Haunted Hustle

Jordan Davis, Nicole Furlage, Jacob Hreshchyshyn, Bradley Potzka, Meghan Vaughn, Yoshihiro Kobayashi

Arizona State University
University Drive and Mill Avenue, Tempe, AZ, 85281, USA
jsdavi21@asu.edu, nfurlage@asu.edu, jhreshch@asu.edu, bpotzka@asu.edu, mvaughn8@asu.edu, ykobaya@asu.edu

ABSTRACT
The goal of this project is to develop a networked multiplayer game that plays like a platform-fighting game like *Super Smash Bros*. This goal also includes the development of a database system that stores player information such as login data and win ratios.

To facilitate collaborative game development across a team of five developers, the Unity game engine was employed during development. Networked play was accomplished through the use of the Photon Unity Networking (PUN) framework while player data storage was accomplished using an ASP.NET API and an AWS server that communicates with a PostgreSQL database, which is ultimately responsible for storing player data. The project made use of these tools to ease integration of network features in the Unity project while also ensuring the utilization of the team's familiarity with database design. In addition to these tools, the team utilized Trello to assist in the process of decomposing the project into individual tasks for easy distribution across team members. GitHub Desktop and GitHub were also utilized for version control.

Leading up to the completion of the project, the earlier versions of the game were heavily playtested by other teams in the CPI441 Gaming Capstone class. The project received high praise for the simple, yet fast-paced gameplay along with the ability to play with friends in lobbies that players set up. The project also received some critiques, such as the indication that the controls were perhaps too simple along with the acknowledgement that certain features, such as leaderboards, were not working since they were playing the Beta version of the project. While the unfinished aspects of the project will be addressed in the final build of the game, we also believe that, based on this feedback, we would seek to improve the game by injecting variety into the gameplay through the additions of gameplay mechanics, such as simple combos, as well as items and more unique stages.

Author Keywords
Network, Online, Multiplayer, Unity, Photon, Database, PostgreSQL, ASP.NET, AWS, RDS, ngrok, Battle Royale

INTRODUCTION
In recent years, the gaming industry has seen a rise in popularity in the battle royale genre, a type of online multiplayer genre in which a usually large collection of players compete with each other simultaneously to be the last man standing. The genre's modern rise to popularity can be attributed to games such as *PlayerUnknown's Battlegrounds*, *H1Z1: King of the Kill*, and *Fortnite*. More recent adaptations of the formula can be seen in various platforming games such as *Fall Guys: Ultimate Knockout* and *Super Mario Bros. 35*.

Following the belief that there remain possibilities to explore the battle royale genre in the space of platform-fighting games like *Super Smash Bros.*, our team decided to create *Haunted Hustle*, a Halloween-themed platform-fighting game that presents battle royale elements in its incorporation of robust online multiplayer support and the pursuit of self-preservation at the expense of others. Aside from the game's unique Halloween-themed art style, *Haunted Hustle* distinguishes itself from other battle royale games in its incorporation of a knockback system, which causes the player receiving a punch from an opponent to be sent flying farther and farther after each subsequent punch received, thereby increasing the frantic energy of game sessions. In addition to these innovative design decisions, the game incorporates more traditional features that one might expect in a multiplayer game, like the ability to create lobbies for hosting game sessions and leaderboards that store player information on a database hosted on a cloud server.

This paper will explore the tools used to implement *Haunted Hustle*, the process of implementing the game's key features, feedback received from those

who playtested the game, and the analysis of that feedback, which will inform the future work to be done for this project.

BACKGROUND

The following is a list of the technologies used during project development:

Unity 2020.3.14f1 and Unity 2020.3.19f1

The Unity game engine is a widely-used game engine developed by Unity Technologies. Its popularity is largely attributed to its relative ease of use due to the large body of online resources on Unity game development as well as the engine's use of C# as a scripting language for Unity developers. Unity also has support for many accessible libraries, such as PUN, that can greatly expand the capabilities of a project with minimal effort from the developer.

Photon Unity Networking (PUN) Framework

PUN is a networking framework developed by Exit Games. *Haunted Hustle* utilizes PUN due to its easy integration with the Unity game engine along with the large quantity of online documentation supporting development with PUN in Unity. PUN is responsible for the multiplayer features currently present in *Haunted Hustle*.

ASP.NET

ASP.NET is an extension of the .NET developer platform that specifically contains libraries for building web applications. These libraries were utilized to create an API that communicates with a cloud-based storage solution to send and receive information about player data.

AWS/PostgreSQL

AWS is an Amazon service that provides a multitude of cloud computing platforms, such as EC2 server instances. Our project relies on this cloud computing service to host our PostgreSQL database, which acts as the main data storage system for player data.

ngrok

ngrok is a command-line-based software that allows web services and APIs to be deployed directly from the programmer's computer, without having to pay for a hosting service. It was used to allow our API to be accessible online for the game to make requests on our database.

Blender

Blender is an open-source 3-D modeling program. The tool is a good choice for our project because it is free to use and because it is relatively easier to model with. The tool has been used in the creation of our Halloween-themed player characters.

Kenney.nl

Kenney.nl is a collection of modular game assets designed to meet a variety of needs among independent game developers. *Haunted Hustle* has made use of the platform pack when fleshing out the main battle stage's layout.

SYSTEM IMPLEMENTATION

The development of *Haunted Hustle* can be broken up into three main components. The first main component is the development of the Unity executable itself, which involved the development of main game features like scenes and player controls along with integration of networked play through Photon. The second main component is the development of the art assets for the game. These assets were sourced from Kenney, sourced from the Unity Asset Store, and developed in-house using Blender. The final main component is the development of the database solution for storing player data. This involved designing and implementing a database in PostgreSQL, hosting the database on AWS, creating an API for interacting with the database, and integrating the API with the Unity executable. This section will explore in more detail the development of each main component.

