System Documentation

"Fog of War"

Bachelor's Thesis 086
Sindre Haugland Paulshus

Date	Version	Description
21.01.2020	0.1	Initial setup.
15.04.2020	0.2	Introduction added. Definitions and references added.
17.04.2020	0.3	Added class and architecture diagrams.
21.04.2020	1.0	First full draft. Added link to source code documentation, wrote about testing, edited the diagrams, added class explanations.

Table of Contents

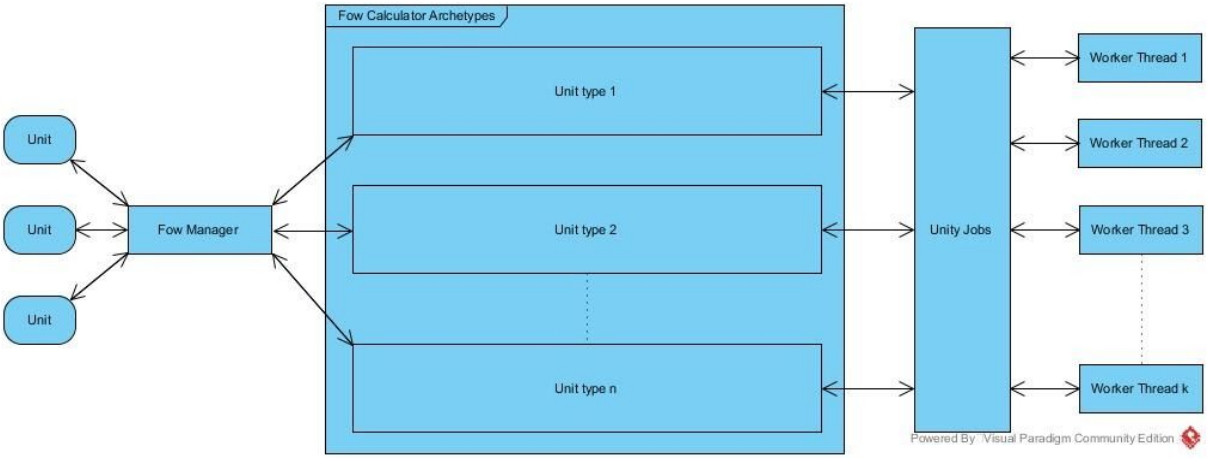
1. Introduction

This document describes the "Fog of War" (herby "FOW") solution created for the game Dwarfheim, being developed by Pineleaf Studio[1]. The project itself is staffed by a single bachelor student for his bachelor's thesis during the first semester of 2020. This document includes the overarching architecture of the solution, its classes, source code documentation and how testing were done.

1.1. Abbreviations, Acronyms and Definitions

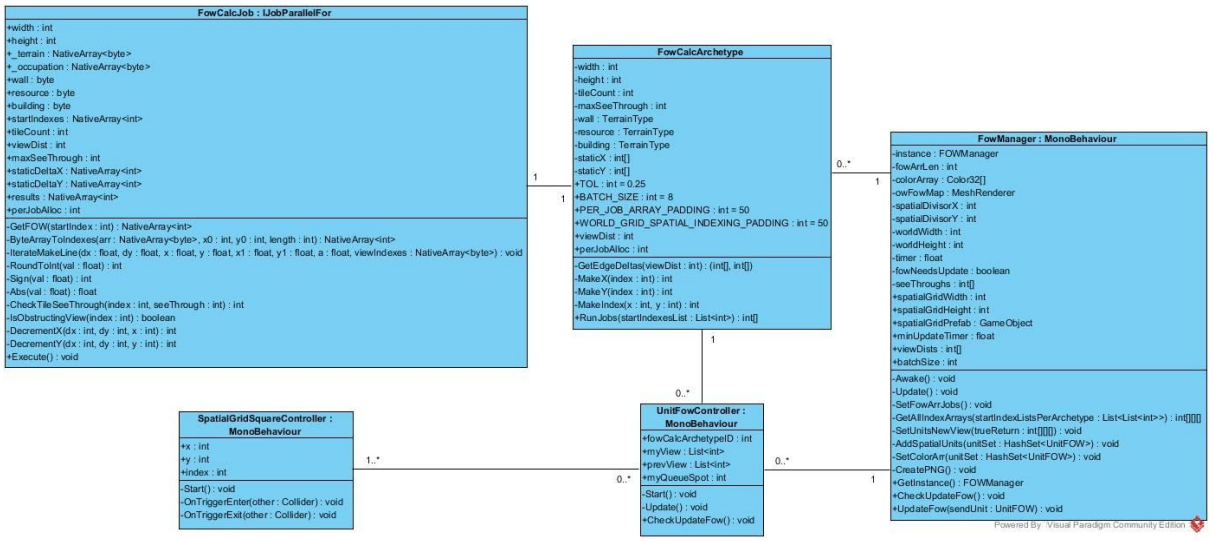
Prefab	A saved configuration of a GameObject. Can be instantiated in a scene.
FOW or Fow	Fog of War
Unity Profiler	A tool within Unity which records cpu and gpu usage, frame rate and various other data for the running game. It also shows which components, classes and functions are responsible for causing said data.

