Twitch Trivia Battle Royale: Interactive Entertainment Engineering Game

A Senior Project
presented to
the Faculty of the Computer Science Department
California Polytechnic State University – San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science

by

Noah Ravetch

March, 2023

Introduction:

As the world of digital media evolves, so too does the way producers and consumers of entertainment content interact with each other. Live streaming is one such evolution. In this format, one person broadcasts their camera and/or their computer screen to a large audience of viewers in real time. People tuning in can communicate with other viewers and the streamer using the chat feature built-in to the streaming platform.

A new type of entertainment has recently entered the marketplace: interactive entertainment. Concerts are being held virtually in games like Fortnite (Epic Games 2021). TV Shows on Netflix are beginning to experiment with choice-based narrative options akin to a "Choose Your Own Adventure" game. Live streaming has also embraced this new direction of engaging content. In the past decade, the chat feature has been utilized as more than just a communication tool. In 2014, a programmer utilized the Twitch API to convert chat messages into game commands, starting the first iteration of "Twitch Plays Pokemon" (Helixpedia 2022).

The goal for my project was to create a trivia game targeted specifically for Twitch streamers to play with their viewers. Rather than have viewers collectively control one character, each viewer has an individual entity they are in charge of, using basic chat commands to move in the game. While the core gameplay is simple, moving one's character to the correct location corresponding to the answer of a trivia question, the competitive element of this game distinguishes it from other Twitch chat controlled experiences.

Background:

The game was developed using Unity. Unity is a 2D/3D game engine that doubles as a cross-platform IDE for game developers. Unity provides several built-in features in its game engine that make games work. Game elements such as collision detection, object management, and physics are all included. Furthermore, Unity has an "Asset Store" which allows developers to search for objects, environments, and more that other people have created.

In addition to being a fully-functional game engine, Unity also serves as an integrated development environment, or IDE. When working in Unity, developers can utilize its visual editor to add objects to game scenes using a simple drag and drop system and manipulate their properties. To further ease development, the game's folder structure has a dedicated UI that is simple to navigate and add new files to. Finally, Unity supports C# scripts that can be attached to game objects that can contain game logic and other game elements that the developer wants to directly control with code (Unity Technologies).

Related Work:

The main source of inspiration for this project came from the Twitch streamer DougDoug (Wreden). In an effort to make his streams as engaging as possible, he utilizes the stream chat to add a level of interactivity to his streams. Most prevalently, he uses a betting program to allow viewers to wager fake currency on different video game outcomes and Python code that converts chat messages into controller inputs. The goal of my project was to combine the two key features from each of these interactive experiences into a single product.

Most iterations of "Twitch Chat Plays" have the entire chat controlling one entity. This lends to a more collaborative experience, with the community coming together to complete a single goal together. Twitch Plays Pokemon is the pinnacle example of this type of experience. Other versions of Twitch interaction include allowing the audience to vote on options within the game and affect certain outcomes. This ranges from voting on funny prompt responses in

Jackbox Games or giving the streamer buffs or additional challenges in games like Streets of Rogue. My game is more individual focused, with each viewer controlling their own entity.

In contrast, the most prominent game that utilizes Twitch chat and has individual entities for each viewer is Marbles on Stream made by Pixel by Pixel Studios Inc. In this game, each viewer is assigned a marble that races other marbles in a gravity-powered marble run. While this game shares features with mine, including simple setup and username displays, the game lacks player agency and does not allow any influence over the viewer's marble while racing.

Another game that focuses on viewer individuality is Kukoro: Stream chat games made by HeyNau. While not as well known as Marbles, this game includes several minigames that give viewers direct control over their character. This game shares the goal of my project: to increase viewer engagement through interactivity. My game distinguishes itself through different mechanics - a focus on trivia instead of real-time inputs - and art style - 3D polygons instead of 2D pixel art.

