# ‘Survive The Infected’

Alankrit Joshi

# Introduction and Project Background

- ## Project Aim and Objectives

    - Create an environment with finite terrain, trees, water bed, day-night cycle and fog in Unity3d for a First Person game
    - Add basic FSM for defining player audio, infected behavior and narration through timed events
    - Bake Unity's in-built NavMesh tool to enable navigation, modify pathfinding algorithm using a custom A* algorithm
    - Implement swarming algorithm for making it a challenge for the player to escape from hoards of zombies in a vicinity
    - Use opponent modelling over a game session by recording frequent player positions and time spent in specific areas

- ## Progress So Far

    - Successfully setup a 3D environment with dynamic lighting, fog, in-game day-night cycle, first person controller and zombie AI agents
    - Used FSMs for zombie behavior, story narration and timed events
    - Applied customized A* algorithm for zombie pathfinding towards flare/player and defined mechanisms according to story
    - Created a swarming mechanism for zombies to hound and corner the player
    - Recorded player occurrences over a gameplay and used data for next game

- What more can be done
    - Replace current Unity's AI controller model with a custom asset from Unity store to make the zombies look more authentic
    - Add multiplayer components (as learnt from Unity's Multiplayer and VR workshop held on 04-22-2016 in Boston) to allow multiple players tactically escape zombies, making it a challenge to incorporate Decision Making
    - Refactor the game to make it VR ready for Oculus Rift, HTC Vive and PlayStation VR (as showcased by Developers at PAX East 2016 in Boston)
    - Use NavMesh off-mesh links for allowing AI agents to traverse looping areas or seemingly inaccessible areas
    - Upgrade swarming algorithm by relatively targeting vectors of zombies residing in a vicinity to hound the player efficiently by reacting to sound
    - Perfect the opponent modelling system by dynamically spawning zombies in directions player is most likely to turn to after seeing existing zombies

- Problems encountered and alternative solutions
    - Unassociating Unity's inbuilt pathfinding with the NavMesh. Solved this by taking guidance from the *A* Pathfinding Project*
    - Transferring game data from one scene to another for allowing opponent modelling. Solved this by using *DontDestroyOnLoad*
    - Swarming algorithm rendering glitchy and uncoordinated movements of zombies. Solved this by fixing the bug of unupdated positions

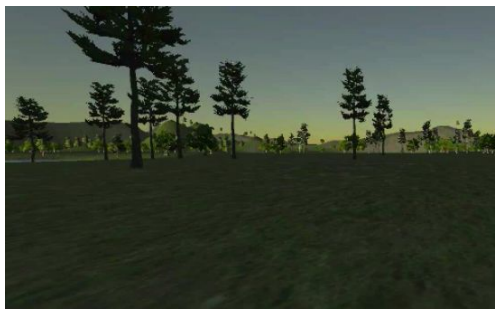
- Prefab instances were breaking and getting inconsistent. Solved this by refactoring code using fresh base prefabs for characters and objects
- Difficulty in writing animator controller for AI. Solved this by following a YouTube guide on creating a basic animator controller for a zombie asset

- Changes in the project plan
  - Procedural Content Generation for ammo and loot was put for future scope as it was overwhelming to implement on top of 4 other AI concepts
  - Project was changed from a Shooter to a Runner due to shooting mechanics in Unity3d being out of scope of this project scale

- Additional creative ventures during project
  - Finished a Blender course on Udemy to get acquainted with immensely sophisticated but rewarding animation tools in Unity3d
  - Attended two Unity workshops for hands-on in Animation, Multiplayer and VR projects
  - Initiated on the path to achieve Unity certification by preparing for Unity Certified Developer Exam to be held in Summer-2/Fall 2016
  - Met Unity representatives at PAX East 2016 for additional guidance on perfecting project scope in near future, extending the learning and possibility of landing a co-op/internship at Unity technologies
  - Began working on incorporating multiplayer elements to the zombie game, currently in development phase

- Start doing research work on refactoring the 3d game to VR, awaiting on a VR kit to identify and test spatial aspects

# Game Description and Mechanics



3D Environment in Unity3d 5.3.4

- Goal : ‘Survive the infected’ is a First-person runner where the goal is to escape hordes of zombies until rescue arrives at an extraction zone by dawn

- Environment : The game environment comprises of trees, waterbed, dynamic directional lighting, lens flare, day-night cycle, fog and varying depth of terrain surface. The game events will be accompanied by voiced dialogues and sound

