Flappy Bird Development Notes

Samuel George

Step 1 – Project Goals

A. Learn to create performant infinite or endless style games by moving the world instead of the player.
B. Learn to Optimize your game by using object pooling to create the illusion of many game objects using only a few.
C. Create 'cartoon style' non-realistic 2D physics using Unity's built in 2D physics system.

- Note: Designed for Unity 5.5; using Unity 2018.3.4f1 – this may require some modification

Step 2 – Adding Sprites

- Drag + Drop asset folders into …/[Project Folder]/Assets

1. Confirm Import Settings
    a. Change Mode from Single to Multiple to handle all 3 bird Sprites in the sample bird
2. Open BirdHero multi Sprite file in the Sprite Editor window
    a. Switch to Trim tool
    b. Select Automatic mode
    c. Apply changes
    - This created three Sprites as children of the original in the Project window
3. Drag BirdHero_0 into the Hierarchy window
    - Spawns at Vector3.zero by default

- Adding GrassThin
    - Drag and drop GrassThin in Hierarchy
    - Reposition along the y-axis so it forms the 'ground' of the scene
- Adding SkyTile
    - Drag onto GrassThin in the Hierarchy window to add it as a child object
        - Plaes the SkyTile over all other Sprites
    - To solve the overlapping issue: add a sorting layer and assign it to the Sprite
        - Sorting Layers are ordered from N to 0, with 0 being the furthest from the Camera.
- Adding Physics2D Components
    - Add Rigidbody2D to Bird
        - Default Settings
        - Needs a Collider to detect collisions
    - Add Colliders(2D) to physics objects
        - Bird: PolygonCollider2D
            - Roughly matches Sprite's outline by default
        - Ground: BoxCollider2D

- Required resizing and offsetting

! PERFORMED TEST (PLAY MODE) ! – PASSED

Step 3 – The Bird Script

Code concepts

A. Class scope variables
B. Stop input if dead via boolean value (control structure)
C. Access events using Unity functions (OnCollisionEnter2D)
D. Introduces MonoBehaviour functionality via OnCollisionEnter2D and GetComponent()

- Note: could have introduced [RequireComponent] flag system
- Note: good intro to physics programming
- Note: Decent explanation of when to use a Coroutine at the end of the segment

Step 4 – Animating the Bird

Note: This is where most students had questions

Window -> Animation -> Animator | docked

Window -> Animation -> Animation | docked

Docking a window is just a matter of dragging and dropping it into an active window area

Save blank animation as “Idle.anim” in Assets/Animations

- Creates the animation file + Animation Controller

In the Animation window

1. Add Property -> Sprite Renderer -> Sprite (**select bird object first**)
2. Delete second keyframe of animation (only one is needed)

Create a new clip and paste in the single keyframe from Idle

- Press Record to begin recording changes to the Bird object

1. While recording: drag BirdHero_1 into the Bird object’s Sprite slot (in the SpriteRenderer component)
2. Stop recording

Created a third clip called “Die” and repeated the steps from the previous clip with BirdHero_2 and without duplication from either existing clip.

Open the Animator window

- Created trigger “Flap”
- Created trigger “Die”

- To prevent waiting for transition from Idle to Death: uncheck “hasWaitTime” in the transition Inspector window
- Add condition “Die” to Idle -> Die transition
- Repeat Transition steps from Idle -> Die but with the “Flap” trigger
- Create return transition, Flap -> Idle, with “hasExitTime” as its only parameter in the Inspector

! PERFORMED TEST (PLAY MODE) ! – PASSED

- Added Animation Trigger Events to Bird.cs

Step 5 – Score + Game Over UI

Hierarchy (Right Click) -> UI -> Text

- Creates a Canvas object with a Text object as a child (Canvases enable UI)
- Tutorial does not require EventSystem – removed
- Set text to desired placeholder value, “Score: 0”
- Set Font to LuckiestGuy
- Set Font Size to 32
- Set Overflow (Horizontal and Vertical) to Overflow
- Renamed object to ScoreText

Centering

1. Select Text object in the Hierarchy
2. Click Anchor Point Presets
3. Hold alt and click Bottom-Center preset
    a. Alt anchors and repositions objects

Added Shadow component to ScoreText

- Add Component -> Shadow
- Set values to 1.8, -1.65

Duplicated ScoreText

- Renamed GameOverText
- Centered: Top-Center with -50 on the y-axis
- Set Text to Game Over
- Set Font Size to 48

Duplicated ScoreText

- Renamed RestartText
- Child of GameOverText
- Set text to Flap To Restart
- Set Font Size to 24

GameOverText is disabled by default

Step 6 – Adding the Game Controller

Note: provides lowest-level game logic

Hierarchy -> Right Click -> Create Empty

Renamed to GameController

Add Component -> GameControl -> New Script

- Create publicly accessible function BirdDead
    - Enables GameOverText
    - Terminates gameplay via boolean value
- Represents a clever MonoBehaviour-Singleton pattern
- Uses Awake() instead of Start() for creating the Singleton instance
    - Destroys other instances if they're found

Good example of SceneManagement basics

- Demonstrates Scene loading and retrieving info about the current scene

! PERFORMED TEST (PLAY MODE) ! – PASSED

Step 7 – Scrolling Repeating Background

Select Ground -> Add Component -> Scrolling Object -> New Script

Select Ground -> Add Component -> RigidBody2D

- Only used for scripted movement and collision detection
- Set Rigidbody2D to Kinematic (script control setting)

! PERFORMED TEST (PLAY MODE) ! – PASSED

Added Scenery as empty GameObject

- Duplicated Ground object and added both objects as children of Scenery object

Added RepeatingBackground.cs to both Ground objects

! PERFORMED TEST (PLAY MODE) ! – FAILED

Ground object position on x-axis set to -26125

- Possible arithmetic error
- Check for erroneous calculations and/or operators
- Found erroneous "-" operator where there should have been a "+" operator. Fixed.

! PERFORMED TEST (PLAY MODE) ! – PASSED

Zeroed out Bird's RigidBody2D velocity on death for aesthetics

Step 8 – Adding Column Obstacles

Note: there were several students with questions on this subject

Added first column Sprite. Sorting Layer = Midground, Height = -4

Added BoxCollider2D with Size 0.85f, 10.24f

Duplicated Column and Inverted it with y-axis position of 8

Added "Columns" object (empty GameObject) at Vector2.zero

Made both Columns children of Columns object

Added trigger BoxCollider2D and positioned at center of airgap in Columns object

Added RigidBody2D with Kinematic mode to Columns object

Added Column.cs to increment score when the Bird enters the trigger BoxCollider2D on Columns object

Added ScrollingObject.cs to Columns object

Added Prefabs folder

- Created Columns object prefab

! PERFORMED TEST (PLAY MODE) ! – PASSED

Step 9 – Recycling Obstacles with Object Pooling

Why object pooling? Pooling is more performant than continuous instantiation.

Why background movement as opposed to player movement? If there were indefinite player movement then we would experience significant overflow errors.

! PERFORMED TEST (PLAY MODE) ! – PASSED

Personal Notes

Unity Systems Used

- 2D Rendering
- Sprite Editing
- 2D Physics
- Sprite Animation

- Various Event Functions
    - OnCollisionEnter2D
    - OnTriggerEnter2D
- Input Events as Cross Platform without EventSystem

Major Development Patterns Used

- Singleton Instancing
- Object Pooling

Other

- Android SDK and NDK have been updated