Primary Objective

Pure ECS DOTS implementation is not practically possible yet with continuous updates being released from unity. This project Leverage the benefits of the Entity Component System (ECS) architecture to achieve optimal performance and scalability

By utilizing the ECS pattern, I aimed to separate the data (components) from the behavior (systems) to enable efficient processing and manipulation of entities.

Furthermore, the project incorporated Unity's Jobs and Burst Compiler packages to maximize performance. By utilizing Jobs, we were able to leverage multi-threading capabilities and execute code in parallel, taking advantage of the available CPU resources and achieving significant performance gains.

In terms of the overall project design, I aimed to provide a solid foundation for the Asteroids game. The separation of concerns, facilitated by ECS, allowed for clear and modular implementation of different game systems such as player control, asteroid movement, collision detection, and scoring.

By dividing these functionalities into distinct systems and components, I achieved a more maintainable and extensible codebase. The project's modular structure and guidelines allows developers to add new features, create additional scenes, or modify existing systems to tailor the game to their specific needs

Implementation Details

Note: All the Data Types / Objects of Components are defined in a file called Components.cs

Player

Data Components

PlayerComponent : Handles type of missile player ship should be shooting and using protection shield to become immune to asteroids and alien ship's attacks

MissileComponent: System will detect this entity type while executing code logic

Systems

PlayerHitSystem: Detects when player is hit by asteroid or alien ship (collision detection system)
Considering Player power ups and game state It updates the Player health and score through events

PlayerMoveSystem: Handles player ship movements using new input system of unity

PlayerShieldSystem: Enables and Disables player shield upon receiving events

HyperSpaceSystem: It is the ability of a player ship to loop through x and y axis movement. If the player is moving forward in positive x and on moving outside the screen bound the position of player will be set to opposite side of the screen bound

Alien Ships And Asteroids

Data Components

EnemySpawnerComponent: Holds different types of alien ship and asteroid enemy prefabs and respective data

Systems

EnemySpawnerSystem: Handles the enemy spawning logic, where type of enemy is randomly picked and instantiated at range of random locations to move in different directions

ExplosionSystem: Instantiates explosion effect prefab upon detecting collision between player missile and enemy given position as input

Power ups

Data Components

PowerUpSpawnerComponent: Holds different types of power up prefabs and respective data

Systems

PowerUpSpawnerSystem: Handled the game logic to instantiate and destroy power prefabs with randomness included and for every x seconds after initial play time of 1-2 minutes

Game Play

Data Components

GameStateComponent: Using the concept of State Pattern here to define and update game state dynamically and act accordingly

Systems

GameStateUISystem: This handles the UI components and entities visibility while Game is in progress, Start or End states

RestartSystem: Responsible for resetting all the parameters to start the game fresh

Displaying Text / Images

Use Case Where We Need to Integrate Mono Behaviors

To implement score display UI and Health display UI I have used Observer pattern by defining few Events which will update score and health of player ship

This made the project an hybrid framework DOTS and Unity’s Default Game Objects work together