GameManager Class

Overview

The `GameManager` class is a Unity MonoBehaviour that manages the game state and controls the interaction between various game components. It follows the Singleton design pattern to ensure only one instance of the class exists in the game. The class is responsible for initializing various game components, handling player inputs, and managing game events such as collisions with obstacles and collectables.

Properties

- `Instance`: A static property that returns the singleton instance of the `GameManager` class.
- `GameStarted`: A static boolean property that indicates whether the game has started.

Fields

- `instance`: A private static field that holds the singleton instance of the `GameManager` class.
- `m_events`: A private field of type `Events` that manages the game events.
- `m_gameManagerData`: A serialized private field that holds the game manager data.
- `m_inputController`, `m_playerController`, `m_movementController`, `m_cameraMovementController`, `m_platformManager`, `m_obstacleManager`, `m_collectManager`, `m_particleManger`, `m_audioManager`: Serialized private fields that hold references to various game controllers and managers.
- `m_hitToObstacleCoolDown`: A private boolean field that indicates whether the player is in a cooldown period after hitting an obstacle.

Methods

- `Awake()`: A Unity callback method that is called when the script instance is being loaded. It calls the `Initialize()` method.
- `Initialize()`: A public method that initializes all managers and controllers and sets up game events.
- `StartGame()`: A public method that starts the game.
- `GetInput(Vector3 newPos)`: A public method that passes a new position vector to the player controller to move the player horizontally.
- `HitToObstacle(PlayerCube playerCube, int obstaclesCubeCount, ObstacleCube obstacleCube)`: A public method that is called when the player cube hits an obstacle cube. It manages the game state based on the hit.
- `HitCoolDown()`: A private method that sets a cooldown period after the player hits an obstacle.
- `HitToFinishPlatform(PlayerCube playerCube)`: A public method that is called when the player cube hits the finish platform. It manages the game state based on the hit.
- `HitToCollectableCube(CollectableCube collectableCube)`: A public method that is called when the player cube hits a collectable cube. It triggers the `OnAddCubeEvent` event.

PlayerController Class

Overview

The `PlayerController` class is a MonoBehaviour in Unity that controls the player's actions. It is responsible for initializing the player, setting the player's position, adding and removing cubes from the player, and determining if the game is over.

Properties

- `Player`: Gets the `Player` object.
- `PlayerTransform`: Gets the `Transform` of the `Player` object.

Fields

- `m_player`: A serialized private field of type `Player`.

Methods

- `Initialize(Color trialColor)`: Initializes the player with a given color.
- `SetPlayerPosition(Vector3 newPos)`: Sets the player's position to a new position.
- `AddCubeToPlayer(CollectableCube collectableCube)`: Adds a collectable cube to the player.
- `RemoveCubeFromPlayer(PlayerCube playerCube)`: Removes a cube from the player.
- `GameIsOver(bool playerIsWinner)`: Determines if the game is over, based on whether the player is a winner or not.

ObstacleManager Class

Overview

The `ObstacleManager` class is a MonoBehaviour that manages the obstacles and finish platforms in the game. It initializes the obstacles and finish platforms with the appropriate colors, and handles the interactions between the player and these game elements.

Properties

m_obstaclesGroups

A serialized private list of `ObstacleGroup` objects. This list contains all the obstacle groups in the game.

m_finishPlatforms

A serialized private array of `FinishPlatform` objects. This array contains all the finish platforms in the game.

Methods

Initialize(List obstacleCubeColorSet, List finishPlatformColors)

This method initializes the obstacle groups and finish platforms with the given color sets. It finds all the `ObstacleGroup` objects in the scene, and initializes each one with the `obstacleCubeColorSet`. It also initializes each `FinishPlatform` in `m_finishPlatforms` with the corresponding color from `finishPlatformColors`.

HitToObstacle(PlayerCube playerCube, int obstacleCubeCount, ObstacleCube obstacleCube)

This method handles the interaction when the player hits an obstacle. It detaches the `playerCube` from its parent, removes its rigid body, and calls the `HitToObstacle` method of the `GameManager` instance.

