```
Talk = {
    Topic: "Game Hacking",
    Who: "Kwan Li",
    Id: 101
}
```

```
Me = {
    Name: "Kwan Li",
    Degree: "Computer Science",
    Major: "Cyber Security",
    Projects: ["Trading Bot",
               "Discord Bots",
               "Game Hacks"]
}
```

Talk.Purpose()

- WHAT IS GAME HACKING?
- A TASTE OF REVERSE ENGINEERING
- BUILDING AN HACK TO MANIPULATE GAME MEMORY WITH WINDOWS API
- INTRODUCE AI HACKING
- ADVANTAGE OF THE NEW HACKING TECHNOLOGY

What is Game Hacking?

What is Game Hacking?

Player using various method to create an advantage beyond normal game play

etc. Vision through walls, in-game data modification….

Traditional Hacking

MEMORY READING AND WRITTING

Computer Memory

0x0	0x1	0x2	0x3
0x4	0x5	0x6	0x7
0x8	0x9	...	

Computer Memory

0x0	0x1	0x2	0x3
0x4	0x5	0x6	0x7
0x8	0x9	...	
	health = 100; 0x12345678		

Reverse Engineering

DEFINITION

UNDERSTANDING THE MEMORY REPRESENTATION IN TARGET PROCESS

IT IS A PROCESS OF MAKING ASSUMPTION
AND VALIDATING THAT ASSUMPTION

Reverse Engineering

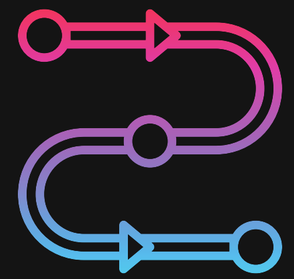

```
class Player {
    public: // player_object_address + offset_health = player_health_address
        int health; // offset: distance measure from give position - 0x02
        int steamId;
        string name;
        Vec3 location;
        .......
};


class Game {
    public:
        int gameMode; // 0 for deathmatch 1 for competitive ....
        Player* playerList; // [player1, player2, player3]
        .......
};
```

Reverse Engineering

COUNTER-STRIKE: GLOBAL OFFENSIVE DEMO TIME

Memory Hacking

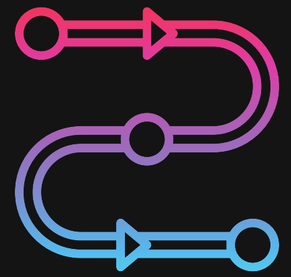
- Process Handle is a tunnel
- Read & Write
- Windows API (Open Process Handle)



Memory Hacking

CREATE A PROCESS HANDLE

```
HANDLE OpenProcess(
  [in] DWORD dwDesiredAccess,
  [in] BOOL  bInheritHandle,
  [in] DWORD dwProcessId
);
```



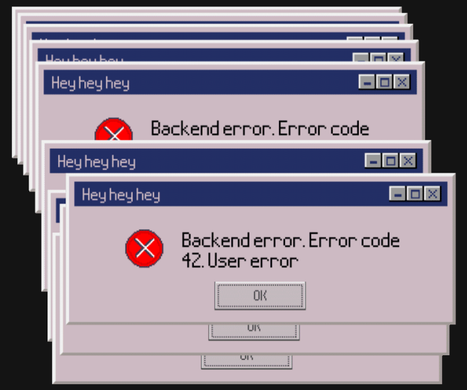
Memory Hacking

CREATE A PROCESS HANDLE

```
DWORD GetProcId(LPCSTR procName) {
    DWORD procId = 0;
    HANDLE hSnap = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
    if (hSnap != INVALID_HANDLE_VALUE) {
        PROCESSENTRY32 procEntry;
        procEntry.dwSize = sizeof(procEntry);

        if (Process32First(hSnap, &procEntry)) {
            do {
                if (!lstrcmpi(procEntry.szExeFile, procName)) {
                    procId = procEntry.th32ProcessID;
                    break;
                }
            } while (Process32Next(hSnap, &procEntry));
        }
```

