

Computer Games Development CW208

Software Requirements Specification

Year IV

[Stephen Ennis]

[C00181305]

[24/04/2017]

[Oisin Cawley]

**Contents**

[Introduction](#_k5vr2u3y5ea)

[Background](#_yvatmhrrrhi2)

[Aesthetics](#_qb3ajb6ab0ps)

[Overview](#_1ci93xb)

[2D Rendering](#_tpx0q5cycn13)

[User Interface](#_2t0lq53x33bg)

[Technical Notes](#_9m422wa12ye7)

[Features](#_hm0scx4n71v9)

[Camera](#_3oa19bbf0oqs)

[Conditions of Satisfaction](#_9pk9k9ltpqhd)

[Health System](#_srb68jsnjbrl)

[Conditions of Satisfaction](#_h618gq9g3a0a)

[Collision Manager](#_h2h9ya2f63c9)

[Conditions of Satisfaction](#_uei673x2fgwy)

[Map](#_dz3utjnnbylm)

[Conditions of Satisfaction](#_2ugolukle9t8)

[Game Over](#_2xldqc7yh25w)

[Conditions of Satisfaction](#_963le5br9t21)

[Menu](#_y46sdo8miluu)

[Conditions of Satisfaction](#_30pfus1t7egi)

[SceneManager](#_2unlul822ip7)

[Conditions of Satisfaction](#_c7vkroynhyw)

[Health and Damage](#_9vvx08fmzmf7)

[Conditions of Satisfaction](#_m1q4s2oqwvnz)

[GOAP](#_gnzsygf0fxyr)

[Conditions of Satisfaction](#_qh79tj3151k4)

[Advanced GOAP](#_mk81xmlza5cr)

[Conditions of Satisfaction](#_vt2c9ma7z46r)

[GameData](#_ec1nwjhky0pl)

[Conditions of Satisfaction](#_oubspsj3x423)

[Game UI](#_yku7of9iojrr)

[Conditions of Satisfaction](#_hkuuc3sb0wkz)

[FSM](#_5qwgq0z4ezqb)

[Conditions of Satisfaction](#_1pky6dplaet3)

[Pathfinding](#_ivlr1iodxcvg)

[Conditions of Satisfaction](#_oa40u6oa7yrk)

[Advanced Pathfinding](#_5r3mieszq2qg)

[Conditions of Satisfaction](#_452mxtcebu4h)

[Create Cover](#_x4b77pi2759h)

[Conditions of Satisfaction](#_23l32ckn2542)

[Game loader](#_fav353u4rtyz)

[Conditions of Satisfaction](#_vkupclfs17yx)

[Ammo Pool](#_yhmwvuhxjk70)

[Conditions of Satisfaction](#_bof4hkhi5fm1)

[Bullets](#_n2ygco6ledwv)

[Conditions of Satisfaction](#_x29pk63117mw)

[Unit](#_ptq5i9j6efqp)

[Conditions of Satisfaction](#_vnwz8liea9zf)

# 

# **Introduction**

This project is a 2D top down arena. There are two teams of four units each. The teams start at opposite ends of the map. They must try to fight survive until the time limit is up.

There are two types of AI; the “random” AI and GOAP AI. The purpose of this project is to investigate which will perform better and behave more realistically. The random AI works by randomly choosing whether to move to cover ot to move to a random location. The GOAP AI works by an implementation of Goal Oriented Action Planning which finds the optimal path to use to survive.

Pressing P will display debug information. Space ends skips to last screen if debug enabled. Minus and Plus change playback speed.

# **Background**

The project design draws inspiration from the XCOM series of games. In these games you control a squad but in a turn based system. I always found this game interesting in how you have to plan out your moves and make sure the squad sticks together and looks out for one another. I decided to implement an AI that could a simplified version of this in real time.

**Where does the project take place?**

The game takes place in an outdoor arena. There are walls around preventing free motion and the AI must pathfind around them. The walls also provide cover.

**What is the main focus?**

The aim of the game is to kill the opponent's units while trying to avoid getting hit.

# **Aesthetics**

## ***Overview***

The game will be rendered using SFML. There are no animations.

## ***2D Rendering***

Textures will be drawn to the screen using SFML sprites. The image file is loaded into a SFML Texture object and that can then be used to create the sprite. Sprites can be drawn to the screen.

***Camera***

The camera can move and zoom when the player presses the w,a,s,d or arrow keys and scroll wheel.

# **User Interface**

The user will see the health and ammo of all the units in both teams along the top of the screen. There is an ID number there and beside the actual position of the unit in the world to see which are which. A cover status icon indicates units in cover and a red x will indicate dead units.

