Lab4 - Randomness and Probability - Class Instructions

COMP396 - Game Programming 2
Class Instructions

Purpose: Use Randomness and Probability to provide richness and variety in your Games.

Due Date(s):

- Class Work portion: in the end of class
- The challenge portion: by end-of-week's Sunday, 11:59PM.

Class Instructions:

1 *Dice Game (s)*

1.1 Regular Dice Game

- Open the Unity Project named *COMP396_001_F24_{YourInitials}*
- Create a new Scene named *ClassWork_Week4_{YourInitials}*
- Create a script named *DiceGame.cs*. Write the following in it.

```
using UnityEngine;
using TMPro;
using UnityEngine.UI;
public class DiceGame : MonoBehaviour {
    public string inputValue = "1";
    public TMP_Text outputText;
    public TMP_InputField inputField;
    public Button button;
    int throwDice() {
        Debug.Log("Throwing dice...");
```

```
        Debug.Log("Finding random between 1 to 6...");
        int diceResult = Random.Range(1,7);
        Debug.Log($"Result: {diceResult}");
        return diceResult;
    }
    public void processGame() {
        inputValue = inputField.text;
        try {
            int inputInteger = int.Parse(inputValue);
            int totalSix = 0;
            for (var i = 0; i < 10; i++) {
                var diceResult = throwDice();
                if (diceResult == 6) { totalSix++; }
                if (diceResult == inputInteger) {
                    outputText.text = $"DICE RESULT: {diceResult} \r\nYOU WIN!";
                } else {
                    outputText.text = $"DICE RESULT: {diceResult} \r\nYOU LOSE!";
                }
            }
            Debug.Log($"Total of six: {totalSix}");
        } catch {
            outputText.text = "Input is not a number!";
            Debug.LogError("Input is not a number!");
        }
    }
}
```

- Click *GameObject → UI → Text - TextMeshPro* to add a *New Text* text on the scene.
- Center it at the top of the canvas
- Similarly, click *GameObject → UI → Button - TextMeshPro* to add a *button* on the scene...
- ...and click *GameObject → UI → Input Field - TextMeshPro* to add an *input field*.
- Arrange them vertically
- Create an empty game object and call it **DiceGame**.
- Select the text inside the button and replace Button with **Play!** in the TextMeshPro component.
- Select the New Text text and replace it with **Result:** in the TextMeshPro component:
- Attach the DiceGame component to the DiceGame object, and connect into the DiceGame component the respective UI elements that we created before
- Wire up the Button's **onClick()** to **DiceGame → processGame()**.
- Playtest.

1.2 *Loaded Dice* (or *Generating a Random Variable from an Arbitrary Distribution*)

Suppose we want to generate a random "throw" of a "loaded dice" with the following probability distribution:

Points (pips)	1	2	3	4	5	6
Probability (%)	12	12	12	12	12	40

- Copy DiceGame.cs to **DiceGameLoaded.cs** and change only the method **throwDice** to the following:

```
int throwDice() {
  Debug.Log("Throwing dice...");
  int randomProbability = Random.Range(0, 100);
  int diceResult = 0;
  if (randomProbability < 40) {
    diceResult = 6;
  } else {
    diceResult = Random.Range(1, 5);
  }
  Debug.Log("Result: " + diceResult);
  return diceResult;
}
```

- Playtest and confirm.

1.2.1 Arbitrary distribution of "loadedness"

The above method can be generalized to the case of arbitrary distribution of "loadedness". See the example below (we need to augment the above table to include a row for cumulative probabilities *CumProb (%)*:

Points (pips)	1	2	3	4	5	6
Probability (%)	12	10	7	15	20	36
Cum.Prob. (%)	12	22	29	44	64	100

- Copy DiceGameLoaded.cs to **DiceGameArbitrary.cs** and change only the method **throwDice** to the following:

```
int throwDice() {
  Debug.Log("Throwing dice...");
  int randomProbability = Random.Range(0, 100);
  int diceResult = 0;
  if (randomProbability < 12) {
    diceResult = 1;
  } else if (randomProbability < 22) {
    diceResult = 2;
  } else if (randomProbability < 29) {
    diceResult = 3;
  } else if (randomProbability < 44) {
    diceResult = 4;
  } else if (randomProbability < 64) {
    diceResult = 5;
  } else {
    diceResult = 6;
  }
  Debug.Log("Result: " + diceResult);
  return diceResult;
}
```

- Playtest and confirm.