- Characters : A first person controller is setup for the player and third person controllers for zombies, all of which have single audio source and behavior script

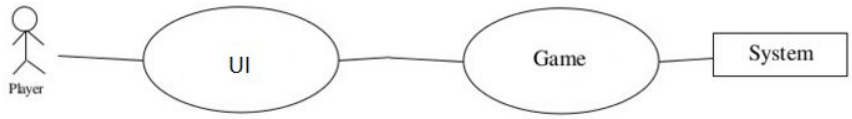
- Narrative : The player has to find a flat area for throwing a flare where the helicopter will arrive after 5 minutes. The flare will trigger zombie chase

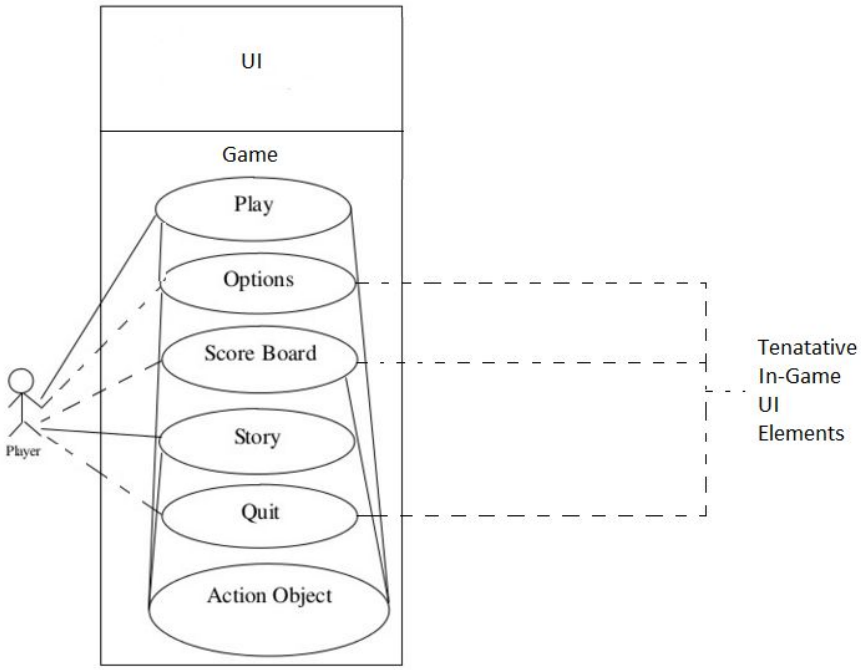
- Challenge : The zombie movement is intelligent and displays realistic coordinated movement. Over a game session, the difficulty increases

