Turning Red: The Top Three Pixels Where Red Was Placed The Most

---

r/place is a recurring online art project hosted by Reddit where users from across the globe can come together and produce images on a canvas one pixel at a time. Upon examining the final canvas from the 2022 event, it became evident that red was one of the most dominant colors. This sparked curiosity about which pixels saw the most red placements throughout the challenge. After conducting a thorough analysis of the r/place 2022 dataset, we identified the top three most-red pixels (#FF4500) as follows: (859, 766) with 31,411 red placements, (860, 766) with 30,925 red placements, and (195, 489) with 10,303 red placements.

The first two coordinates are positioned adjacent to each other, forming two pixels in the tail of the “p” in the GameStop design. Our research revealed that this design (Figure 1)

Figure 1: GameStop Logo Proposal

was proposed by a user named u/ORaNGeTechPB within the Reddit community, r/Superstonk, a place for theoretical discussions about GameStop stock. The community

has over 1.1 million members and remains active today. The post showcasing the idea received over 900 upvotes. Once a design was created, another user helped increase its visibility (Figure 2) by sharing a link to it, https://halfdane.github.io/rplace/, in a post with 7.2k upvotes. This post also provided a guide on how to overlay the design (Figure 3) so it would be easier for others to replicate. Throughout the event, it seems that these two

Figure 2: Spreading Design Awareness

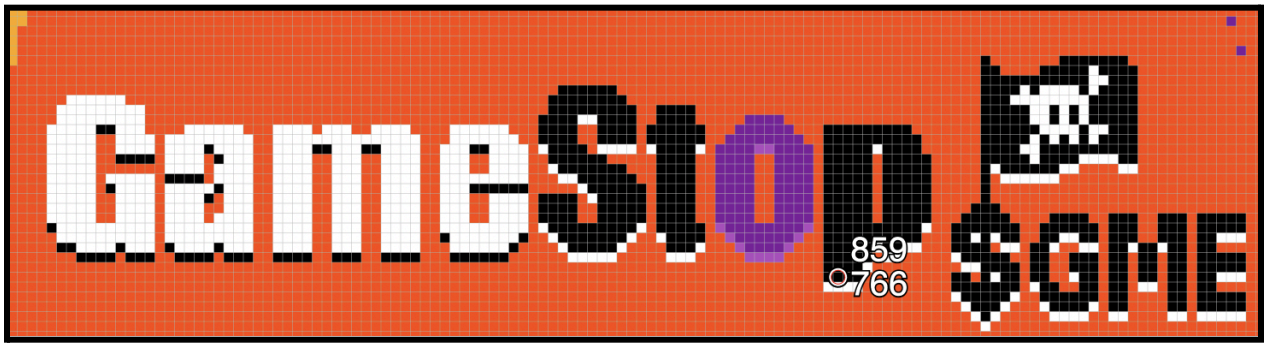

Figure 3: Design Overlay (Updated) Example

pixels were among the most placed in red, likely due to users being unsure of where the tail of the “p” should end (Figure 4 and Figure 5),. This ambiguity led to a design change as seen in a response from user u/Kyle772, who encouraged others to double check their pixel placements (Figure 6). The timelapse shown in the link, https://2022.place-atlas.stefanocoding.me/#/152/861/775/6.431 clearly illustrates how the

Figure 4: GameStop “p” Version 1

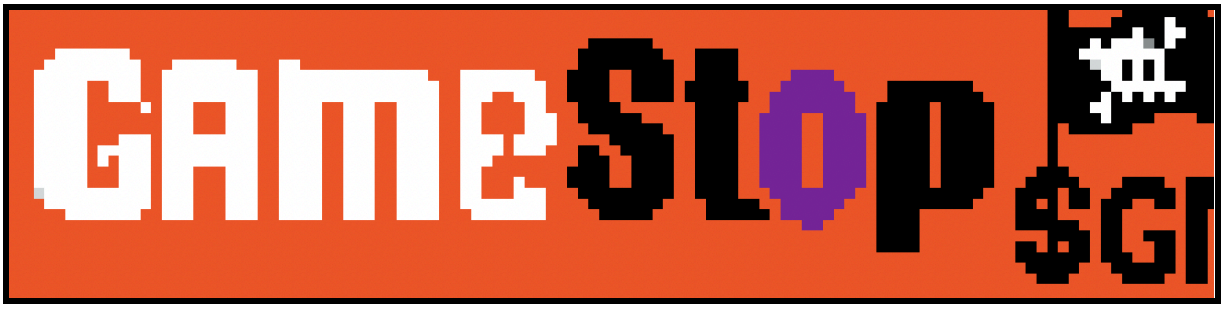

Figure 5: GameStop “p” Version 2

Figure 6: New Design Update Response

design evolved from its original form to one that involved shading (Figure 7). With the new changes, the uncertainty around the tail of the “p” was addressed, and users no longer placed their pixels where they thought the “p” should end. However, with all of the confusion prior to the change, it’s easy to see why so many red pixels were placed in order to change the length of the “p” due to confusion.