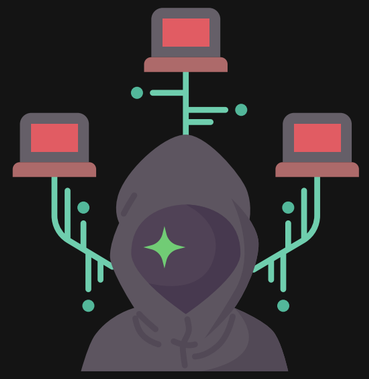
Live code anti-flash hack

COUNTER-STRIKE: GLOBAL OFFENSIVE



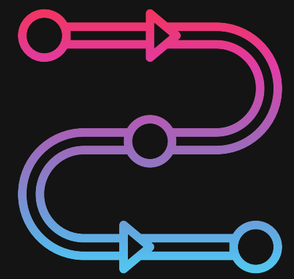
Memory Hacking

COUNTER-STRIKE: GLOBAL OFFENSIVE HACK DEMO TIME



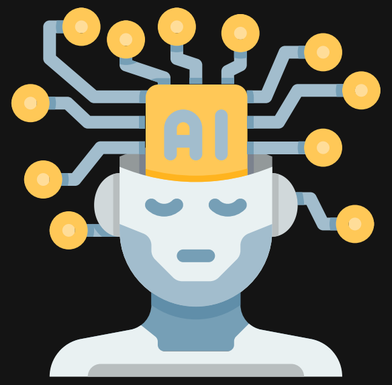
Memory Hacking

- Implement features by reading & writing at target memory
- Gain Advantage by illustrating information in ram
- Top Tier Anti-cheat protected game disable handle creation



AI Game Hack

BUILD AIMBOT WITH OBJECT DETECTION AI



Inconsistency

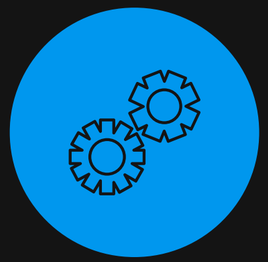
Player reaction time could be fluctuate dramatically