HitToFinishPlatform(PlayerCube playerCube)

This method handles the interaction when the player hits a finish platform. It detaches the `playerCube` from its parent, removes its rigid body, and calls the `HitToFinishPlatform` method of the `GameManager` instance.

PlatformManager

Overview

The `PlatformManager` class is a Unity MonoBehaviour that manages a list of `Platform` objects. It provides methods to initialize the platforms and set the currently hit platform. It also exposes a property to get the `BoxCollider` of the hit platform.

Properties

- `HittedPlatformBoxCollider`: This is a `BoxCollider` property that returns the `BoxCollider` of the currently hit platform.

Fields

- `m_platforms`: This is a private serialized field of type `List<Platform>`. It holds the list of all `Platform` objects in the scene.

- `m_hittedPlatform`: This is a private serialized field of type `Platform`. It holds the reference to the currently hit platform.

Methods

- `Initialize()`: This method initializes the `PlatformManager`. It finds all the `Platform` objects in the scene, initializes them, sorts them based on their distance from the camera, and sets the first platform in the sorted list as the hit platform.

- `SetPlatform(Platform platform)`: This method sets the provided `Platform` object as the currently hit platform.

CollectManager Class

Overview

The `CollectManager` class is a `MonoBehaviour` that manages the collection of `CollectableCube` objects in a Unity scene. It provides methods for initializing the collection of cubes with a random color from a provided set, and for handling the event of a cube being hit.

Properties

- `m_collectableCube`: A private serialized field of type `List<CollectableCube>`. This list holds references to all the `CollectableCube` objects in the scene.

Methods

Initialize(collectableCubeColorSet)

This method initializes the `m_collectableCube` list with all the `CollectableCube` objects found in the scene. It then selects a random color from the `collectableCubeColorSet` list and assigns it to all the cubes in the `m_collectableCube` list. The method returns the selected color.

Parameters

- `collectableCubeColorSet`: A list of `Color` objects from which a random color is selected for the cubes.

Returns

- `Color`: The randomly selected color.

HitToCollectableCube(collectableCube)

This method is called when a `CollectableCube` is hit. It calls the `HitToCollectableCube` method of the `GameManager` singleton instance, passing the hit cube as a parameter.

Parameters

- `collectableCube`: The `CollectableCube` object that was hit.

MovementController

Overview

The `MovementController` class is a Unity MonoBehaviour that controls the movement of a player object in the game. It uses the DOTween library to animate the player's movement and the Cysharp.Threading.Tasks library to handle asynchronous tasks.

Properties

- `m_trailReference`: A serialized private field of type `SplineTrailRenderer`. This is used to render a trail behind the player as they move.
- `m_playerTransform`: A serialized private field of type `Transform`. This is the transform of the player object that the `MovementController` is controlling.
- `m_moveSpeed`: A serialized private field of type `float`. This is the speed at which the player object moves.
- `distance`: A private field of type `float`. This is used to calculate the distance the player has moved.

Methods

- `Initialize(Transform playerTransform)`: This public method is used to initialize the `MovementController` with a player transform.
- `Update()`: This public method is a Unity callback that is called every frame. It updates the player's position if the game has started.
- `SpeedUp()`: This public async method is used to temporarily increase the player's speed. It uses the DOTween library to animate the player's movement and the Cysharp.Threading.Tasks library to handle

the delay.

AudioManager Class

Overview

The `AudioManager` class is a MonoBehaviour that manages the audio in a Unity game. It has two audio clips, one for sliding and one for adding a cube, and an audio source from the player. The class provides methods to initialize the player's audio source and to play the slide and add cube sounds.

Properties

There are no public properties in this class.

Fields

- `m_slideAudioClip`: An AudioClip that holds the audio for the slide sound.
- `m_addCubeAudioClip`: An AudioClip that holds the audio for the add cube sound.
- `m_playerAudioSource`: An AudioSource that is used to play the audio clips.

Methods

- `Initialize(AudioSource playerAudioSource)`: This method is used to initialize the player's audio source. It takes an AudioSource as a parameter and assigns it to `m_playerAudioSource`.
- `PlaySlideSound()`: This method is used to play the slide sound. It assigns `m_slideAudioClip` to the clip of `m_playerAudioSource` and then plays it.
- `PlayAddCube()`: This method is used to play the add cube sound. It assigns `m_addCubeAudioClip` to the clip of `m_playerAudioSource` and then plays it.

