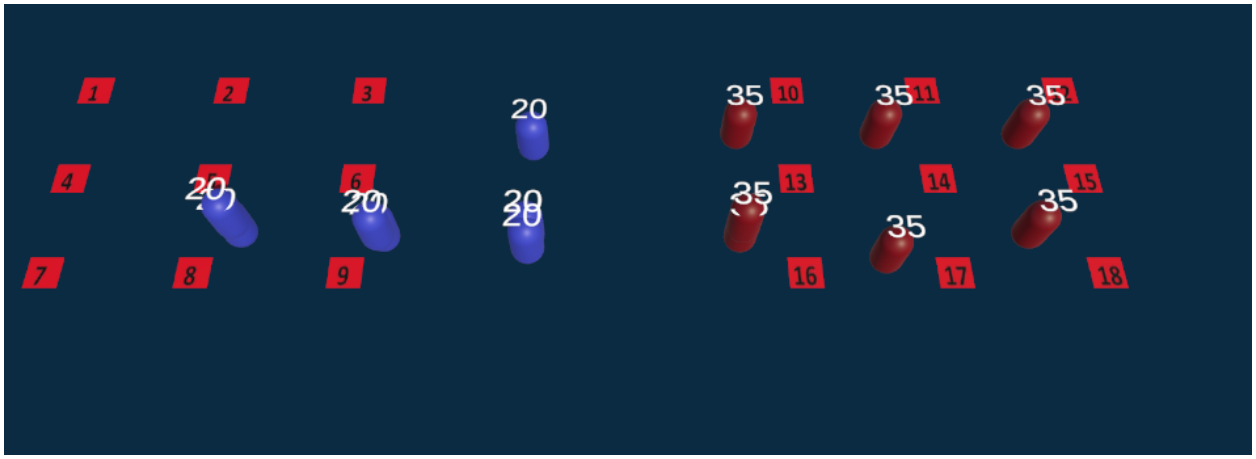
ECS Battle



Technical Documentation & Installation Guide

Introduction

This is the *Technical Documentation and Installation Guide* for ECS Battle. This is a small Unity demo that demonstrates working with Unity DOTS and ECS. It is a battle simulator where two teams battle. Unit attributes and properties are configured in a JSON configuration file.

Installation Guide

Installation Guide - Unity Project

- Clone the repository to your Windows/Mac machine using the following link: https://github.com/RamiB1234/ECS-Battle
- Download **Unity 2021.3.7f1 LTS** using Unity Hub
- Add and locate the unity project from the downloaded repository. Path of the project folder is ECS-Battle\unity_project\ECSBattle
- Launch the project from the Unity Hub projects tab

Installation Guide - Android APK

- Clone the repository to your Windows/Mac machine using the following link: https://github.com/RamiB1234/ECS-Battle
- Copy the binary file called **ECSBattle.apk** from ECS-Battle\binary\android to your Android device
- Install the apk and run the game

Technical Documentation

Configuring Team Properties

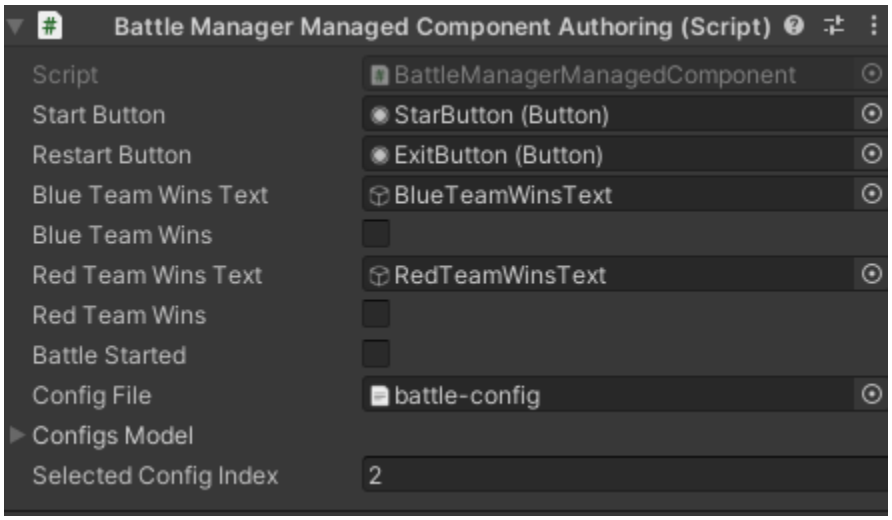
Each team has a 3x3 grid and each grid cell acts as a spawning point for units.

