**University of Ulster**

**BSc (Hons) Computing Systems**

**COM547 - Computing Systems Project**

**Project Plan**

**Dynamic cover-based system for Unity Game AI Players**

**Student Name: Groza Leontin**

**Student Number: 8888888**

**Supervisor: Tendai Mhlanga**

Contents

[Introduction 3](#_Toc528934907)

[Project Purpose 3](#_Toc528934908)

[Background 3](#_Toc528934909)

[Context 3](#_Toc528934910)

[Field of View (FOV) 4](#_Toc528934911)

[A\* Pathfinder 5](#_Toc528934912)

[Edge Detections 6](#_Toc528934913)

[Aims and Objectives 7](#_Toc528934914)

[Project Aims 7](#_Toc528934915)

[Project Scope 8](#_Toc528934916)

[Unity Profiler 8](#_Toc528934917)

[Stakeholders 9](#_Toc528934918)

[Project Objectives 9](#_Toc528934919)

[Conclusion 10](#_Toc528934920)

[Deliverables 10](#_Toc528934921)

[Project deliverables and timeline 10](#_Toc528934922)

[Indicative hardware and software resources 11](#_Toc528934923)

[Research 11](#_Toc528934924)

[Unity Game Engine 12](#_Toc528934925)

[Visual Studio 12](#_Toc528934926)

[C# Language 12](#_Toc528934927)

[Development Life Cycle 13](#_Toc528934928)

[Iterative Model 13](#_Toc528934929)

[Spiral Model 14](#_Toc528934930)

[Waterfall Model 14](#_Toc528934931)

[Agile Model 15](#_Toc528934932)

[Life Cycle Implemented 15](#_Toc528934933)

[Evaluation 16](#_Toc528934934)

[Risk Assessment 16](#_Toc528934935)

[References 17](#_Toc528934936)

## 

## Introduction

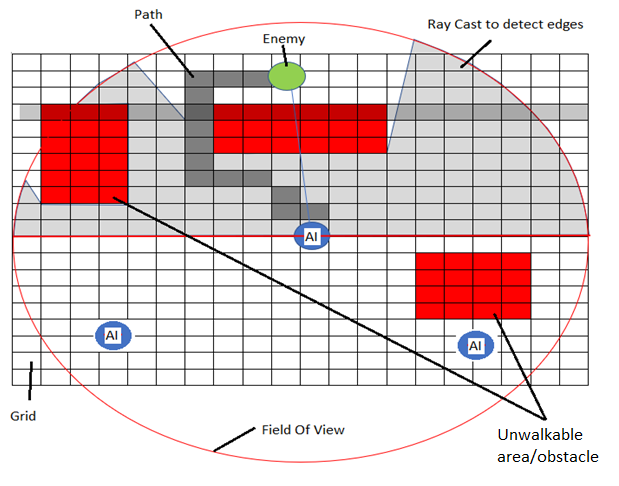
### Project Purpose

The objective of the project involves a number of different tasks designed to temper with the foundation of multithreading and create stages to house thread-switching hardware in a game development software responsible for 50% of the world’s games development.

### Background

Patient Zero is a proposed Zombie based 3rd person shooter project being developed in the Unity game engine. On a battle scene, AI agents operate using the A\* path finding algorithm to calculate a path toward the target in real-time. Each agent has their own A\* pathfinding script which is calculated per frame, so a scene with several AI opposition is taxing to the CPU as graphics rendering, physics calculations and the AI Agents A\* pathfinding are all done on the main thread in real time making obstacle detection dynamic meaning if new obstacles are introduced on the scene enemy AI will be aware of it. All these real-time calculations however are taxing and scenes have to be reduced in complexity by reducing the graphics rendering tax.

### Context



The game frame is constructed on a grid. All the game objects are positioned on the grid on different coordinates. The grid will calculate the real time location for all the game objects, either static or mobile according to the field of view of the AI. The AI game object is implemented with several programming methods that allow ray cast static object edge detection, A\* pathfinding, ray cast enemy detection and AI detection. The AI is implemented with a field of view method that reveals the game scene in the AI’s field of view. All the detection methods implemented to the AI will take place only in the AI’s field of view. The computations for each game object along with the rest of the elements are done in the main thread in real time taxing the capabilities of the CPU. The desired prototype method will implement the removal of the AI threads form the main thread and create individual threads for each AI game object that will later be synchronised with the main thread.