First Person Shooting Game Break down

Observe screen

Recognize and react to the enemy

Move cursor to the enemy



Fire when the crosshair on the enemy

Let's build a AI Hack

- OBJECT DETECTION ALGORITHM TO LOCATE THE ENEMY

- CROSSHAIR TO TARGET PATHWAY CALCULATION

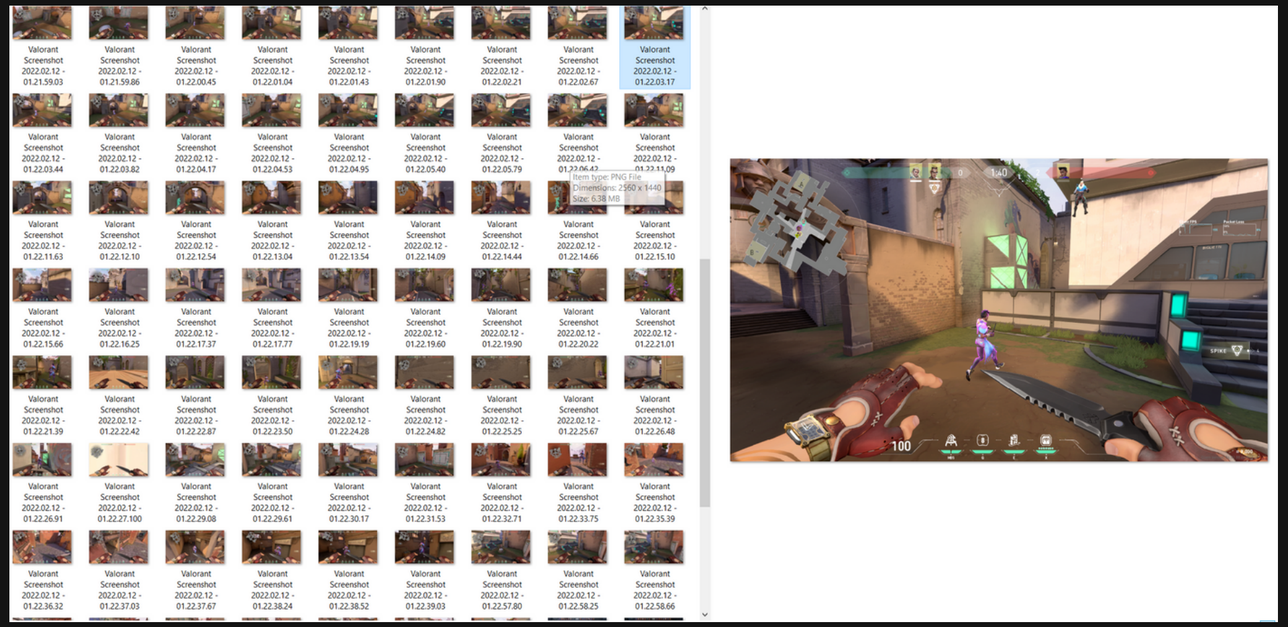
- EMULATE MOUSE INPUT

Let's build a AI Hack

OBJECT DETECTION ALGORITHM

TRAIN A MODEL THAT DETECT VALORANT PLAYER

Screenshots In-game player



Let's build a AI Hack

TRAIN A MODEL THAT DETECT VALORANT PLAYER

OBJECT DETECTION ALGORITHM

Label Images

Let's build a AI Hack

OBJECT DETECTION ALGORITHM

TRAIN A MODEL THAT DETECT VALORANT PLAYER

Model Training

```
!python train.py --img 414 --batch 16 --epochs 300 --data {dataset.location}/data.yaml --weights yolov5s.pt --cache --nosave --name valo
                 all        355        620      0.664      0.234      0.168       0.06

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     1/299     1.97G   0.06995   0.01957  0.009711         8       416: 100% 126/126 [00:18<00:00,  6.81it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 12/12 [00:02<00:00,  5.98it/s]
                  all        355        620       0.59      0.523      0.556       0.22

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     2/299     1.97G   0.06467   0.01713  0.007574         8       416: 100% 126/126 [00:18<00:00,  6.99it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 12/12 [00:02<00:00,  6.00it/s]
                  all        355        620      0.431      0.498      0.441      0.158

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     3/299     1.97G   0.05957   0.01724  0.007202        11       416: 100% 126/126 [00:18<00:00,  6.91it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 12/12 [00:02<00:00,  5.77it/s]
                  all        355        620      0.621       0.57      0.572      0.235

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     4/299     1.97G   0.05513   0.01662  0.006462        10       416: 100% 126/126 [00:18<00:00,  6.96it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 12/12 [00:02<00:00,  5.87it/s]
                  all        355        620       0.66      0.583      0.618      0.258

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     5/299     1.97G   0.05344   0.01567  0.005854         7       416: 100% 126/126 [00:18<00:00,  6.92it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 12/12 [00:02<00:00,  5.87it/s]
                  all        355        620      0.703      0.681      0.688      0.301

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     6/299     1.97G   0.05105   0.01591  0.005646        12       416: 100% 126/126 [00:18<00:00,  6.79it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 12/12 [00:02<00:00,  5.95it/s]
                  all        355        620      0.794      0.707      0.759      0.333

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     7/299     1.97G   0.05016   0.01617  0.005509         5       416: 100% 126/126 [00:18<00:00,  6.67it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 12/12 [00:02<00:00,  5.83it/s]
                  all        355        620      0.785       0.68      0.704      0.298

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     8/299     1.97G   0.04982   0.01599  0.005265         6       416: 100% 126/126 [00:18<00:00,  6.74it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 12/12 [00:02<00:00,  5.96it/s]
                  all        355        620      0.822      0.722      0.779      0.358

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     9/299     1.97G   0.04822   0.01528  0.005102         3       416: 100% 126/126 [00:18<00:00,  6.82it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 12/12 [00:02<00:00,  5.84it/s]
                  all        355        620      0.797      0.727      0.765      0.334

     Epoch   gpu_mem       box       obj       cls    labels  img_size
    10/299     1.97G   0.04824   0.01528  0.004928        11       416: 100% 126/126 [00:18<00:00,  6.97it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 12/12 [00:02<00:00,  5.98it/s]
                  all        355        620      0.777      0.682      0.747      0.349

     Epoch   gpu_mem       box       obj       cls    labels  img_size
    11/299     1.97G   0.04726   0.01547   0.00511         7       416: 100% 126/126 [00:18<00:00,  6.88it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 12/12 [00:02<00:00,  5.92it/s]
```

Train Custom Data · ultralytics/yolov5 Wiki

YOLOv5 □ in PyTorch > ONNX > CoreML > TFLite. Contribute to ultralytics/yolov5 development by creating an account on GitHub.