# Technical Notes

SFML

Tiled

RapidJson

# Features

## Camera

User can move the camera and also zoom it

### Conditions of Satisfaction

* User can use arrow keys or w,a,s,d to pan the camera
* User can use the mouse wheel to zoom in and out

## Health System

Units have a health and this is displayed to the user. Damage is applied and drops health

### Conditions of Satisfaction

* Visible to user
* Drops when damage is taken

## Collision Manager

Collision manager contains a vector of pointers to all collidable objects in the world. It can use this to get a rectangle from it and check collisions. It can also do raycasting. There is collision filtering using bit masks to prevent unnecessary collision checks. Create Collidable class to give objects the ability to collide.

### Conditions of Satisfaction

* Bullets destroyed on collision with wall or unit
* Collision detection between ammo pick up and unit
* Ray Casting can be done to detect enemies in range of sight of unit

## Map

Map created in Tiled. Loaded in with RapidJSON and drawn to the screen.

### Conditions of Satisfaction

* Can see tilemap

## Game Over

A game over condition that shows stats for each team

### Conditions of Satisfaction

* Can see game time
* Can see type of AI each team was
* Can see kills,deaths,shots and nodes traversed for each team
* Can press space to return to the menu and start again

## Menu

Start menu that shows options

### Conditions of Satisfaction

* Shows 3 options you can select by pressing 1,2,3
* 1 starts with Random vs Random
* 2 starts with Random vs GOAP
* 3 starts with GOAP vs GOAP

## SceneManager

Each scene has its own class. Can do code on start,end,update,draw,poll events. Delta time is passed down to updates, event passed from pollevents.

### Conditions of Satisfaction

* Can push scenes without replacing the current scene
* Can pop scenes by replacing with specified scene
* Only update and render scene on the top of the stack

## Health and Damage

Units have health. Bullets have damage.

### Conditions of Satisfaction

* Lose health on collision with bullet
* If in cover for long enough then start to regain health

## GOAP

Have GoapActions which can be subclassed to define specific behaviour. Have GoapAgent which a AI that wants to use GOAP must inherit from. GOAP planner to plan actions.

### Conditions of Satisfaction

* Can define action preconditions and effects
* Can add actions to goap agent
* Can get world and goal state
* Can plan best queue of actions to reach goal state

## Advanced GOAP

Create actions for the unit to take cover, flank, get ammo and move to a random node.

### Conditions of Satisfaction

* If not in range to perform action to get a path to where we need to be to complete action
* Unit can take cover if enemies are spotted or not in cover
* Unit can try to flank an enemy (moving to cover that is closer) if hidden in cover for 5 seconds
* Unit can move to and get ammo pick up

## 

## 

## GameData

Singleton for storing data that is loaded in. Stores test data and test conditions. Stores debug conditions.

### Conditions of Satisfaction

* Holds all loaded in data so it can be used wherever it is needed
* Store textures so only 1 copy is used for multiple sprites of the same image

## Game UI

Draw portraits at the top of the screen for each team and every unit in the teams.

### Conditions of Satisfaction

* Displays units health bar and health value
* Displays ammo remaining
* Displays shield icon to indicate if in cover
* Displays red x to indicate if dead
* Has a border and transparent background color

## FSM

Create FSM for use with GOAP planning.

### Conditions of Satisfaction

* Idle state will decide on the goal
* MoveTo will move to within range of action requirement
* Perform Action will perform the action
* None is used for Random AI to choose randomly to move to cover to random non-cover

## Pathfinding

Create nodes and connections in tiled and load them in correctly. Set up a graph that is used for finding the best paths with A\*.

### Conditions of Satisfaction

* Nodes contain list of connecting nodes
* Can find best path from one node to another

## Advanced Pathfinding

Pathfinding to nearest cover,ammo etc.

### Conditions of Satisfaction

* Can find cover closest to starting position that isn't in range of enemies
* Can find cover from a certain direction with the above criteria
* Can find closest ammo node

## Create Cover

Create cover nodes and connecting pathfinding nodes

### Conditions of Satisfaction

* Load in nodes correctly and have cover slots
* Cover slots can only be occupied by 1 unit
* Each cover node has multiple potential cover slots (west,north,east,south)

## Game loader

Use rapidJSON to load all data from json file and map json file.

### Conditions of Satisfaction

* Sets up pathfinding
* Sets up cover
* Sets up tilemap
* Sets up game data stored in GameData class
* Sets up collision rects for collision manager

## Ammo Pool

A pool of ammo pickups.

### Conditions of Satisfaction

* Spawns a new one every 3 seconds
* Ammo despawns after 10 seconds
* Is collidable

## Bullets

Have a bullet pool to limit unnecessary recreation of bullets.

### Conditions of Satisfaction

* Can fire from and to a position

## Unit

Unit implements GOAP planner and functionality. It will fire and take cover.

### Conditions of Satisfaction

* Apply GOAP actions as described in GOAP features
* Fire at nearby target
* Moves to cover
* Moves to pathfinding nodes
* Uses pathfinding to find shortest path between start and goal
* Has health and can heal in cover