**Project Game Automation**

Documentation

Components Used:-

1. Live Log displayer.exe 🡪developed in c++(for efficiency)
2. ADB console
3. Emmulator
4. HayDay Game
5. opencv
6. Google’s Teachable Machine
7. Google’s Tensorflow api
8. Bash commands

Working Idea of Main Module

* Start live Log monitor
* Start Memu manually or by command
* Make a continuous loop
* Take screenshot 1920\*720 convert to 244\*244
* Feed to the model gather the info
* According to that make moves

Working Idea of executing touch commands

* For single tap , swipe we already have predefined commands that can be executed directly through adb command line
* For multiple touch input at same time we need to think:-

🡪In Case you want to zoom in or zoom out or do a sequence of execution 🡪

🡪the solution for this is using adb shell getevent command to get record the input in the touchscreen

🡪This will generate a sequence of event recorded and print in the screen

Then copy them

🡪using sendevent we can send event to the devices

🡪But sendevent requires decimal values but we have the values in hexadecimal format

🡪convert them to decimal and create a bash script

🡪In order to make them work

🡪push that file to that device /data/<FOLDERNAME>/

🡪install esFileExploler then enable root access

🡪Go to that directory and open that file properties and change to read ,write ,execute permission enabled [**this step is much much important**]

🡪Call that file in the android shell inorder to execute that bash script

Working Idea of live Log Monitor

* Script file, Log csv file
* Script file:

This file contains the information like which file to read and how much waiting time is to allot

* Log File ->

This is the main file updated continuously by Python ,updating the current state

In the form of csv file format

* This will continuously read that file in the interval of 500ms
* This binary file is working in the following way:

First it will read the file line by line

🡪 Extracts the information

🡪 Displays according to my wish

🡪Then sleeps for 500 ms

🡪again reads the file line by line from 0th to nth  line

🡪If any new line i.e, (n+1)th line is found it will executes the above operation again