Tiny Muncher! Information

Repository: https://github.com/Goltred/Unity-2DEater

Description

Tiny Muncher is a 2D that challenges the player to beat their high score. The game features simple controls and pixel art graphics, with the main character “Roach” running from side to side eating food that falls from the sky, acquiring different amount points depending on the type of food eaten.

Main Features

- Pixel art style
- Simple left/right controls: Use A/D or Left Arrow/Right arrow to move “Roach” around
- Multiple food types, each with their own point value and visual graphic
- Randomized food drops, size and music tracks. No two gaming sessions will be the same!

Implementation details

Easy to change game settings

The game uses ScriptableObjects to store data relevant to game settings and food data. Adding a new setting is as easy as creating another GameSettingsSO and then setting up the values you would like your level to have.
The GameManager keeps a reference to the selected Game Settings, making it easy to swap the asset for new configurations.

Event System

The code makes heavy use of an event system that allows us to detach the game design from code.
Creating a new event is as simple as creating a new “GameEvent*” asset, and then using said object to trigger the event when they are needed.
Another set of scripts: “EventListener*” matches these events, allowing you to set in the editor which game object/script needs to listen to a specific event.

This allows designers to easily connect new features to existing events. For example, an event listener could be used when OnCooldownEvent is triggered by the Game Manager, to execute some effects that could happen in the last seconds of the level. This is currently used by the MusicManager object, who will start fading out the in-game music when the countdown event is triggered

Gameobject pooling

The game will reduce the amount of gameobjects being instantiated by using a simple pooling system in **EdibleSpawner**.
The pooling system stores a dictionary with the gameobjects that have been created for a specific food id. Once the same type of food id is requested to be spawned, we first check if there’s a free gameobject to re-use. If there’s no free objects then a new one is instantiated.

Flexible play area

The code that drives food spawning and character movement is not limited by gameobjects, but by the camera viewport instead.
This allows you to expand the game area, or play with the background, without having to worry about setting up extra objects to retain functionality.

Music Manager Queue

The Music Manager handles 3 different lists of music tracks, one for the “Main Menu”, another one for in-game and another one for the “Game Over” state.

During Main Menu and Game Over, the manager will handle music tracks in a queue, so that there will always be something playing in case that users decide to spend extra time in these screens.

Main Menu animation

To provide some extra flavoring in the main menu, an animation was created to display “Roach” playing around with some food.
This animation is also programmed to replay every one minute, making sure that something always happens in there in case that players decide to stay in it.