Asteroid Documentation

Project Overview

A. Introduction

Asteroid 2d is a top down arcade-style endless survival game inspired by the classic video game Asteroids. The game features a player-controlled spaceship that can rotate, accelerate and shoot projectiles in order to destroy incoming asteroids and enemies. The objective of the game is to survive as long as possible and obtain a high score. The game becomes progressively more challenging as the number of asteroids and enemies increases. Players are also awarded with various power ups throughout the gameplay which adds an extra layer of excitement and challenge to the game thus increasing the chances of survival.

B. Game Design

a. Game Flow & Working

i. Main menu scene with high score
ii. On play, load game scene
iii. Play cutscene where ship arrives from left with camera zoomed in ,then camera pans out with asteroid coming from right, wait for user input to begin game. (Implemented using Unity Timeline Tool)
iv. Spawn Asteroids and Enemies with defined/random properties.
v. Spawn Power ups with defined/random probabilities.
vi. Players must dodge & destroy targets while collecting power ups to survive as long as possible and obtain high score.
vii. Users can pause & resume game through the pause menu.
viii. Game ends when the player runs out of health/lives.
ix. Players are prompted with a game over panel displaying current score and options to restart or quit the game.

b. GamePlay

i. UI Controls -
   1. Escape key - Pause/ Resume/ Restart/ Go to Menu/ Quit game.

ii. Ship Controls -
   1. Up Arrow key - Accelerate & Move forward
   2. Left/Right Arrow keys - Rotate ship in left / right direction
   3. Spacebar - Shoot weapon

c. Game Rules & Mechanics

i. User Stories

1. Player controls the ship with arrow keys. Designers can tweak ship controls.
2. Player ship and Asteroids are clamped within the camera bounds to ensure a smooth and uninterrupted gameplay.
3. Player ship has three properties: health, fuel, lives/chances.
4. Health will deplete based on collisions and damage taken.
5. Fuel will deplete as per forward movement (not while turning).

6. Lives/chances are the amount of chances a player can get to continue. On health reaching zero one life is reduced.
7. Pressing the spacebar allows you to shoot projectiles with the default weapon which shoots 3 bullets in burst mode.
8. Asteroids are spawned with defined/random properties with default set to 2 count and the count increases as the player scores more points making the game challenging.
9. Large Asteroids are splitted into two smaller ones until min split value is reached, each sharing half the size/damage to player /points to player.
10. Destroying the smallest parts of the asteroid will score points.
11. Enemies are spawned with defined/random properties with default set to 1 count and the count increases with difficulty.
12. Spawned enemies move constantly around the play area while attacking the player.
13. Player ship takes damage from collisions between asteroids & enemies and enemy bullets.
14. Power ups are spawned with time delays during gameplay as well as when asteroids are destroyed with certain probabilities with designer defined properties which will hover around the play area and expire with time limit if not collected.
15. Implemented Powerups (6 powerups)
    a. Health drop - Permanent effect powerup which will add health to the player ship with defined value.
    b. Fuel drop - Permanent effect powerup which will add fuel to the player ship with defined value.
    c. Live drop - Permanent effect powerup which will add Extra Life to the player ship with defined value.
    d. Crescent moon weapon - Temporary effect powerup which will allow the player to shoot moon shaped bullets with designer defined values.

    e. Spread weapon - Temporary effect powerup which will allow the player to shoot Multiple bullets in circular pattern with designer defined values.
    f. Shield - Temporary effect powerup which will allow the player to avoid damage taken from asteroids/enemies with designer defined values.

16. As the player scores more points and gets accustomed to the gameplay the difficulty increases with increase in the number of asteroids and enemies spawning.
17. Pause menu can be assessed by pressing the escape key, which will pause the game and display the following options - Resume, Menu, Quit, Restart.
18. On Player reaching zero health and lives, the game will end and the player will be prompted with a game over panel with current score and options to play again or quit the game.

ii. Designer Stories

1. Designer tweakable Scriptable Configuration Files

a. Player Ship Config -

- i. max player health
- ii. max player fuel
- iii. fuel depletion rate
- iv. max player lives
- v. ship speed
- vi. ship rotation speed.

b. Asteroid Config -

- i. Asteroids points (distributed among childs)
- ii. Max Asteroid Size
- iii. Max Asteroid Hits/Steps to Destroy
- iv. Min Size till the asteroid can be splitted.
- v. Move Speed of asteroid
- vi. Damage to player on collision (scales with size)
- vii. Spawn Distance from camera viewpoint

viii. Trajectory Variance
ix. Designers can also select random properties which will set the size and speed of asteroids randomly, rest will scale as per size.
x. Asteroid image are randomly selected from list of sprites

c. Enemy Config -
   i. Points on killing enemy
   ii. Max enemy hit points
   iii. Enemy damage to player on collision
   iv. Time between next fire
   v. Enemy move speed
   vi. Spawn distance from camera viewpoint
   vii. Designers can also select random properties which will set the above properties randomly.