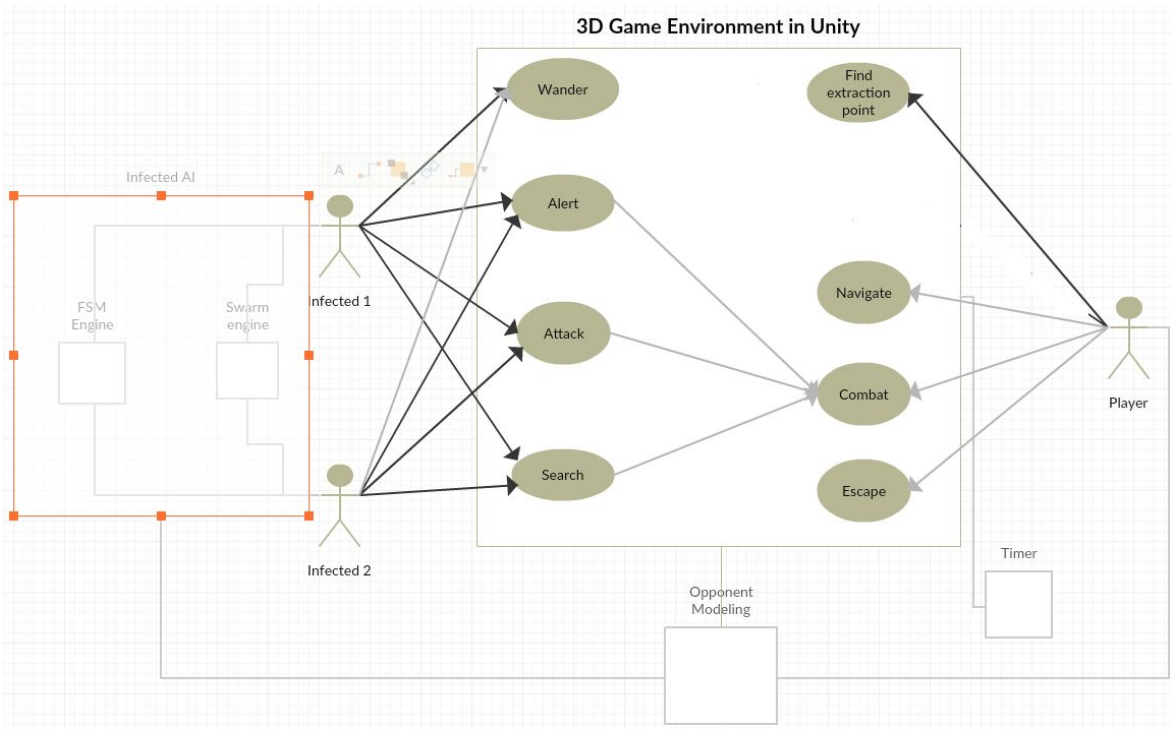
# Analysis and Design of Project

- Requirements Analysis
    - Normal Requirements
        - User friendly
        - Efficient
        - Lucrative
        - Low maintenance
        - Easy to reconfigure or retest
        - Simplified coding standards and modelling
    - Expected Requirements
        - Develop system with finite resources within 7 weeks
        - Reasonable hardware requirements
        - Free-to-play
        - Professional design strategy
        - Readable code
        - Implement AI concepts and exhibit knowledge
    - Exciting Requirements (additional to user expectations)
        - Provide tweakable codebase
        - Resources that can be referred for branched projects
        - Integration with network elements of Unity
        - VR compatible framework

- Use Cases




Game Design Outline

- Design Architecture
    - *Modelling* : Iterative system design and object oriented programming loosely based on MVC architecture to allow loose coupling of scenes and logic
    - *Scripting* : Scripts are attached as a component to GameObject and follow Object Oriented programming concepts. Scripting is done in tandem with Unity's Editor
    - *Hierarchy* : An order of GameObject is organized with appropriate parent and child nodes to improve design understanding and ease of changing things in future
    - *Non-physical GameObject* : When scripting behaviors and game rules if there is no particular physical GameObject that the scripting has to be run for then an empty GameObject is created at (0, 0, 0) and script is added to it as a component
    - *Abstraction and hiding* : Careful consideration is given to fields and methods that have to be accessed globally or within local scope of the class. Methods are called using MessageUp or Broadcast wherever appropriate to encourage modular interaction between game objects
    - *Asset Management* : Unity's standard assets are kept that are required and audio files and third party assets are organized separately. Scenes consist of the UI, game and the pause menu

- Testing
    - Functionality Testing : The project was developed using Iterative System Design borrowed from on-going course CS 5010: Program Design Paradigms, which allowed a testing model involving both white box and black box testing for functionality
        - Internal (White Box Testing) :
            - Unit testing : Each functional module was individually tested during development to ensure correctness
            - Integration testing : Two or more than two modules were always tested using standard integration tests like playability, viewability, render quality, frame rate and performance
            - Regression testing : Involved post integration tests after every unit tests were completed for new modules
            - Process : Requirement Analysis > Risk Analysis > Test Results
        - Overall (Black Box Testing) :
            - Error Guessing : Since the white box testing was a thorough and time consuming process, the black box testing was restricted to error guessing for eeking out remnant bugs
    - Compatibility Testing
        - Visual and render testing on all platforms (Windows, Linux, WebGL)
        - Backwards compatibility identified to Unity 5.3.0 as the game engine to edit project files

        - Hardware testing was done to test drawbacks of using on low-end PCs and identifying minimum and recommended requirements
        - Code compilation : C# and JavaScript transcoding was tried to identify changes in performance by changing native scripting language, however further reading suggested that code compilation was independent of scripting language in a project of such small scale
    - Soak Testing : Game was left running for over 48 hours to identify any game breaks or performance issues while monitored by FPS recorder to identify minimum, maximum and average frame rates
    - Beta Testing : Minor beta tests were conducted using college peers enrolled in non-game AI courses to enable transparency and unbiased reviews
    - Multiplayer Testing : <*Reserved for future*>

