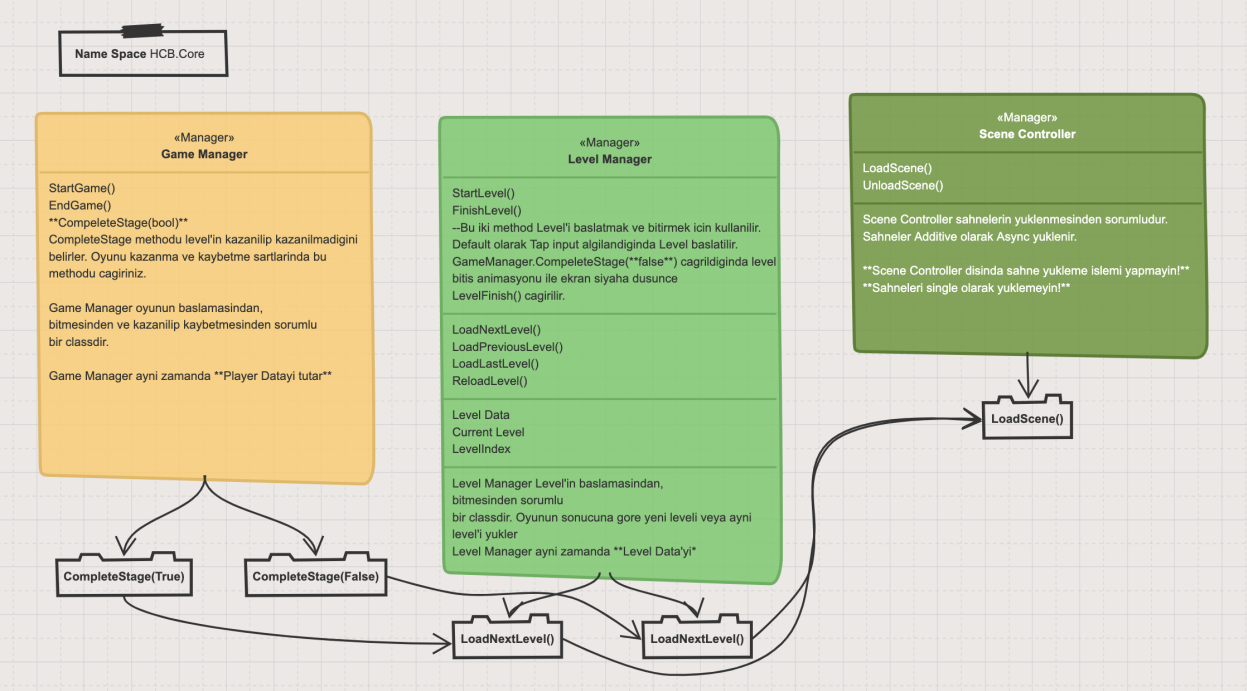
Hyper Casual Base Core

Created	@October 7, 2021 5:14 PM
Created By	
Last Edited By	
Last Edited Time	@October 11, 2021 2:50 AM
Stakeholders	
Status	In Progress 🙌
Type	Architecture Overview

Description

This is the core component of the Hyper Casual Base Project. Hyper Casual Base and all of its components follow the SOLID principles and Observer Patterns. Be sure to understand what they are and how they are being used in order to use this package. Core components contain the base architecture to create the game loop and handle analytics, ads, and other common tasks.

Diagram



Core Components

To have a game loop you must have the following components in the game

- GameManager.cs
- LevelManager.cs
- SceneController.cs
- SaveLoadManager.cs

Common Task Handlers

These components handle SDKs, analytics, ads, and so on.

- **AnalyticsManager**
  - LogEvent()
    - Sends custom events to analytic SDKs
- **Haptic Manager**
  - Haptic(HapticType)

        - Vibrates the mobile device with different vibrations

- **Input Manager**

  Handles input with various options

    - Input Direction
        - This is a Vector2 variable that is edited externally. By default, Joystick modifies the input direction.
        - You should always keep your input from the logic. Having a class that modifies the input is always a good idea.
    - Swipe Input
        - Input Manager supports 8 directional swipe inputs. By default, 4 directions are enabled. Go to InputManager and enable UseEightDirections bool.
        - Input Manager also gives swipe delta as Vector2.
        - OnSwipeDetected(Swipe, Vector2) event will give you the last detected swipe's direction and delta.
    - Tap Input
        - Tap input is detected when tapping on the screen.
        - Tap input ignores tapping on UI objects
        - OnTapInput triggers when tap Detected
    - Tap and Hold
        - Tap and Hold input is detected when finger down detected.
        - Tap and Hold Ignores tap holding on UI objects
        - OnTapDown and OnTapUp events trigger when detected.
    - Touch Input
        - Touch input will provide the information of how many fingers are being touched to screen
        - OnTouch event will trigger.

- **Exchange Manager**

  The exchange Manager takes care of the earning and spending of the game currency.

  - DoExchange(ExchangeType, int)
    - This method updates the currency data in player data. You can handle the earnings and spending from the same method, just make sure to send negative numbers when spending currency.
  - GetData(ExchangeType)
    - This method will return how much currency of type player has. You can use it to check if the player has enough money to buy an item etc.

- **Save & Load Manager**
  - This Manager works in the background and most of the time there is no need to interfere with this manager. Game Manager automatically loads the data using this manager and saves it when the game is closed.

Miscellaneous Components

Interfaces

Some interfaces require additional functions that need to be done in the background. If you are implementing an interface from the base you will like to inherit from "**InterfaceBase**". It's a monobehavior that does some job for the interfaces. You can do what you will do regardless.

Note that **OnDestroy** method is being used by the **InterfaceBase** component. Be advised if you are going to use the method.

Singleton.cs

This is the base class to create singletons in the base. All Singletons inherit from this script.

If you are going to create a singleton, you should inherit from this script.

Player Data

Player data is a serialized class that allows us to keep track of the player's progress. If you are going to save some data related to the player's data you should save it here.