

***Bilkent University***

***Department of Computer Engineering***

**Senior Design Project**

**Project Analysis Report**

**Project Name:** Espionage

**Project Group Members:**

Özgür Öney 21101821

Ahmet Safa Kayhan 21001532

Selçuk Gülcan 21101231

Ateş Balcı 21202771

Fırat Özbay 21201683

**Supervisor:** Can Alkan

**Jury Members:** Öznur Taştan, Uğur Güdükbay

**Project Website:** espionage-game.github.io

# Introduction

Video games that get involved our lives in pre 60’s, from these day to nowadays confront us as not only a basic show business element but also software that is used by people from every age and worth millions and billions worth market share. Such software that plough through software world with the help of devices called game consoles at the very beginning became very popular just after personal computers proliferation such that there is at least one PC in every single house.

Espionage, basically is an two dimensional (2D) stealth-arcade game which has character left into a map to go from one point to another with the help of supplied inventory and where users directs this character. Map, that player will follow to finish the particular level is not constant structure of course; it will be designed as layered and will be looking like a lateral section of a building. In addition to its lateral section structure, it will contain elements that have different characteristics; such as boxes will be set down to hide behind them, stairs will be there to elevate between floors/layers, vent holes to pass through enemies without getting attention etc. Needless to say that, number of elements will differ from level to level with the help of machine learning algorithm described below to change difficulty.

On player’s way to the end, computer-controlled units that have own adaptive intelligence attempts to fail him whereas player attempts to complete the game by reacting with reflexive and strategic moves. As character is controlled by player, enemy units that aim to fail player will be controlled by artificial intelligence algorithm, which means that they will also behave in natural ways to give reaction to player’s character. For example, whenever character is seen by an enemy unit, such enemy unit will alert all other enemy as well as change their constant formation (such as patrol route or condition of being armed etc.). In addition to that, a machine learning algorithm will detect the player’s style by gathering data like time spent, number of tools used or number of enemies neutralized. Such processed data will be used to determine the level of difficulty of course, hereby play a critical role on playability and enjoyment.

Most importantly, a genetic algorithm will be implemented for the game. As the name suggest, a spectrum basic artificial intelligences for every enemy unit will be implemented first and game will be opened to many players to play. As time goes and players play the game to finish, just enemy units that have adequate intelligence will be survived, while others were simply eliminated. To do this, of course a score mechanism for enemy units to detect ones those are able to survive more. At last, a combination of surviving ones will be created to construct a sound basis artificial intelligence.

## Object Design Trade-Offs

### High quality graphics vs. Performance

Espionagé game aims to satisfy the potential customer-gamers during gameplay. To do such, graphical content -including character that player control, non-playable enemy units and even objects to interact- of the game should be drawn in high resolution. To satisfy this requirement, a professional help from a graphic designer has taken. However, GPU of a single computer is directly responsible from drawing of elements to the screen and due to high load on drawing, performance problem can occur. Hereby, balance of such trade-off should be achieved through a wise policy to sacrifice neither performance nor quality.

### Cost vs. Scalability

Espionage game keep the data of the users in its community such that it records every single player’s level passing time, number of gadgets used and enemies eliminated to optimize the difficulty level of the game. As number of players and number of levels played increases, machine learning algorithm’s approximation for difficulty approaches to optimal value. However, keeping track of such attributes simply requires a working database on the background. As pre-defined numbers increase, since expenses are directly proportional, they are also increases. To cope with such trade-off, optimal number of players should be determined.

### Complexity vs. Playability/Usability

Espionage game includes much more traits compared to a basic adventure game: enemy dynamics that uniquely affects level difficulty, gadgets that changes playability of the game from head to toe as well as environmental factors that stands here to determine level structure. As combination of these values are getting complex, playability of the game is dramatically decreases for the player because of the fact that people tend to understand basic combinations much more than complex ones. Hereby, as a trade-off, elements defined above should be spread over the game so that they do not make game difficult to understand and easy-to-play.

