Table of Contents

- Task 1
  - PrefabsDataChanger
- Task 2
  - MiniGameManager
  - MiniGameUIManager
  - LuckyWheelController
  - LuckyWheelGiftItemView
  - MultiplyInfoUI
  - ObstacleObject
  - MiniGameName
  - LuckyWheelEditor
- Task 3
  - MiniGameManager
  - MiniGameUIManager
  - GameManager
  - GridManager
  - TileType
  - Extentions
  - Struct
  - DataScriptableObject

Task 1

PrefabsDataChanger

The `PrefabsDataChanger` class is an `EditorWindow` script in Unity that allows users to modify data of prefabs based on a JSON file. It provides functionality to search for prefabs, fill them with data, apply changes, revert changes, and clear all changes. The script also includes methods for editing prefab assets.

```
namespace Assessment.EditorTools.Scripts.Editor
{
    public class PrefabsDataChanger : EditorWindow
    {
        // Class members and methods are defined here
    }
}
```

```
# MiniGameManager

This script defines a `MiniGameManager` class that manages the mini-game
```

```
functionality. It initializes the Gameplay API, shows UI elements, and handles
player balance updates.
```

Task 2

MiniGameUIManager

The `MiniGameUIManager` class is a manager for initializing the wheel controller in a mini-game UI. It contains methods for showing and hiding server response animations, initializing the mini-game, and handling actions after the spin finishes.

```
using System;
using System.Collections;
using System.Collections.Generic;
using DG.Tweening;
using TMPro;
using UnityEngine;
using UnityEngine.UI;

public class MiniGameUIManager : MonoBehaviour
{
    [SerializeField] private Image m_blocker;
    [SerializeField] private LuckyWheelController m_luckyWheelController;
    [SerializeField] private TextMeshProUGUI m_waitingForServerResponse;

    [SerializeField] private long m_playerBalance;
    [SerializeField] private MultiplyInfoUI mMultiplyInfoUI;
    private Tween waitingTextTween;

    [SerializeField] private Transform m_startTransfrom;
    [SerializeField] private Transform m_endTransfrom;

    private Action<long> m_afterFinishMiniGameCallback = null;

    private MiniGameName m_miniGameName;

    public void ShowWaitingForServerAnimation()
    {
        // Method implementation
    }

    public void HideWaitingForServerAnimation()
    {
        // Method implementation
    }

    public void Initialize(MiniGameName miniGameName, long playerBalance,
Action<long> afterMiniGameFinished)
```

```
    {
        // Method implementation
    }

    private void AfterSpinFinished(Int16 hittedNumber)
    {
        // Method implementation
    }

    public void MiniGameShow()
    {
        // Method implementation
    }
}
```

Task 3

GameManager

This Unity C# script defines a `GameManager` class that manages the grid size and initializes the grid manager with the specified data.

(Note: Properties, Fields, and Methods sections are not present in the script and are excluded from the output.)

GridManager

This script represents a `GridManager` class that manages a grid of tiles for a Minesweeper game. It includes methods for initializing the grid, creating tiles, placing mines, setting neighbors for each tile, showing tile neighbors, handling game over events, and managing remaining unrevealed tiles.

(Note: Properties, Fields, and Methods sections are not present in the provided script and have been excluded from the output documentation.)

TileType

This script defines two enums, `TileType` and `NeighborName`, within the `ProductMadness.Minesweeper` namespace. `TileType` represents different states of a tile in a Minesweeper game, including `Revealed`, `Unrevealed`, `Mine`, `Flag`, and `Number`. `NeighborName` enumerates the possible neighboring positions of a tile, such as `Up`, `UpLeft`, `UpRight`, `Down`, `DownLeft`, `DownRight`, `Left`, and `Right`.

Extentions

This Unity C# script defines a static class called `Extentions` that contains extension methods for GameObject and Camera objects. The `GetBoundsWithChildren` method calculates the combined bounding box of a GameObject and its children renderers. The `FocusOn` method adjusts the camera's position and near clip plane to focus on a specific GameObject with a given margin percentage.

(Note: The script does not contain properties or fields, so they are not included in the documentation.)

Struct

This script defines several structs related to tile textures, tile texture lists, and neighbors in a Minesweeper game. It includes the `TileTexture`, `TileTextureList`, and `Neighbour` structs.

Note: The script contains only the overview section.

DataScriptableObject

This script defines a `DataScriptableObject` class that inherits from `ScriptableObject`. It is used to store data related to a Minesweeper game, such as the `TilePrefab`, `TileSize`, and `TileSprites`.

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

namespace ProductMadness.Minesweeper
{
    [CreateAssetMenu(fileName = "Data", menuName = "ScriptableObjects/Data", order
= 1)]
    public class DataScriptableObject : ScriptableObject
    {
        public Tile TilePrefab;
        public float TileSize;
        public TileTextureList TileSprites;
    }
}
```