### Field of View (FOV)

The AI characters have a number of sensory processing implementations. Field of View (FOV) is the simulation of sight sensory implementation and every frame runs in real time to ensure real-time detection of a target. When a target is found it triggers the A\* path finding algorithm to establish paths to the target and to cover locations.

Area of sight. Any target within this range of the AI’s FOV not obscured by an obstacle, will be seen by the AI.

### A\* Pathfinder

The A\* pathfinding algorithm is passive and is not engaged until a target has been identified within the FOV. It consists of a grid with a size of 45\*45 with each cell size set to 1/45 the size of the grid to improve performance but in the detriment of reducing the precision of the walkable paths to non-walkable areas, particularly on scenes with obstacles that are close together. Ideal size of the cells would be 0.5/45 of the grid. The A\* pathfinder works in real-time with the grid attached to the AI character, the AI being directly positioned in the centre of the grid and it moves instant with the AI character maintaining his central location within the grid. Every frame in the grid recalculates the path, detects obstacles making the scene dynamic as new obstacles can be introduced at runtime and will be detected by the AI. This however is a taxing process for the graphics rendering system as FOV and A\* pathfinder are sharing this thread.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

### Edge Detections

Ray casts are cast out for the purpose of detecting edges of obstacles. The goal of this is to have the AI agent dynamically choose cover locations in a fire-fight with the target. Originally empty game objects with a script attached with a single Boolean variable and public getter method were at design time placed in cover locations and then sorted in an array in the AI script. The getter method would return the Boolean value, if it returned true, then that cover was occupied by an AI colleague and if it returned false the cover location was available to be occupied. The AI would then loop through this array calling the getter method and when a false was returned, they would choose that cover and turn its global Boolean variable to false, and other AI’s would not attempt to occupy that position.

This made the AI rigid to the pre-set cover locations and not able to use new obstacles introduced at runtime thus reducing the dynamism of the battle scene. The job of the ray casts is to detect obstacles and edges and rather have the AI choose their obstacle edges based on what is on the scene. Again, these ray casts are done in real-time on the main thread along with A\* pathfinder and FOV algorithms making this taxing as graphics rendering, physics calculations are also sharing this thread.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Currently to reduce the calculation tasks, all Update() methods have been removed from the AI agents’ scripts. An empty game object has been created with a single script attached with an Update method and the AI agents set in an array of this script. A loop iterates over each agent and calls an update method in which all the calculation methods the AI agent needs to perform are called.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Agent 1 | Agent 2 | Agent 3 | Agent 4 | Agent 5 | Agent 6 | Agent 7 | Etc.. |

Among the A\* path finding, agents produce hundreds of ray casts to dynamically detect obstacles and more importantly detect edges of these obstacles which are now envisioned and will be used to dynamically allow AI agents to set fire cover positions and inform colleagues of these new positions.

This can be achieved by saving the new location in a public static global variable allowing all AI agents using the same script to see the static Array List. Unity in 2018 have introduced a jobs system which is a process of scheduling jobs to be executed.

## Aims and Objectives

### Project Aims

1. To critically review the new Unity Jobs multi-threading system in relation to the multi-threading objective.
2. To remove the AI agents from the main thread and have them run on individual threads.
3. To use the edge detection ray cast framework of the enemy game object to allocate dynamic cover at run time as opposed to the current framework of pre-set cover positions.
4. To propose an enemy AI coordination framework e.g. enemy AI provide cover fire for colleagues designed for close cover to approach the target.