## 1.3 Engineering Standards

**UML:** Unified Modeling Language is a standard that is intended to provide a way to visualize the design of a system.

**OpenGL:** Basically a standard maintained by the OpenGL Architecture Review Board (ARB) for rendering [2D](https://en.wikipedia.org/wiki/2D_computer_graphics) and [3D](https://en.wikipedia.org/wiki/3D_computer_graphics) [vector graphics](https://en.wikipedia.org/wiki/Vector_graphics).

**IEEE Xplore Standard:** Naming conventions from web service to machine learning algorithm is covered by this standard.

## 1.4 Definitions, Acronyms and Abbreviations

**UML**: Unified Modeling Language.

**SQL**: Structured Query Language.

**DB**: Database

**AI:** Artificial Intelligence

**GPU**: Graphical Processing Unit, a part of computer which is responsible for rendering graphical objects.

**Python**: Object-oriented extensive programming language

**ML**: Machine Learning GUI: Graphical User Interface

**2D:** Two dimensional. Designed in a way such that its only possible to see width and length of a single object.

# **2.0 Packages**

## 2.1 Artificial Intelligence (AI) package

Package that is responsible for every single action of non-controllable enemies.

Explanation of classes in this package is given below.

**Node:** Grids defined in each frame of a level which is used in calculations of possible paths that non-playable enemy unit can follow.

**Graph:** Class that manages interactions contain nodes, such as choosing a route from current location to destination or selecting a place to hide.

**ReactionAI:** Class that keeps algorithms responsible for reactions of non-playable characters.

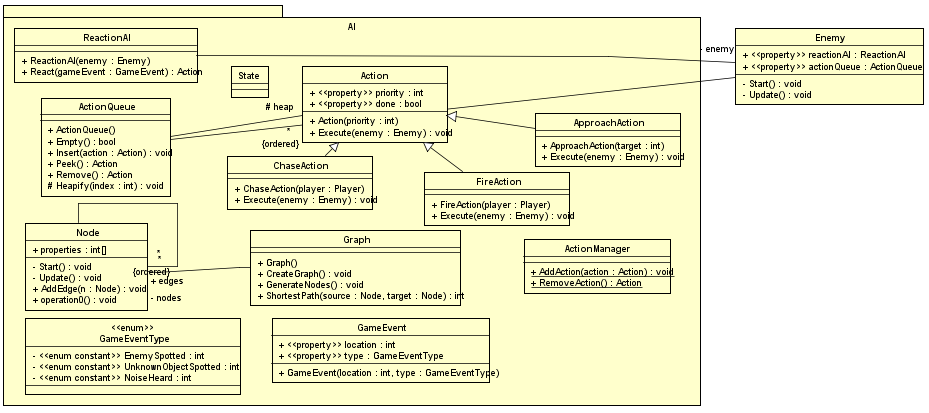
**State:** Non-playable characters have their unique states depending on the player’s interaction with them, such as one enemy should be in alert state whenever our player-controlled character is seen by him. This class herein is responsible for hosting algorithms that determines states of non-playable characters.

**ApproachAction:** Class responsible for guardians’/enemies actions whenever they meet our player controlled character such as going towards source of a unidentified sound.

**ChaseAction:** Class responsible for guardians/enemies whenever they meet a player who is escaping from them.

**FireAction:** Class responsible for guardians/enemies whenever they are forced to use their weapons. In a scenario when player escapes from shoots, non-playable units will be managed in a way that they will shoot in the direction where player escapes to.

**ActionManager:** Class that contains a priority queue for the events so that event mechanism is arranged in a logical order. For example, whenever a series of events occur, their respective drawings and triggers should be materialized in a chronological order.



