Introduction

This documentation provides a comprehensive overview of the Unity-based component scripts provided. Each section includes a description of the classes, properties, methods, and specific use cases for each script to help developers understand and implement these components effectively.

ColorPicker.cs

Purpose: Implements a user interface for selecting colors using a graphical texture (wheel or gradient).

Classes & Methods:

• ColorPicker Class: Extends MonoBehaviour and implements IDragHandler, IPointerDownHandler.

- Properties:
  - cursor: RectTransform object representing the cursor.
  - onColorChanged: Event fired when the color is changed.
  - colorTexture: The texture representing the color spectrum.
- Methods:
  - Start(): Initializes the ColorTextureGenerator and fires an initial color change event.
  - OnPointerDown / OnDrag(): Handles pointer and drag events to update the cursor and the selected color.
  - GetColorFromTexture(Vector2 localPoint): Gets the color from the texture based on the cursor's position.

Use Case: A visual tool for users to select colors dynamically in the game.

ColorTextureGenerator.cs

Purpose: Generates the texture for the ColorPicker based on the chosen texture type (wheel or gradient).

Classes & Methods:

• ColorTextureGenerator Class: Extends MonoBehaviour.

- Properties:
  - TextureType: Enum property defining if the texture is a Gradient or Wheel.
  - textureWidth, textureHeight: Dimensions of the generated texture.
- Methods:
  - GenerateTexture(): Creates a texture based on the selected texture type.
  - GenerateGradientTexture(): Generates a square gradient texture.
  - GenerateWheelTexture(): Generates a circular color wheel texture.

Use Case: Helps create flexible color selection interfaces.

TextureType.cs

Purpose: Defines the different types of textures that can be generated by ColorTextureGenerator.

Classes & Methods:

• TextureType Enum: Contains two values:

- Gradient: Represents a square gradient texture.
- Wheel: Represents a circular color wheel texture.

Use Case: Used in conjunction with ColorTextureGenerator to determine the type of color picker texture displayed to the user.

GameEvent.cs

Purpose: Represents a custom Unity ScriptableObject to manage and trigger game events.

Classes & Methods:

• GameEvent Class: Extends ScriptableObject.

- Properties:
  - listeners: A list of listeners registered to this event.
- Methods:
  - Raise(): Raises the event with optional data and sender.
  - RegisterListener(GameEventListener listener): Registers a listener.

- UnregisterListener(GameEventListener listener): Unregisters a listener.

Use Case: A core part of a flexible event system, useful in decoupling components that need to react to specific changes in the game state.

GameEventListener.cs

Purpose: Listens to a GameEvent and triggers a response when the event is raised.

Classes & Methods:

• GameEventListener Class: Extends MonoBehaviour.

- Properties:
  - gameEvent: The GameEvent to register with.
  - response: The UnityEvent that is invoked when the event is raised.
- Methods:
  - OnEnable(): Registers this listener with the event.
  - OnDisable(): Unregisters this listener from the event.
  - OnEventRaised(Component sender, object data): Invokes the response.

Use Case: Great for implementing game state changes that need to be tracked and responded to in different parts of a game.

Vehicle.cs

Purpose: Represents a basic vehicle object with properties for texture and color updates.

Classes & Methods:

• Vehicle Class: Extends MonoBehaviour to interact with Unity's GameObject components.

- Properties:
  - baseMaterial: Primary material for the vehicle.
  - textureMaterial: Material to update the vehicle's texture.
- Methods:
  - SetTexture(Texture texture): Assigns a texture to the vehicle's textureMaterial.
  - UpdateColor(Component sender, object data): Updates the color of the vehicle based on the received data if it is of type Color.

Use Case: This class could be used to dynamically change the appearance of vehicles in a game.

CarsManager.cs

Purpose: Manages a collection of cars, allows switching between them, and changes their textures.

Classes & Methods:

• CarsManager Class: Extends MonoBehaviour.

- Properties:
  - cars: A list of Vehicle objects.
  - OnCarChanged: Event triggered when the active car changes.
  - carTextures: List of textures available for the cars.
- Methods:
  - OnEnable(): Initializes the car and texture indices and hides inactive cars.
  - SetPreviousTexture() / SetNextTexture(): Changes the active car's texture.
  - SelectPreviousCar() / SelectNextCar(): Changes the active car.
  - HideAllCars(): Disables all car game objects.

Use Case: Ideal for managing multiple car models in a car customization game or a showroom application.

RotateComponent.cs

Purpose: Provides a simple way to rotate a GameObject based on a specified rotation speed.

Classes & Methods:

• RotateComponent Class: Extends MonoBehaviour.

- Properties:
  - rotateSpeed: A vector specifying rotation speed in three dimensions.
- Methods:
  - Update(): Continuously rotates the object using the specified rotation speed.

Use Case: Useful for creating dynamic visual effects like rotating a car model in a 3D preview or turning wheels on a moving vehicle.

ImageColorUpdater.cs

Purpose: Updates the color of a UI Image component based on external data.

Classes & Methods:

- ImageColorUpdater Class: Extends MonoBehaviour.
  - Properties: None explicitly mentioned.
  - Methods:
    - OnEnable(): Gets and caches the Image component reference.
    - UpdateColor(Component sender, object data): Updates the Image color if the data type is Color.

Use Case: Commonly used for updating the UI, such as changing the color of a button or panel based on an external event.