ParticleManager Class

Overview

The `ParticleManager` class is a `MonoBehaviour` script in Unity that manages the particle systems in the game. It is responsible for showing different particles when certain events occur, such as adding a cube, removing a cube, or hitting the finish platform. The class contains three private `ParticleSystem` fields and three public methods to control these particle systems.

Fields

- `m_playerAddCubeParticle`: A private `ParticleSystem` that is shown when a cube is added.
- `m_playerRemoveCubeParticle`: A private `ParticleSystem` that is shown when a cube is removed.
- `m_hitToFinishPlatformParticle`: A private `ParticleSystem` that is shown when the player hits the finish platform.

Methods

- `ShowAddCubeParticle(CollectableCube lastCollectableCube)`: This method sets the position of the `m_playerAddCubeParticle` to the position of the last collected cube and plays the particle system.
- `ShowRemoveCubeParticle(PlayerCube hittedCube, ObstacleCube obstacleCube)`: This method sets the position of the `m_playerRemoveCubeParticle` to the position of the hit cube and plays the particle system.
- `ShowHitToFinishPlatformParticle(PlayerCube hittedCube)`: This method sets the position of the `m_hitToFinishPlatformParticle` to the position of the hit cube and plays the particle system.

InputController Class

Overview

The `InputController` class is a MonoBehaviour that handles the user input for the game. It is responsible for detecting mouse events and updating the player's position accordingly. The class uses the UniRx library to handle events in a reactive programming style.

Fields

- `private bool startDragging`: A boolean value that indicates whether the user has started dragging the mouse.
- `private float startMouseXPosition`: A float value that stores the initial x position of the mouse when the user starts dragging.
- `[SerializeField] private Player m_player`: A serialized private field that references the Player object.
- `[SerializeField] private BoxCollider m_groundCollider`: A serialized private field that references the BoxCollider object representing the ground.

Methods

- `public void Initialize(Player player,BoxCollider groundCollider)`: This method initializes the `InputController` with a `Player` and a `BoxCollider`. It sets up the necessary UniRx observables to handle mouse events and update the player's position. The method uses the `UpdateAsObservable` and `LateUpdateAsObservable` methods from the UniRx library to create observables that emit values every frame and every late update, respectively. It also uses the `Where` method to filter the emissions based on whether the mouse button is down or up. The `Subscribe` method is used to specify what actions to take when the observables emit values.

CameraMovementController

Overview

The `CameraMovementController` class is a Unity MonoBehaviour script that controls the movement of a camera in relation to a player's position. The camera's position is updated every frame during the game's update loop, maintaining a specified offset from the player's position.

Properties

- `m_camera`: A private serialized field of type `Camera`. This is the camera that will be controlled by this script.
- `m_playerTransform`: A private serialized field of type `Transform`. This is the transform of the player object that the camera will follow.
- `m_offset`: A private serialized field of type `float`. This is the distance that the camera will maintain from the player's position.

Methods

- `Initialize(Transform playerTransform)`: This public method sets the `m_playerTransform` field to the provided `playerTransform` argument. This method is used to set the player object that the camera will follow.
- `Update()`: This public method is called every frame by Unity's game loop. If the game has started, it updates the camera's position to maintain the specified offset from the player's position.

GameManagerData Class

Overview

The `GameManagerData` class is a `ScriptableObject` that stores the game manager data. This class is used to store various settings and configurations for the game manager. It includes properties for move speed and different color sets for collectable cubes, finish platforms, and obstacle cubes.

This class can be created from the Unity Editor via the "Create" context menu, under the "GameManagerData" option.

Properties

- `MoveSpeed` (float): This property stores the move speed of the game object.

- `CollectableCubeColorSet` (List): This property stores a list of colors that can be used for collectable cubes in the game.

- `FinishPlatformColorSet` (List): This property stores a list of colors that can be used for the finish platforms in the game.