- Results and Evaluations

    - Functionality :
        - The prototype works with known minimal bugs whose resolutions is out of the scope of the course, for example, avoiding navigation of first person controller under a water bed
        - Zombie's animator controller needs a rework to incorporate an actual zombie model but Unity's AI third person controller works effectively for the purpose of this project
        - A* algorithm is extremely efficiently by core and is made even more light on resources using directional vector as heuristic

        - Swarming is relatively easily to comprehend and pursue as majority of the inputs the algorithm directly come from positions of objects in the game world
        - Opponent modelling is the toughest AI concept to implement properly and without mathematical opponent modelling it largely relies on error and trial picking of defined regions and time scales
    - Playability :
        - The game is fun to play and serves as a base First Person game to develop bigger projects on
        - Duration of the one game session is 5 minutes and can be easily shortened down to 2-3 depending on lack of content or can be increased to 10 minutes depending on additional things to do
        - The game in essence is made to showcase AI algorithms and suffers from non-replayability after few game sessions
    - Completeness :
        - The game is complete to project the 4 AI concepts : Pathfinding, FSMs, Swarming and Opponent Modelling
        - The zombie character controller needs further work to make the zombies look more scary
        - The terrain can be smoothed out further and can also include shooting mechanics and procedurally generated content
    - Reliability
        - The game runs smoothly from 45-60 fps even on low-budget PC, Linux or Mac machine

- General Game AI Evaluation Criteria
    - Intelligent Pathfinding using A*
        - Heuristic : Should be simple and effective > Used simple directional vector and distance as heuristic for A* algorithm
        - Walkability : Should disable falling off terrain or going through unwalkable surfaces > NavMesh enabled intelligent walkable areas on terrain with varied depth
        - Collision Avoidance : Nav agents should not transpose or overlap with each other > Used rigid body for controllers and draw distance to enable collision avoidance
        - Smooth Corrections : AI should not chase in jagged path when target changes > AI Third person controller blend trees enabled smooth walking and running transitions whenever AI behavior for chase is triggered
    - Finite State Machines
        - Zombie behaviors : Wander, Alert, Search and Attack are defined as standard states of a FSM whose transitions are triggered by detection/non-detection of a player
        - Story narration : Story is a narrated as a single direction FSM with no loops as a sequence of timed events
    - Swarming

        - Detection : Zombie detect the player if in a swarm by sharing the found status of the player to make them move towards the player even if the other zombies aren't in direct vicinity of the player
        - Targeting : The velocity vectors are dynamically changed governed by the A* algorithm values to make the zombies not follow a waypoint-like path to the player
    - Opponent Modelling
        - Compiling game data : Location and time spent are recorded in designated areas to use in next play in the current session
        - Using recorded data : The game strategically places zombies in more frequently visited places where the player might be successfully evading the zombies, allowing rise in difficulty and challenge

## AI Algorithms/Methods Used

- Intelligent Pathfinding : Implementation of a custom A* algorithm where the heuristic is last known directional vector provided by field of view; inclusion of obstacle avoidance using Unity's NavMesh and; correction techniques for smooth transitions
- Finite State Machines : Behaviors of players, zombies and in-game objects are scripted with a FSM. The narration of the story is also designed as a FSM with audio clips and events played as a sequence of events met under certain conditions
- Swarming behaviors : The zombie GameObjects interact with each other in proximity by 'broadcasting' player location and updating their own knowledge

using Push-Subscribe model. This allows zombies to converge even if not in defined proximity

- Opponent Modelling : Regions are defined and frequency of each region is increased if player ventures into it often and spends more time. This information is conveyed to the next round where zombie spawns are more in number
- Integration of AI concepts : Iterative system design allows module creation stepwise to allow implementing one methodology after another for successful integration

## Design and Implementation Tools

- Technology : Unity3d 3.3.4 as the game development engine, Unity Editor for creating and modifying GameObjects, MonoDevelop for scripting
- Scripting Language : C# is used as it has simplified support for Object Oriented concepts and likeliness with C/C++
- Code Repository : GitHub (version control done natively on PC due to problems with GitHub registering changes in Unity project files)
- Unity Asset Store : For importing custom assets created by the Unity community
- Other tools : *Recast+Detour* used as a reference to model the customized pathfinding technique in a NavMesh. *A* Pathfinding Project* to get started with implementing pathfinding algorithms using Unity's NavMesh

