Community-based Play in Twitch Plays Pokémon

Max Mallory

Introduction

On February 12th, 2014, an Australian programmer and fan of the *Pokémon* series began a live stream of a program he was running on his computer. The program was a Game Boy Advance emulator entitled Visual Boy Advance, running a hacked copy of *Pokémon Red*, the first JRPG in the now-famous *Pokémon* series. The start screen displayed the classic black-and-white image of trainer Red and starting Pokémon Charmander.

Alongside the emulated screen, there was a list of usernames. Next to the usernames were either "b", "a", or the directional buttons found on a Game Boy Advance, buttons that the emulator thought were being pressed, which caused actions to be taken by the player in *Pokémon Red* (see figure 1). These actions were witnessed by hundreds of thousands of people with frustration, laughter, and sometimes even triumph.

Nobody was aware of what was going to happen. Barely anyone had heard of the Australian programmer's idea on February 12th, but for those who had, the concept was enticing. "Enter button inputs via chat!!"the title read. It seemed bizarre that this existed. People logged on and tried it out. It seemed even more bizarre that it worked. Multiple people were playing the same game, on the same system, together. This wasn't a multiplayer game, this was something different, something new.

Figure 1. Twitch Plays Pokemon Interface

The output was calculated by a simple IRC bot written in Python. It connected to the chat server of Twitch as well as the emulator running the game, taking inputs from people in chat and executing them in the emulator. It was just a small social experiment, but it seemed like a great way to get a few hundred people to collaborate on a game in a new and interesting way, as long as all of them communicated with each other.

But then, the people who had found the channel told their friends about it. And then those friends posted the stream's URL on Facebook and Twitter. And then a media outlet or two wrote a post about it. And then, just six days after the stream went live, 121,000 people were witnessing the chaos. It was a flurry of commands that were being processed so fast, Red was spinning around, walking in a completely unpredictable fashion, opening and closing his inventory, saving, and checking the Pok*é*dex. It was chaos, it was brilliance. It was *Twitch Plays Pokémon*.

Many players began to ask questions. How the hell did this work? Why did people actually care about beating Pokémon by typing in commands? If you want the short answer: *Twitch Plays Pokémon* was

weird. No previous games had the same mechanics in play that this did. Nobody had considered using an IRC server as a control system. The tools for something like this have only recently become popular. However, *Twitch Plays Pokémon* was also the birthplace of a new gameplay mechanic that I've dubbed "MMO input": the concept of many, many players all contributing to one game's inputs. With some creativity and a bit of hard work, creating a new gameplay mechanic is not as daunting as it sounds.

Part 1: The Creation

> *I didn't really have any plans for it from the beginning. I just wanted to put it up to see how people would respond. I put it together and put it up on a dedicated server all within a few days (McWhertor, 2014, para. 6)*

Very little is known about the person who began this entire fad, and even less is known about what their future projects will be. The creator has said that the stream is "a social experiment"(Wayling, 2014), but after the popularity boost, it's hard to believe that they won't be doing something with MMO input again. There are plenty of interviews with the creator from many reputable sources (such as Polygon, CNet, Game Informer, the Guardian, and Escapist Magazine, just to name a few), which gives us some insight into the creator's development process. There's even a few Python scripts that are based off of the programming for *Twitch Plays Pokémon*. These allow for the easy addition of MMO input to any game.

In many of these interviews, there is a mention of another popular Twitch stream/game called Salty Bet. In the interview with Polygon, the creator talked about their inspiration from Salty Bet in the very early stages of *Twitch Plays Pokémon*. "Originally I was creating my own version of Salty Bet … but after interest in that type of content died down I decided to do something different."(McWhertor, 2014, para. 7) Salty Bet was popular for a short time in the summer of 2013, but I'll expand on this later. First, we need to talk about the platform this was broadcast on: Twitch.

Twitch.tv has skyrocketed in popularity recently. The rising eSports community needs a platform to broadcast on, and Twitch dominates that market. Twitch didn't start out as a games service, though.

