**DESCRIPTION OF WORK**

**for**

**ISE 309**

**MULTIMEDIA SYSTEMS**

**COURSE PROJECT**

**Target Detection in FPS (First person shooter) games**

**Sarp Özdemir 150180207**

29/03/2021

**Table of Contents**

[1 EXECUTIVE SUMMARY 3](#_Toc350329061)

[2 INTRODUCTION 3](#_Toc350329062)

[3 PROJECT DESCRIPTION 3](#_Toc350329063)

[3.1 Goals of Project 3](#_Toc350329064)

[3.2 Impact of Solution 3](#_Toc350329065)

[3.3 SOTA 3](#_Toc350329066)

[3.3.1 Novel contributions 3](#_Toc350329067)

[3.4 Risk Assessment 3](#_Toc350329068)

[4 PROJECT SCOPE 4](#_Toc350329069)

[4.1 Work Breakdown Structure (WBS) 4](#_Toc350329070)

[4.2 Work Packages 4](#_Toc350329071)

[4.3 Out of Scope 6](#_Toc350329072)

[5 ASSUMPTIONS 6](#_Toc350329073)

[6 MILESTONES and DELIVERABLES 7](#_Toc350329074)

[6.1 Deliverables and Milestone Tables 7](#_Toc350329075)

[6.2 Project Schedule (Gantt Chart) 7](#_Toc350329076)

[7 7](#_Toc350329077)

# EXECUTIVE SUMMARY

In this project I aim to create an object detection tool which can detect target objects in first person shooter games. I aim to make this tool as lightweight as possible which will allow it to run in real time.

# INTRODUCTION

I have been playing first person shooter games for long time and I have observed usage of cheats by several people. A surface research on the tools used for those cheats will show us that most of these cheats depend on reading from the virtual space of the game using tools such as the ‘ReadProcessMemory’ function from Microsofts’s memoryapi library. These cheats are effective and are real time yet they can be caught very easily by tools such as Valve Anti Cheat (VAC). Despite not being interested in cheating in online games the idea of an “uncatchable” cheat seems interesting to me.

# PROJECT DESCRIPTION

## Goals of Project

This project aims to provide a target detection tool which can work in real time with minimal overhead and maximum true positive rate. I aim it to be able to capture the screen and analyze it 24 times per second.

## Impact of Solution

This solution will create and undetectable cheat for certain games if it comes out successfully. In case the program isn’t working as fast as I expected then it still has the potential of creating datasets for deep learning problems.

## SOTA

State of the art solutions for this problem seem to be either resorting to previously mentioned methods of hooking the game processes memory. Or they aim to fix the aim of the player by capturing the target inside a very small radius around the players crosshair (the cross in the middle which the player shoots at by left click).

### Novel contributions

I aim to beat the speeds of previous solutions and create a solution which can do this effectively for fullscreen. I also aim to use only the input taken from the screen to achieve this.

## Risk Assessment

The main risk related to this project is the possibility of someone using this in real life and not getting caught in the game they are using it which may lead to unfair advantages during events with monetary prize pools.

# PROJECT SCOPE

## Work Breakdown Structure (WBS)

Figure 1 Work Breakdown Structure (WBS)

