GameManager Class

- Overview
- Properties
- Fields
- Methods
- CameraController
  - Overview
  - Properties
  - Methods
- InputController Class Documentation
  - Overview
  - Properties
  - Fields
  - Methods
- MenuManager Class
  - Overview
  - Properties
  - Methods
- UIManager Class
  - Overview
  - Fields
  - Methods
- LevelManager Class Documentation
  - Overview
  - Properties
    - Levels
- LevelManagerEditor Class
  - Overview
  - Fields
  - Methods
- Level Class Documentation
  - Overview
  - Properties
    - BackgroundColor
    - MainBarColor
    - GoalPlatformColor
    - FoulPlatformColor
    - NeutralPlatformColor
    - BallColor
    - BallMass
    - PlatformData
- BallController Class Documentation
  - Overview
  - Properties
  - Fields

    - Methods
- Ball Class Documentation
    - Overview
- PlayableObject Class Documentation
    - Overview
    - Properties
    - Methods
- Object Class Documentation
    - Overview
    - Properties
        - Color
- Unity C# Script Documentation
    - Overview
    - Structures
        - PlatformData
            - Fields
        - JsonColor
            - Fields
            - Properties
- Unity C# Script Documentation
    - Overview
    - Enumerations
        - InteractiveObjectType
- InteractiveObject Class
    - Overview
    - Properties
    - Methods
- Platform Class Documentation
    - Overview
    - Properties
    - Fields
    - Methods
- MainObject Class Documentation
    - Overview
    - Properties
    - Methods
- NonInteractiveObject Class Documentation
    - Overview

GameManager Class

Overview

The `GameManager` class is a `MonoBehaviour` that manages the game state, including initialization, scoring, and game state changes (start, end, reset). It also handles the interaction between different game objects and controllers such as `MainObject`, `BallController`, `CameraController`, `LevelManager`, `InputController`, and `UIManager`.

Properties

- `Instance`: Singleton instance of the `GameManager` class.
- `Score`: The current score of the game. When the score changes, it triggers the `OnScoreChange` event of the `UIManager`.

Fields

- `m_mainObject`: The main object of the game.
- `m_ballController`: The controller for the ball.
- `m_cameraController`: The controller for the camera.
- `m_levelManager`: The manager for the game levels.
- `m_inputController`: The controller for the input.
- `m_uiManager`: The manager for the UI.
- `passedPlatform`: A list of platforms that have been passed.
- `score`: The current score of the game.

Methods

- `Initialize(int levelIndex)`: Initializes all managers and controllers of the game and levels platforms depending on the selected level. Each platform is generated randomly.
- `StartGame()`: Starts the game by activating the gravity of the ball.
- `EndGame()`: Ends the game by deactivating the gravity of the ball, disabling the camera, and triggering the `OnFinishLevel` event of the `UIManager`.
- `ResetPlatforms()`: Resets the platforms to their initial state.
- `ResetGame()`: Resets the game by setting the score to 0 and restarting the game.
- `AddForce()`: Adds force to the ball when it hits a Neutral piece for bouncing movement.
- `Pass()`: Called when the ball passes through a gap piece. It increments the score.
- `Foul()`: Called when the ball hits a foul piece. It deactivates the input controller and the ball, and triggers the `OnFoulPlay` event of the `UIManager`.

CameraController

Overview

The `CameraController` class is a script that controls the camera movement in a Unity game. It allows the camera to follow a ball with a specified offset. The camera's background color can also be set.

Properties

- `m_mainCamera`: A private serialized field of type `Camera`. This is the main camera that the script controls.
- `m_mainCameraTransform`: A private field of type `Transform`. This is used to store the transform of the main camera.

- `m_offset`, `m_initialOffset`: Private serialized fields of type `float`. These are used to control the offset of the camera from the ball. `m_initialOffset` is set to 2.4f by default.
- `m_ballTramsform`: A private field of type `Transform`. This is used to store the transform of the ball that the camera is following.

Methods

- `Initialize(Transform ballTransform)`: This public method initializes the `m_ballTramsform`, `m_mainCameraTransform`, and `m_offset` fields. It takes a `Transform` as an argument, which is the transform of the ball that the camera will follow.
- `SetBackgroundColor(Color color)`: This public method sets the background color of the main camera. It takes a `Color` as an argument.
- `Update()`: A private method that is called every frame. It updates the position of the camera to follow the ball with the specified offset.

InputController Class Documentation

Overview

The `InputController` class is a Unity script written in C# that is responsible for the input controlling of the main object to rotate. It is attached to a GameObject in a Unity scene, and it listens for mouse input to rotate the main object in the scene.