Twitch.tv was actually spawned by a social experiment of its own: Justin.tv. This was a website launched in 2007 by a man named Justin Kan and his friends. It was a live stream with one channel: Justin's life, broadcasted 24/7 for anyone with a computer to watch. Kan wore a webcam attached to his hat, which was then fed into a laptop that he wore in his backpack and broadcasted across the internet for everyone to see. It was a big deal, landing Kan interviews and invitations to talk about Justin.tv on mainstream television (and yes, he did wear the camera during these interviews). Eventually, the team behind Justin's "life stream", as it was deemed, decided to let other people live stream their lives and activities, and they split these new streams up into different categories. As is usual with many internet video startups, the "gaming" section grew rapidly, and the team decided it was big enough to warrant its own website. Naming the domain after the common FPS term "Twitch gameplay", Twitch.tv was born.

Live streaming in games is currently spiking in popularity. The average viewer of a Twitch.tv stream watches over 90 minutes of video a day on the site. More minutes of live streams are watched than minutes of YouTube videos (DiPietro, 2013), and most of the large eSports tournaments are now using Twitch as a sponsor and/or a broadcasting service. Twitch was the obvious choice to host the experiment, as no other live streaming site offers a focus on gaming or an audience as large as Twitch. While it was extremely popular before *Twitch Plays Pokémon* started, it garnered an amazing 36 million views during the streaming of the *Twitch Plays Pokémon*'s first game, *Pokemon Red* (Chase, 2014). People who had never heard of the site became daily visitors, and even the executives at Twitch took notice of thes stream in the form of blog posts detailing data about the channel and praising the creator. They were just as excited as the players and watchers of the stream.

This Australian programmer was not the first person to think of an

idea similar to MMO input. There are a few concepts and experiments that are very similar to *Twitch Plays Pokémon*. One of the most comparable is the Infinite Monkey Theorem, which states that a number of monkeys hitting keys on a keyboard will almost surely type a complex text, such as a famous novel or play. In the videogame realm, Loren Carpenter's 1991 *Pong* experiment, in which hundreds of unknowing participants collaborated on a game of *Pong* in a movie theater, is an older generation's version of *Twitch Plays Pokémon*. But Salty Bet, a platform that allows people to bet fictional currency on AI fighting matches, was the only one that the creator directly cited as an influence. Salty Bet is another stream that started in the summer of 2013. It's a 24/7 battle royale that takes place in the fighting game *M.U.G.E.N.*

Here's how it works. Two randomly chosen characters in the game, controlled by the AI, fight each other. Those who bet on the winning fighters with a fictional currency called "Salty Bucks"get their payout, and the odds are based on how much money is placed on a character. For example, let's say fighter B wins a match. If Fighter A has a total of 200,000 Salty Bucks wagered on them, and Fighter B has a total of 100,000, then everyone who bet on Fighter B will win double their bets. Salty Bucks are a completely fictional currency that the player obtains for free. In fact, there isn't even an option to pay real money for them. Even though the stakes aren't actually tangible, it's more exciting than you'd think.

A large part of Salty Bet's success resided in how simple the interface was (see figure 2). It was very easy to use, and the odds were automatically calculated for viewers once the fight began. In figure 2, players have bet significantly more on Ren idagawa then Cirno, thus the odds are heavily in his favor (189.6:1). TPP's interface was even simpler, with just one sidebar accompanying the game.

Since characters in *M.U.G.E.N.* are player created and uploaded (there are *thousands* of characters in the game), there's a lot of unpredictability in what a character's move set is, how the AI is coded, and who will win a match. Without real money involved there

isn't any legal gray area. What viewers found most entertaining were the ridiculous matchups that would come up, such as SpongeBob vs. Super Saiyan Goku, or, more appropriately, Pikachu vs. Ash Ketchum.