### Project Scope

In a single-threaded computer system the time to complete a program depends of the amount of work a CPU has to do, considering one instruction is executed at one time and the result is displayed at one time. Multithreading is a specific programming type that profits from the Central Processing Unit ability to run multiple threads at the same time. The instructions are no longer executed one after another, they run concurrently. Once a program starts, a default thread will start, called *Main Thread*. The main thread will create multiple new threads to handle future new tasks. The threads run simultaneously and will synchronize the result with the main thread once the task has been completed. The multi-threading approach works well if a program contains few several tasks that can run for a long time, however by having multiple tasks with threads running for a short period of time and all of them synchronise with the main thread, it will limit the capabilities of the CPU and the OS.

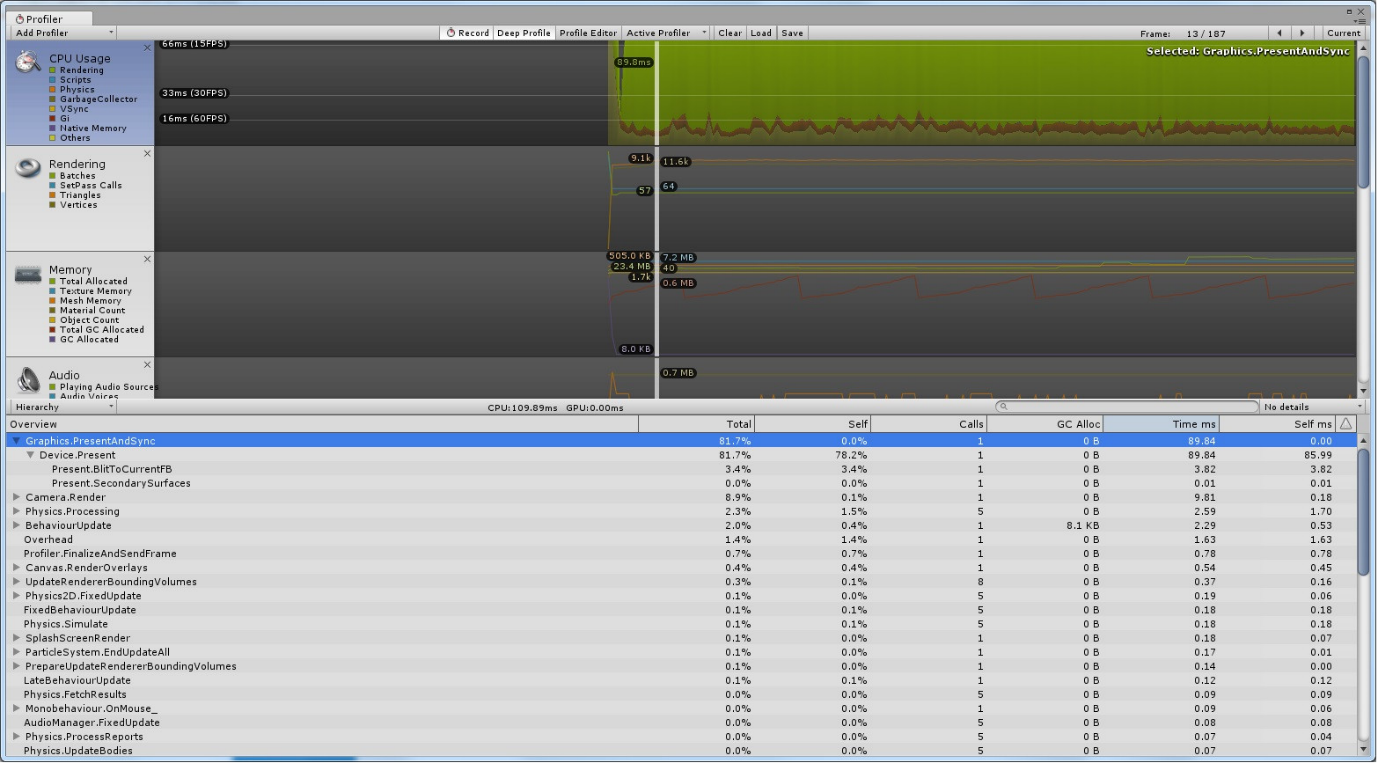
The newly introduced Unity Jobs System intends to separate the data from functions, enable multicores processing and save multi-threading by controlling the functionality of multiple game objects into a single job and spread it across multiple threads to speed up the whole process. The Unity Jobs System will allow the developer to create similar game objects with the same purpose and functionality as long as the functionality of the game object will be defined at the beginning and not at run-time.