Unity Executable Development

The development of the Unity executable involved a joint effort to implement the main features that players would interact with and to integrate networked play through Photon. To help accomplish these tasks, the team adapted a set of platforming controller scripts previously developed by one of the team members so that players can move using a 2D sidescroller means of playing. Additionally, a simple collision object was attached close to the front of the player model. This collision object, when enabled with a mouse click, would simulate a punch and would cause players interacting with it to enter a knockback phase, which would increase the percent-based damage of that player and cause the player to fly away without control for a period of time that scales with the amount of damage taken. All of these features were achieved with the standard Unity game development pipeline of creating scripts as components to add behavior in a modular fashion.

The integration of Photon provided an additional challenge due to Photon's use of remote procedure

calls. The implementation of these remote procedure calls was vital in conveying interactions between players on the network as well as the moving spiked walls in the scene. Data such as player transform information, animation data, collision data, etc. not only had to be rendered for the player controlling their own character, but also had to be rendered for other players in the network. The use of PUN RPC's helped accomplish the data passing necessary to render that data and enable players to play with each other online.

Art Sourcing and Development

*Haunted Hustle'*s art style is simplistic, low poly, and heavily inspired by horror themes. The main level is set on an abandoned graveyard with an underground crypt, and each of the playable characters are based on popular Halloween monsters. The decision to go with this art style was partly due to the variety of free low poly assets available online, and partly due to the ease of creating low poly models in-house. Several asset packages on Kenney.nl contained grass platforms and various graveyard or spooky props that the team could use in building the main level. Other art assets not sourced from Kenney were found on the Unity Asset store, most notably the particle system pack and a lava-distortion texture.

The use of free, pre-made assets allowed time for the three playable character models to be developed. The wolf, vampire, and pumpkin models were created using Blender and texture maps painted with Clip Studio Paint. All three characters share the same height, build, and rig. Character animations were added using Mixamo's automatic character rigging. By simply defining where key joints are on a model, Mixamo generates a humanoid skeleton rig with proper weight values assigned to the geometry. Further polish was added to the animations in Unity, where a Blend Tree was used to smoothly transition the idle animation into the running animation during player movement.

Other models created by the team's artist in Blender include the Death Wall and the background terrain in the main level. Additionally, user interface elements were custom made using a combination of Adobe Illustrator, Clip Studio Paint, and free-to-use fonts.

Player Data Storage Development

The database was created using an AWS Database, PostgreSQL, and ASP.NET. The AWS service we chose was the Relational Database Service (RDS) of a db.t2.micro size. We then connected to the database through the PostgreSQL desktop app. The columns of the table are the id, username, number of games won, and number of games played by a particular player. Finally, an ASP.NET API was created. It was coded on VisualStudio 2019, and as such already had Swashbuckle for ASP installed. This allowed us to visualize the API calls when the program ran on localhost. Four API calls were created: GET, PUT, POST, DELETE. GET returns the full table of all players and their win data. PUT adds a new player to the table. POST updates a single player's data. DELETE removes a single player from the table. These API calls were integrated into the Unity game at appropriate times; PUT when a player enters a username that is not registered in the table yet, POST at the end of a match, and GET when the player enters the leaderboard screen. DELETE was not integrated, as there is no true "account" for the players to delete (i.e. we did not gather passwords or any other data outside of a username). However, it was useful in removing the usernames we registered while testing the leaderboard system. The API was hosted through ngrok, a software that allows APIs to be hosted locally, directly from the command line.

CASE STUDY

Our team took every opportunity to learn how players responded to the game as it proceeded through its development increments. The most valuable of these opportunities were the times each of the gaming capstone teams were given hands-on experience of the other teams' projects and provided feedback. Using our own project as a case study, we worked to improve our project using the feedback provided on the project's progress. Examples of improvements to the game from such feedback include criticism on a piece of the player's movement behavior that caused the player to stick to the wall after jumping into it.

There were also numerous suggestions about additional features to include in the game. These suggestions helped immensely in informing how the game can be improved beyond the gaming capstone class.

CONCLUSION AND FUTURE WORK

In conclusion, *Haunted Hustle* represents a successful implementation of a platform-fighting game with battle royale elements. The success of this project can be attributed to the clever adaptation of good player movement scripts, the use of a robust framework for network play, expertise in database design, and the employment of foresight in standardizing player character behaviors and properties to minimize the effort of developing corresponding art assets.

Using the feedback received from the gaming capstone teams in earlier project development increments, the game can be improved further by developing more unique stages on which players can compete. Additionally, the main gameplay features can be expanded to include different ways of attacking players, such as executing simple combos. Finally, to further emphasize the party game nature of *Haunted Hustle*, items and power-ups can be developed.

REFERENCES

1. Amazon Web Services. (2022). AWS, Retrieved April 15, 2022, from https://aws.amazon.com
2. The Blender Foundation. (2022). Blender. Retrieved April 15, 2022, from https://www.blender.org
3. Exit Games. (2022). Photon. Retrieved April 15, 2022, from https://www.photonengine.com/pun
4. Kenney Game Studio. (2022). Kenney. Retrieved April 15, 2022, from https://www.kenney.nl
5. Microsoft. (2022). ASP.NET. Retrieved April 15, 2022, from https://dotnet.microsoft.com/en-us/apps/aspnet
6. The PostgreSQL Global Development Group. (2022). PostgreSQL: The World's Most Advanced Open Source Relational Database. Retrieved April 15, 2022, from https://www.postgresql.org
7. Unity Technologies. (2022). Unity. Retrieved April 15, 2022, from https://unity.com/