DTT
APPS WEB GAMES

SINCE 2010

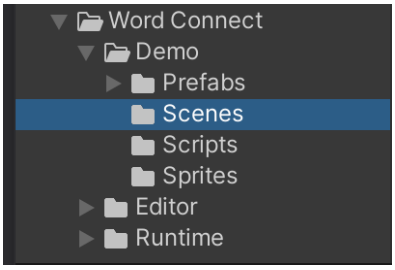
Word Connect

Documentation | 24-10-22

Table of Contents

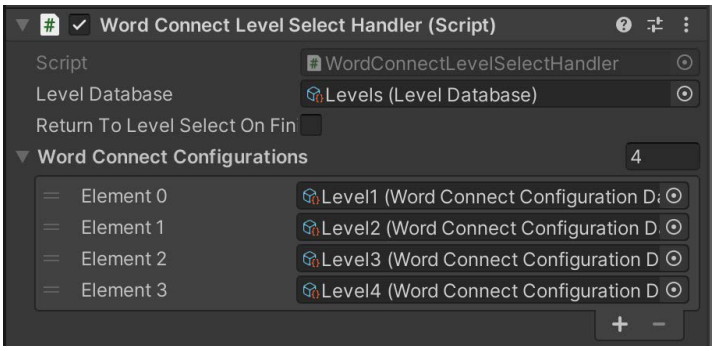
1. Get started quickly

To get started, open either the “Demo - Landscape” or “Demo - Portrait” scene in the Demo folder and press play.

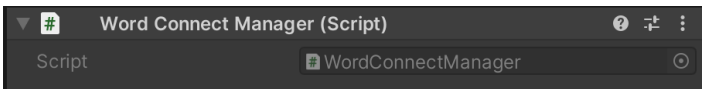


Part of the demo scene is a **WordConnectManager** object. This object holds important components that help manage the game.

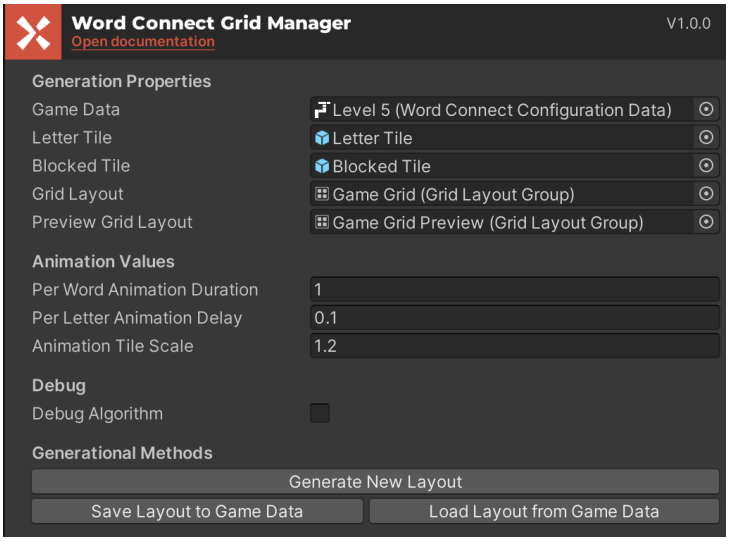
In the **WordConnectLevelSelectHandler** component, you can modify your levels, which are saved as configuration scriptable objects.



If you are not using the **WordConnectLevelSelectHandler**, you can change which level is currently active in the **WordConnectGridManager** component by changing the ‘Configuration’ variable to a specific level.

