Mini-Project 1: 2D Sprite Flight Game

Lucas P. Cordova, Ph.D.

Table of contents

1 Overview

For this project, you'll build a 2D game in Unity — starting from a completely blank project with no assets provided, you'll create the entire game from start to finish. In the game, you fly a simple triangle-shaped ship using mouse clicks (or screen taps on mobile). Your goal: stay alive for as long as you can by dodging flying obstacles while your score increases over time.

Along the way, you'll learn the fundamentals of working in 2D, from navigating the Scene view to coding simple gameplay interactions. You'll also explore key Unity systems like a user interface (UI), particle effects, audio, and publishing. By the end, you'll not only have a complete, playable game, but you'll also have the skills to start building your own original 2D projects.

2 Game Interface Preview

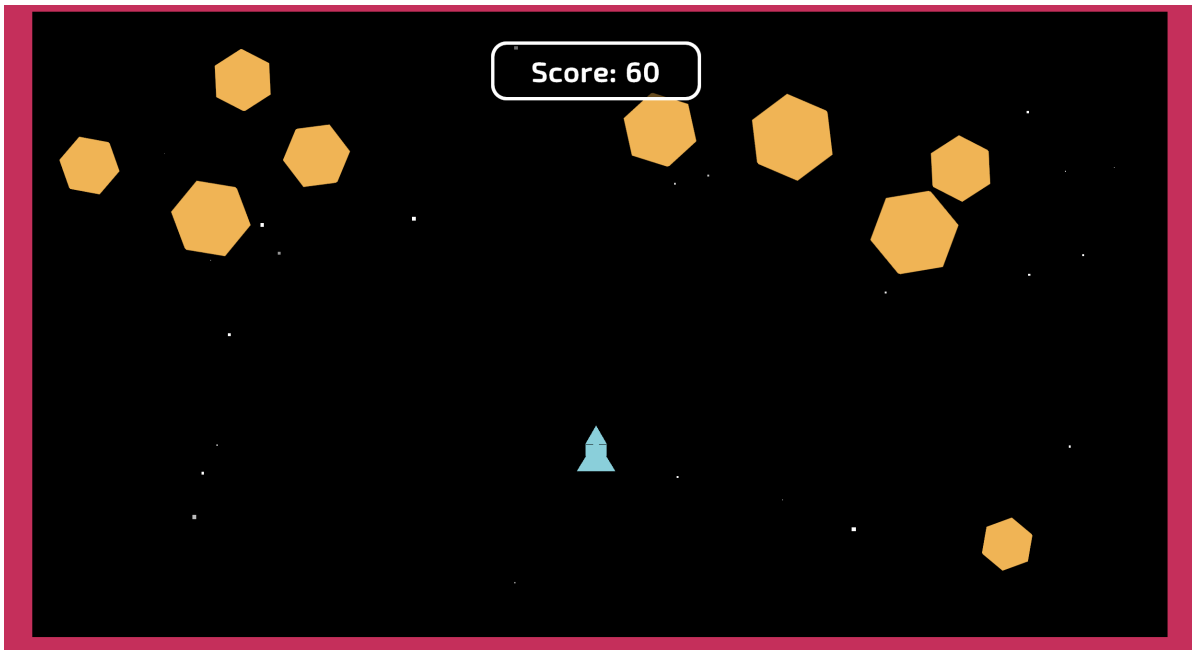

Figure 1: Game Interface Preview

3 Learning Objectives

- Implement 2D movement and physics in Unity
- Create prefabs and manage GameObjects
- Implement collision detection and game states
- Work with Unity's UI system for score display
- Extend a base game with custom features
- Document technical work professionally

4 Grading Rubric

5 Total Points: 100

Component	Points	Criteria
Base Game Implementation	**70**	

Component	Points	Criteria
Player Movement	15	- Ship moves correctly with mouse/touch input- Ship rotates to face cursor- Movement feels responsive
Obstacle System	15	- Obstacles spawn randomly- Physics/bouncing works correctly- Proper prefab implementation
Collision Detection	15	- Collisions properly detected- Game ends on collision- Explosion effect triggers
UI & Score System	15	- Score increments properly- UI displays correctly- Game over screen functional
Core Game Loop	10	- Game starts properly- Can restart after game over- No major bugs in flow
Extensions	**20**	
Extension Implementation	20	- Minimum 20 points from extensions- Features work as described- No breaking of base game
Documentation	**10**	
README Quality	10	See detailed rubric below

6 Documentation Rubric (10 points)

Aspect	Excellent (2 pts)	Good (1 pt)	Needs Work (0 pts)
Game Overview & Link	Unity Play link works, clear description, complete controls/instructions	Basic description and link provided	Missing link or inadequate description
Base Implementation Status	Thorough status checklist, all bugs documented, limitations explained	Basic status provided, some bugs noted	Incomplete or missing status

