SWE 402 - Game Development

Assignment 1: Sprite Flight - 2D Game

Overview

In this assignment, you will complete the Unity Learn "2D Beginner Game: Sprite Flight" course to build a complete 2D arcade game from scratch. You will create a triangle ship that players control with mouse clicks to dodge flying obstacles while their score increases over time.

Course Link: https://learn.unity.com/course/2d-beginner-game-sprite-flight

Section 1: Project Setup & 2D Game World

Complete the tutorial: "Set up your 2D game world"

Objectives:

- Create a new Unity project using the Universal 2D template
- Set up the 2 by 3 Editor layout for efficient 2D development
- Navigate the 2D Scene view using pan and zoom
- Add a hexagon obstacle sprite to the scene at position (0, 0, 0)
- Configure the Main Camera size and background color
- Create four screen borders using square sprites
- Lock the aspect ratio to 16:9 and adjust borders accordingly
- Organize the Hierarchy with a "Borders" parent GameObject and create an Assets folder structure

Section 2: Physics & Prefabs

Complete the tutorial: "Make a flying obstacle prefab"

Objectives:

- Add a Rigidbody 2D component to the obstacle
- Add Polygon Collider 2D to the obstacle and Box Collider 2D to all borders
- Create a 2D Physics Material with Bounciness = 1 and Friction = 0
- Apply the physics material to the obstacle
- Test zero gravity behavior (Gravity Scale = 0)
- Convert the obstacle into a reusable prefab
- Create a "Prefabs" folder and organize your project structure

Section 3: Obstacle Scripting

Complete the tutorial: "Random obstacle size, direction, and speed"

Objectives:

- Create a C# script named "Obstacle" and attach it to the prefab

- Implement randomized obstacle size using Random.Range() and transform.localScale
- Create public variables for minSize, maxSize, minSpeed, and maxSpeed
- Use GetComponent<Rigidbody2D>() to access the physics body
- Apply random force using AddForce() with randomized direction
- Use Vector2.normalized to ensure consistent direction calculations
- Expose variables in the Inspector for runtime tuning
- Place multiple obstacle instances in the scene for testing

Section 4: Player Controls

Complete the tutorial: "Steer the player"

Objectives:

- Create a multi-part Player ship using grouped 2D sprites (triangles, squares)
- Add Rigidbody 2D (Gravity Scale = 0) to the parent Player GameObject
- Add compound colliders (Polygon Collider 2D) to each child sprite
- Create a PlayerController script with mouse input detection
- Convert mouse screen position to world space using Camera.main.ScreenToWorldPoint()
- Rotate the player to face the mouse position
- Apply thrust force in the direction the player is facing
- Implement OnCollisionEnter2D() to destroy the player on collision

Section 5: Scoring System

Complete the tutorial: "Add a scoring system"

Objectives:

- Track elapsed survival time using Time.deltaTime in the Update() method
- Create a score multiplier variable to calculate the final score
- Use Mathf.FloorToInt() to display clean whole number scores
- Create a UI Document GameObject using UI Toolkit
- Add and style a score Label element in the UI Builder
- Get a reference to the UIDocument component in your script
- Update the score label text dynamically during gameplay
- Configure Panel Settings for screen size scaling

Section 6: Game Over Effects & Restart

Complete the tutorial: "Restart the game with a bang"

Objectives:

- Create an explosion Particle System effect
- Configure Emission, Shape, and randomization settings for the explosion
- Convert the explosion effect into a prefab
- Instantiate the explosion prefab from script when the player is destroyed

- Create a styled Restart button using UI Toolkit
- Hide the Restart button at game start
- Show the Restart button when the player collides
- Implement scene reload functionality using SceneManager.LoadScene()

Section 7: Build & Publish

Complete the exercise: "Build and publish to web"

Objectives:

- Configure scenes in the Build Settings
- Switch the build platform to WebGL
- Build your project for web deployment
- Publish your game to Unity Play
- Test your published game in a web browser

Bonus Features (Required: Choose at least 2)

Implement at least TWO of the following bonus features to personalize your game and demonstrate additional Unity skills:

- ☐ **Change Entire Game Concept:** Re-theme your game with a completely new concept (e.g., mouse escaping cats, alien dodging meteors) using simple 2D shapes
- ☐ **Swap Out Sprites:** Replace default shapes with custom sprites or assets from Unity Asset Store, OpenGameArt.org, or Kenney.nl
- ☐ **Increase Difficulty Over Time:** Adjust physics material bounciness above 1.0 to gradually increase obstacle speed; optionally clamp maximum velocity
- ☐ **Add Sound Effects & Background Music:** Add Audio Source components with looping background music and explosion sound effects
- ☐ **Ambient Background Particles:** Create a background particle system (stars, dust, sparkles) with randomized velocity and fade effects
- ☐ **Animated Booster with Graphics:** Add a visual booster flame that activates/deactivates on mouse press/release with accompanying thruster audio
- ☐ **Make Game Compatible with Mobile:** Replace mouse input with InputAction system using Pointer bindings for touch support; test in Simulator view

Resources

Unity Learn Course: 2D Beginner Game: Sprite Flight

Unity Documentation: https://docs.unity3d.com/

Unity Play (for publishing): https://play.unity.com/

Free Assets: Unity Asset Store, OpenGameArt.org, Kenney.nl