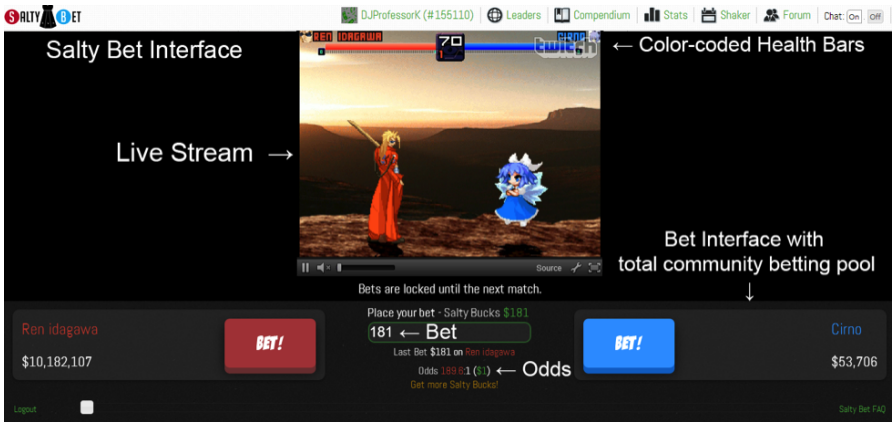

Figure 2. Salty Bet's Interface with Main Components Labeled and Leaderboards / Chat Omitted

Salty Bet took off in the summer of 2013. It wasn't nearly as popular as *Twitch Plays Pokémon* was nine months later, but it set the tone for user interaction through live streaming. The popularity of Salty Bet was equally as surprising as *Twitch Plays Pokémon*. While both *Pokémon* and *M.U.G.E.N.* are games that anyone could play on their own computers, the people behind Salty Bet and *Twitch Plays Pokémon* saw another area of potential; a way to play a game in a way completely different from what was intended. This brings life back to titles that have lost it.

In retrospect, it makes a lot of sense. Both live streams are moderately easy to set up, manage, and monitor. They're both broadcasted on a very popular website that thousands of people visit every day, and they both use very simple control schemes for people to participate in. Yet, they're still loosely-orchestrated pits of chaos.

Part 2: Managing 120,000 Button Mashers

It's to make the game beatable without modifications. I didn't want to babysit the stream and nerf [reduce the power of] the game. I'd rather the game be beaten on its own terms (McWhertor, 2014, para. 5)

"Weird"is a pretty broad term, so I'm going to clarify what I mean. When a game is weird, it is doing two things:

1. 1. Being played in a way not intended by the developer
2. 2. Showing results that are unexpected by the player

Let's take speedruns, for example. Those are pretty weird. Tool-assisted speedruns (TAS) show the completion of games in ways that are seemingly impossible. This creates some very odd-looking playthroughs of games where players take advantage of glitches and bugs: Ways unintended by the developer. These playthroughs are also unexpected by the player. While their inputs are responsive and intentional, the results clearly don't fit into the game's world. This could be anything that breaks the player's immersion: from using glitches as I mentioned above, to things like intentional ludonarrative dissonance (e.g. killing all of your allies in a first-person shooter).

Unintended by the developer, and unexpected by the player. This definition of weird slots right into *Twitch Plays Pokémon*. If a game with MMO input was attempted with face-to-face communication, it would be quite literally impossible and probably result in a few injuries. Imagine 100,000 people all trying to press buttons on one Game Boy. 20,000 of these people are also shouting instructions, complaints, and religious chants at the rest of the community. Nothing like this was ever considered by the creators of *Pokémon* when the game was made. It is weird in its beauty. None of it was ever possible without the rise of the internet and the popularity of live-streaming.

With a small city's worth of players constantly watching and participating in the stream, problems were inevitable, and happened

at various points during the stream. Sometimes the input would glitch out, or the emulator would drop frames and players would start to get confused. But the largest and most-debated problem was easily the control system. The first control system for *Twitch Plays Pokémon* was as basic as could be: when a command was said in chat, the game would display the username of the "player"with the command and then execute it in the game all at the same time. It was purposely chaotic—there was no buffer for any controls, only a 20 to 30 second input lag because of the ridiculous amount of people participating. Every control would eventually go through, which means that (of course) there were large trouble spots where one wrong button press would set the team back a sizable amount. But, it was fair. No command took precedent over the other, and every player would get their command through eventually.

There came a time where the system failed. During *Pokémon Red*, there was a section of the game the community dubbed "The Ledge", which was a one-tile-wide path with a ledge below the player that, when the player pressed down, forced them to go back to the beginning of the area and walk the path. (Neilan, 2014) Crossing the ledge took a full day to pass, as one wrong move would negate all progress, and force the player to move back to the beginning of the area. Many people lost their temper and showed it in the chat. Some even quit because of the frustration this was causing. This is a great example of how *Twitch Plays Pokémon* altered the gameplay of *Pokémon*: an obstacle that is seen as a minor part of the normal game can become a huge difficulty.

