Exercise 1: Your first simple game

Introduction

In this exercise, you'll create a simple timing-based minigame in which a frog tries to catch falling food with its tongue. The main objects of the game are:

- The frog
  The frog contains two sub-objects
  - The body
    Just sits there. Contains a SpriteRenderer component to draw the picture of the frog
  - The Tongue
    Reaches out toward the food and detects when it touches the candy. Controlled the the **Tongue** component, which is defined in Tongue.cs in the Code folder. Also contains a trigger collider to detect when the food hits the tongue. It also defines an event called **YummyCaught** that the sound and score systems subscribe to, that it calls when the tongue grabs some food. The tongue also contains some other components for drawing and rendering the tongue.
- Yummies
  The falling pieces of food. Contains the **Yummy** component, which is defined, not surprisingly, in the Yummy.cs file (again, in the Code folder).
- The spawner
  Contains a **YummySpawner** component (defined in YummySpawner.cs) and another trigger collider. YummySpawner does two things:
  - It periodically drops pieces of food, gradually accelerating the rate at which it drops.
  - It notices when food falls of the bottom of the screen, destroying it and notifying the rest of the game by signaling the **YummyMissed** event. The collider is used to detect when food falls off the bottom of the screen.

The game also contains the following game objects that you can mostly ignore:

- Main camera
  Contains the Camera component that renders the scene. Don't worry about this.
- Sound
  Contains an AudioSource component (a built-in Unity type) and a SoundController component (defined in SoundController.cs). Subscribes to the YummyCaught and YummyMissed events.
- Canvas
  This is used by Unity's GUI system. It's used here only to display the score. The only relevant thing is the Text object inside it that contains a **ScoreDisplay** component (defined in ScoreDisplay.cs). It subscribes to the YummyCaught and YummyMissed events to keep the score up to date.

What you should do

For this assignment, you'll just be filling in functionality in the methods of the components we've already created. Fill in the following methods in the following files:

- Yummy.cs
  This is a warmup: fill in the **Update**() to make the yummy spin at the rate specified in the field RotateSpeed (in degrees per second). Compute the number of degrees you need to rotate relative to the previous frame. The call the Rotate method of the game object's transform component.[1] Rotate takes the arguments: the amount to rotate around the X, Y, and Z axes. Since this is a 2D game, you'll want to rotate only around the Z axis, so leave the first two arguments zero.

- YummySpawner.cs
  This needs to keep track of whether it's time to spawn a yummy, and if so do so. To do that, it keeps a various called **nextTime** that holds the time at which the next yummy will spawn. It also has a field **SpawnRate** to keep track of the number of seconds between spawns. And to make sure the game gets more challenging, it decreases SpawnRate each time it spawns. The field **SpawnAcceleration** is how many seconds to decrease SpawnRate by each time you spawn..
  - **Update**()
    This should call SpawnYummy whenever Time.time > nextTime.
  - **SpawnYummy**()
    This should instantiate a new copy of the prefab stored in the field YummyPrefab, then update SpawnRate and nextTime.
  - **OnTriggerEnter2D**(yummy)
    Called when a yummy hits the spawner's trigger zone (at the bottom of the screen). Should destroy the yummy, and signal the YummyMissed event (i.e. just call YummyMissed()).

- Tongue.cs
  - **Update**()
    For this, you want to see if the player has just pressed the spacebar, and if so, start the tongue animation. The tongue animation is controlled by the Animator component packaged alongside the Tongue component. So you need to get that component, and then call its SetTrigger("Licking") method.
  - **OnTriggerEnter2D**(yummy)
    This will be called whenever the tongue hits an object. Since the only objects to hit are yummies, you know said object is a yummy, so destroy it and signal the CaughtYummy() event.

That's it! Now try out your game!

[1] Remember that every Component has a field named transform that Unity automatically sets to be the Transform of the component's game object. So you can just say transform.Rotate(arguments …).

Turning it in

For this assignment, you should just turn in the source files you modified: Yummy.cs, YummySpawner.cs, and Tongue.cs. Make a zip file containing just those files, and upload it to Canvas.