In a Unity engine game, the best functionality can be given by separating the individual game logic threads from the main-thread game processing logic. To best improve the rendering of the game, the main thread must drop small individual threads that overlap taxing the resources of the CPU.

The proposed prototype allows the creation of individual and separate single-thread game processing logic that will be synchronised with the main thread at finish but will not start from the main thread. The prototype method could improve the rendering speed of the game by not overlapping the queue of the threads generated by the main thread.

### Unity Profiler

Unity has recently introduced an important feature that will help control and view the optimisation of the software. The Unity Profiler is in place to help optimize the software by reporting the time spent in different stages of the application. It will give important feedback on the time percentage spent with rendering, animation and game logic. The profiler also records the data performance of the software, allows observations in performance improvement or downsize according to the prototype methods implemented. There can be cases where a prototype could have a negative impact on the frame rate. During the development process focus with the Unity Profiler will stand to help measure the optimisation and the progress set with development.



### Stakeholders

Contact with stakeholders will be kept via regular meetings and email exchange.

|  |  |
| --- | --- |
| **Name/Organisation** | **E-mail** |
| Tendai Mhlanga  Ulster University | Tendai.Mhlanga@qa.com |

### Project Objectives

Main Objectives:

1. Evaluate the difference between the classic multi-threading system and the Unity Jobs system, to gain an understanding about steps taken in order to achieve this specific upgrade and based on the acquired documentation push towards a prototype coding method that will allow individual threading to be created separately and not start from the main thread. Once a prototype is created it will be implemented into the main frame of the game and tested to evaluate the functionality and if it meets the purpose for which it has been developed.
2. Unity AI development gives the programmer the tools to create a game character that behaves similar to a living organism. Regardless of the native programming, a game object, AI, needs to have supplementary coding implemented to achieve self-learning.

Unity will provide basic game object skills for characters created but it will be limited to the standard programming such as basic pathfinding, *NavMesh*. A major objective of the project is to create environmental self-awareness for AI characters. Prototype supplementary coding can be constructed to *teach* a game character how to read and use the environment in pathfinding or completing a task without supplementary guidance from the user.

A number of activities will need to be performed to achieve the project objectives:

1. Develop programming skills in in *C#*  programming language, a programming language developed by Microsoft.
2. Become familiar with Visual Studio, a Microsoft development environment used to write and manage code written for Microsoft Windows and used to develop GUI and consoles.
3. Understand the concept of single-thread, multi-thread and their dependencies in regard to the resources generated by the CPU and the availability of the bandwidth.
4. Set up a development environment to create a simple game frame with multiple game objects extracted from *Patient Zero* each implemented with separate coding, to allow tempering and prototyping without actually tempering with the main game at this stage.
5. Prototype different coding methods in regards with the objective of the project.
6. Research for existing AI dynamic programming.
7. Grow in size the game frame to allow more flexibility in the programming and functionality of game objects and characters.
8. Identify and review similar game development platforms that have the necessary specification to be allowed in the development of the project.
9. Research different programming languages and their library extensions to analyse resource allocation in terms of multi-threading.
10. Research a software development cycle to be used in the development of the project.
11. Carry out extensive testing and documentation throughout the whole development process.
12. Debug any coding errors arising from the testing process.
13. Create comment throughout the code to allow a light debugging process.
14. Integrate the prototype methods into the game frame and towards the end of the project implement the new methods and functions onto *Patient Zero*.
15. Create a coding report and instructions.

### Conclusion

The due-date of the project will be remain within the time constrains if the main objectives will be properly identified to allow the project to be kept in scope and therefore achieve the overhaul aims.

## Deliverables

The tasks that must be completed by the end of the project are intitled deliverables.

### Project deliverables and timeline

The proposed aims, which include the conception, design and development, characterise a full academic year project commencing on September 2018.