In response, the creator implemented a new system a few days later. This system introduced the settings of "anarchy" and "democracy" to the available list of chat commands. A small meter was introduced at the top of the stream, one end being labeled "anarchy" and the other, "democracy." (TwiliSailor, 2014) When a player typed in one of the two choices, a bar would sway the direction of the system the player typed in. Once the bar got far enough towards one side, the entire stream would switch over to the new system. Anarchy was the original system, and democracy was the new implementation. In

anarchy, every chat command was executed soon after it was typed, the stream just had to catch up. In democracy, there was a chance your command would not be executed. Every 30-45 seconds, the stream collected all inputs from all players and tallied them up. The highest voted command was executed. Then the process would repeat.

Players were enraged at this new system, and in response they abused another added feature to stop it. Along with the update to the voting system, players were now able to add a number after any command (in democracy mode only) to vote for its execution. Huge masses of players began typing the command "start9″into chat, which would cause the system to press the start button 9 times, thus stopping all progress of the game. (notacatchyname, 2014) Players were so upset at this system that they staged an in-game protest, and it was successful. Progress of the stream stopped for hours, and eventually the anarchy system became heavily favored by viewers for the rest of the playthrough, with democracy only popping up a few times after that.

It's amazing that things like these can happen in the fashion they do. There was a political protest staged on a website, for a game, that was actually successful. I see the start9 riot as one of the most important lessons in *Twitch Plays Pokémon*: players will get what they want, regardless of what systems are put in place to stop them. There's a common physics concept in play here: the path of least resistance. Whatever is easiest for people to use, they will use it. In *Twitch Plays Pokémon*, the easiest path was that of anarchy. There was almost no complexity and a limited set of commands. It was the easiest mode for players to play, thus it was chosen.

The control system was changed a few more times. During the second run of the stream (where players participated in *Pokémon Red*'s sequel *Pokémon Crystal*) democracy mode was activated once every hour in an attempt to get the mode more popular, but it was almost always voted back into anarchy in a few minutes. This became frustrating for many players who preferred democracy, so as a result, a much more complex system was installed: a command had to

be entered 10 times in 500 milliseconds to be executed. It was a confusing system that was abolished after just one day of use. These confusing and ineffective control schemes could have contributed to the drop in the stream's popularity. However, it seems the creator has started to understand what the players want; the controls as of this writing are in complete anarchy mode.

This community is the beating heart of *Twitch Plays Pokémon* — and communities thrive on participation. 1,165,140 people entered at least one input towards the completion of *Pokémon Red* (Chase, 2014) is what will influence future developers to explore this as an option. In fact, it already has. San Francisco game developer Michael Molinari has crowdfunded his game *Choice Chamber*, which has mechanics that change based on what people watching (and technically, playing) decide. Molinari raised over $34,000 for the development of this game. (Molinari, 2014) If it gets popular, that is just the tip of the iceberg.

Part 3: Praise Helix

> *How did they ever once open the menu and select the right item for the right moment, let alone complete the entire game? Some might attribute the stream's success to enthusiasm for the game, sheer determination, blind luck, boring post-game interview stuff like that. Most of them, though—the Hivemind, as they call themselves—they'll tell you something else. They'll say it was the will of their risen lord Helix, through His champion Bird Jesus, the sacrifice of The Twelve, and renouncing the false dogma of The Dome (Davis, 2014, para. 3)*

So let's get this straight.

Over 100,000 people are playing a two-decade old Japanese RPG at various places around the world with no interaction other than a text box. The game that these 100,000 people are playing is being run

by a machine not intended to run the game, and the screen is being broadcasted to anyone with access to a computer.

Typing that whole sentence seems like it would sum this up, doesn't it? There's a lot more to it, though. It's not just the stream itself that was weird, but everything before (or nothing, in this case) and everything after. The "game" itself was weird, but the culture is weirder.