## Work Packages

|  |  |  |  |
| --- | --- | --- | --- |
| **WP 1** | **Obtaining screenshots** | | |
| Start Date | 29/03/2021 | End Date | 21/04/2021 |
| **Objectives: Write the code required for obtaining the screenshot of the screen and prepare it to be in a useable state.** | | | |
| **Tasks**   * *Obtaining the screenshot by use of windows API’s* * *Finding a way to move it into memory of the main program* * *Parse the photo to be manipulation-ready* | | | |
| **Deliverables and Milestones:**  MS1: Obtain a version of a screenshot in programs memory.  D1.1: Obtain a version of screenshot in RGB format in an array between ranges 0-255 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **WP 2** | **Gausian Blur** | | |
| Start Date | 21/04/2021 | End Date | 20/05/2021 |
| **Objectives: Write the code required to blur the screenshot obtained from.** | | | |
| **Tasks**   * *Implement function to apply masks to pictures.* * *Implement gausian function in 2d.* * *Implement gausian blur mask.* | | | |
| **Deliverables and Milestones:**  D2.1: 2d Gausian function.  D2.2: Gausian blur mask. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **WP 3** | **Excess Pixel Removal** | | |
| Start Date | 20/05/2021 | End Date | 05/06/2021 |
| **Objectives: Downscale the picture by a factor of two.** | | | |
| **Tasks**   * *Implement downscaling alghoritm* | | | |
| **Deliverables and Milestones:**  D3.1: Downscaled picture | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **WP 4** | **Object Detection Functionality** | | |
| Start Date | 05/06/2021 | End Date | 25/06/2021 |
| **Objectives: Write the code required to detect objects.** | | | |
| **Tasks**   * *Chose an object detection framework to use for object detection.* * *Implement the framework.* * *Test the framework* | | | |
| **Deliverables and Milestones:**  MS2: Screenshots with objects marked.  D4.1 Screenshots with objects marked “correctly” | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **WP 5** | **Mouse Movement Functionality** | | |
| Start Date | 25/06/2021 | End Date | 01/07/2021 |
| **Objectives: Write the code required to make mouse movements according to input images taken.** | | | |
| **Tasks**   * *Write code which extracts co-ordinates from the picture.* * *Write geometric code to compute how much the mouse should move.* * *Write code which sends computer signal to move the mouse.* | | | |
| **Deliverables and Milestones:**  MS3: Obtain coordinates from marked images.  D5.1 Program able to correctly calculate crosshair movement required to get on target | | | |

## Out of Scope

The following are considered OUT OF SCOPE for this contract:

Producing a generalized solution which can work in plug and play fashion for multiple games is not in the scope of this project. An other thing that is not in the scope of this project is creating a user friendly ui for the user as the purpose of this project is not to create a “cheat” but to test an idea.

# ASSUMPTIONS

This project depends heavily on the hardware constraints of the system for the “real-time” calculation goal as that is a hard task. This project also assumes that the task at hand could be done in 4 months.

# MILESTONES and DELIVERABLES

## Deliverables and Milestone Tables

Example:

|  |  |  |
| --- | --- | --- |
| **Deliverable (D)** | **Description** | **Date** |
| D1.1 | Obtain a version of screenshot in RGB format in an array between ranges 0-255 | 21/04/2021 |
| D2.1 | 2d Gausian function. | 10/05/2021 |
| D2.2 | Gausian blur mask. | 20/05/2021 |
| D3.1 | Downscaled picture | 05/06/2021 |
| D4.1 | 1 Screenshots with objects marked “correctly” | 25/06/2021 |
| D5.1 | Program able to correctly calculate crosshair movement required to get on target | 01/07/2021 |

Table 2 Deliverable Table

|  |  |  |
| --- | --- | --- |
| **Milestone (MS)** | **Date** | **Deliverables** |
| MS1 | 15/0/2021 | Obtain a version of a screenshot in programs memory |
| MS2 | 20/06/2021 | Screenshots with objects marked. |
| MS3 | 28/06/2021 | Obtain coordinates from marked images. |

## Project Schedule (Gantt Chart)

Example:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8 | M7 | M8 | M9 | M10 | M11 | M12 |
| WP 1 |  |  |  | D  1.1 |  |  |  |  |  |  |  |  |  |  |
| WP 2 |  |  |  | D  2.1 | D  2.2 |  |  |  |  |  |  |  |  |  |
| WP  3 |  |  |  |  |  | D  3.1 |  |  |  |  |  |  |  |  |
| WP  4 |  |  |  |  |  | D  4.1 |  |  |  |  |  |  |  |  |
| WP  5 |  |  |  |  |  |  | D  5.1 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| WP 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| WP 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| WP 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Figure 1 Gantt chart

# 