - Kunoichi Ninja Character
        - https://www.assetstore.unity3d.com/en/#!/content/64665
    - Asia-Far East Environment
        - https://www.assetstore.unity3d.com/en/#!/content/21298
    - Universal Sound FX
        - https://www.assetstore.unity3d.com/en/#!/content/17256
    - JAPANESE ORIENTAL Free
        - https://www.assetstore.unity3d.com/en/#!/content/18495
    - AVPro Video Capture
        - https://www.assetstore.unity3d.com/en/#!/content/2670
    - YOZAKURA-Regular
        - http://www.dafont.com/yozakura.font