Nightwatch

Eric Wang - Computer Science
Michael Abelar - Computer Science
Arjun Guru - Computer Science
Dhruv Karthik - Computer Science
Matt Lebermann - Computer Science

Executive Summary

Battle royale game modes comprise some of the most popular video games in the market currently, with marquee titles such as Fortnite, Warzone, PUBG, and more. Most of the offerings in this category are first-person shooter games. However, there is potential to integrate the widely popular battle royale game mode with gameplay elements of popular single player games such as Assassin's Creed that focus on stealth among crowds and slower, more clever gameplay. Currently, no offerings combining these two popular elements of video games are available on the market.

With **Nightwatch**, we are developing a stealth-focused multiplayer battle royale game, where players are surrounded by crowds of bystanders. Each player must track and eliminate his target while remaining inconspicuous by blending into the crowd, and the last man standing wins. This game combines the elements of stealth that brought titles such as Assassin's Creed and Hitman to widespread popularity with the competitive and challenging nature of battle royale games such as Fortnite or Warzone. This game is being developed with Unreal Engine 4.

Our game relies on mechanics by which crowds of people interact with their surroundings. Towards this, we have designed the behavior of non-playable characters (NPCs) to be reactive to their environment. In particular, NPCs may gravitate to points of interest in their environment; they will also panic and run when they witness an attack occurring nearby.

Because our game will rely heavily on the multiplayer experience, we are also working on a scalable and reliable full multiplayer system to allow for up to 100 concurrent players in a given game. This multiplayer system will be responsible for intelligently allocating users based on skill and location into a game lobby. To implement this, we are using a microservice architecture to create services on AWS that automatically scale up and down the number of game servers, striking a balance between cost effectiveness and multiplayer availability.

Description

Gameplay Mechanics

Players have a few key abilities. The first is to attack other players or NPCs. An attack can kill an NPC or eliminate a player, and it will also alert NPCs in the area and cause them to panic. Players can also disguise themselves by changing their costume and apparel to mimic that of the surrounding NPCs or players.

Each player is assigned a target that he must find and eliminate. Upon elimination of his target, the player will receive a new target (a player alive and remaining in the game) to eliminate. The process of target assignment and elimination continues until only one player is left standing and declared the winner.

Crowd Mechanics and AI

The first component of the crowd mechanics is the passive behavior of NPCs. NPCs have a few distinct passive behaviors. In the first, NPCs simply remain stationary until something in their environment agitates them. In the second, NPCs walk along predefined paths in groups.

The second component of crowd mechanics involves points of interest in the environment. For example, NPCs may see a bar and wish to go take a seat. These “triggers” in the environment are called hotspots, and upon perception of a hotspot with space available (i.e. not already filled with NPCs), an NPC will go toward that hotspot and behave in a manner prescribed by the type of hotspot.

The final component of crowd mechanics involves NPC’s reactions to aggression by players or, in other words, their “panic mode.” When an NPC witnesses an attack, their panic state is triggered; at this point, the NPC will run to the nearest exit from the scene.

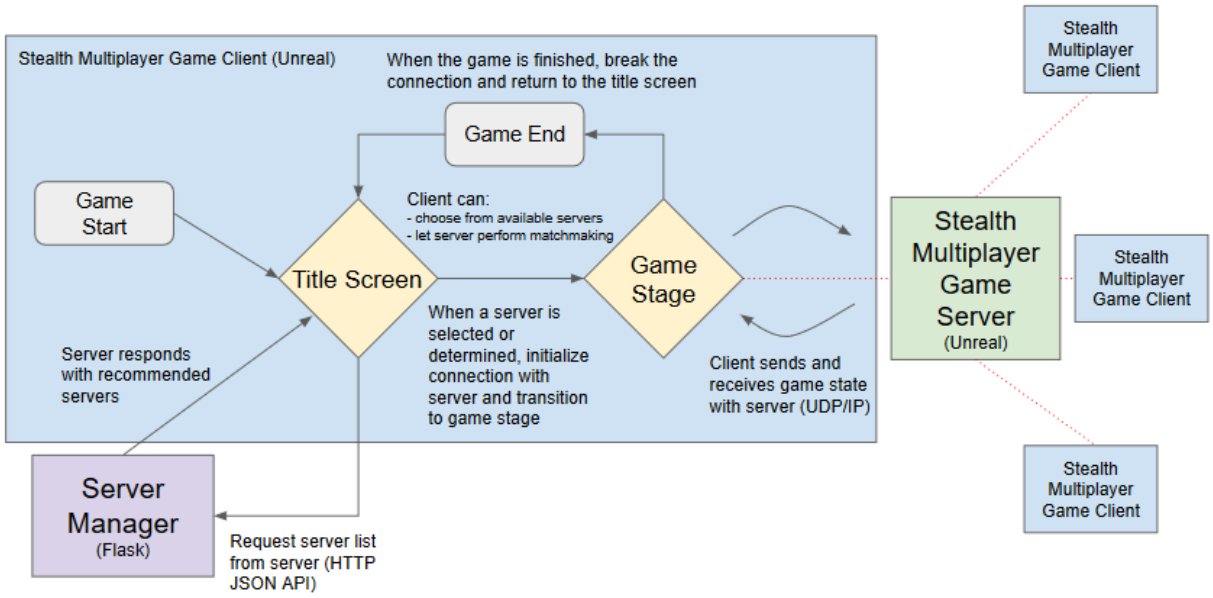
All of these crowd mechanics allow for varied gameplay strategies and game progressions. A player may visit a bar with a crowd of NPCs to blend in while tracking a target, or he may blend in with a moving crowd to make an approach. Alternatively, the player may directly rush his target, thus scaring the crowd and drawing attention from other players in the area. There are many opportunities in the game that give players the freedom to pursue their own paths and incur the risks and rewards of those respective choices made. Ultimately, these crowd mechanics allow scenes to come