2. Architecture



The solution works in layers. Separate units communicate with the FowManager when their view needs to update. This happens when either the unit is spawned or the unit moves. The manager then requests the view for each unit that needs to update using Fow Calculator Archetypes and the units' spot (index) in the world grid. Each archetype schedules one Unity Parallel For Job, where each index it works on is a separate unit of that archetype. Therefore, each unit's view is calculated on a unique thread, which takes advantage of multiple CPU cores and boosts performance.

3. Class Diagram



The project consists of 5 classes:

SpatialGridSquareController	A component class of the SpatialGridSquare prefab. An amount of these prefabs are spawned in a square grid and help with spatial indexing the units in the game. Has two main functions, OnTriggerEnter() and OnTriggerExit(). They are called when collider enters and exits the collider attached to the SpatialGridSquare prefab respectively. It checks if the collider belongs to a unit and then adds or removes the unit from its list.
UnitFowController	This script is attached to the units in the game. Each frame it checks if itself has moved from the last frame. If it has, it tells the FowManager to update. Contains an ID int of which FowCalcArchetype it should use.
FowManager	A singleton class. Controls the flow of information for the fog of war. It checks on a configurable interval if any units have told it to update. It calculates the fow of the units that told it to update by separating them into archetypes and using the FowCalcArchetype class.
FowCalcArchetype	A unique set of a view distance and a maximum number of resources/walls it can see through. When created, it calculates the edges of this archetype's view in relation to the unit. Its main function is RunJobs, which initializes and schedules FowCalcJob.

FowCalcJob	Not actually a class, but a struct inside FowCalcArchetype. As such, it acts as a private inner class of the aforementioned. This is a Unity Parallel For Job, which means when scheduled, threads allocated by the Unity Jobs system will work on each index of an array separately using this struct. Each index of the array is references a unique unit of this archetype.

4. Documentation of Source Code

Source code documentation is written in markdown and available here: https://github.com/Sindrex/Bachelor086-fow-doc

5. Testing

Testing was done manually through the Unity Editor. There were two types of tests conducted during the development: Function testing and performance testing.

5.1. Function testing

Function testing is to check that the core functions of the system works, that the fog of war is created correctly. This is done by using two test units of two different archetype and moving one or both. The fog of war is observed as it changes depending on unit location and obstacles.

5.2. Performance testing

Performance testing aims to test the performance of the system in a worst case scenario. To do so, 120 test units (120 is the upper limit of units in the game) split into 4 archetypes are used and moved around simultaneously for a duration of 10-20 seconds. This forces the FowManager to update the fog of war and update the view of each unit on each check it does. During this, the Unity Profiler is recording the performance. Generally, a game frame rate of at least 60 fps is considered good, but this needs to be compared to the frame rate when the units are not moving to be able to see the impact of the fog of war. The observer can then identify which parts of the solution that is causing problems and set out to fix it.

6. References

[1] Pineleaf Studio. Dwarfheim [internet]. Pineleaf Studio; 07.02.2020 [updated 07.02.2020; 07.02.2020]. Available from: https://dwarfheim.com