The proposed aims and objectives will be divided in smaller, individual tasks, each connected to another according to the specific aim. The project will be conducted in different stages, each stage corresponding to a different aim.

Immediately as the Unity Jobs system has been reviewed in comparison with the multi-thread system, a report will document the findings together with the proposed methods of upgrade to modify the thread system such as to correspond the project needs. Several prototype programming methods will be developed and implemented inside the game code to modify the current development method.

If the prototype method is successful, the AI game object will be removed from the main thread to individual threads and the second aim will be completed. Otherwise, new methods will be designed and developed to improve the rendering without creating individual threads but modifying the main thread.

With stage completed the main focus will turn towards the behaviour of the game objects. Programming methods will be developed and implemented to attribute AI game-objects self-awareness with the surrounding environment.

Developing methods to allow the AI to respond to the surrounding environment will have an immense impact for the last part of the project, at this stage the AI game objects will be implemented supplementary coding learning how to make use of the rest of the game objects found in the game frame considering that at this point the AI game object will be responsible for its individual game play.

A project management is mandatory to be implemented at the beginning of the project to increase the percentage of the success rate, keep the project on track and meet the requirement of the project.

The main procedures contained by a process management are:

* Initiating
* Planning
* Executing
* Controlling
* Monitoring
* Closing

Several markers are to be considered in the project planning to achieve the overhaul objectives:

* Analysis
* Planning
* Information Gathering
* Design
* Development
* Prototype
* Testing
* Implementation
* Evaluation.

A project management delivers an exclusive attention moulded by the aims, resources and time allocated to a project.

### Indicative hardware and software resources

The mandatory resources for project delivery are as follow:

* Personal PC - Intel Core i7-7500U (4M Cache, 2.7 GHz), 8GB DDR4 2133MHz, 1TB SATA HDD
* Programming software (Visual Studio IDE, Code Editor C# programming language)
* Unity Game engine platform.

All resources are available.

## Research

The indicative hardware and software resources have been researched with the particular implemented features to create a basic understanding of the pros and cons in regard to functionality and resource allocation.

The proposed game, *Patient Zero*, has been developed strictly in Unity game engine, the development is mandatory to be continued on the same platform. Research will be conducted for different game engine platforms but will not have a big extent as only basic functions will be inquired such as threading.

### Unity Game Engine

Unity is a popular game development tool used intensively by amateur and professional developers enabling features such as android game development.

Advantages:

* Dedicated 2D tools
* Dedicated 3D tools
* Object-oriented drag and drop UI
* Basic programming knowledge required
* Extensive documentation available

Disadvantages:

* Script writing requires training
* Engine takes a large amount of storage
* Editor limited only for Windows
* Enabling all of the features is taxing
* Extensive developing requires paid subscription

### Visual Studio

Visual Studio is a code editor created by Microsoft supporting code-completion and code refraction and it uses Microsoft software platforms such as Windows Store and Microsoft Silverlight. Approximately 36 programming languages are supported by Visual Studio and the debugger and editor will support all programming languages.

Advantages:

* Friendly interface
* Easy to customize
* Large number of plugins available through marketplace
* Extremely good for small applications
* Limited programming required

Disadvantages:

* Slow in searching files and folders
* Time to launch the app is higher than most competitors
* Microsoft proprietary language
* Limited compatibility with non-Microsoft platforms

### C# Language

C# (sharp) is a programming language developed by Microsoft. It is intended to be simple, OOP language suitable for programming software components for different environments. Suitable for writing application and capable of high processing power features.

Advantages:

* Intended for RAD (Rapid Application Development)
* Safe to run
* Capable of developing mobile apps by using Xamarin
* Robust and flexible

Disadvantages:

* Lack of automated methods such as getters and setters
* Cannot be used in real-time systems
* Limited multiple inheritance
* Compiler warning/error messages are extremely limited

## Development Life Cycle

The intended use of the development life cycle process is to build a high-quality, efficient and effective product.

According to the specifications of the project several software life cycles can be considered in the development of the project:

* Iterative model
* Spiral model
* Waterfall model
* Agile model

### Iterative Model

The process begins with modest implementations of developer objectives which will be enhanced along the process and evolved to a point where a complete prototype can be build and implemented on the existing software. In an iterative model the development is not bound to start with full requirements .