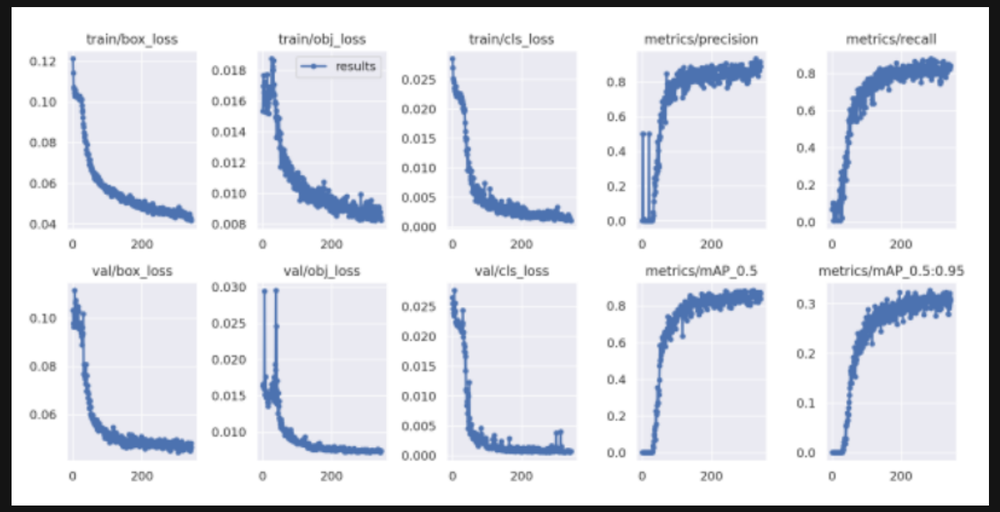
GitHub

Let's build a AI Hack

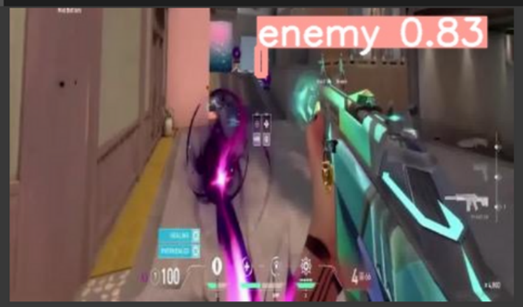
TRAIN A MODEL THAT DETECT VALORANT PLAYER

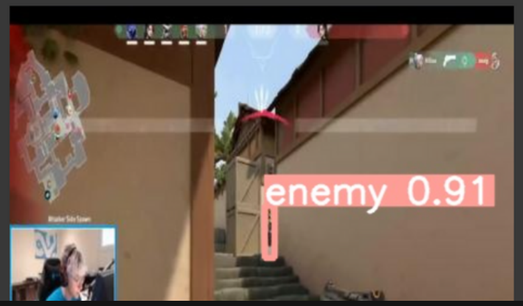
OBJECT DETECTION ALGORITHM

Model Training




```
#display inference on ALL test images

import glob
from IPython.display import Image, display

for imageName in glob.glob('/content/yolov5/runs/detect/exp/*.jpg
    display(Image(filename=imageName))
    print("\n")
```





Let's build a AI Hack

RUN OBJECT DETECTION

input { CONFIDENCE_THRESHOLD, MAX_OBJECT, gameScreenshot}

OUTPUT

```
0 = {dict: 7} {'xmin': 198.3563537598, 'ymin': 189.4714813232, 'xmax': 227.5848846436, 'ymax': 232.7379150391, 'confid
  'xmin' = {float} 198.3563537598
  'ymin' = {float} 189.4714813232
  'xmax' = {float} 227.5848846436
  'ymax' = {float} 232.7379150391
  'confidence' = {float} 0.8806430101
  'class' = {int} 0
  'name' = {str} 'enemyBody'
  __len__ = {int} 7
1 = {dict: 7} {'xmin': 206.322555542, 'ymin': 159.7570953369, 'xmax': 221.9651947021, 'ymax': 183.2277374268, 'confide
  'xmin' = {float} 206.322555542
  'ymin' = {float} 159.7570953369
  'xmax' = {float} 221.9651947021
  'ymax' = {float} 183.2277374268
  'confidence' = {float} 0.8289935589
  'class' = {int} 1
  'name' = {str} 'enemyHead'
  __len__ = {int} 7
__len__ = {int} 2
```