- Machine : The game was developed on an Intel i7 machine with nVidia 960m for graphics. The game is playable on Windows, Mac and Linux and supports WebGL play using Unity Web Player

## Project Instructions and User's Manual

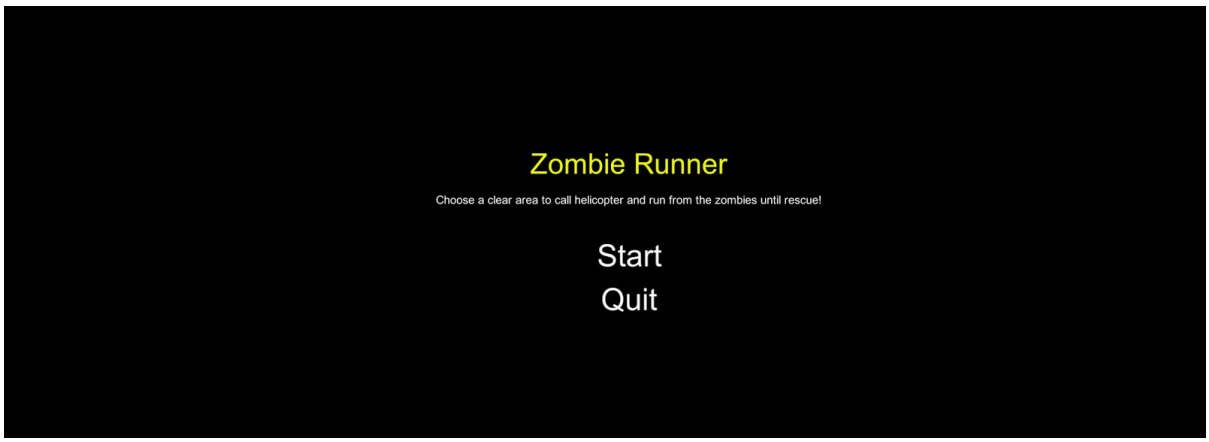
- Project Files :
  - Unity 5.3.4 (or higher) > Open > <Zombie Runner>
  - Select Layout 2 by 3 in top right corner using drop down
  - In the Project pane, navigate to the Project files in the Assets folder, distributed to Scenes, Scripts, Custom Assets, Prefabs and other dependencies
  - Open the Scene by double clicking on it and observe its GameObjects in Hierarchy window and inspect each GameObject using Inspector
  - Open *Start Menu* scene to successfully play the game using Unity3d engine. Loading *Game* scene will also play the game but without UI
  - To play the game, press on Play button on Center Top. Play can be maximized by pressing the option on top of the Game window
- Game Executable : Double click on the executable and choose resolution and quality to play in PC, Linux or Mac
- WebGL : Install Unity web player to play game. Run the game in Firefox (recommended); more information in References
- Navigation : W - Move Forward, A - Move Left, S - Move Backward, D - Move Right, Shift - Sprint. <Mouse> - Look Around

- UI Guide : Click on *Start* to play the game. If dead, click on *Try Again?* to retry the game. Similarly, *Play Again?* to play the game again after winning
- Game Rules : Find clear area to throw flare for the helicopter (flare thrown is part of scripted narrative). Do not get touched by infected zombies until helicopter arrives. You win when you are in proximity of the helicopter

