PROJECT BRIEF

Author: Pablo Sforsini

Date: September 8th, 2021

Version: 1.0

1) Introduction

This project was made as a mvp that required the development of a simple game prototype inspired by the classic Asteroids game.

This prototype does not include vfx, sfx, game visual indicators (like shield equipped, power up cool time, hit indicator, etc.) or any other visual design aspects such as good UI for proper user experience.

2) Game architecture

Since the game requires the use of DOTS, the prototype was built around Data Components and System Components which interacts which each other.

For further information about the entities and components used in the project, please see the **Entity-Component Relationship.pdf file.**

Despite being a DOTS Project, 3 Monobehaviours are used:

* GameManager: A simple Game Manager that tracks the Game Over condition for closing the World for reloading the scene.
* UIManager: A simple UI Manager that shows the player lives and a restart button when the game is over.
* Entities Spawner: Responsible for creating entities from game object prefabs and spawn them in the screen.

You will find other Monobehaviours in the project’s script folder, but they were used for testing purposes (I will keep them there just in case 😊 )

Systems used and their responsibilities (Alphabetical order):

* AsteroidDirectionAsignmentSystem : System that generates random directions to the new Asteroids that are spawned after a bigger one is hit (“Child Asteroids” or the fragments of a hit asteriod).
* AsteroidHitSystem: Systems that creates the “Child Asteroids” when the parent is hit.
* ChangeDirectionSystem: Allows the Enemy Ship to change direction over a period of time.
* CollisionDetectionSystem: System responsible for detecting entities collisions using the Unity Physics package.
* DestroySystem: Responsible for destroying entities in game.
* EnemyHitSystem: System responsible for actions when Enemy Spaceships are hit.
* HyperSpaceSystem: Responsible for translating the player ship to any point in the screen.
* InputSystem: System responsible for capturing the player input data.
* LaserTraversalPowerUpSystem: System responsible for creating a shot variation (required mechanism in the test) when the proper Powerup is picked up. This system allows the lasers to appear on the other side of the screen for a limit period of time instead of being destroyed outbounds.
* LinearMovingSystem: Allows all entities with a Force Component to move in a linear fashion.
* PlayerShipHitSystem: System that manages the player ship hit event, sending events to the GameManager and the UI for updating lives display and game over conditions.
* PowerUpInScreenSystem: Controls when a Powerup should disappear if it is not picked up.
* RotationSystem: Responsible for the rotation of entities (Asteroids).
* ScreenTraversalSystem: System that allows entities to traverse or not the screen when they are outbounds.
* ShieldSystem : Responsible for allowing the player to avoid death after a collision for a period of time. As aforementioned, despite the mechanism being implemented, there is no visual or UI indicator.
* SpaceshipMovingSystem: Responsible for moving the player ship.
* SpaceShipShootingSystem: System responsible for instantiating lasers.
* TriggerDetectionSystem: System responsible for detecting entities colliders triggers using the Unity Physics package (Powerups and player)
* VulnerabilitySystem: System that allows the player to avoid being hit temporarily when the ship is positioned in the center of the screen after being hit and a life is lost. As aforementioned, there is no visual indicator.

3) Game Controls:

* Ship Movement: ASWD Keys / Arrow Keys.
* Ship Rotation: Mouse
* Shooting: Left Mouse Button.
* Hyperspace: Right Mouse Button.