Fortune Spin Wheel Documentation

Overview

The JSG Fortune Spin Wheel is a comprehensive and simple system designed to create and manage a fortune spin wheel in Unity. It includes the main FortuneSpinWheel script, the Spin script for animating the final shown rewards, and the RewardData ScriptableObject for managing different types of rewards.

Components :

1. RewardData

The RewardData component is a ScriptableObject that defines the data structure for rewards in the Fortune Spin Wheel system. This component allows you to create and manage different types of rewards, such as coins or gems, along with their associated properties like title, type, count, and icon.

Features

- Data Storage: Stores reward information in a centralized and reusable format.

- Customizable: Easily customizable via the Unity Inspector.

- ScriptableObject: Leverages Unity's ScriptableObject to create asset files that can be used across multiple scenes and objects.

Fields

- `public string m_Title = "coin";` - The title of the reward.

- `public string m_Type = "coin";` - The type of reward.

- `public int m_Count = 10;` - The amount of the reward.

- `public Sprite m_Icon;` - The icon representing the reward.

Creating RewardData Assets

1. **Navigate to Create Menu:** In the Unity Editor, go to the Project window, right-click, and navigate to `Create -> CustomObjects -> RewardData`.

2. **Create and Configure:** Select `RewardData` from the menu to create a new RewardData asset. Configure the fields in the Inspector.

Example Usage

Create instances of `RewardData` for each type of reward you want to include in your spin wheel and assign these instances to the `m_RewardData` array in the FortuneSpinWheel script.

RewardData Script:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

namespace JSG.FortuneSpinWheel
{
   [CreateAssetMenu(fileName = "RewardData", menuName = "CustomObjects/RewardData", order = 1)]
   public class RewardData : ScriptableObject
   {
      public string m_Title = "coin";
      public string m_Type = "coin";
      public int m_Count = 10;
      public Sprite m_Icon;
   }
}
```

```

## 2. Spin

The Spin component is responsible for rotating the panel-reward (spark images). It rotates the GameObject it's attached to based on the specified speed and axis.

### Fields

- `public float Speed;` - The rotation speed of the wheel.
- `public Vector3 Axis;` - The axis around which the wheel will rotate.

### Methods

- Update(): Rotates the wheel every frame based on the specified speed and axis.

### Example Usage

Attach the Spin script to the GameObject representing your spark images in panel-reward in the Unity Editor. Set the speed and axis values in the Inspector.

### Spin Script:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

namespace JSG.FortuneSpinWheel
{
```
```

```
public class Spin : MonoBehaviour
{
    public float Speed;
    public Vector3 Axis;

    void Update()
    {
        transform.rotation = Quaternion.Euler(Time.deltaTime * Speed * Axis) * transform.rotation;
    }
}
}
```

3. FortuneSpinWheel

The FortuneSpinWheel component is the main script controlling the behavior of the fortune spin wheel. It manages the spinning logic, reward distribution, and UI updates.

Fields

- `public RewardData[] m_RewardData;` - Array of reward data.
- `public Image m_CircleBase;` - The base image of the circle.
- `public Image[] m_RewardPictures;` - Array of images representing rewards.
- `public Text[] m_RewardCounts;` - Array of texts showing reward counts.
- `public GameObject m_RewardPanel;` - Panel showing the final reward.
- `public Text m_RewardFinalText;` - Text showing the final reward count.
- `public Image m_RewardFinalImage;` - Image showing the final reward icon.
- `public Image m_SpinButton;` - Button to start the spin.

Hidden Fields

- `public bool m_IsSpinning = false;`
- `public float m_SpinSpeed = 0;`
- `public float m_Rotation = 0;`
- `public int m_RewardNumber = -1;`

Methods

- Start(): Initializes the wheel and rewards.
- Update(): Manages the spinning logic and updates the UI.
- HandleReward(): Handles the reward distribution based on the reward type.
- ShowRewardMenu(int seconds): Coroutine to show the reward panel after the spin.
- StartSpin(): Starts the spin.
- Reset(): Resets the wheel and UI to the initial state.

Example Usage

Attach the FortuneSpinWheel script to a GameObject in your scene. Assign the necessary references in the Inspector and configure the reward data.