Building The AI Hack

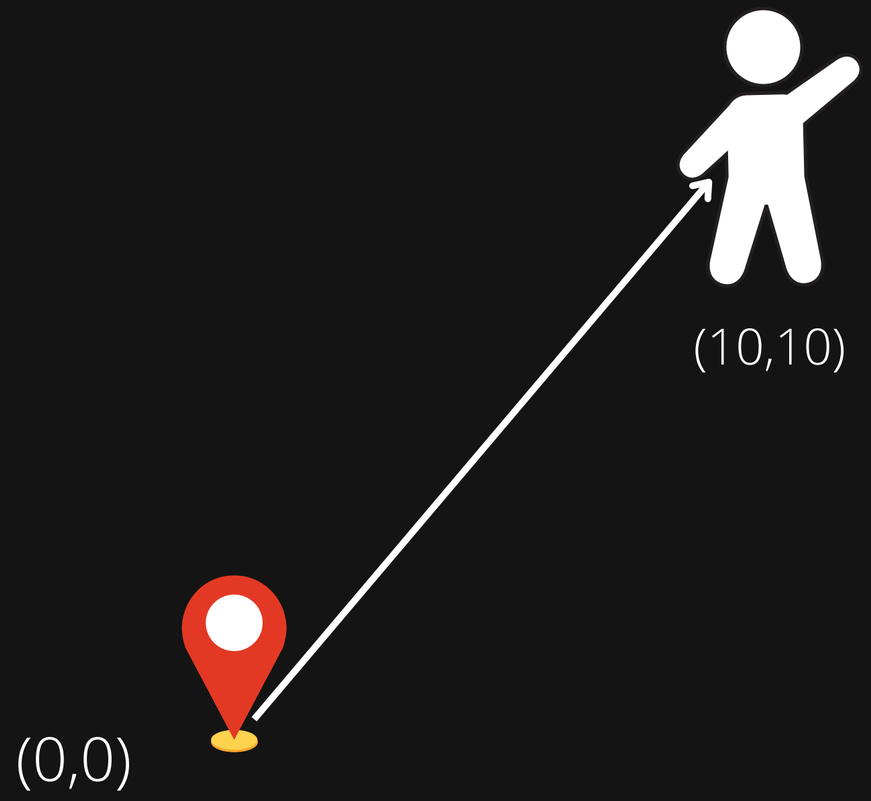
USE OBJECT DETECTION ALGORITHM TO LOCATE THE ENEMY