Properties

There are no public properties in the `InputController` class.

Fields

- `private MainObject m_mainObject`: This field is a reference to the main object that will be rotated.
- `private Vector3 m_startRotation`: This field stores the initial rotation of the main object.
- `private Vector2 m_lastTapPos`: This field stores the position of the last tap or click.

Methods

- `public void Initialize(MainObject mainObject)`: This method is used to initialize the `InputController`. It sets the `m_startRotation` and `m_lastTapPos` fields to zero, and sets the `m_mainObject` field to the provided `mainObject` parameter.
- `void Update()`: This method is called once per frame by Unity. It checks if the left mouse button is being held down or released, and performs actions accordingly. If the left mouse button is being held down, it calculates the difference in x-position between the current and last mouse position, and rotates the main object by this amount. If the left mouse button is released, it resets the `m_lastTapPos` field to zero.

MenuManager Class

Overview

The `MenuManager` class is a Unity MonoBehaviour script that manages the main menu of the game. It is responsible for starting the game and handling the transition from the main menu to the game scene.

Properties

- `MainMenuPanelCanvasGroup`: A public property of type `CanvasGroup`. It represents the main menu panel's canvas group in the Unity UI system.

Methods

- `StartGame()`: This is a public method that is responsible for starting the game. It first initializes the game manager with the last played level (or 0 if it's the first time the game is played). Then, it fades out the main menu panel over 1 second. Once the fade out is complete, it deactivates the main menu panel game object and starts the game.

UIManager Class

Overview

The `UIManager` class is a `MonoBehaviour` that manages all in-game UI in a Unity game. It includes several serialized private fields for UI elements such as score text, foul play panel, restart button, and finish level panel. The class also includes several public methods to handle game events like score change, foul play, finish level, and restart game.

Fields

- `m_scoreText` : A `TextMeshProUGUI` object that displays the score in the game.
- `m_foulPlayPanelCanvasGroup` : A `CanvasGroup` object that represents the foul play panel in the game.
- `m_restartBtn` : A `Button` object that represents the restart button in the game.
- `m_finishLevelPanelCanvasGroup` : A `CanvasGroup` object that represents the finish level panel in the game.

Methods

- `Initialize(Action resetGameAction)` : Initializes the `UIManager` by adding a listener to the restart button that calls the `resetGameAction` when clicked.
- `OnScoreChange(int newScore)` : Updates the score text with the `newScore`.
- `OnFoulPlay()` : Activates the foul play panel and fades it in.
- `OnFinishLevel()` : Activates the finish level panel and fades it in.
- `OnRestartGame(Action afterHideCallBack)` : Fades out the foul play panel and deactivates it, then invokes the `afterHideCallBack` action.

LevelManager Class Documentation

Overview

The `LevelManager` class is a Unity MonoBehaviour script used for creating a level in the Unity inspector. It is a public class that inherits from the `MonoBehaviour` class provided by the Unity engine. The main purpose of this class is to manage the levels in a game.

Properties

Levels

```
public List<Level> Levels;
```

This is a serialized field, meaning it can be set in the Unity inspector. It is a list of `Level` objects, which presumably represent individual levels in a game. The details of what a `Level` object contains are not provided in this script.

LevelManagerEditor Class

Overview

The `LevelManagerEditor` class is a custom editor for the `LevelManager` class in Unity. It is used to change the view of the level manager and add buttons to it. This class is part of the `DropBallEditor` namespace.

The class includes methods for creating and updating the inspector GUI, showing array properties, and getting the resources directory. It also includes functionality for loading and saving level data, placing and resetting levels.

Fields

- `LevelManager levelManager`: An instance of the `LevelManager` class.
- `private int selectedLevelNumber`: The number of the selected level.
- `string strSelectedLevel`: The string representation of the selected level number.

Methods

- `public override VisualElement CreateInspectorGUI()`: Overrides the `CreateInspectorGUI` method from the `Editor` class. It assigns the `target` to the `levelManager` and returns the base `CreateInspectorGUI`.

- `public override void OnInspectorGUI()`: Overrides the `OnInspectorGUI` method from the `Editor` class. It updates the serialized object, shows array properties, loads and saves level data, and places and resets levels.

- `public void ShowArrayProperty(SerializedProperty list)`: Shows the array properties of the serialized object. It also allows for editing of the level properties.

- `public static string GetResourcesDirectory()`: Returns the path to the resources directory. If the directory does not exist, it creates it.