Fortune Spin Wheel Script:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

namespace JSG.FortuneSpinWheel
```

```
{
    public class FortuneSpinWheel : MonoBehaviour
    {
        public RewardData[] m_RewardData;
        public Image m_CircleBase;
        public Image[] m_RewardPictures;
        public Text[] m_RewardCounts;
        public GameObject m_RewardPanel;
        public Text m_RewardFinalText;
        public Image m_RewardFinalImage;
        public Image m_SpinButton;

        [HideInInspector]
        public bool m_IsSpinning = false;
        [HideInInspector]
        public float m_SpinSpeed = 0;
        [HideInInspector]
        public float m_Rotation = 0;
        [HideInInspector]
        public int m_RewardNumber = -1;

        void Start()
        {
            m_Rotation = 0;
            m_IsSpinning = false;
            m_RewardNumber = -1;

            for (int i = 0; i < m_RewardData.Length; i++)
            {
```

```
        m_RewardPictures[i].sprite = m_RewardData[i].m_Icon;
        if (m_RewardData[i].m_Count > 0)
        {
            m_RewardCounts[i].text = "+" + m_RewardData[i].m_Count.ToString();
        }
        else
        {
            m_RewardCounts[i].gameObject.SetActive(false);
        }
    }
}

void Update()
{
    if (m_IsSpinning)
    {
        m_RewardPanel.gameObject.SetActive(false);
        if (m_SpinSpeed > 2)
        {
            m_SpinSpeed -= 4 * Time.deltaTime;
        }
        else
        {
            m_SpinSpeed -= .3f * Time.deltaTime;
        }
        m_Rotation += 100 * Time.deltaTime * m_SpinSpeed;
        m_CircleBase.transform.localRotation = Quaternion.Euler(0, 0, m_Rotation);
        for (int i = 0; i < 6; i++)
        {
```

```
            m_RewardPictures[i].transform.rotation = Quaternion.identity;
        }
        if (m_SpinSpeed <= 0)
        {
            m_SpinSpeed = 0;
            m_IsSpinning = false;
            m_RewardNumber = (int)((m_Rotation % 360) / 60);
            StartCoroutine(ShowRewardMenu(1));
            HandleReward();
        }
      }
      else
      {
        if (m_RewardNumber != -1)
        {
            m_RewardPictures[m_RewardNumber].transform.localScale = (1 + 0.2f * Mathf.Sin(10 *
Time.time)) * Vector3.one;
        }
      }
    }


    public void HandleReward()
    {
      RewardData reward = m_RewardData[m_RewardNumber];
      switch (reward.m_Type)
      {
        case "coin":
            // add coin count to your inventory
            break;
```

```
        case "gem":
            // add gem count to your inventory
            break;
    }
}

IEnumerator ShowRewardMenu(int seconds)
{
    RewardData reward = m_RewardData[m_RewardNumber];
    yield return new WaitForSeconds(seconds);
    if (reward.m_Type != "nothing")
    {
        m_RewardPanel.gameObject.SetActive(true);
        m_RewardFinalText.text = reward.m_Count.ToString();
        m_RewardFinalImage.sprite = reward.m_Icon;
        yield return new WaitForSeconds(2);
    }
    yield return new WaitForSeconds(.1f);
    Reset();
}

public void StartSpin()
{
    if (!m_IsSpinning)
    {
        m_SpinSpeed = Random.Range(4f, 14f);
        m_IsSpinning = true;
        m_RewardNumber = -1;
        m_SpinButton.gameObject.SetActive(false);
```

```
            }
        }

        public void Reset()
        {
            m_Rotation = 0;
            m_CircleBase.transform.localRotation = Quaternion.identity;
            m_IsSpinning = false;
            m_RewardNumber = -1;
            m_SpinButton.gameObject.SetActive(true);
            m_RewardPanel.gameObject.SetActive(false);
        }
    }
}
```

Conclusion

The Fortune Spin Wheel system is a versatile and easy-to-use solution for creating a fortune spin wheel in Unity. By combining the RewardData, Spin, and FortuneSpinWheel components, you can manage rewards, handle spinning mechanics, and update the UI seamlessly. Use this documentation to integrate and customize the fortune spin wheel for your game's needs.