CALCULATE THE PATH FROM CROSSHAIR TO TARGET

EMULATE MOUSE INPUT

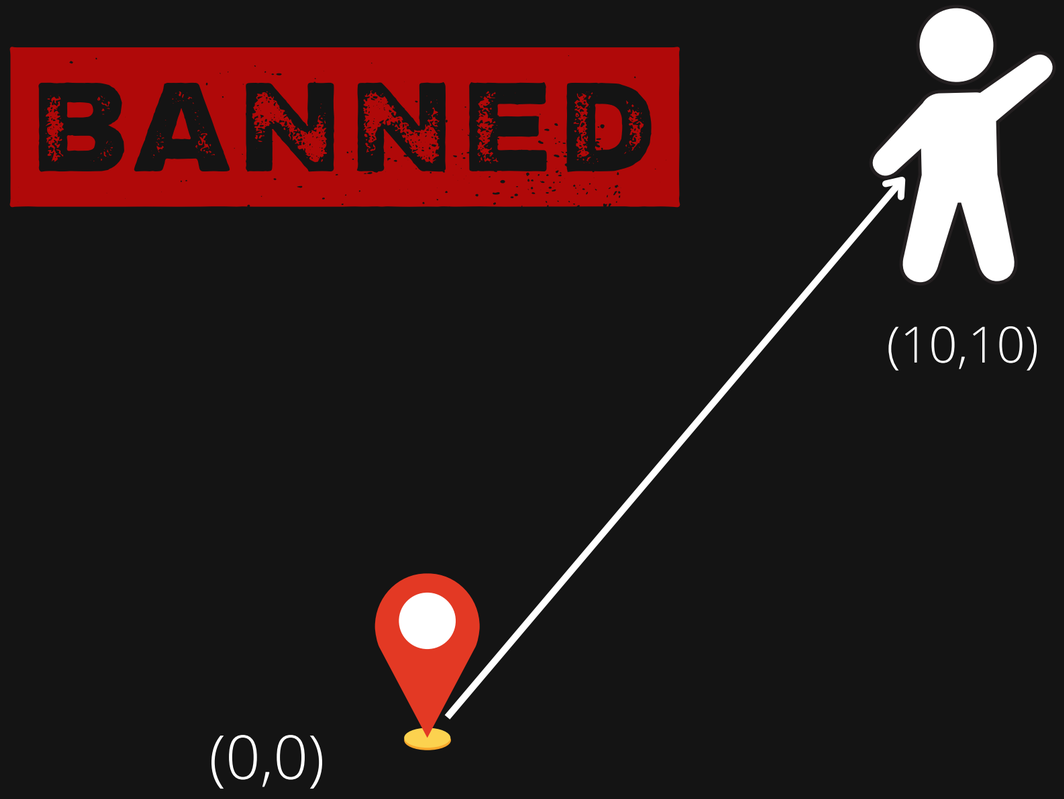
Let's build a AI Hack

Path Point A To Point B



Let's build a AI Hack

BANNED

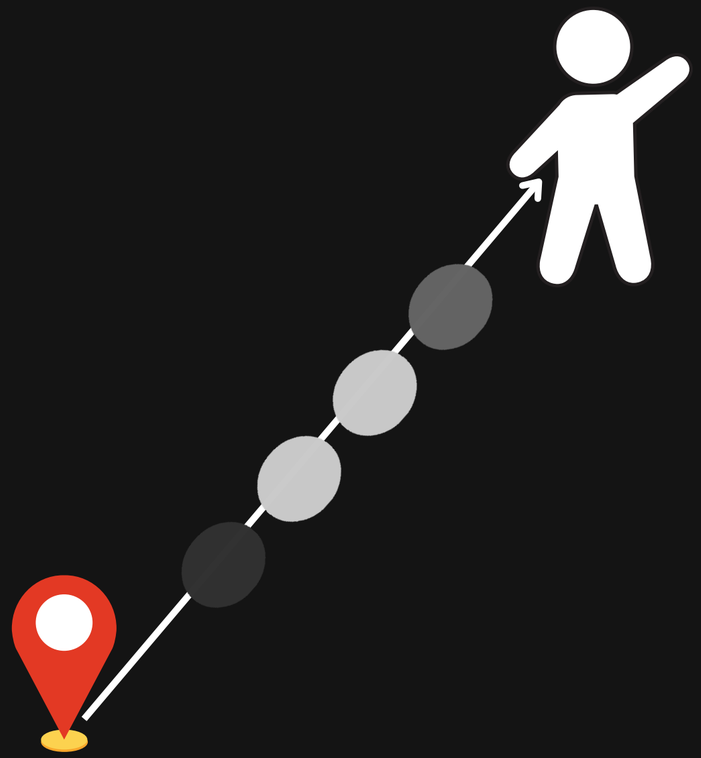


This company is using AI to track players' inputs and detect bots, boosters, cheaters and smurfs in games like Valorant, aims to go beyond traditional anti-cheat setups like Riot Vanguard

Posted on March 2, 2022 by Dom Sacco

Let's build a AI Hack

STOPS



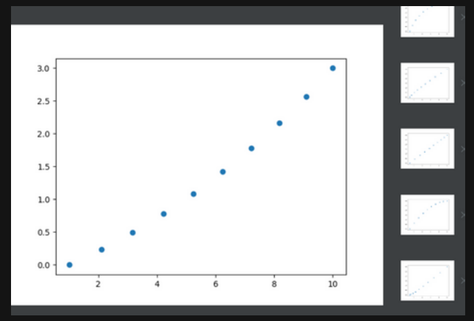
Let's build a AI Hack

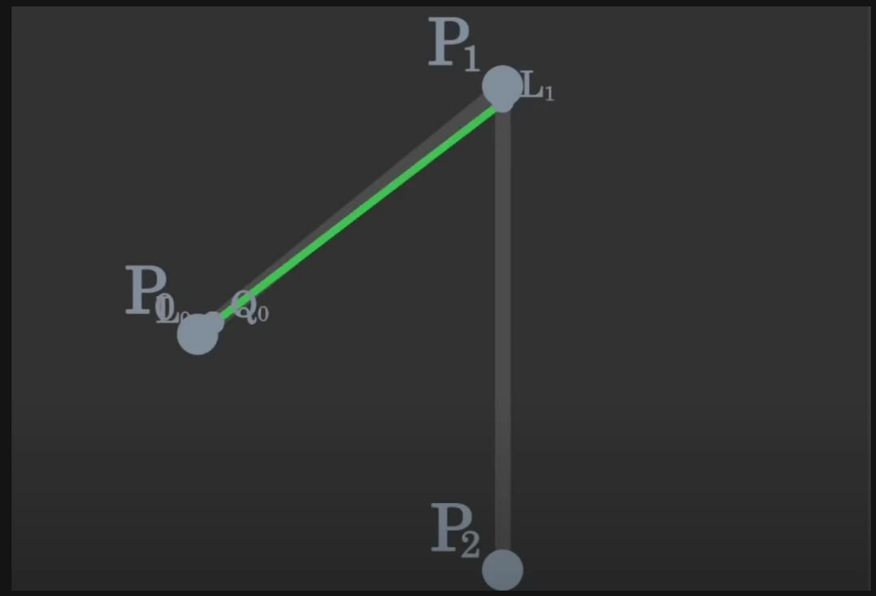
BEZIER CURVES

Quadratic

$$L_1(t) = (1 - t)P_1 + tP_2$$
$$Q_0(t) = (1 - t)L_0(t) + tL_1(t)$$
$$Q_0(t) = (1 - t)^2P_0 + 2(1 - t)tP_1 + t^2P_2$$

```
def create_path(ori, dest, stop):
```





https://www.youtube.com/watch?v=pnYccz1Ha34
Guidev

Let's build a AI Hack

EMULATE MOUSE INPUT

Win32 API?