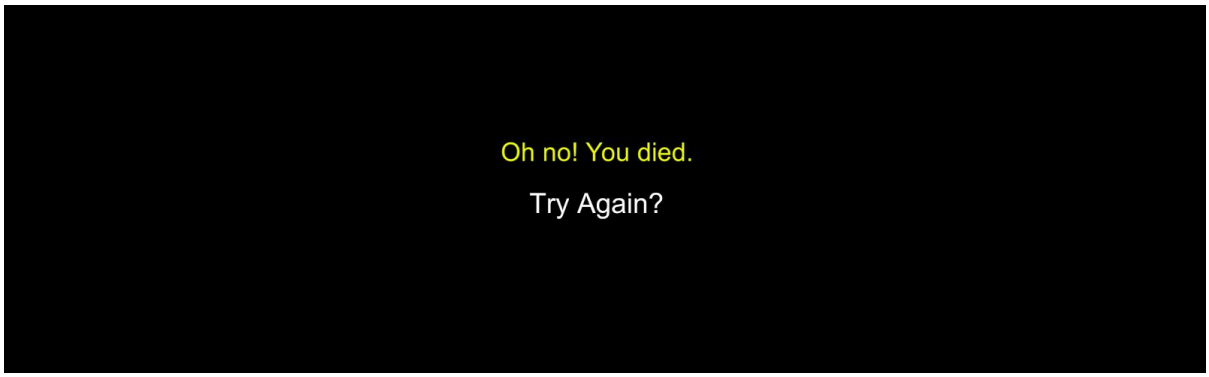
## Work Distribution

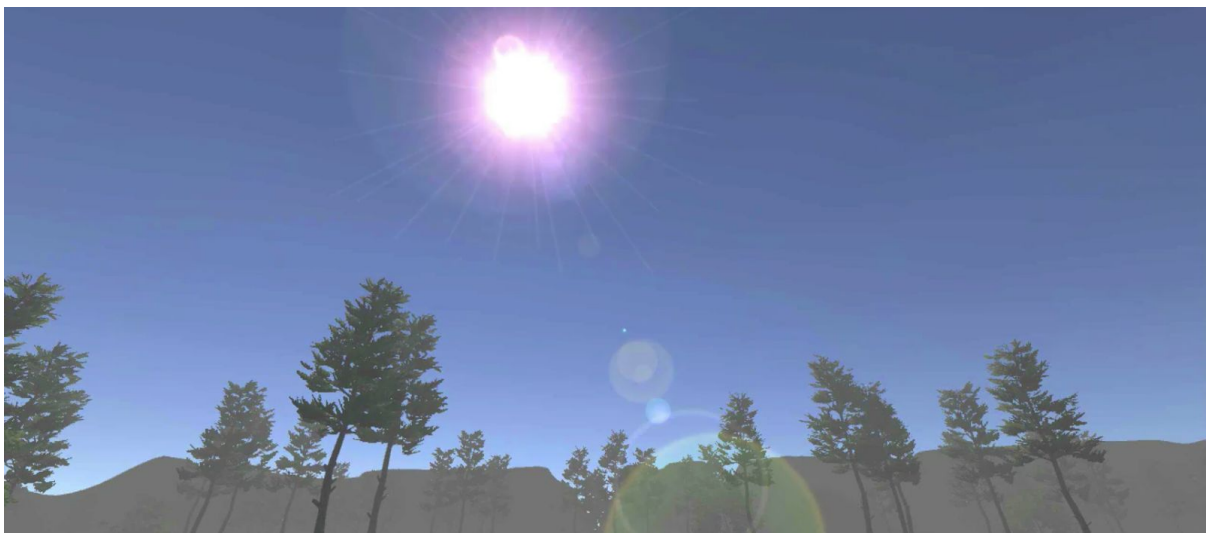
- Week 1: Research work on proposed AI techniques
- Week 2: Hands-on Unity3d and setting up terrain
- Week 3: Creating initial world, setup controllers and basic FSM for infected
- Week 4: Import assets for content and design narration using timed events
- Week 5: Modify pathfinding technique to include field of view
- Week 6: Implement swarming behavior of zombies
- Week 7: Create modules for opponent modelling and begin testing/debugging