Design:

The gameplay will be based on a combination of the children's game Four Corners and trivia. In the game Four Corners, there is one designated person who stands in the center of a room with their eyes closed. Everyone else playing then chooses to hide in one of the four corners of the room, numbered 1 to 4. The person in the center then calls out one of the numbers, eliminating any player standing in that corner. The game repeats itself until only one player is left. This game works well with the Twitch Chat format, only requiring one command from each viewer per round.

Naturally, the streamer plays the role of the person in the center. However, because the analog of closing one's eyes to prevent bias for which corner to choose does not translate well, trivia is used to give meaning to each of the corners. Rather than have three safe corners, each incorrect answer eliminates players who selected that corner. These eliminations are permanent until one player is left, preserving the spirit of the original game.

Instead of including only factual questions as part of the question pool, the game also includes the ability to write streamer-specific trivia. These questions provide the opportunity to ask something subjective about the streamer, allowing the streamer to come up with questions with their own correct and incorrect answers. An example question might be "What is the streamer's favorite color?" with the various corners including things like "Red, Green, etc.", with only one color being the correct choice..

Joining to participate in the game operates similarly to Marbles on Stream, where a command similar to "!join" will read in the user's Twitch username and add it to the list of participating players. A data structure stores a list of current players and keeps track of various attributes, such as whether the player is still alive in the game and where they are located in the play area.

The final product is a downloadable Windows program that the streamer can launch and live stream to Twitch. With the game running locally on the streamer's computer, their personal internet will handle all of the API calls with no external servers required. The Twitch API is accessed by leveraging the code written by GitHub user LuckyNoS7evin to convert chat commands into game commands.

Implementation:

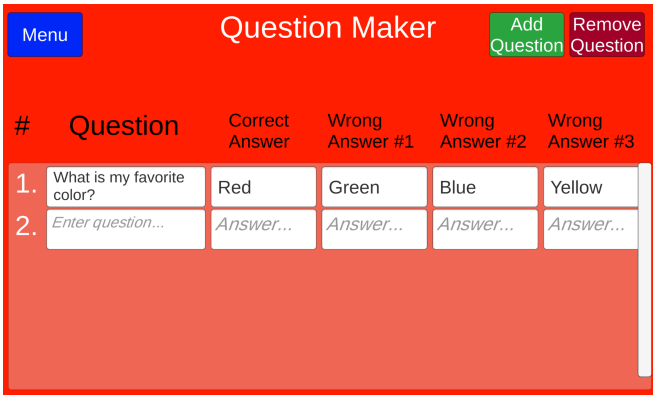
In my implementation, I created several scripts divided into three main categories: backend, data handling, and game object handling. The backend scripts are responsible for communicating with the Twitch API, allowing for the game to both access messages sent through Twitch chat and send messages via a moderation bot. The moderation bot can be used to send real-time messages, which can be used, for example, to let players know that the game is no longer accepting new game commands. Accessing chat messages allows them to be converted into game commands. The messages are processed by the following pseudocode:

```
Client_OnMessageReceived(TwitchEvent event):
	String message = event.message
	String username = event.username
	if (scene is JoinScene and message == "!join" and user has not yet joined):
		Add user to list of players
		Update UI to display player
	else if (scene is GameScene)
		chosenAnswer = message // "a" or "b" or "c" or "d"
		Move player to chosen answer corner
```