Level Class Documentation

Overview

The `Level` class is a serializable class in Unity that represents the color attributes for different segments of a game level. It includes properties for the background color, main bar color, goal platform color, foul platform color, neutral platform color, and ball color. It also includes properties for the ball mass and platform data.

Properties

BackgroundColor

A `JsonColor` object that represents the background color of the level. The default value is black.

MainBarColor

A `JsonColor` object that represents the color of the main bar in the level. The default value is black.

GoalPlatformColor

A `JsonColor` object that represents the color of the goal platform in the level. The default value is black.

FoulPlatformColor

A `JsonColor` object that represents the color of the foul platform in the level. The default value is black.

NeutralPlatformColor

A `JsonColor` object that represents the color of the neutral platform in the level. The default value is black.

BallColor

A `JsonColor` object that represents the color of the ball in the level. The default value is black.

BallMass

An integer that represents the mass of the ball in the level.

PlatformData

An array of `PlatformData` objects that represents the data for the platforms in the level. The array has a length of 14.

BallController Class Documentation

Overview

The `BallController` class is a MonoBehaviour in Unity that controls the behavior of a ball object in the game. It is responsible for managing the ball's position, gravity, initialization, and force. The class uses the DOTween library to animate the ball's scale when force is applied.

Properties

- `BallTransform`: A public property with a private setter and public getter. It represents the transform of the ball.

Fields

- `m_ball`: A private serialized field of type `Ball`. It represents the ball object that the controller manages.
- `m_ballStartPosition`: A private serialized field of type `Transform`. It represents the starting position of the ball.

Methods

- `ActiveGravity()`: A public method that activates the gravity on the ball's Rigidbody.
- `DeactiveGravity()`: A public method that deactivates the gravity on the ball's Rigidbody.
- `Initialize(Level level)`: A public method that initializes the ball's position, transform, mass, and color based on the provided level.
- `AddForce(int impulseForce)`: A public method that applies an upward force to the ball's Rigidbody and shakes its scale. The amount of force applied is determined by the `impulseForce` parameter.

Ball Class Documentation

Overview

The `Ball` class is a subclass of `PlayableObject`. It currently doesn't have any properties, fields, or methods of its own. It inherits all the characteristics and behaviors of the `PlayableObject` class. The purpose and functionality of this class depend on the implementation of the `PlayableObject` class.

Please refer to the `PlayableObject` class documentation for more information about the inherited properties, fields, and methods.

PlayableObject Class Documentation

Overview

The `PlayableObject` class is an abstract class that inherits from the `Object` class in Unity. It represents an object in the game that can be played with. This class is responsible for initializing the object with a certain mass and color. The class contains properties for the `Rigidbody` and `Material` of the object, and a method to initialize these properties.

Properties

- `Rigidbody Rigidbody`: This property is used to get and privately set the `Rigidbody` component of the `PlayableObject`. The `Rigidbody` component is used to give the object physics-based behavior in the game.

- `Material Material`: This public field is used to store the `Material` component of the `PlayableObject`. The `Material` component is used to define how the object looks in terms of its color and texture.

Methods

- `public virtual void Initialize(int mass,Color color)`: This is a virtual method that is used to initialize the `Rigidbody` and `Material` properties of the `PlayableObject`. It takes two parameters: an integer `mass` to set the mass of the `Rigidbody`, and a `Color` object `color` to set the color of the `Material`. The method first gets the `Rigidbody` and `Material` components of the object, and then sets their mass and color properties respectively.

Object Class Documentation

Overview

The `Object` class is an abstract class that serves as the parent class for all playable (interactive and non-interactive) objects in the game. It is derived from the `MonoBehaviour` class which is the base class every script derives from. This class is used to set and get the color of the game objects.

Properties

Color

The `Color` property is used to get and set the color of the game object. It uses the `Renderer` component's `sharedMaterial.color` property to set the color of the object. The `get` accessor returns the color of the object.

```
public Color Color
{
    set
    {
        GetComponent<Renderer>().sharedMaterial.color = value;
    }
    get => Color;
}
```

Please note that this class is abstract and cannot be instantiated directly. It is meant to be inherited by other classes.

Unity C# Script Documentation

Overview

This script defines two serializable structures, `PlatformData` and `JsonColor`.

`PlatformData` is a structure that contains information about a platform, including whether it is active and the number of neutral, gap, and foul objects it contains.

`JsonColor` is a structure that represents a color in JSON format. It contains four float values representing the red, green, blue, and alpha channels of a color. It also includes a `Color` property that gets and sets the color using Unity's `Color` structure.