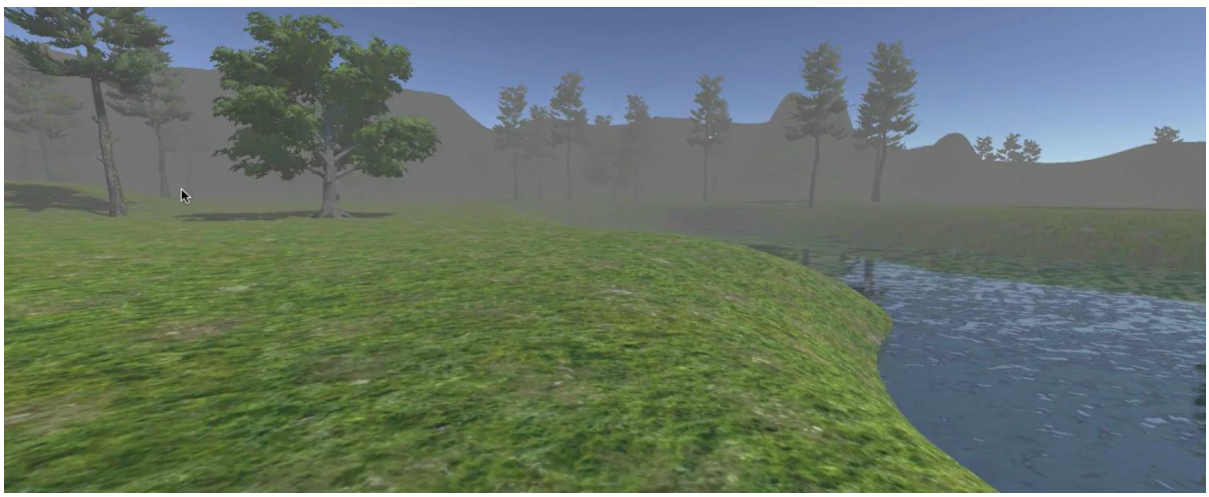
## Screenshots

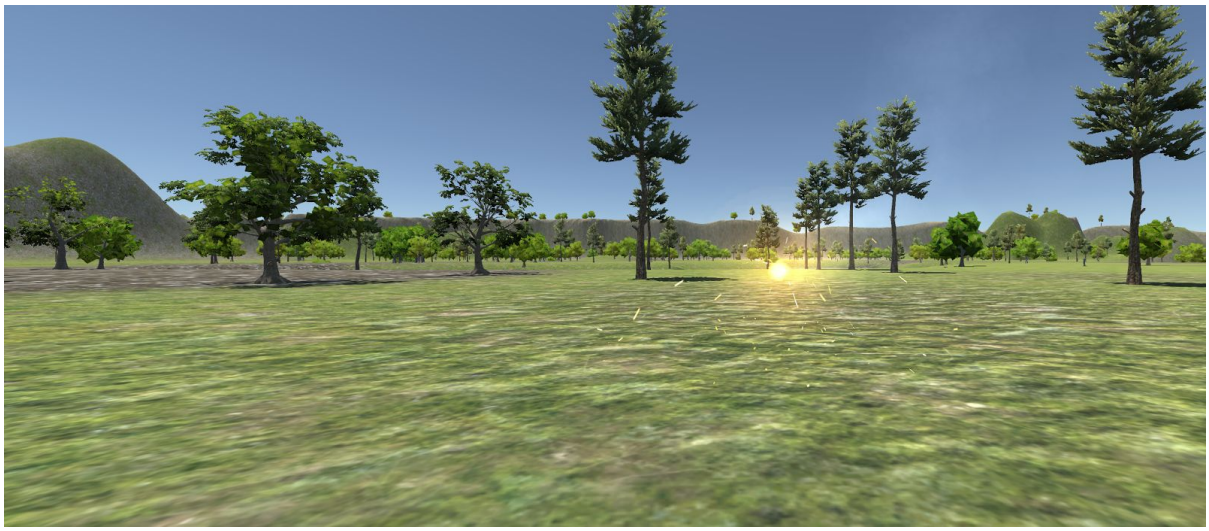



You escaped successfully!
Play Again?


Oh no! You died.
Try Again?







- References and Resources
    - Game Design Documentation : http://goo.gl/7edVYS
    - Unity Tutorial : http://unity3d.com/learn

- Pathfinding for Beginners : http://goo.gl/vWVmFe
- Pathfinding techniques : http://www.hindawi.com/journals/ijcgt/2015/736138/
- Pathfinding in Unity YouTube playlist : https://goo.gl/f9IZEm
- Off-mesh links : http://goo.gl/wIy8dz
- How to make Zombie AI : http://goo.gl/baVthZ
- Swarming : http://staff.washington.edu/paymana/swarm/krink_01.pdf
- Opponent Modelling : https://goo.gl/Y7wby7
- Usage of Blend Trees in Unity vs FSMs : http://goo.gl/odyRMJ
- Recast+Detour : https://goo.gl/Obs1fR
- A* Pathfinding Project : http://arongranberg.com/astar/
- Get Unity3d : http://unity3d.com/get-unity
- GitHub : https://github.com/
- Unity Asset Store : https://www.assetstore.unity3d.com/
- Unity Web Player : https://unity3d.com/webplayer
- Mozilla Firefox : https://www.mozilla.org/en-US/firefox/new/
- Audacity : http://www.audacityteam.org/

## Acknowledgements


- Professor Yetunde Folajimi for guidance during project and for review
- Game AI peers during in-class discussions and Piazza shares
- Guest lecturers (Julian Togelius, Britton Horn, Casper Harteveld) for additional information useful for modelling future utilization of project
- Fellow peers enrolled in Non-Game AI courses for beta testing