*Figure 1: Detailed component diagram of AI package of the game.*

## 2.2 Machine Learning (ML) package

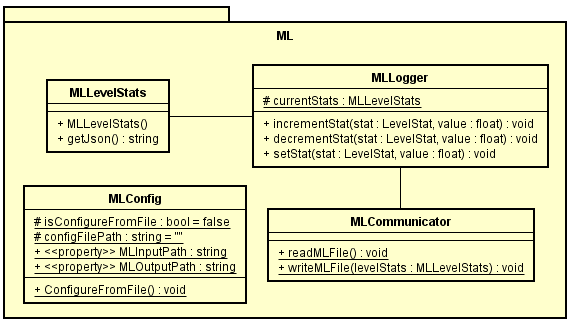
This package contains classes that are responsible for every single Machine learning action. Explanation of classes in this package is given below

**MLCommunicator:** Class required to communicate with machine learning module.

**MLConfig:** Class that contains basic configuration information needed to communicate with ML module.

**MLLevelStats:** Class keeping data of level statistics like time spent, number of enemies eliminated and number of gadgets used during a single level.

**MLLogger:** Class that keeps the log information which will be later used to update MLLevelStats’ information.



*Figure 2: Detailed component diagram of ML package of the game.*

## 2.3 Enemy package

Enemy package can be seen as a bridge between AI package and game itself. Decisions taken by AI package are implemented/drawn/applied through Enemy package. Therefore, respective classes to connect AI & game each other will be implemented for each AI package class.

**IApproachable:** Interface responsible for executing commands taken from PathingAI class defined above.

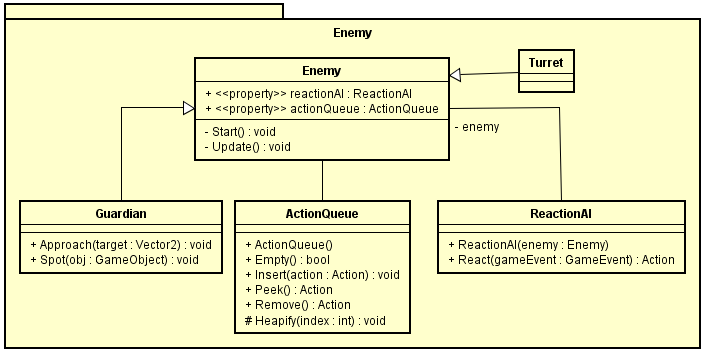
**IFirable:** Interface responsible for executing commands taken from ReactionAI class defined above.

**ISpotable:** Interface responsible for executing commands taken from ChaseAction & FireAction classes defined above.

**Enemy:** Class responsible from initializing & updating enemy attributes.

**Turret:** Subclass responsible for actions of turrets, which are unshifted enemies that attacks whenever it sees player controlled unit.

**Guardian:** Subclass responsible for actions of shifting enemies that not only attacks whenever they see player controlled unit but also chase him down and warn any other guardian.



*Figure 3: Detailed component diagram of Enemy package of the game.*

## 2.4 Gameplay Package

Package that contains necessary classes for precise player-game interaction.

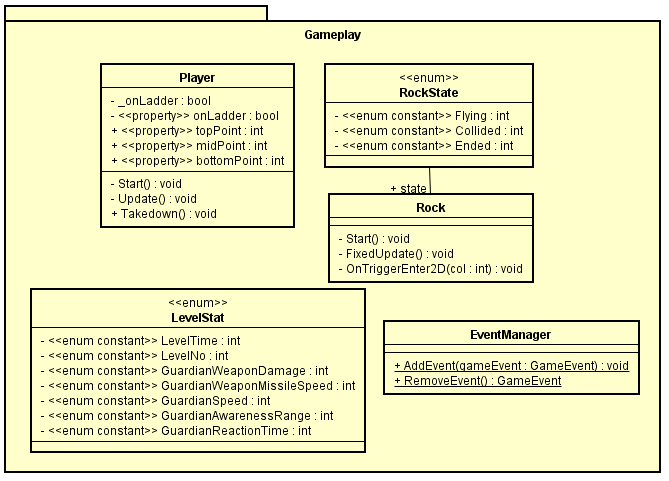
**Player:** Class that keeps the information about player’s visual attributes such as width & length as well as necessary methods/operations to start visualizing.

**RockState:**

**Rock:**

**EventManager:**

**LevelStat:**



# 3.0 Class Interfaces

## 3.1 AI package

## 3.2 ML package

## 3.3 Enemy package

## 3.4 Gameplay Package