Aspect	Excellent (2 pts)	Good (1 pt)	Needs Work (0 pts)
Extensions Documentation	Each extension fully documented with implementation details, impact, and issues	Extensions listed with basic descriptions	Extensions poorly documented or missing
Technical Writing	Professional tone, well-organized, proper markdown formatting, no spelling/grammar errors	Generally clear with minor issues	Difficult to read or understand
Credits & Reflection	All assets credited, thoughtful reflection on learning and challenges	Basic credits and reflection	Missing credits or reflection

7 Extension Point Values

Extension	Points	Description
Cohesive Color Scheme	2	Unified visual theme using color palette
Change Game Concept	3	New theme while keeping mechanics
Swap Sprites	3	Replace shapes with custom/free sprites
Destroy Borders on Game Over	4	Obstacles fly off-screen when game ends
Ambient Background Particles	4	Atmospheric particle effects
Increase Difficulty Over Time	5	Progressive difficulty scaling
Sound Effects & Background Music	5	Audio feedback and ambiance
Animate Booster with Audio	6	Visual and audio thrust feedback

8 Grade Calculation Examples

Minimum Passing (70%) - Base Game: 70 points (all features working) - Extensions: 0 points (none implemented) - Documentation: 0 points (missing/inadequate) - Total: 70/100 = C-

Good Submission (85%) - Base Game: 65 points (minor bugs) - Extensions: 15 points (implemented 3-4 extensions) - Documentation: 5 points (adequate documentation) - Total: 85/100 = B

Excellent Submission (95%+) - Base Game: 70 points (polished, no bugs) - Extensions: 20+ points (creative implementations) - Documentation: 10 points (professional quality) - Total: 100/100 = A+

9 Part 1: Base Game Requirements (70%)

10 Complete the Unity Tutorial Series

Follow the complete "2D Beginner Game: Sprite Flight" tutorial course on Unity Learn. This includes:

1. Setting up the project and player movement
2. Creating obstacle prefabs with physics
3. Implementing collision detection and game over state
4. Adding UI for score tracking
5. Creating an obstacle spawning system

The base game must be fully functional with:

- Player-controlled ship movement
- Randomly spawning obstacles
- Collision detection that ends the game
- Score tracking system
- Game over UI

11 Part 2: Extensions (20%)

Choose extensions from the list below to earn up to 20 points. You may implement any combination that totals at least 20 points.

12 Available Extensions 1-4

12.1 1. Create a Cohesive Color Scheme (2 points)

Use an online color palette generator to create a unified visual theme. Apply consistent colors to your player, obstacles, background, and UI elements using hex codes.

12.2 2. Change Your Entire Game Concept (3 points)

Transform the theme while keeping the mechanics. Examples: mouse escaping cats, alien dodging meteors, or pizza slice flying through space. Use Unity's 2D primitives (triangles, rectangles, capsules) to create new visuals.

12.3 3. Swap Out Your Sprites (3 points)

Replace default shapes with custom sprites or free assets from Unity Asset Store, OpenGameArt.org, or Kenney.nl. Add sprites as child GameObjects and properly configure sprite renderers.

12.4 4. Destroy the Borders on Game Over (4 points)

Modify the PlayerController script to disable screen borders when the player dies, allowing obstacles to fly off-screen. Requires creating a reference to the Borders GameObject and calling `SetActive(false)` on collision.

13 Available Extensions 5-8

13.1 5. Add Ambient Background Particles (4 points)

Create a particle system for atmospheric effects (stars, dust, sparkles). Configure emission area, particle size, lifetime, and velocity to create subtle background motion.

13.2 6. Increase Difficulty Over Time (5 points)

Make the game progressively harder by adjusting obstacle physics. Set the Physics Material's bounciness slightly above 1.0 (e.g., 1.05) so obstacles gain speed with each bounce. Optionally clamp maximum velocity in code.

13.3 7. Add Sound Effects and Background Music (5 points)

Implement audio feedback including:

- Looping background music at reduced volume
- Explosion sound effect on player death
- Audio Source components properly configured
- Free audio resources from Unity Asset Store or Freesound.org

13.4 8. Animate the Booster Graphic with Audio (6 points)

Create responsive controls with visual and audio feedback:

- Add booster flame GameObject that activates on input
- Implement code to show/hide booster on button press/release
- Add looping thruster sound effect
- Optional: animate the flame with scaling or flickering effects

14 Extension Combination Examples

Example 1: Visual Overhaul (20 points) - Color scheme (2) + Change concept (3) + Swap sprites (3) + Particles (4) + Sound effects (5) + Change concept (3) = 20 points

Example 2: Gameplay Enhancement (21 points) - Destroy borders (4) + Difficulty scaling (5) + Sound effects (5) + Animated booster (6) = 20 points

Example 3: Complete Package (24 points) - Color scheme (2) + Swap sprites (3) + Particles (4) + Sound effects (5) + Difficulty (5) + Borders (4) = 23 points

15 Part 3: Documentation (10%)

Your README.md must demonstrate professional technical writing and thorough documentation. This is worth 10% of your grade and will be evaluated on completeness, clarity, and professionalism.