Iterative model requirements:

* System requirements are defined
* Major objectives must be defined
* Time to deliver constraint

Advantages:

* Partial functionality can be developed at the beginning of the life cycle
* Progress measurement possibility
* Functional prototype delivered with every incrementation
* Early and periodical results
* Identified risks from each increment can be applied to the following one
* Software is produced early offering feedback possibilities
* Requirements changing is supported

Disadvantages:

* Supplementary management attention is required
* Extra resources would need to be implemented
* Design issues arise from gathering requirement throughout the whole process
* Requires extensive management attention
* The risks may not be known at the end of the project

### Spiral Model

The spiral model is a combination of the iterative model with waterfall model with a high focus on risk analysis. Widley used in the industry, it’s focused on the natural development of the product representing a minimum risk for the customer as it develops sideways with maturity. The spiral model is divided in four stages. The project repeatedly passes through the four stages in iterations called spirals.

The stages of the spiral model are:

* Identification – identify and gathering requirements at the baseline of the spiral
* Design – conceptual design, architectural design and final project design are defined
* Construct – the development of the actual prototype takes shape to get feedback from the customer
* Evaluation and Analysis – evaluation and risk management

Advantages:

* Accommodates requirements alteration
* Prototypes are broadly implemented
* The project development can be seen in early times
* Increased risk management
* Project development can be divided in smaller tasks

Disadvantages:

* Highly complex process
* Increased volume of documentation needed considering the project needs to pass through various stages
* High cost can occur in small projects

### Waterfall Model

The very first process to be introduces in product development life cycle is the waterfall model. In a waterfall model the development process is presented in a linear way, every step in the development process starts only if the previous one has been completed.

The stages of the waterfall model are:

* Requirements gathering – documenting the requirements of the system
* Design – the gathered requirements are evaluated, and a system design is produced accordingly
* Implementation – following the design phase, the system is built in small units which are first tested and later implemented in a system
* Testing and Implementation – the units built in the implementation stage are integrated in a system
* Deployment – once the system has been tested it will be released to the customer or on the market
* Maintenance – issues will arise once the system is implemented in the customers environment. System patches are developed to repair the issues and system updates are implemented to create improved versions of the system.

Advantages:

* Simple to use and easy to understand
* Phases are developed one at a time
* Task are easy to organise
* Highly functional for very small projects
* Deliverables are seen after each stage
* Phases are well defined

Disadvantages:

* Not a suitable model for long time frame projects
* Working software is not developed until the final stages of the process
* The requirements cannot be changed
* Cannot measure the progress in stages
* Not suitable for OOP projects

### Agile Model

The agile model divides the process in small tasks. Iterations will provide the builds; each iteration can last up to several weeks. Concurrent work will be done in different stages such as planning, information gathering, design, coding, testing.

The prime key in agile model is that each project is different from the rest and methods need to be developed and expanded to suit the best needs of the project. In agile model, following one iteration, working software is developed in builds and the final built holds all of the customer requirements.

Advantages:

* Team work is promoted
* Minimum planning required
* Early solution developed
* Minimum resources are required
* Functional for rapid changing environments
* Allows flexibility

Disadvantages:

* Minimum documentation leads to team individual dependency
* Having an accomplished leader is imperative
* Challenging transfer of technology considering the lack of documentation
* The customer must interact with the project

### Life Cycle Implemented

The comparison conducted among the different models of software development life cycle points out several advantages and disadvantages of each model depending of the requirements of the project. Taking in consideration the aims and objectives of the project, early software development and implementation is required to uncover early prototype issues and faults. Developing prototype AI game objects programming methods will inevitably lead to new issues arising from implementation, issues that can affect the full or partial functionality of the software and will require attention to maintain the functionality of the system.

The iterative model will be the development life cycle implemented for this project. The possibility of early builds, testing in small iterations, parallel development and the prospect of measuring the progress are suitable for the development of the project.