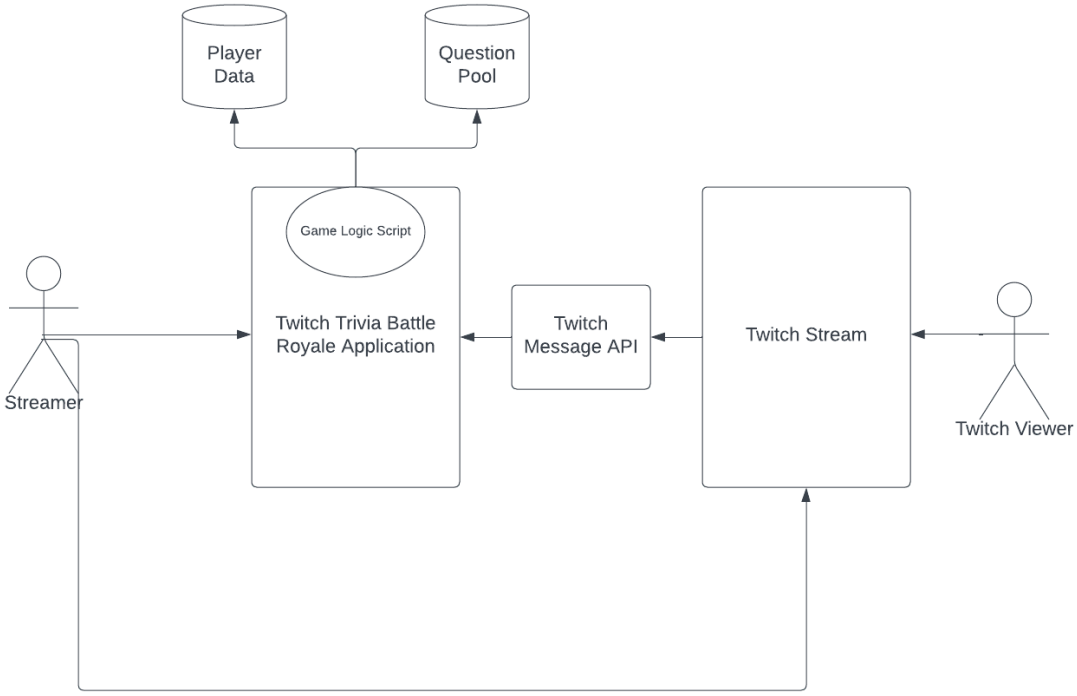
The data handling scripts control the core logic of the game. The "GameData" script keeps track of the participating players in a dictionary. This dictionary stores information about the player such as their Twitch username, and game-specific data like their position in the arena.

```
Dictionary<string, PlayerInfo> players
class PlayerInfo:
	string username
	GameObject characterModel
	Color color
	float xOffset # used for positioning in the game
	float zOffset # used for positioning in the game
```

The "QuestionData" script holds the pool of questions to prompt the players with. There is a UI screen in the game that allows the streamer to write custom questions. This feature allows them to swap the pre-written trivia question pool with their own. In tandem with this one, the "QuestionManager" script keeps track of which questions have already been answered and



randomly picks the next question from the remaining ones. Finally, the “GameLogic” script keeps track of the various game states and transitions between them. It fires the calls to both the player data and question pool data structures when the current game state requires them.



The game object scripts communicate directly with both of the other categories, moving the players and various other environment objects to convert the data into a playable experience. These range from moving player objects to the proper answer corner when a twitch command is processed and updating the UI to display new questions.

In addition to the standard Unity development the game required, I had the opportunity to work with my music major friend, Alexander Word, to compose music tracks for the game. The project features three custom songs to accompany the gameplay. There is one that plays when the question is prompted and players choose their answer choices, a second that plays when the answer to the question is revealed and players get eliminated, and a third that triggers on the victory screen when one player is left alive. These songs greatly improve the energy of the gameplay and add to the overall presentation of the game.

Analysis / Verification:

Based on the playtests conducted throughout the game’s development cycle and at the end of its completion, I have deemed the project a success. The main goal of the project was to produce a game that could be used by streamers to play with their chat and increase the interactivity of their streams. Playtesters of the game all enjoyed participating in the trivia and especially favored the streamer-specific trivia to the standard questions that they were accustomed to.

Diving deeper into the playtester feedback, the first iteration included the most notes for things to improve. Several of the UI screens were accompanied by bright background colors and

were difficult for playtesters to look at. In the initial build of the game, there were also bugs that made the game unplayable, such as the questions sometimes failing to load into the UI. The second iteration of the build fixed these issues, but introduced player collisions when the movement of player objects changed from teleporting to the correct location over to translating across the map. Playtesters actually enjoyed this unintended feature, having the ability to push other players around by running into them. Ultimately, I felt that this detracted from the intended design of the game and removed collisions. It did give me inspiration for a future game project though.