16 Submission Requirements

16.1 1. GitHub Repository Setup

1. Clone the provided repository
2. Create your Unity project in the cloned directory
3. Commit your work regularly with meaningful commit messages
4. Include an appropriate .gitignore file for Unity projects

16.2 2. Unity Play Publishing

1. Build your game for WebGL following the Unity tutorial instructions
2. Use the "Share Your Project" feature to publish to Unity Play (Unity's free web hosting platform)
3. Select the **public link option** when publishing
    - This creates portfolio pieces visible to potential employers
    - If you prefer not to make it public, please see the instructor for alternatives
4. Test your published game to ensure it works correctly in a web browser

16.3 3. GitHub Release

1. Create a GitHub Release/Tag when your project is complete
    - Use version naming like `v1.0`, `v1.1` for iterations
    - Your final submission should be the latest release
2. Include in your release notes:
    - The Unity Play URL for your game
    - Summary of game and features
    - Quick list of completed extensions with point totals
    - Any special instructions or known issues

16.4 4. README.md Documentation

Create a comprehensive README.md file in your repository root with the following sections:

17 Required README Sections

17.1 1. Game Overview

- Game title and brief description
- **Unity Play Link**: [Your game URL]
- How to play instructions
- Controls explanation

17.2 2. Base Game Implementation

- **Completion Status**: Confirm which tutorial sections were completed
- **Known Bugs**: List any bugs or issues in the base implementation
- **Limitations**: Describe any features that don't work as expected
- **Deviations**: Note any intentional changes from the tutorial

17.3 3. Extensions Documentation

For each extension implemented, include:

- **Extension Name and Point Value**
- **Implementation Description**: How you implemented the feature
- **Game Impact**: How this extension changes the gameplay experience
- **Technical Details**: Key code changes or Unity configurations
- **Known Issues**: Any bugs or limitations specific to this extension
- **Assets Used**: Credit any external resources (sprites, sounds, etc.)

17.4 4. Reflection

- Total points claimed (base + extensions)
- Most challenging aspect of the project
- What you learned from this project

18 README.md Template

```
# [Your Game Title]

## Play the Game
**Unity Play Link**: [Your Unity Play URL]

## Game Overview
[Brief description of your game and its objective]

### Controls
- [List controls here]

### How to Play
[Instructions for playing the game]

```

```
## Base Game Implementation

### Completion Status
- [x] Player movement and controls
- [x] Obstacle spawning system
- [x] Collision detection
- [x] Score system
- [x] Game over state
- [ ] [Any incomplete features]

### Known Bugs
- [List any bugs in base game]

### Limitations
- [List any limitations]

## Extensions Implemented

### 1. [Extension Name] (X points)
**Implementation**: [How you implemented it]
**Game Impact**: [How it changes gameplay]
**Technical Details**: [Key technical changes]
**Known Issues**: [Any bugs specific to this extension]

### 2. [Extension Name] (X points)
[Repeat format for each extension]

## Credits
- [List any external assets used]
- [Credit sources for sprites, sounds, etc.]

## Reflection
**Total Points Claimed**: [Base: 70 + Extensions: X + Documentation: X = Total]
**Challenges**: [What was difficult]
**Learning Outcomes**: [What you learned]

## Development Notes
[Any additional notes about your development process]
```

19 Submission Checklist

Before submitting, ensure you have:

- ☐ Completed the base game tutorial (70%)
- ☐ Implemented extensions totaling at least 20 points
- ☐ Published game to Unity Play with public link
- ☐ Created comprehensive README.md with Unity Play link
- ☐ Created GitHub Release with version tag
- ☐ Tested the Unity Play version in a web browser
- ☐ Committed all project files to GitHub
- ☐ Added appropriate .gitignore for Unity

20 Tips for Success

- Complete the base tutorial first before attempting extensions
- Test your game frequently as you add features
- Test your WebGL build locally before publishing to Unity Play
- Use version control to save working states before major changes
- Document bugs as you find them rather than trying to remember later
- Combine extensions that complement each other
- Consider using Unity Muse or ChatGPT for coding help, as suggested in the tutorial
- Spend time on your documentation - it's worth 10% of your grade!

21 Resources

- Unity Learn: 2D Beginner Game: Sprite Flight
- Unity Play Publishing Guide
- Unity Asset Store
- OpenGameArt.org
- Kenney.nl
- Freesound.org

22 Academic Integrity

While you may use tutorials and external assets, all code modifications and game design decisions should be your own work. Properly credit any external resources in your README file.

23 Portfolio Development

Remember that this project becomes part of your professional portfolio. A well-documented, polished game with thoughtful extensions demonstrates your ability to:

- Follow technical specifications
- Extend existing systems creatively
- Document your work professionally
- Deploy projects for public access

Take pride in creating something you'd be excited to show potential employers!

24 Activity

1. Launch the Demo Version of the game on Unity Play: Sprite Flight Demo.
2. Play the game to understand its mechanics and features for 5 minutes.
3. Turn to your neighbor and discuss:
    - Have you played similar games before? Which one is this most like?
    - Look over the extensions list. Which ones would you be excited to implement? What ideas do you have for changing the game concept that would be fun?