### Evaluation

The software evaluation will be carried out through regular testing and implementation testing. Prototype, findings and issues will be documented and evaluated to set the general direction of the project development.

## Risk Assessment

Software development projects are a collection of programs with high number of dependencies. Involves the development of a product that has never been done before even though the development process is similar to other projects.

Monitoring the risks of the project throughout the system is mandatory. Different types of risks will affect budgets, performance and user satisfaction. If a risk has been identified urgent actions are to be taken to limit or eliminate the cause or issue that could ham the overhaul performance of the software.

Software development risks:

* Lack in personnel
* Developing wrong prototypes
* Adding more functionality than necessary
* Real-time performance issues
* Continuous requirements change
* Ignoring resource capabilities
* Shortfall in performance
* Incorrect detailing
* Absurd time to market

Effective risk management is an important tool to be employed for the success of a project.

Risk management is not extensively used and can prove to be a competitive advantage for developers which implement risk management processes in their projects.

# References

Anon., 2016. *Tutorials Point.* [Online]   
Available at: https://www.tutorialspoint.com/sdlc/sdlc\_iterative\_model.htm  
[Accessed 1 November 2018].

Anon., 2016. *Unity.* [Online]   
Available at: https://unity3d.com/  
[Accessed 28 10 2018].

Anon., 2018. *C# Job System Exposed.* [Online]   
Available at: https://docs.unity3d.com/Manual/JobSystemMultithreading.html  
[Accessed 29 10 2018].

Anon., . *C# Language Specification.* ed. (): Ecma International.

Anon., . *What is Project Management?.* [Online]   
Available at: http://www.pmi.org/About-Us/About-Us-What-is-Project-Management.aspx  
[Accessed 31 10 2018].

Anon., . *What's new in C# 7.2.* [Online]   
Available at: https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-7-2  
[Accessed 1 11 2018].

Bennett, C. & Sagmiller, D. V., 2014. *Unity AI Programming Essentials.* [Online]   
Available at: https://amazon.com/unity-programming-essentials-curtis-bennett/dp/1783553553  
[Accessed 29 10 2018].

Birrell, A., 2005. *An Introduction to Programming with C# Threads.* [Online]   
Available at: http://birrell.org/andrew/papers/035-threads.pdf  
[Accessed 29 10 2018].

Gajjar, A., 2013. *Learning C# by Developing Games with Unity 3D Beginner's Guide.* s.l.: Packt Publishing Ltd.

Kwak, Y. H. & Stoddard, J., 2004. Project risk management: lessons learned from software development environment. *Technovation,* , 24(11), pp. 915-920.

Phillips, K., 2016. *Developing a 3D game engine using multi-threading.* [Online]   
Available at: http://ssudl.solent.ac.uk/3400  
[Accessed 29 10 2018].

Rhalibi, A. E., Merabti, M. & Shen, Y., 2006. *Improving game processing in multithreading and multiprocessor architecture.* [Online]   
Available at: http://dblp.uni-trier.de/db/conf/edutainment/edutainment2006.html  
[Accessed 29 10 2018].

Snell, M. & Powers, L., 2010. *Microsoft Visual Studio 2010 Unleashed.* [Online]   
Available at: https://amazon.com/microsoft-visual-studio-2010-unleashed/dp/0672330814  
[Accessed 1 11 2018].

Subashni, S. & Kumar, N. S., 2010. *Software Testing Using Visual Studio 2010.* [Online]   
Available at: https://amazon.com/software-testing-using-visual-studio/dp/1849681406  
[Accessed 30 10 2018].

Yoo, J.-K. & Kim, Y., 2015. *Packet Discard Policy of Network Thread in an Unity Engine for Multi-player Online Games.* [Online]   
Available at: http://ocean.kisti.re.kr/downfile/volume/kgames/kgohcl/2015/v15n6/kgohcl\_2015\_v15n6\_97.pdf  
[Accessed 29 10 2018].

Zander, J., . *Couple of Historical Facts.* [Online]   
Available at: http://blogs.msdn.com/jasonz/archive/2007/11/23/couple-of-historical-facts.aspx  
[Accessed 30 10 2018].