Cells are numbered from 1 to 18. You will use these cell numbers to configure whether or not the cell should spawn a unit. To configure team properties, open the file named **battle-config.json**

```
{
    "Configs": [
        {
            "TeamAProperties":{
                "HP": 100,
                "Attack": 10,
                "AttackSpeed": 2,
                "AttackRange": 1.5,
                "MovementSpeed": 3
            },
            "TeamBProperties":{
                "HP": 150,
                "Attack": 20,
                "AttackSpeed": 1,
                "AttackRange": 4,
                "MovementSpeed": 1
            },
            "Units":[1,2,3,10,14,16]
        },
        .
        .
        .
    ]
}
```

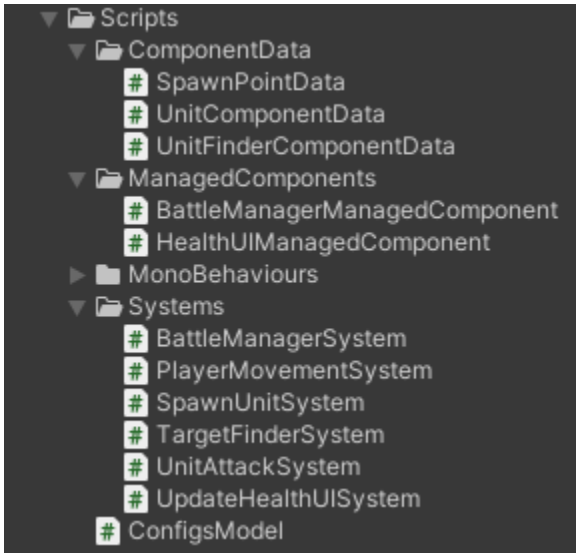
You can configure attributes like HP and Attack Speed for each team. You can also use grid cell numbers to spawn units in those cells. You can define an array of configurations. You can easily add/remove elements or modify attributes.

After finishing editing the configurations, open the **EmptyECS** scene and click on **BattleManager** gameObject in the Hierarchy Window. Check the inspector window:



Pay attention to the last field “Selected Config Index”. This is the index of the configuration element that we defined in the configuration json file. Please note that the index is zero based.

Script Folder Structure



Folder	Script Files	Description
Root of Scripts	- ConfigsModel	This class is used as a model object for deserializing the json config file
ComponentData	- SpawnPointData - UnitComponentData - UnitFinderComponentData	These are component data for the spawn point entity and for the unit entity
ManagedComponents*	- BattleManagerManagedComponent - HealthUIManagedComponent	These are managed component for the battle manager and the health UI entities
MonoBehaviours	- ExitButton - StartButton - StartButtonMenu	These are scripts to handle button clicked
Systems	- BattleManagerSystem - PlayerMovementSystem - SpawnUnitSystem - TargetFinderSystem - UnitAttackSystem - UpdateHealthUISystem	These are all the systems that operate on our data components

*Managed components are classes not struct like component data and they can hold any monobehaviour references like UI elements or references to game objects

ECS

Below are all entities and their corresponding data component and their related systems

Entity	Components	Systems
Unit	- UnitComponentData - UnitFinderComponentData	- PlayerMovementSystem - TargetFinderSystem - UnitAttackSystem
Spawn Point	- SpawnPointData	- SpawnUnitSystem
Health UI(managed)	- HealthUIManagedComponent	- UpdateHealthUISystem
Battle Manager(managed)	- BattleManagerManagedComponent	- BattleManagerSystem

Features

Feature	Status	Remarks
All units can see every unit in the game, friendly and enemy both	**Done**	
Each unit will choose a random target from the enemy and start the movement towards the acquired target	**Done**	
The unit will move toward that target until it comes into attack range.	**Done**	
The unit will start the attack and do damage based on the attack speed and attack	**Done**	

It keeps on doing damage until the unit dies, and then it will acquire another random target until all the enemy unit dies	**Done**	
The initial setup of the team is a 3X3 grid where you can define which grid position a unit is on. One grid can only hold one unit, and each unit can only take one grid slot	**Done**	
You should be able to change all the unit properties and the initial setup	**Done**	All unit properties and unit numbers are configured in battle-config.json
The right side is the enemy team and is also read from the config file	**Done**	All unit properties and unit numbers are configured in battle-config.json
You can also change the enemy team by clicking on Team 1, Team 2, team 3, etc	**Not implemented**	Due to time constraint, I couldn't implement the preset config button list, the only way to change teams by manually modifying battle-config.json
The list is also read from the config file, and we should be able to add new teams, and the list should adapt to that	**Done**	You can add/remove config element from the list in battle-config.json and change the selected index in the battle manager object
The Start battle should hide the UI, and the battle will begin	**Done**	
Each unit has the HP number of top and should decrease as they receive damage	**Done**	

THE UNIT WILL DISAPPEAR when HP is zero; you don't have to add any transitions or death effects	**Done**	
After the battle ends, it should display a victory screen and the option to go to the main menu to start another battle.	**Done**	
You are required to use the ECS pattern to implement the above game	**Done**	
It's optional to use Unity ECS, but it's highly encouraged.	**Done**	I have used unity ECS
You can use scriptable objects, JSON, or XML to implement configurations	**Done**	Configurations are defined in JSON format

Conclusion

That was a little fun project to make in one week. I have learned a ton of new skills as this is the first time I work with *data oriented approach* rather than *object oriented*. I had a slight learning curve specially in the first couple of days but now I'm confident in my skills in ECS.

I'm sure that I haven't implemented everything in the most optimized way, but due to the time constraint and the limited documentation and resources available for unity ECS as it's still in preview, I did my best to implement all features the best way possible. I had to implement *UI button click events* in the good old *MonoBehaviour*, I'm not sure if that's possible yet using pure ECS.

Thanks for reading the document.