- `ObstacleCubeColorSet` (List): This property stores a list of colors that can be used for obstacle cubes in the game.

Unity C# Script

Overview

This script defines two abstract classes, `CubeBase` and `Cube<T>`, that are used to create and manage cube objects in a Unity game. The `CubeBase` class is a `MonoBehaviour` and contains a serialized field for the cube's material. The `Cube<T>` class inherits from `CubeBase` and contains a serialized field for a generic manager object. It also includes a method to initialize the cube with a manager and a color.

Properties

There are no properties in this script.

Fields

- `CubeBase` class:
  - `m_material` (Material): A serialized field that stores the material of the cube.
- `Cube<T>` class:
  - `m_manager` (T): A serialized field that stores a generic manager object.

Methods

- `Cube<T>` class:
  - `Initialize(T t, Color cubeColor)`: This method sets the manager object and the color of the cube. It takes two parameters:
    - `t` (T): The manager object.
    - `cubeColor` (Color): The color of the cube.

CollectableCube Class

Overview

The `CollectableCube` class is a derived class from the generic `Cube` class with `CollectManager` as the type parameter. This class represents a collectable cube in the game. When a player's cube collides with a collectable cube, the cube is collected and the player's cube is repositioned.

Fields

- `private bool m_collected`: A private boolean field that keeps track of whether the cube has been collected or not.
- `[SerializeField] private BoxCollider m_boxCollider`: A private serialized field of type `BoxCollider`. This is the collider for the cube.

Properties

- `public Vector3 Size`: A public property that returns the size of the cube's box collider.

Methods

- `private void OnCollisionEnter(Collision other)`: A private method that is called when the cube collides with another object. If the other object is the player's cube and the collectable cube has not been collected yet, the cube is marked as collected, the player's cube is repositioned, and the `HitToCollectableCube` method of the `m_manager` object is called with this cube as the argument.

Events Class

Overview

The `Events` class in Unity C# is a public class that contains several delegate definitions and corresponding event declarations. These events are designed to handle different scenarios in a game, such as adding a cube, hitting an obstacle, hitting the finish platform, and the game ending.

Delegates and Events

AddCubeEvent

This delegate takes an argument of `CollectableCube` type. The corresponding event `OnAddCubeEvent` is triggered when a new cube is added to the game.

HitToObstacleCubeEvent

This delegate takes two arguments, `PlayerCube` and `ObstacleCube`. The corresponding event `OnHitToObstacleCubeEvent` is triggered when the player's cube hits an obstacle cube in the game.

HitToObstacleRunOneTime

This delegate does not take any arguments. The corresponding event `OnHitToObstacleRunOneTime` is triggered when the player's cube hits an obstacle for the first time.

HitToFinishPlatform

This delegate takes an argument of `PlayerCube` type. The corresponding event `OnHitToFinishPlatform` is triggered when the player's cube hits the finish platform.

GameIsOverEvent

This delegate takes a boolean argument `playerIsWinner`. The corresponding event `OnGameIsOverEvent` is triggered when the game is over, indicating whether the player has won or lost.

FinishPlatform Class

Overview

The `FinishPlatform` class is a derived class from the generic `Cube` class with `ObstacleManager` as the type parameter. This class is used to handle the collision events of the finish platform in a game. When a player's cube hits the finish platform, it triggers a hit event.

Properties

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

Fields

- `hitted`: A private serialized field of type `bool`. This field is used to check if the finish platform has been hit by the player's cube.

Methods

- `OnCollisionEnter(Collision other)`: A private method that is called when the finish platform collides with another object. It checks if the object it collided with has the tag "PlayerCube" and if the finish platform has not been hit before. If both conditions are met, it sets `hitted` to `true` and calls the `HitToFinishPlatform` method of the `ObstacleManager` instance, passing the `PlayerCube` component of the other object as an argument.

Group and GroupBase Classes

Overview

The provided script defines two abstract classes: `Group` and `GroupBase`. These classes are used to manage groups of objects in a Unity game. The `Group` class is a simple `MonoBehaviour`, while the `GroupBase` class extends `Group` and adds additional functionality for managing a list of `CubeBase` objects and initializing them with a random color from a provided list.

