Making program faster and efficient

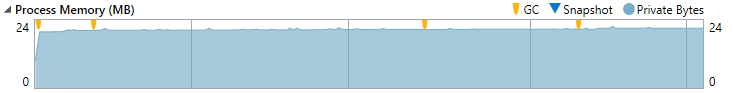
We are interested in doing analysis for computational requirements of the program.

Computational Requirements of the program:

When we first completed the implementations, we were amazed to see the ineffectiveness of the software. On our machine, it was taking 20% of CPU usage and 3GB of RAM usage. That is way too much. It was a stage which we even can’t think for releasing out. Even we won’t use such a software. So, solution was to be found. We started analyzing the program line by line. Seeing every line of code again. Checking how much resources each of the statements are taking. After doing a thorough analysis we found lots of memory leaks. We fixed them by explicitly destroying the objects that we declared. By doing such a simple step the memory usage was brought down to 600 megabytes. But CPU usage was still quite high. If found the reason for that was the fact the animations that we made were of high resolution. And the final canvas size that we were using was quite smaller. So, the frames were rescaled before displaying. Moreover, more number of frames means more amount time will be taken to process them. So, we looked for solutions.

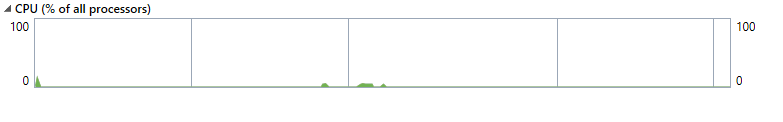
To solve this problem, we decided to start over. That is, to again produce the animation sprites. Doing this would be difficult. So, we go for other more reasonable choice. What we did was to write a new program which would transform all the frames to smaller frames along with giving the locations of the eyes. So, this program would traverse through all the frames one by one, scale them down, and then call the eye detection program. Once this much is done, program would output them to new resource folder. That way we removed the repeated processing.

With CPU utilization was impressively reduced to just 2% of total and RAM to 350 megabytes. We were happy with this much. But later we realized that even this too much for a program which will run for all the time in parallel with the all other programs. So, we decided to improve it further.



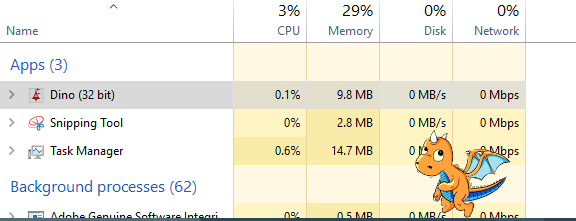
*Graph showing RAM usage. It can be seen that the program never crosses the 24 mb mark.*

Earlier what we were doing was to load all the resources into the RAM. We were thinking that it should improve performance. But that was not happening. The resource size on disk was just 25 megabytes but after loading them into the RAM it was 350 megabytes. It took us few days to understand the real issue. The thing is that the PNG file format, which were using for resources, is compressed file format. When images are stored in PNG file format they compressed down. But when they are loaded into the RAM, they would be decompressed. That is why the 25mb resources were converted into 350 mb RAM usage.



*Graph of CPU utilization. Green line shows the CPU used by our program.*

To solve this issue, we improved the implementations and instead of loading the files into the RAM, we would read them directly from disk whenever required. That way only partly resources would be decompressed hence lower RAM usage. So this happened. The final CPU usage on our machine is 0.1% on overage and RAM usage is 24 mb only.



*Snapshot of Windows Task Manager*