alive and give players the sense of interacting with a lively and rich environment that they can dramatically influence with their actions.

Multiplayer System

Game Server

The first component of the multiplayer system was implementing a multiplayer client within the game to interact with the multiplayer system. An overview of this game integration with the multiplayer can be found in the diagram below:

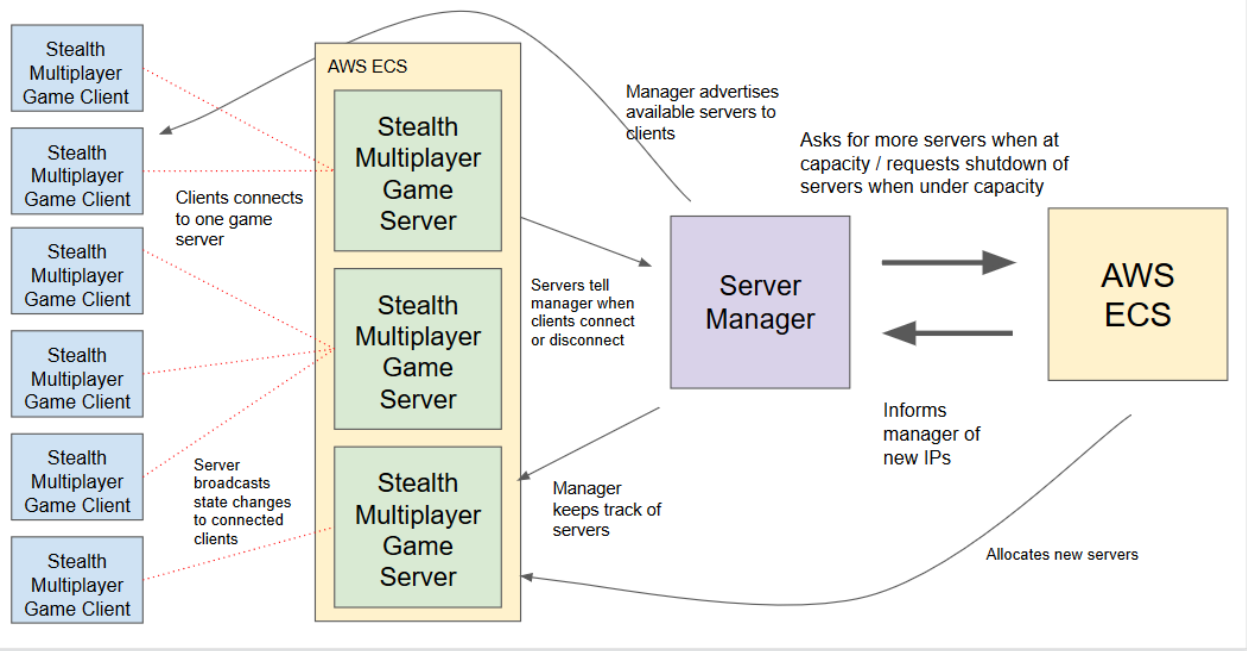


Once the client was in place, a continuous integration and continuous delivery pipeline was set up such that the development code can be compiled into a Linux game server file. This compiled Linux game server file, when run, is able to accept game client connections and host a multiplayer game. Once a Linux game server file is compiled, it is automatically deployed to the fleet of cloud Linux servers hosted on AWS EC2.