Classes

Group

This is an abstract class that inherits from `MonoBehaviour`. It does not contain any properties, fields, or methods.

GroupBase

This is an abstract class that inherits from `Group` and is generic, taking two type parameters: `T1` and `T2`. `T2` must be a type that inherits from `CubeBase`.

Fields

- `m_manager`: A protected field of type `T1`. This is used to manage the group.
- `m_cubes`: A public field of type `List<T2>`. This is used to store the cubes in the group.

Methods

- `Initialize(T1 manager, List<Color> colorSet)`: This public method takes a manager of type `T1` and a list of colors. It sets the `m_manager` field to the provided manager, gets all the `CubeBase` components in the children of this `MonoBehaviour`, converts them to a list, and stores them in the `m_cubes` field. It then calls the `Initialize(Color color)` method with a random color from the provided list.
- `Initialize(Color color)`: This is an abstract method that takes a color. The implementation of this method should initialize the group with the provided color.

Obstacle Class

Overview

The `Obstacle` class is a Unity script attached to a GameObject, making the GameObject behave as an obstacle in the game. This class inherits from the `MonoBehaviour` class which is the base class from which every Unity script derives. The `Obstacle` class currently has no properties or fields, and it only contains two methods: `Start` and `Update`.

Methods

Start

```
void Start()
```

The `Start` method is a Unity-specific method that is called before the first frame update. Currently, this method does not perform any actions.

Update

```
void Update()
```

The `Update` method is another Unity-specific method that is called once per frame. Currently, this method also does not perform any actions.

ObstacleCube Class

Overview

The `ObstacleCube` class is a subclass of the `Cube` class with `ObstacleGroup` as its generic parameter. This class is used to handle the behavior of obstacle cubes in the game. The main functionality of this class is to detect collisions with player cubes and respond accordingly.

Methods

OnCollisionEnter(Collision other)

This is a Unity-specific method that is called when this object starts colliding with another object. The method checks if the object it collided with has a tag of "PlayerCube". If it does, it calls the `HitToObstacle` method of the `m_manager` object, passing in the `PlayerCube` component of the other object and `this` as arguments.

```
private void OnCollisionEnter(Collision other)
{
    if (other.transform.CompareTag("PlayerCube"))
```

```
        m_manager.HitToObstacle(other.transform.GetComponent<PlayerCube>(),this);
}
```

Parameters:

- `other` - The `Collision` object that contains all the information about the collision. This is automatically passed in by Unity when the collision happens.

Returns:

- This method does not return any value.

ObstacleGroup Class

Overview

The `ObstacleGroup` class is a part of the Unity game engine and is used to manage groups of obstacle cubes in a game. It inherits from the `GroupBase` class, with `ObstacleManager` and `ObstacleCube` as its generic parameters. The class contains methods for initializing the obstacle group and handling hits to the obstacles.

Methods

Initialize Method

```
public override void Initialize(Color color)
```

The `Initialize` method is used to set up the obstacle group. It takes a `Color` object as a parameter and applies this color to all the cubes in the group.

HitToObstacle Method

```
public void HitToObstacle(PlayerCube playerCube,ObstacleCube obstacleCube)
```

The `HitToObstacle` method is used to handle the event of a player's cube hitting an obstacle cube. It takes a `PlayerCube` object and an `ObstacleCube` object as parameters. The method then calls the `HitToObstacle` method of the `ObstacleManager` class, passing in the player's cube, the count of cubes in the group, and the obstacle cube.

Platform Class

Overview

The `Platform` class is a Unity MonoBehaviour script that is used to manage the behavior of platforms in a game. It includes a reference to a `BoxCollider` component and a `PlatformManager` object. The

`BoxCollider` is used to detect when an object enters the platform, and the `PlatformManager` is used to manage the platform's behavior when an object enters it.

Properties

- `BoxCollider`: This is a public property that returns the private `BoxCollider` field `m_boxCollider`.

Fields

- `m_boxCollider`: This is a private serialized field of type `BoxCollider`. This is used to detect when an object enters the platform.

- `m_manager`: This is a private field of type `PlatformManager`. This is used to manage the platform's behavior when an object enters it.