Structures

PlatformData

Fields

- `IsActive`: A boolean value indicating whether the platform is active.
- `NeutralObjectsNumber`: An integer value in the range of 0 to 11 indicating the number of neutral objects.
- `GapObjectsNumber`: An integer value in the range of 0 to 11 indicating the number of gap objects.
- `FoulObjectNumbers`: An integer value in the range of 0 to 11 indicating the number of foul objects.

JsonColor

Fields

- `_r`: A float value representing the red channel of a color.
- `_g`: A float value representing the green channel of a color.
- `_b`: A float value representing the blue channel of a color.
- `_a`: A float value representing the alpha channel of a color.

Properties

- `Color`: A property of type `Color` that gets and sets the color using the `_r`, `_g`, `_b`, and `_a` fields.

Unity C# Script Documentation

Overview

This script defines an enumeration `InteractiveObjectType` in Unity using C#. The `InteractiveObjectType` enum is used to categorize different types of interactive objects within the game. It has four different values: `Neutral`, `Gap`, `Foul`, and `Goal`.

Enumerations

InteractiveObjectType

This enumeration is used to categorize different types of interactive objects within the game. It has four different values:

- **Neutral**: This value represents a neutral interactive object in the game.
- **Gap**: This value represents a gap in the game.
- **Foul**: This value represents a foul in the game.
- **Goal**: This value represents a goal in the game.

InteractiveObject Class

Overview

The `InteractiveObject` class is a type of `Object` in Unity that represents an interactive object in the game. Each interactive object has an action that is triggered when it is hit. The class provides functionality for initializing the object, adding listeners to various actions, and handling collision and trigger events.

Properties

- `AddForced`: A static boolean property that indicates whether force should be added to the object.
- `PassCoolDown`: A static boolean property that indicates whether the object is in a cooldown period after passing through a trigger.
- `Type`: An `InteractiveObjectType` property that specifies the type of the interactive object.
- `Renderer`: A `Renderer` property that renders the object in the game.
- `Collider`: A `Collider` property that handles collision detection for the object.
- `OnHitAction`: An `Action<Object>` property that defines the action to be performed when the object is hit.
- `OnPassAction`: An `Action<Object>` property that defines the action to be performed when the object passes through a trigger.
- `OnFoulAction`: An `Action<Object>` property that defines the action to be performed when a foul occurs.
- `OnGoalAction`: An `Action<Object>` property that defines the action to be performed when a goal is scored.

Methods

- `InteractiveObject(InteractiveObjectType type)`: Constructor that initializes a new instance of the `InteractiveObject` class with the specified type.
- `Initialize(InteractiveObjectType type, Color color, Action<Object> onHitAction, Action<Object> onPassAction, Action<Object> onGoalAction, Action<Object> onFoulAction)`: Method that initializes the interactive object.
- `AddListenerToOnHit(Action<Object> hitAction)`: Method that adds a listener to the `OnHitAction`.
- `AddListenerToOnPass(Action<Object> passAction)`: Method that adds a listener to the `OnPassAction`.
- `AddListenerToOnFoul(Action<Object> foulAction)`: Method that adds a listener to the `OnFoulAction`.
- `AddListenerToOnGoal(Action<Object> goalAction)`: Method that adds a listener to the `OnGoalAction`.
- `OnCollisionEnter(Collision other)`: Method that handles the collision event.
- `OnTriggerExit(Collider other)`: Method that handles the trigger exit event.

Platform Class Documentation

Overview

The `Platform` class is a Unity MonoBehaviour script that is used to manage a list of `InteractiveObject` instances. This class is typically attached to a GameObject in the Unity scene to provide functionality for managing multiple interactive objects.

Properties

There are no properties in the `Platform` class.

Fields

- `Objects` : A serialized field that is a list of `InteractiveObject` instances. This field is public, so it can be accessed and modified directly from other scripts.

Methods

There are no methods in the `Platform` class.

MainObject Class Documentation

Overview

The `MainObject` class is a subclass of `NonInteractiveObject`. It is used to represent a main object in the game that has a list of platforms associated with it and can have its color set.

Properties

- `Platforms`: A public list of `Platform` objects associated with the `MainObject`.

Methods

- `SetColor(Color color)`: A public method that sets the color of the `MainObject`. It takes a `Color` object as an argument and applies it to the `Renderer.sharedMaterial.color` of the `MainObject`.

NonInteractiveObject Class Documentation

Overview

The `NonInteractiveObject` class is a subclass of Unity's `Object` class. This class is used to represent objects in the game that are not interactive, meaning the player cannot interact with them directly. It contains a single public field, `Renderer`, which is used to control the rendering of the object in the game.