Server Manager

Any multiplayer game should always allow users to connect to a game server and play with other users. However, a game developer should also aim to not waste resources running unused servers. To solve this problem, we create a dedicated service called the Server Manager (SM). The SM is hosted on AWS EC2. The SM is responsible for overseeing all game servers that are running the Linux game server file

and informing clients which servers are available. Based on the number of users connected to various game servers, the SM then scales up or down the number of servers. To scale up, the SM uses AWS ECS to provision new game servers hosted on EC2 according to a predefined Docker file that contains the necessary dependencies for a game server. To scale down, the SM uses the same service to shut down game server instances. Because AWS ECS takes around 2 minutes to create a new server, we built a smart allocation algorithm to ensure that users will always have a server available and are not waiting for servers to be created. Below is a diagram, highlighting what is described above:



Finally, this multiplayer system is responsible for keeping track of users and key user statistics such as the number of games played and number of game victories. An authentication service is created that allows users to sign up and sign in such that when they play, key statistics mentioned above are recorded. The game client can then interact with this system to inform the user of these statistics. This system will also be responsible for keeping track of microtransactions and other purchases.

Business Analysis

Value Proposition:

This game combines two popular genres of video games that have not been combined to date: battle royales and stealth games. It gives players a fun environment where they can showcase their skill and cleverness in outsmarting their competitors.

Stakeholders:

Video game players
Distribution platforms (e.g. Steam)

Competition:

Though no games currently exist at the intersection of these two genres (i.e. battle royales and stealth games), our game will be in competition with titles from the individual genres that it combines.

Battle Royale Games
Fortnite
Call of Duty: Warzone
PUBG
Fall Guys

Stealth Games
Assassin's Creed
Hitman
Among Us

Comparison of Competitors:

Fortnite, Warzone, PUBG: Fast-paced, competitive battle royale came with looter shooter style mechanics

Fall Guys: Alternative battle royale based on platforming

Assassin's Creed: Singleplayer, story-driven game with focus on slow, stealthy play

Among Us: Mobile device-friendly, multiplayer, team-based game focused on stealth and remaining inconspicuous, detecting unusual behavior

Our Game: Slow-paced, competitive battle royale with focus on stealth, remaining inconspicuous, and detecting unusual behavior

Market Opportunity:

Our core market includes battle royale and general multiplayer game players in addition to players of singleplayer stealth-focused games. Our game specifically provides a refreshing new experience that moves away from the typical fast-paced shoot and loot paradigms of existing battle royales. We therefore believe there is a great opportunity in the segments of general multiplayer and battle royale game players. We plan to drive early adoption and gain publicity through forums such as r/gaming on Reddit as well as word of mouth; as a result, initial adoption will be slow. However, as our product gains traction on distribution platforms such as Steam, we believe adoption will quicken. Additionally, much of the virality of these games comes from popular streamers on twitch creating content about the games; we believe this will be a key dynamic in the growth of our game.

Games in the battle royale genre have shown promising growth over the previous years. Fortnite, a popular battle royale game, has had a 6.1% CAGR over the past four years. We estimate that our game would have similar growth to others in its respective genres.

Costs:

After the game is developed, the only costs would be supporting the multiplayer infrastructure and ongoing maintenance. Ongoing maintenance of the game will hopefully be done by the senior design team; however, there is a chance that an outside developer will have to be brought in to help with any patches or content updates. This developer would likely be paid hourly. According to Upwork, a freelance platform, the rate for a Unreal Engine developer appears to be $30 an hour.

In terms of supporting the ongoing multiplayer infrastructure, the cost of hosting the server manager would be around $35 a month as the server needs to be replicated across different AWS data centers to ensure reliability. The server manager being down would mean the multiplayer functionality is not available, making the investment into reliability essential. Each of the game servers hosting the game would cost around $15 a month, assuming the game server is running and not shut down by the server manager. Each of these game servers is able to serve up to 100 clients so the server costs would roughly be linear with the number of active players at a given time. AWS ECS costs would be negligible as the service is a wrapper for deploying new EC2 instances.

Revenue Model:

We plan to use the microtransaction model employed by a number of other players in the battle royale space. This revenue model focuses on enticing players with the world of our game before giving them competitive and/or non-competitive

advantages in our game as a result of paying a premium. Players can buy a “battle pass” for a set price each season (where a season lasts for a number of months), and that battle pass would allow users to unlock exclusive cosmetic items (such as unique taunts or dances). This is a proven strategy; it has been used to a great degree of success by titles such as Fortnite.