Save = K key
Load = L key

Interface

- IJsonSaveable

Classes

- SaveController
- TransformSaver
- TransformData
- ScoreKeeper
- ScoreData

Saving of player and enemies transforms implementation

The player can press K to save all data and L to load the last saved data. The SaveController script is responsible for detecting these inputs, then searching the level for all monobehaviors that implement the IJsonSaveable interface, and calling their Save or LoadSave functions. Player and enemy transforms are saved using the TransformSaver script, which is added as a component on anything that needs to save and load its transform. The TransformSaver class contains a public subclass called TransformData, which is used to store the saveId along with its associated position, rotation, and scale values. TransformSaver also implements the IJsonSaveable interface, requiring it to implement a saveID, a SaveData function, and a LoadData function. Upon saving, the TransformSaver assigns the values in TransformData with the current object's transform values and saveID, returning that information back to the SaveController script, which then writes that information to the json file created with the Application.persistentDataPath. Upon loading, the SaveController will read from that json file, and apply the TransformData to the object with the matching saveID.

Saving of score implementation

The Save Controller is also responsible for saving the last score. This is done in ScoreKeeper, which has a subclass called ScoreData. ScoreData has a score variable that is always assigned the score from ScoreKeeper whenever it is instantiated. I also made a new directory for the score binary file to be saved in the Save Controller. Whenever Save is called in the Save Controller, if a monobehaviour is a ScoreKeeper, a new instance of ScoreData is created with the saved score and is written into a binary file. On load, the score data updates to the old saved version of the score data, and updates the ScoreKeeper score to show up in game.