Due to the gameplay's chaotic nature, it's hard for people to make predictable actions. This leads to a lot of things becoming bizarre that are expected to be normal by the game, such as nicknames, logical movements, and battles. Something that's usually an easy in-game task proves to be difficult. The best example of this is the nicknaming system, which has led to nicknames such as "ABBBBBBK(" and "JLVWNOOOO". It's also led to jokes caused by the useless movements of the players, such as the phrase "This isn't the time for that!"which would pop up when the player tried to use an item in the middle of a battle.

Since so many people were playing at once, it was almost impossible to get some of these simple actions to happen right away. Giving a nickname to a Pokémon brings up an on-screen keyboard with almost no input delay, which allows most of the commands coming in to be executed. Nicknames in Pokémon are a way to make players feel more affection for their party, but these bizarre nicknames are unexpected, and unintended. In short, they're really freaking weird.

Battles were a challenge for players early in the game. A good portion of these battles were spent trying to back out of the accidentally-activated item menu. This was also part of why people started drawing religious connotations to the helix fossil, as it was the first item selected when the menu was opened for the beginning of the game. This became more frustrating than weird, but it did contribute to the weirdness that the community would create later on.

Both nicknames and early-game battles are inconsequential things,

though. *Pokémon* is generally not a very punishing game, but there were quite a few problems that the community ran into, namely the Team Rocket Base and the Elite Four. Both of these spawned some of the greatest moments in *Twitch Plays Pokémon* history, such as Digrat's (Rattata's) horribly-timed dig out of the Team Rocket Base just before the community obtained the silph scope, a crucial item that the player needs to advance the game. (Gorilla Airplane, 2014) The Elite Four took quite a few tries to finally defeat, but during one of those runs, the party's weakest pokémon took out one of the strongest elite four members in the game. It's moments like these that the community lives for, and everyone is always striving to participate in and/or witness the next one.

However, we haven't even touched on the weirdest part, which is assuredly how the community has reacted to these events. There's plenty of different in-game events I can talk about, but the most unpredictable happening in *Twitch Plays Pokémon* is definitely the fan "religion"that has spawned. It started with a simple choice the player has to make near the start of the game. At a certain point inside a cave, a scientist offers you a choice between two fossils he found: The dome fossil or the helix fossil. The player chose the helix fossil, which the player attempted to use in many battles after it was obtained. Because of this choice, many people started praising the helix fossil, citing that because the player chose it, the community was progressing through the game and winning battles (see figure 3).

Figure 3. Left: Twitch Plays Pokémon's Party Upon Completing Pokémon Red, Dubbed "The Chosen Six". Right: Twitch Plays Pokémon Graffiti at University of British Columbia

This only escalated when the helix fossil was finally evolved into the pokémon Omanyte. Players referred to the pokémon as "Lord Helix"and many wanted him to be at the front of every battle to assure victory. A religious trapping was formed around this pokémon, and continued even after *Pokémon Red* had finished. Fans clamored around this and defended their "religion"throughout the duration of *Twitch Plays Pokémon*. The community was almost always excited, and it's every developer's dream to have fans so passionate about your project. This passion fueled the hype surrounding *Twitch Plays Pokémon* by connecting fans with each other in ways beyond just playing/watching the stream.

Lots of these trappings became the public image of *Twitch Plays Pokémon*. Articles mentioning the stream undoubtedly mentioned the players' infatuation with "Lord Helix"or "Bird Jesus"(see figure 3). As a result, those who didn't see the stream found these religious statements being posted along with news about this phenomenon. This doesn't just make *Twitch Plays Pokémon* seem a bit weird. It puts 'weird' right on the cover. During the later parts of Pokémon Red, it was more accurate to mention *Twitch Plays Pokémon* with these religious community aspects rather than exclude them. With over a million people contributing over the month, it's hard to think of the "weird"label as a bad one.

Conclusion: What can we learn?

Perhaps it was the popularity of the early *Pokémon* games that brought *Twitch Plays Pokémon* to the spotlight. Perhaps it was the nostalgia that our generation had for those games. Perhaps it was simply a fad that was out of control, and the waves it made in the videogame community will taper out. The thing is, we don't know. And we won't know. If there's one thing we learned, it's that we never know what games people want. There's never going to be a way to tell what games will get played, what games will get popular, and what type of games will become a trend. The best we can do is make educated guesses towards emergent ideas.

