DDA FPS in Unity: Game Development Using Unity

Supervisor: Kemi Ademoye

Programme: MSc Computer Science (AI)

Word Count:

**Abstract**

In this thesis I developed a FPS game developed in Unity, with two Dynamic Difficulty Adjustment strategies, Dynamic Scripting and Reinforcement Learning.

**Table of Contents**

[**Introduction 4**](#_dq0mw3lq4ccj)

[**What is Unity? 5**](#_itjvgeyp3110)

[**What is DDA? 5**](#_texopfcjeofg)

[Dynamic Scripting 5](#_bvgsfe79dhgj)

[Reinforcement Learning 6](#_e4fcoafx4g53)

[**Literature Review 6**](#_n2qwuvhl5ye5)

[**Methodology and Results 6**](#_3io21slyl9xn)

[**Agile & DevOps 7**](#_yy4qut2rhvf6)

[Agile Scrum Methodology 7](#_wv1vr4tzelgx)

1. Introduction
2. Project Management
   1. Google Drive
3. Gameplay & mechanics
   1. Shooting & Pickup
   2. Enemy AI Behaviour
   3. PCG
   4. DDA
      1. Scoring system
4. Developing in Unity
5. Agile/Scrum methodology
   1. Sprints
   2. Backlog
6. Development and Operation (DevOps)
   1. Source Control
   2. Deploying the game
   3. Continuous Integration/Development
7. Game Design/Software Design
   1. Good Practices
   2. Software Design Choices
   3. Software Testing
   4. Performance
      1. Multi-threadingx
      2. Memory management
8. Player Testing
9. Conclusion

Appendix

# Introduction

The project involves developing a Unity game with well defined software design patterns with the main aim including elements of Procedurally Generated Content (PCG) and Dynamic Difficulty Adjustment (DDA) mechanic. Well defined software patterns allow the developers to quickly test and implement new features.

The theme of the game will be a First Person Shooter (FPS) that is based in a dark dungeon environment. The game consists of two stages. The first stage will be a PCG maze and the second stage will be a predefined map layout which contains the FPS & DDA elements.

The project will be developed using the Agile/Scrum methodology.

Topics

* Developing an efficient and modular Unity game
* Procedurally generating mazes
* Dynamic difficulty adjustment
* Agile/Scrum methodology

# What is Unity?

# What is DDA?

## Dynamic Scripting

Dynamic Scripting (TBF) is a form of reinforcement learning which was implemented on a RPG game, Baldur’s Gate. A rulebase was proposed for each type of character (wizard, fighter, etc.) and each character had a script that was updated in real-time composing rules from its specific rulebase. (TBC) - Each rule contains two values, priority value and weight, which are ranked respectively. The script size for each character type is set to a fixed number of rules.

The two pseudocode demonstrated by Dynamic Scripting, “Script Generation” and “Weight Adjustment”, (is responsible for maintaining the core concept of dynamic scripting). “Script Generation” generates a new script for a character type by using a roulette wheel selection of its rules; rules with higher weights are most likely chosen. “Weight Adjustment” updates the adjustment needed for the weights of its selected rules in the current script by computing the fitness of the agents and its team.

The “Calculate Adjustment” function of “Weight Adjustment” pseudocode includes the primary logic of rewarding or penalizing rules by utilizing the agent’s fitness and a specified break-even value.

Though the Dynamic Scripting has been shown to be successful in various situations (TBF), the strategy had to be adapted to fit into the genre of the game. As opposed to various “rules” which have variable weights, there will be 3 parameters tweaked which are health of enemies, damage of enemies and number of enemies.

## Reinforcement Learning

# Literature Review

# Methodology and Results

There are 3 main components to the game’s dynamic difficulty , Player’s Fitness, Enemy Fitness & Dynamic Difficulty Adjustment. The player’s fitness is compared against the fitness of the enemies and depending on the difference between the two, the difficulty is adjusted accordingly.

Player Fitness Function

The player’s fitness equation consists of 5 components, (i) number of kills, (ii) actions per minute, (iii) time elapsed, (iv) hit miss ratio and (v) hits taken.

Enemy Fitness Function

The enemy fitness function takes into account the fitness level of the current enemies and approximates the fitness of previous enemies.

The current fitness is calculated by taking into account the number of live agents and their distance from the player.

The previous fitness involves the calculation of dead agents and the last distance of the agents from the player.

# Agile & DevOps

## Agile Scrum Methodology

For the software development of the project, the scrum methodology was adopted. There were a total of 6 sprints throughout the lifecycle development of the project. The various tasks were named as the Product Backlog which is listed in a priority order in the Scrum spreadsheet and written in a “user story” format. Specific “user stories” were selected as the primary focus of respective sprints based on their priority order.

A backlog item in a sprint further contained a priority level (low, medium & high) of the backlog item against other items in a specific sprint, the number of estimated days to develop the backlog item, sprint cycle number, current status (In Progress/Done) of and user ownership. This allowed a structured and straightforward collaboration tool for the various members of the team.

## Development Operations & CI/CD

Rapid Deployment (TBI) allows us to both quickly test our new build environments and reliably deploy our game. The Continuous Integration (CI) tool used, CirleCI, hooks into the Github repository via a setting and a config file. The config file, which is stored on Git inside the Github repository, specifies two jobs, test and build. The test job ensures the test written in Unity runs and passes. The build job ensures the Unity application successfully builds and runs in the specified environment, WebGL.

Following a push into Github, triggering CirleCI runs the build and test. Upon successful build and tests on CircleCI, a deploy hook is fired which pushes the new compile build onto CloudFlare pages.

The average time to build and test on a CircleCI environment is approximately 2-3 minutes.

The Unity application communicates regularly with an internal Web JSON API, which generates a guest token and stores game statistics as persisted information. The internal Web API is ran on Ruby on Rails with a Postgres database hosted on a DigitalOcean Droplet with the Dokku application sitting as layer on top of the droplet. Dokku is a docker-powered Paas platform that allows us to easily manage applications on a deployed server. The database used is Postgresql.

The game statistics stored on Postgres contains the various states of the game when the game’s difficulty is dynamically adjusted and final scores of players which culminates into a global leaderboard.