Data gathering from a game

by Pepijn Muskens

A picture containing diagram

Description automatically generated

# Problem:

In the game Dead by Daylight players can start a match with a variety of Perks, Items and Addons to gain an advantage during the match. Some of these have more effect on the gameplay as others but so naturally some are more used as others. Some players want to know what the most used options are in the game, but the developers only release these statistics around once a year (usually also with just with data from the last 2 weeks). To get these statistics for myself I have used the DOT research framework.

# Library:

During the Library stage I started off by searching for a way to get the data I wanted from existing api’s. there is a steam api(steam is one of the platforms on witch the game is available) but this api only tracks the data of achievements and doesn’t contain any of the data I needed. My second idea was to try and get the data from the end screen after a match. In this screen everything a player takes with them to a match is shown as seen in figure 1. To get the data from the image I needed a way to recognize the icons and save them to a database. I read up on a few different techniques which looked promising.



Figure 1 Result Screen

**Template matching:**

The idea of template matching is to get a small a sample image of what you are looking for (the template) and shift it over the image you are trying to find it on. Every time you shift the template 1 pixel you look how much pixels it has in common with the image it is laying on top of and give it a percentage, move over 1 pixel and repeat the process. This method has a few downsides, if the image you are looking for has a different size on the whole image as your template, for example if the image your template is form was cut out of an image with a higher resolution, you won’t get a good match. Also, most of the templates we are looking for have a diamond shape and the background behind these diamonds is transparent and looks out on a scene which can change every match. This makes it harder to get a good score on these shapes.

**Image recognition:**

Image recognition works by training a program by giving it a lot of data with the correct answer. After it has been trained you can feed it a new image and it can quickly tell you what it thinks it is. This method is not able to find it on a larger image so it would still have to be combined with a way to find the images on the screen.

# Workshop:

I decided to try out template matching for a prototype because I am more familiar whit C# and there was an easy to implement library. In my first tests it worked great, but this was to find a template cut from the image I was searching on. When I started experimenting with different a different resolution the results became less certain. The scale of the image could differ by around 10% I found but this was just for the base image, if I tried to match it to an image with a different background the resolution mattered even more. To try and fix the resolution issue I decided to start of by searching for a part of the image with a template which does not change on the full image. The text at the top of the image “Score Status” has a nice solid black background. By trying the template in different scales, I could find the scale of the image compared to my templates with in the worst case a match of 97%. Once I had the scale, I resized the image and I could look for my templates.

With this solution I could find most of the square templates on the image. The next problem was the transparent background of the to try and solve this I tried to filter out only the white parts of the image all the templates have a white icon in the center so I could get these zoned out of the rest of the image. But to try and fix all the issues with this method was taking to much time but I expect I could have used new templates with only the white icon on a black background to get a good match on the new image.

Graphical user interface

Description automatically generated

Figure 2 White Filter

# Conclusion:

In the end I was able to get the data from the square icons from an image but to get all the different icons would have required to use different techniques for which I did not have the time to implement them all. If I would do this again, I would like to try the image recognition combined with a way to locate the position of the icons on the image to see if I can get better results for all of the icons.

# Links:

Template matching:  
<https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html>

Image recognition:  
<https://pythoncursus.nl/image-recognition-python/#image-recognition-python>

Template matching code:  
<https://github.com/PepijnMuskens/Templatematching-test>