SetCursorPos function (winuser.h)

Article • 10/13/2021 • 2 minutes to read

Moves the cursor to the specified screen coordinates. If the new coordinates are not within the screen rectangle set by the most recent ClipCursor function call, the system automatically adjusts the coordinates so that the cursor stays within the rectangle.

Syntax

C++ Copy

```
BOOL SetCursorPos(
  [in] int X,
  [in] int Y
);
```

Let's build a AI Hack

EMULATE MOUSE INPUT

Win32 API?

SetCursorPos function (winuser.h)

Article • 10/13/2021 • 2 minutes to read

Moves the cursor to the specified screen coordinates. If the new coordinates are not within the screen rectangle set by the most recent ClipCursor function call, the system automatically adjusts the coordinates so that the cursor stays within the rectangle.

Syntax

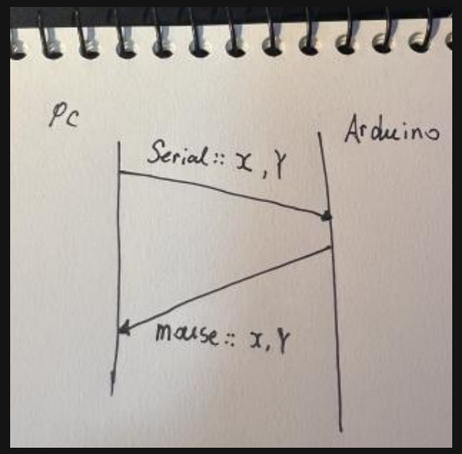
C++ Copy

```
BOOL SetCursorPos(
  [in] int X,
  [in] int Y
);
```

Let's build a AI Hack

EMULATE MOUSE INPUT

Arduino Leonardo




```
arduino = serial.Serial(SERIAL_PORT, 115200, timeout=0)
```

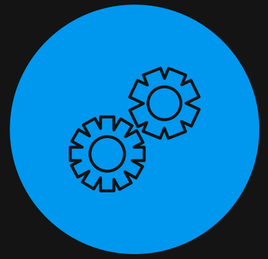
```
def move_cursor(arduino, x, y):
    data = str(x) + ':' + str(y)
    arduino.write(data.encode())
    arduino.flush()
```

