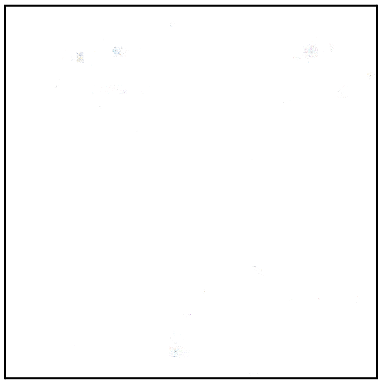
r/Place 2022 Most Dedicated Users Analysis

This analysis aims to examine the most dedicated users from the r/place 2022 to determine which users placed the most pixels and what communities were they from. Using data visualizations, image regeneration, and statistical analysis, I explored user behavior and community influences on pixel placement.

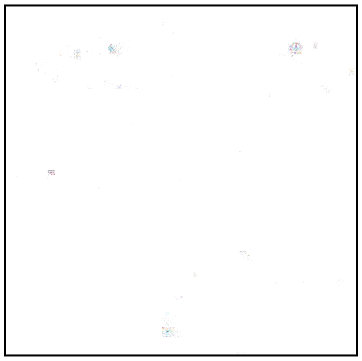
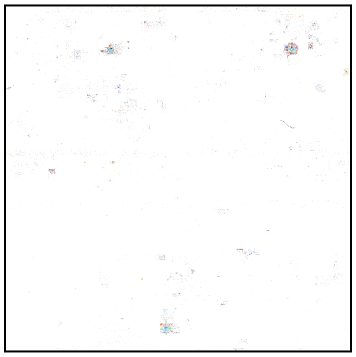
To begin my analysis, I first started out with finding the users that painted the most pixels which resulted in the following:

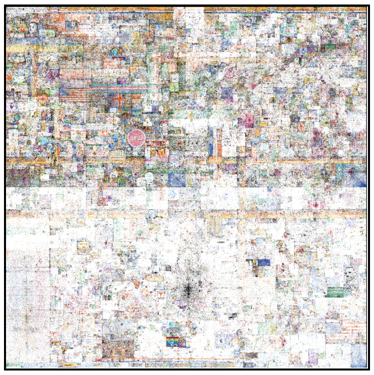
user_id_hashed	pixel_count
593296176866929730	795
18116963203247821516	781
8159728121586574741	777

Next, I reconstructed the canvas to observe the content these users contributed. Initially, focusing on the top users, I identified some patterns but couldn't fully decipher the images they were constructing. Initially, I focused on the top three users, and while I could identify some general patterns, it was difficult to determine the exact content they were constructing as you can see by the picture to the right. This led me to expand my analysis.

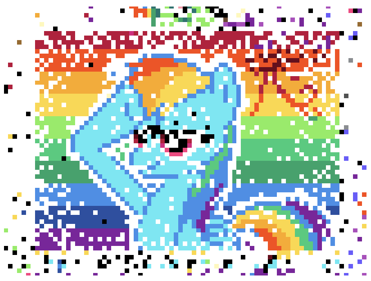


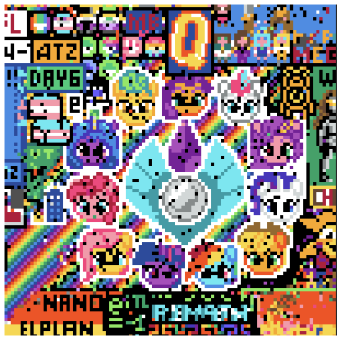
I began regenerating the canvas with increasing numbers of top contributors, starting with the top 3, then expanding to the top 10, 50, 100, and even 10,000 users. Through these iterations, it became evident that one consistent image stood out: a depiction of Rainbow Dash, a prominent character from My Little Pony: Friendship is Magic. It also became clear that this image was repeatedly attempted in various areas of the canvas and was continuously painted over by other users.



This led me to begin exploring the lore behind the placement of Rainbow Dash and why it was so contested over. After referring to the r/place2022 atlas, I found out that *Rainbow Dash* represents the element of loyalty and is shown saluting the future. This artwork was initially part of a collaborative project by the MLP fandom but faced constant interruptions. Over time, the character's image was erased by the Ukraine flag and raids from various Twitch streamers. Despite these challenges, the MLP fandom persisted, relocating and adapting their artwork until they were able to finalize it on the canvas as shown to the right.





Another major hotspot on the r/place canvas for the community was the Pegasus Crystal, which, according to the r/place2022 atlas, was created by r/mylittlepony and the Manechat Discord server. Initially located in one spot, the artwork was overtaken by Asmongold's Twitch community, who replaced it with a Guts portrait. In response, the community relocated eastward and reestablished the Pegasus Crystal. Furthermore, it became a symbol of unity and resilience, rallying the fandom and their surrounding allies to defend against the Twitch raids, including attacks from xQc's and Mizkif. Despite all the chaos and raids, the *Pegasus Crystal* and Rainbow Dash were displayed on the final canvas.

DuckDB vs. Pyspark

After reviewing both frameworks, I found that DuckDB outperforms PySpark in terms of runtime. In running both my analyses, the runtime for PySpark was significantly slower compared to DuckDB. After looking into the framework, I thought this was a bit unexpected since PySpark is often favored for large-scale distributed processing. Furthermore, I also find DuckDB's syntax much more appealing and user-friendly. This could also be because I just started using PySpark though and I am sure I could become more comfortable with it in time.

After researching both frameworks, I found that while PySpark excels in distributed processing and can handle massive datasets across clusters, DuckDB shines in scenarios where you need fast, in-memory processing on a single machine. PySpark's setup and performance might not be optimal for smaller-scale tasks, especially when working with individual data processing workflows.

Ultimately, I found DuckDB to be my preferred option for analyzing the r/place2022 dataset due to its speed and ease of use. However, PySpark seems to be the better choice for large-scale, distributed computing tasks that require parallel processing.