Figure 7: New Design With Shading

The last coordinate was in the design of the Canadian flag (Figure 8) and was located diagonally from the lower tip of the "C" in the word “CANADA”. After some investigation, we discovered that the Canadian flag was engaged in a unique battle with the "Banana" community, whose main objective was to replace the iconic maple leaf with a

Figure 8: Canada Flag Design

banana and change the word "CANADA" to "BANANAS." At one point, the Banana community succeeded in altering the flag, swapping out the familiar leaf for a banana and transforming "CANADA" into "BANANAS" (Figure 9). During this altercation, a significant portion of the effort was focused on modifying the word "Canada" to "Banana," leading to the creation of the term "BANADA" (Figure 10) for a period of time. The most frequently altered pixel was located at the starting point of the lower curve of the letter "B." The fight over pixel colors eventually became a meme sensation on Reddit (Figure 11). Given that the

Figure 9: “BANANA” Flag

“C” was the primary target for changing the word to "BANANA," it’s understandable why the pixel was colored red to take back control.

Figure 10: “BANADA” flag

Figure 11: Canada Flag to Banana Flag Meme

Contrasting PySpark and DuckDb

When completing this analysis, I observed that DuckDB not only provided a simpler solution but also outperformed PySpark in terms of execution time. Specifically, for each script, DuckDB executed in just 2.6429 seconds, while PySpark required 8.4921 seconds. In the case of larger datasets with more complex queries, I anticipate that the difference in execution time would only increase, with DuckDB likely offering a significant advantage in processing speed. In addition to its better execution time, the output generated by DuckDB (Figure 12) was notably more aesthetically pleasing and organized. The clean format of the results makes it much easier to read. In contrast, the output from PySpark (Figure 13) appeared less organized and took longer to read and understand which is the last thing you want.

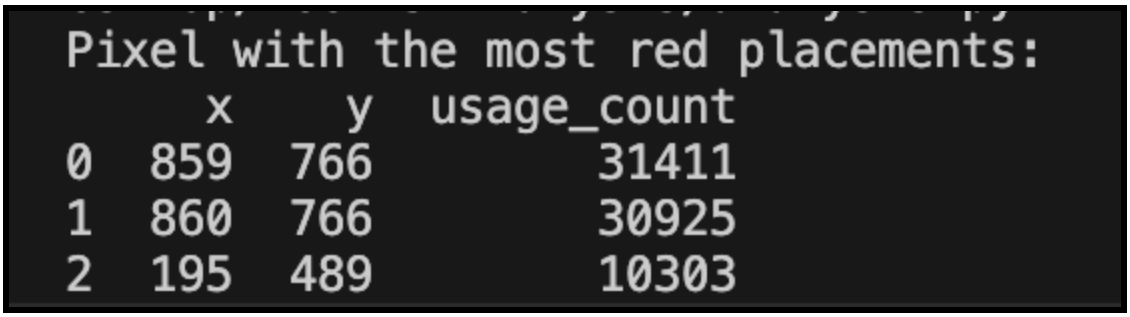

Figure 12: DuckDb Output

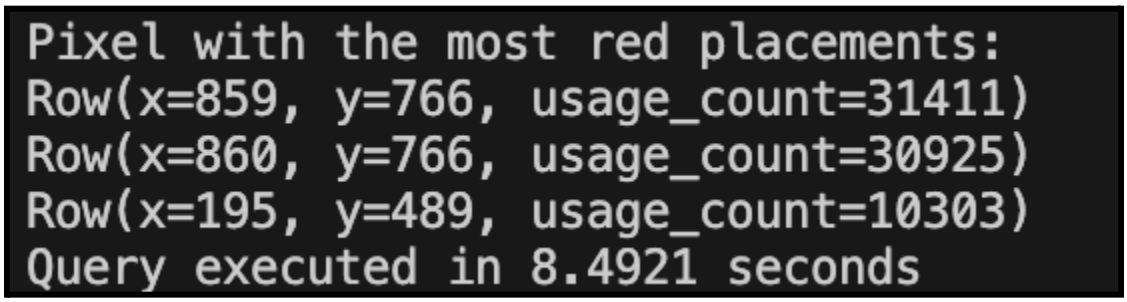

Figure 13: PySpark Output