Methods

- `Initialize(PlatformManager manager)`: This is a public method that takes a `PlatformManager` object as a parameter. It is used to initialize the `m_manager` field.

- `OnTriggerEnter(Collider other)`: This is a private method that is called when an object enters the `BoxCollider`. It sets the platform in the `PlatformManager` to `this` platform.

Player Class

Overview

The `Player` class is a Unity MonoBehaviour that represents the player in the game. It handles the player's animations, position, interactions with collectable cubes, and game state (winning or losing). The player has a series of serialized fields for components such as an Animator, CapsuleCollider, Transform, PlayerCubes, AudioSource, and Material. It also exposes some properties for accessing the AudioSource, the count of PlayerCubes, and the size of the first cube.

Properties

- `AudioSource AudioSource`: Gets the AudioSource component attached to the player.
- `int PlayerCubeCount`: Gets the count of PlayerCubes.
- `Vector3 FirstCubeSize`: Gets the size of the first cube in the PlayerCubes collection.

Methods

- `void Initialize(Color trialColor)`: Initializes the player with a given trial color.
- `void SetPosition(Vector3 newPos)`: Sets the player's position to a new position.
- `void AddCubeToPlayer(CollectableCube collectableCube)`: Adds a collectable cube to the player and adjusts the player's position.
- `void RemoveCubeFromPlayer(PlayerCube collectableCubeTransform)`: Removes a cube from the player.
- `void GameIsOver(bool playerIsWinner)`: Handles the end of the game, setting the appropriate animation based on whether the player won or lost.

- `private IEnumerator FallingFinish()`: Coroutine that waits until the player has finished falling before setting the Falling animation state to false.
- `private void OnDrawGizmos()`: Unity method for drawing gizmos in the editor. This method draws a ray from the player's position downwards.

PlayerCube Class

Overview

The `PlayerCube` class is a Unity script written in C# and is attached to a GameObject to control its behavior. This class is responsible for managing the BoxCollider and Rigidbody components of the GameObject. It provides methods to initialize these components and to remove the Rigidbody from the GameObject.

Properties

- `Size`: This is a read-only property that returns the size of the BoxCollider component.

Fields

- `m_boxCollider`: This is a private serialized field of type BoxCollider. It represents the BoxCollider component of the GameObject.
- `m_rigidbody`: This is a private serialized field of type Rigidbody. It represents the Rigidbody component of the GameObject.

Methods

- `Initialize()`: This method is used to initialize the BoxCollider and Rigidbody components of the GameObject. It gets these components from the GameObject and assigns them to the `m_boxCollider` and `m_rigidbody` fields respectively.
- `RemoveRigidBody()`: This method is used to remove the Rigidbody component from the GameObject. It destroys the `m_rigidbody` field.

PlayerCubes Class

Overview

The `PlayerCubes` class is a `MonoBehaviour` that manages the cubes associated with a player in a Unity game. It provides methods for initializing, adding, removing, and sorting the cubes based on their distance from the player.

Properties

- `Cubes`: A public read-only property that returns the list of `PlayerCube` objects associated with the player.

Fields

- `m_player`: A private field of type `Player` that represents the player associated with the cubes.

- `m_cubes`: A private serialized field of type `List<PlayerCube>` that stores the list of cubes associated with the player.

Methods

- `Initialize()`: This method initializes the `m_cubes` list by getting all the `PlayerCube` components in the children of the transform.
- `AddCube(CollectableCube collectableCube)`: This method adds a `PlayerCube` to the `m_cubes` list. It takes a `CollectableCube` as a parameter, adds a `PlayerCube` component to it, initializes it, sets its tag to "PlayerCube", sets its scale and position, sets its parent to the transform, destroys the `CollectableCube` component, and inserts the `PlayerCube` at the beginning of the `m_cubes` list.
- `RemoveCube(PlayerCube playerCube)`: This method removes a `PlayerCube` from the `m_cubes` list. It takes a `PlayerCube` as a parameter.
- `SortCubeByDistance()`: This method sorts the `m_cubes` list based on the square magnitude of the distance between the cube's position and the player's position.