d. Power Up Config -
   i. Power spawn probability (This is a cumulative probability so adjust accordingly)
   ii. How long power up stays alive in the play area
   iii. Power hover speed
   iv. Power up effect type Permanent/Temporary
   v. Power up effect alive time (only for temporary power ups)

e. Weapon Type Config -
   i. Weapon image
   ii. Number of bullets to fire
   iii. Delay between next shot fire
   iv. Bullet speed
   v. Fire rate for weapon
   vi. Damage on collision/hit

2. Designer tweakable Gameplay Settings

a. Asteroid Spawner
   i. Maximum number of asteroids to spawn at a given time, starts with low count and increases as per difficulty.

ii. List of config files for asteroids , can add new SO files here. These are randomly selected while spawning.

b. Enemy Spawner

i. Maximum number of enemies to spawn at a given time, starts with low count and increases as per difficulty.

ii. List of config files for enemies, can add new SO files here. These are randomly selected while spawning.

c. Power Ups Spawner

i. List of Config files for powerups which contain spawn properties, add new SO files here for new power ups.

ii. Time delay at which new power up will spawn.

iii. Power up spawn probability, This probability is for spawning power ups on destroying asteroids.

d. Difficulty Settings

i. Asteroid Spawn Threshold - Set the score threshold at which the maximum no. of asteroids will increase. (ex. 50, so every 100 points scored max no of asteroid will increase)

ii. Asteroid Increase Count - Count at which the number of asteroids will increase every threshold. (default 2)

iii. Enemy Spawn Threshold - Set the score threshold at which the maximum no. of enemies will increase. (ex. 50, so every 100 points scored max no of enemies will increase)

iv. Enemy Increase Count - Count at which the number of enemies will increase every threshold. (default 1)

C. Architecture & Principles

a. Single Responsibility Principle

i. Implemented SRP, which ensures that each function in a class is responsible for a single task or responsibility. For example, the player input class only handles input-related tasks, player stats only handles stats-related tasks, and so on.

b. Open Closed Principle

   i. Implemented Scriptable Objects to store and expose tweakable variables to designers. This separates data from code, which allows to adjust the variables without requiring to change code logic.

c. Interface Segregation Principle

   i. Implemented ISP, where the collision between projectiles and target checks for any object implementing the “damageable enemy” interface to damage them. This promotes flexibility and modularity in the code by not forcing dependency on specific tags or object types.

d. Singleton Pattern

   i. Game Manager - Game manager object has only one instance through the game and it holds necessary objects references and is responsible for handling game state and difficulty settings.

   ii. Audio Manager - Audio manager object has only one instance through the game is responsible for handling all the sfx/music control settings such as audio source, clips references, volume, pitch, loop etc.

e. Observer Pattern

   i. Game State events are observed among multiple gameobjects to keep them decoupled from one another.

   ii. Events like player health bar, player fuel bar, player lives, player score are observed in UI. which keeps the game logic decoupled from the UI updates.

   iii. Events like Asteroid destroyed, Enemy Destroyed, Power up status are observed in respective spawner classes.

f. Object Pooling Pattern

   i. Following object prefabs are pooled for better reusability and performance efficiency.

      1. Bullet prefabs
      2. Asteroid prefabs
      3. Enemy prefabs
      4. Explosion particle effect

        5. Power up prefabs

g. Factory Design Pattern
    i. The Factory design pattern provides abstraction to the power up base class, making it easy to scale the power up system and add new power ups in future independently.
    ii. The Weapon system implements the factory pattern using an abstract base class for weapons to provide a common interface for creating new weapons with shared properties and methods.

h. Strategy Design Pattern
    i. The strategy pattern is used to create a flexible and scalable weapon system in which a delegate points to different fire methods. When powerup is picked up, the delegate is updated to point to a different method, allowing for easy modification of the weapon's behavior without changing the underlying code.

D. Testing Techniques Followed

a. Smoke Testing - Followed smoke test after every build to identify critical issues at early stage and move forward with advanced techniques.

b. Unity Testing - Tested individual units/components of the game to ensure that they perform as expected.

c. Integration Testing - Tested the integration of various units/components of the game to ensure that they perform as expected.

d. System Testing - Tested the game as a whole to ensure that it meets the desired System specifications and requirements.

Note :

- Please find the Project Archive File which includes the Asset/Packages/ Project settings. (Application Build contains the executable files for the game).
- The Project is implemented in Unity Version 2020.3.13f1 LTS.
- All other requirements mentioned in the Assignment are also implemented.