VALORANT DEMO TIME

Observe screen

Recognize and react to the enemy

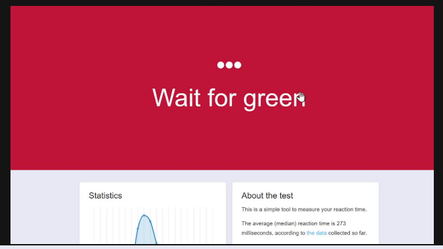
Move cursor to the enemy

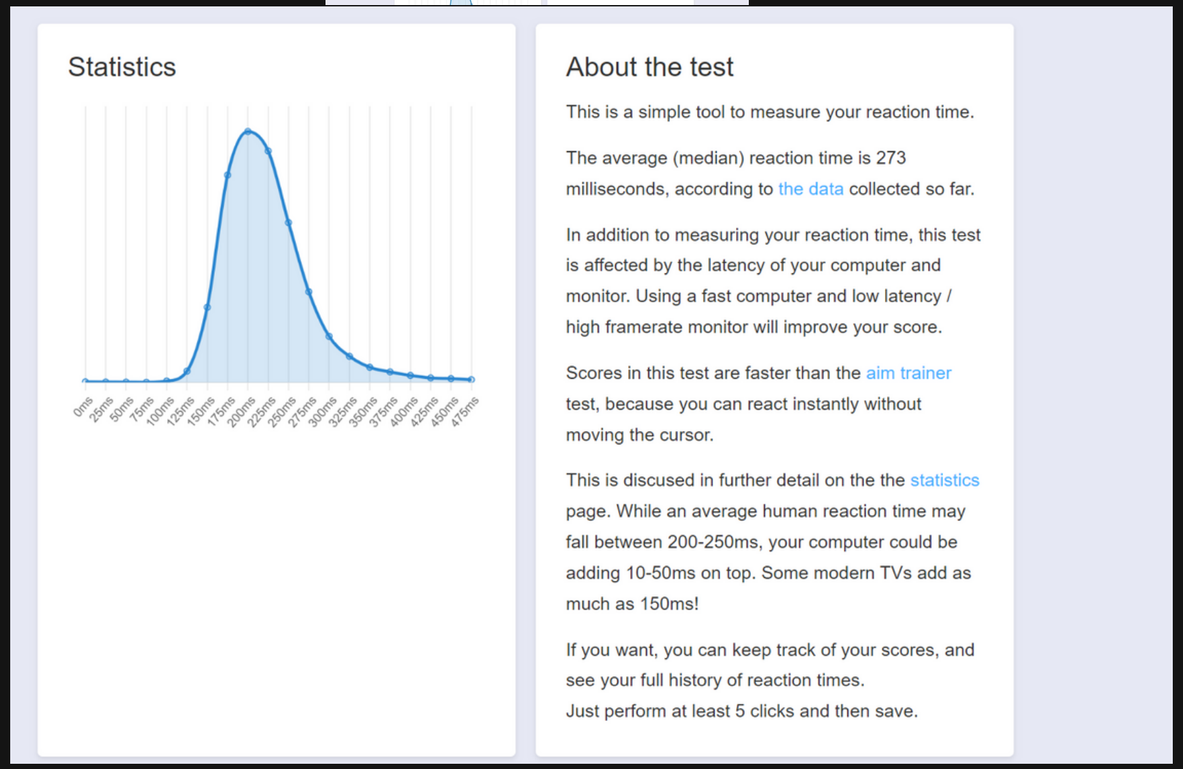


Fire when the crosshair on the enemy

Human Player Average reaction time

273ms





https://humanbenchmark.com/tests/reactiontime

Observe screen

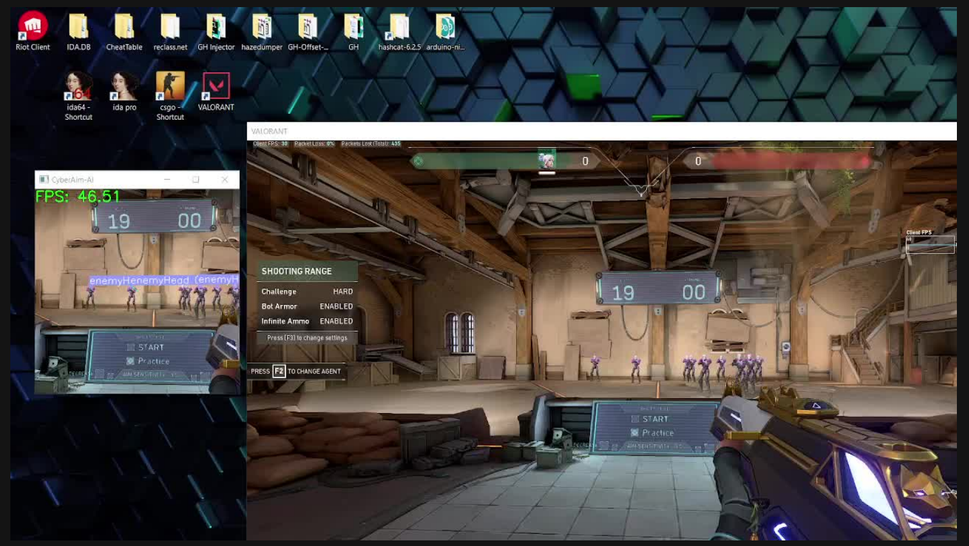
Recognize and react to the enemy

Move cursor to the enemy

Fire when the crosshair on the enemy

Time took aimbot to move crosshair on enemy head

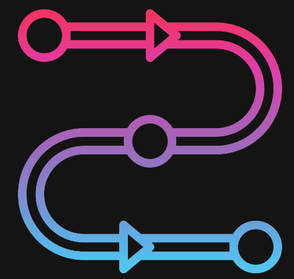
60ms



https://humanbenchmark.com/tests/reactiontime

AI GAME HACK

- Anticheat Safe (Not touching game memory)
- Limited functionality (Aimbot, Auto Trigger)



Kwan Li

LINKEDIN

https://www.linkedin.com/in/kwan-li-a517671a6/

EMAIL

lihungkwan72@gmail.com