2 FSM with probability

- Create a class named **FSMWithProbability.cs** and write the following:

```
using UnityEngine;
using System.Collections;
using System;
using System.Linq;
public class FSM : MonoBehaviour {
    [Serializable]
    public enum FSMState {
        Chase,
        Flee,
        SelfDestruct,
    }
    [Serializable]
    public struct FSMProbability {
```

```
        public FSMState state;
        public int weight;
    }
    public FSMProbability[] states;
    FSMState selectState() {
        // Sum the weights of every state.
        var weightSum = states.Sum(state => state.weight);
        var randomNumber = UnityEngine.Random.Range(0, weightSum);
        var i = 0;
        while (randomNumber >= 0) {
            var state = states[i];
            randomNumber -= state.weight;
            if (randomNumber <= 0) {
                return state.state;
            }
            i++;
        }
        // It is not possible to reach this point!
        throw new Exception("Something is wrong in the selectState algorithm!");
    }

    // Update is called once per frame
    void Update () {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            FSMState randomState = selectState();
            Debug.Log(randomState.ToString());
        }
    }
}
```

- Attach the script to the NPC
- Set up the weights as 80, 19 and 1 respectively for states Chase, Flee and SelfDestruct.
- Playtest and Confirm.

3 Creating a Random Slot Machine

- Setup the scene with the following UI elements:
    - txtReel1, txtReel2 and txtReel3
    - txtBetResult (to show “YOU WIN!” or “YOU LOSE!”)
    - txtCredits (players' credits - $$$s)
    - inputFieldBet (for the bet)

    - btnPullLever (button to imitate the lever of the slot machine)
- Create the file SlotMachine.cs with the following code:

```
using UnityEngine;
using UnityEngine.UI;
public class SlotMachine : MonoBehaviour {
  public float spinDuration = 2.0f;
  public int numberOfSym = 10;
  public Text firstReel;
  public Text secondReel;
  public Text thirdReel;
  public Text betResult;
  public Text totalCredits;
  public InputField inputBet;
  private bool startSpin = false;
  private bool firstReelSpinned = false;
  private bool secondReelSpinned = false;
  private bool thirdReelSpinned = false;
  private int betAmount;
  private int credits = 1000;
  private int firstReelResult = 0;
  private int secondReelResult = 0;
  private int thirdReelResult = 0;
  private float elapsedTime = 0.0f;
  public void Spin() {
    if (betAmount > 0) {
        startSpin = true;
    } else {
        betResult.text = "Insert a valid bet!";
    }
  }
  private void OnGUI() {
    try {
        betAmount = int.Parse(inputBet.text);
    } catch {
        betAmount = 0;
    }
    totalCredits.text = credits.ToString();
  }
  void checkBet() {
    if (firstReelResult == secondReelResult && secondReelResult == thirdReelResult) {
        betResult.text = "YOU WIN!"; credits += 500*betAmount;
    } else {
        betResult.text = "YOU LOSE!"; credits -= betAmount;
```

```
  }
}
void FixedUpdate () {
  if (startSpin) {
      elapsedTime += Time.deltaTime;
      int randomSpinResult = Random.Range(0, numberOfSym);
      if (!firstReelSpinned) {
          firstReel.text = randomSpinResult.ToString();
          if (elapsedTime >= spinDuration) {
              firstReelResult = randomSpinResult;
              firstReelSpinned = true;
              elapsedTime = 0;
          }
      } else if (!secondReelSpinned) {
          secondReel.text = randomSpinResult.ToString();
          if (elapsedTime >= spinDuration) {
              secondReelResult = randomSpinResult;
              secondReelSpinned = true;
              elapsedTime = 0;
          }
      } else if (!thirdReelSpinned) {
          thirdReel.text = randomSpinResult.ToString();
          if (elapsedTime >= spinDuration) {
              thirdReelResult = randomSpinResult;
              startSpin = false;
              elapsedTime = 0;
              firstReelSpinned = false;
              secondReelSpinned = false;
              checkBet();
          }
      }
  }
}
```

- Create an empty named **GameController** and drop the above script on it.
- Wire up the button's **onClick()** event handler to the **Spin()** method.
- Add a **GAME OVER** message when the player runs out of credits.
- Playtest and confirm.

4 Challenge(s)

- Implement the Weighted Slot Machine (using the techniques of Loaded Dice) (See Ch.4)
- Implement the **Near Miss** (See Ch.4).

- [Optional: Nicer graphics, animations, sounds.]