By the end of all the iterations, all of the large bugs and major issues playtesters had with the game were removed from or fixed for the final deployment. This allowed the final playtest of the game to run without any technical issues.

While one of the potential goals was to have this project be scalable for use with any sized live stream, this element was ultimately scrapped. It would have been difficult to test for scalability without having additional tools or a big playtesting audience to simulate a large player base. In addition, that would have taken development time away from more important features.

Future Work:

The game as it stands can be expanded upon to improve its mechanics, ease of use, and presentation. Mechanically, there is space to be explored with respect to how trivia questions are generated. For example, a very generic question such as “What food does the streamer like” could be asked. The people chatting could then submit possible options for the answers with the Twitch API finding the most popular responses or choosing randomly. These chosen options would populate the four choices. This feature would additionally require the streamer to manually select which answer choices would be considered “incorrect.” This variant of the question format could add additional interactivity and engagement to the game.

In terms of ease of use, there is currently no way to write custom questions besides manually entering them in before each round. An option to use a text file to upload custom questions could speed up this part of game setup. The game’s static question pool is also small and could use additional questions. Finally, the game currently only works using my own Twitch channel. This decision greatly sped up development as the game really only needs to use a chat server to function properly. However, having the game be compatible with the particular streamer’s channel would make the game easier to use.

Because the majority of my development time was focused on gameplay and features, my strong suit, there are several presentation aspects that can be improved. The UI used in the game uses the bare minimum utilization of Unity’s TextMeshProGUI. Each of the UI elements could be improved aesthetically. The players are also currently represented by capsule objects and can be updated to have full character models and walking animations. The text in the game can be reworked to avoid awkward multi-line usernames, for example. And, while music is present in the game, sound effects are absent.

Conclusion:

I am very proud of the final build of this game. It fulfills all of the initial requirements I had in mind when brainstorming the game concept and it is something I genuinely enjoy playing with other people. I think this project demonstrates the ease of creating games or other programs that interface with the Twitch API. As live streaming as a medium of entertainment continues to expand, the opportunities to create games that complement the content, increasing interactivity

and engagement will only grow in number.

Bibliography

Dabrowski, M. (2019, July 12). Streets of Rogue. Steam. tinyBuild. Retrieved October 7, 2022, from https://store.steampowered.com/app/512900/Streets_of_Rogue/.

Epic Games. (2021, August). *Rift Tour | A Musical Experience Unlike Any Other*. Fortnite. Retrieved October 6, 2022, from https://www.epicgames.com/fortnite/en-US/rift-tour.

Helixpedia. (2022, January 4). *Gen 1 (Pokemon Red)*. TTPedia Wiki. Retrieved September 30, 2022, from https://helixpedia.fandom.com/wiki/Gen_1_(Pokemon_Red).

HeyNau. (2020, November 6). Kukoro: Stream chat games. Steam. Retrieved October 7, 2022, from https://store.steampowered.com/app/1166990/Kukoro_Stream_chat_games/.

Jackbox Games, Inc. (2014). Jackbox Games. Steam. Retrieved October 7, 2022, from https://store.steampowered.com/developer/jackboxgames/#browse.

LuckyNoS7evin. (2021, September 25). TwitchLib. *GitHub*. Retrieved October 5, 2022, from https://github.com/TwitchLib/TwitchLib.Unity.

Mann, S. B. (2019). Four Corners Game. Icebreaker Ideas. Retrieved October 7, 2022, from https://icebreakerideas.com/four-corners-game/.

Nolan. (2021, November 3). Games with Twitch Integration. StreamScheme. Retrieved September 30, 2022, from https://www.streamscheme.com/games-with-twitch-integration/.

Pixel by Pixel Studios Inc. (2018, March 28). Marbles on Stream. Steam. Retrieved October 5, 2022, from https://store.steampowered.com/app/1170970/Marbles_on_Stream/.

Unity Technologies. Unity. Retrieved October 5, 2022, from https://unity.com/.

Wreden, D. DougDougW. Twitch. Retrieved October 5, 2022, from https://www.twitch.tv/dougdougw.