The first iteration of the stream lasted only 17 days before the players were able to defeat their in-game rival and beat the game. (Chase, 2014) It was a short-lived fire that flared up to extreme proportions very quickly, then died out even quicker. While it did continue to other *Pokémon* games after completion, they have not been nearly as successful as the first.

There's a lot of possible inspiration in the *Twitch Plays Pokémon* phenomenon. No current games utilize simultaneous input from thousands of players for one character. In theory, it's a poor mechanic. The chaotic entry of commands seemed unsurmountable and beating the game always felt like a pipe dream. How was the community able to do this?

Chaos is two things: disorganization and motivation. People enjoy the chaos. It's a great feeling when you see your username on the list of executed moves. It's a sign of power, a sign that you had the ability to command one character that thousands of others are trying to control at the same time. There's a constant battle between the server load and the players' inputs. Everyone wants to be able to say they 'played' *Twitch Plays Pokémon*, and a good amount of those players wanted to beat this game and prove that they could do the unthinkable by toppling one of the biggest challenges in the video gaming community.

The disorganization that was introduced with so many people in chat became nullified by the motivation it caused. Common goals started to form. Series of commands were laid out for the less experienced players to use. People created maps and objectives for the community to use and posted links repeatedly in the chat. Without time constraints, chaos only becomes unmanageable when people don't have the willpower to overcome it. The *Twitch Plays Pokémon* community had that willpower.

We may never see something like this again. Maybe the idea of "Twitch Plays"will just die out over time. On the other hand, we are seeing games being influenced by this (like *Choice Chamber*). We

might see new input systems being experimented with and the trend of MMO input appear in gaming. Regardless of which paths the fans take, we can remember what we had. *Twitch Plays Pokémon* was an amazing experience for everyone who witnessed it, and it proves to us that implementing weird ideas can become something valuable.

References

Wayling, A. (2014, Feburary 16). An Interview with the Creator of TwitchPlaysPokemon!. *BADatVIDEOGAMES.* Retrieved from http://badatvideogames.net/2014/02/16/an-interview-with-the-creator-of-twitchplayspokemon/

Davis, D. (2014, March 4). Unraveling the Cult of Twitch Plays Pokemon. *Kill Screen*. Retrieved from http://killscreendaily.com/articles/articles/feature/unraveling-cult-twitch-plays-pokemon/

Airplane. (2014, February 18). Twitch Plays PokÃ©mon VS. Giovanni (Full Battle + DigRat) HD. *YouTube*. Retrieved from https://www.youtube.com/watch?v=0QSFavoEU38

McWhertor, M. (2014, February 14). How Twitch is Crowd-sourcing an Amazing Pokemon Multiplayer Game. *Polygon*. Retrieved from http://www.polygon.com/2014/2/14/5411790/twitch-plays-pokemon-creator-interview-twitchplayspokemon

McWhertor, M. (2014, February 20). Twitch Plays Pokemon Creator Tweaks Channel to Make Game More 'Beatable'. *Polygon*. Retrieved from http://www.polygon.com/2014/2/20/5428398/twitch-plays-pokemon-democracy-vs-anarchy

Molinari, M. (2014, March 26). Choice Chamber. *Kickstarter*. Retrieved from https://www.kickstarter.com/projects/1451486150/choice-chamber

Neilan, S. (2014, February 20). Twitch Plays Pokemon – The Ledge. *YouTube*. Retrieved from https://www.youtube.com/watch?v=OkyhPYNwH14

Chase. (2014, March 1). TPP Victory! The Thundershock Heard Around the World – The Official Twitch Blog. *The Official Twitch Blog*. Retrieved from http://blog.twitch.tv/2014/03/twitch-prevails-at-pokemon/

Twilisailor. (2014, February 18). Twitch Plays Pokemon – Democracy VS Anarchy. *YouTube*. Retrieved from https://www.youtube.com/watch?v=zSq0VLgcjHQ

DiPietro, Matthew. (n.d.). Twitch 2013 Retrospective. *Twitch.tv*. Retrieved from http://www.twitch.tv/year/2013

notacatchyname. (2014, February 18). Twitch Plays Pokemon: Start9 Revolt. *YouTube*. Retrieved from https://www.youtube.com/watch?v=7z0nlvjt3BU