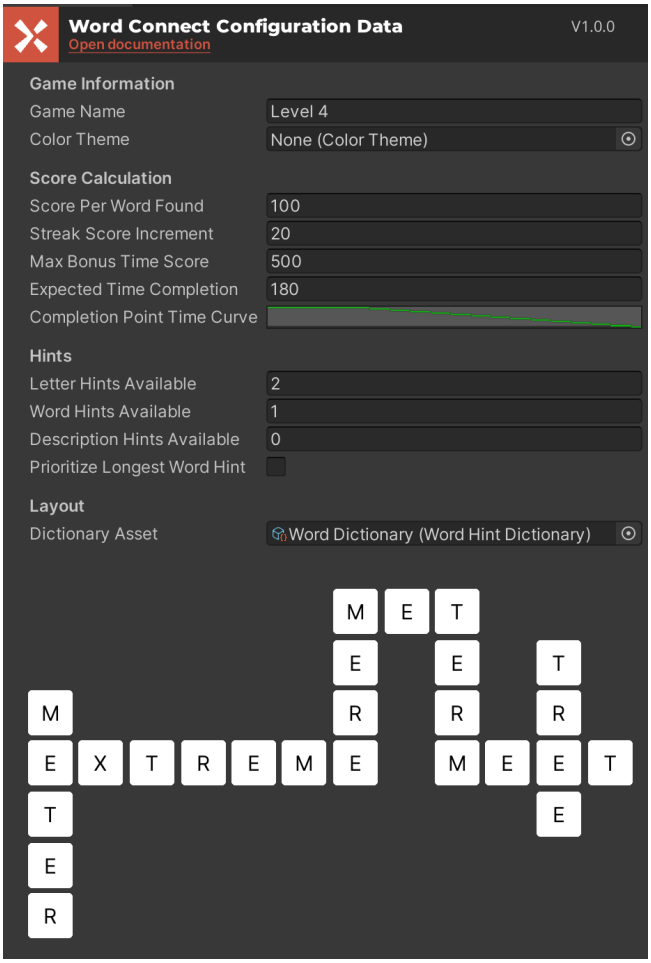


In the WordConnectGridManager component, you can customize the game and UI elements used in the demo, and connect other relevant game objects and components, like the prefabs for the grid tiles.

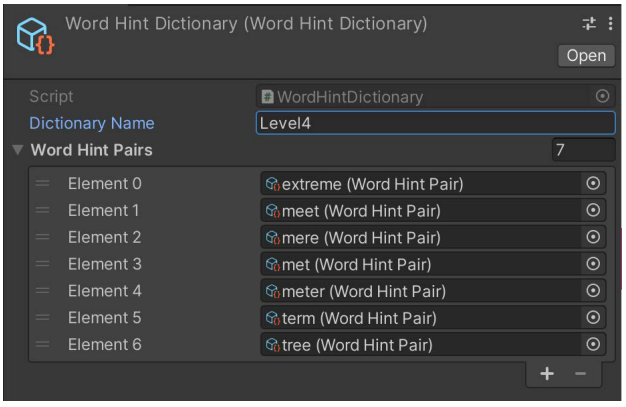


2. Introduction

DTT Word Connect is a Unity asset that allows you to easily implement a Word Connect minigame as part of your larger project. The asset allows for the customization of gameplay elements like prefabs and data as scriptable objects.



It supports generating puzzles by providing a dictionary asset with the words you would like to include. The WordConnectGridManager will then try to generate a layout with as many of these words as possible,



3. Set-Up

Down below is a basic procedure described on how to implement the asset in a working example, without using the demo layout.

1. Setting up all parts of the WordConnectConfigurationData. This step can be skipped by using the premade data from the template demo.
    a. Create multiple WordHintPair objects in your assets by clicking **Create/DTT/Word Connect/Word Hint Pair** in the context menu of your project and fill in the values to your liking.
        i. Aim to create pairs with common letters in order to improve layout generation and reduce the number of letter inputs.
        ii. See chapter 4. *Editor* for more information on its properties.
    b. Create a new WordHintDictionary object in your assets by clicking **Create/DTT/Word Connect/Word Hint Dictionary** in the context menu of your project and add in the newly created WordHintPair objects.
    c. Create a new WordConnectConfigurationData object in your assets by clicking **Create/DTT/Word Connect/Config** in the context menu of your project and fill in the values to your liking.
        i. See chapter 4. *Editor* for more information on its properties.
2. Create a **Canvas** object in your scene and create the following **GameObjects** as children to the **Canvas**.
    a. Create two empty GameObjects with the **GridLayoutGroup** and **ContentSizeFitter** component.
        i. The **GridLayoutGroup** will allow you to format the game grid. You can modify the padding, cell size and spacing, while the other settings are set by the WordConnectGridManager.
        ii. The **ContentSizeFitter** will automatically size the **GameObject** to match the size of the crossword puzzle. The horizontal and vertical fit should be set to the preferred size option.
        iii. One of these objects will hold the preview grid, while the other will hold the grid when the game is running.
    b. Create an empty **GameObject** with the LetterInputUI component. This **GameObject** will contain and generate the input UI.
        i. When adding the LetterInputUI component, the InputHandler component should be added automatically. If not, add this

manually. This component will handle the input for the specified input platform variable.

ii. This component requires a Letter Input Button Prefab and an Input Line Prefab. An example of these can be found in the project at **Word Connect > Demo > Prefabs > Input System**

iii. Fill in the other values of this component to your liking.

3. Create a manager GameObject with the components WordConnectLevelSelectHandler, the WordConnectGameManager and the WordConnectGridManager. This **GameObject** will be the entry point for the asset and will handle the overall flow of the minigame.
    a. Fill in the values of these components to your liking, keeping in mind that the previously created **GameObjects** in the **Canvas** will be used for some of the parameters.
    b. The WordConnectGridManager requires a Letter Tile Prefab and a Blocked Tile Prefab. An example of these can be found in the project at **Word Connect > Demo > Prefabs > Grid**
    c. See chapter 4. *Editor* for more information on its properties.
4. If you are using a custom WordConnectConfigurationData not provided from the Demo, set the GameData parameter in the WordConnectGridManager inspector to your new data object. Then under the Generational Methods portion of the inspector, generate game layouts until it results in one you like. If you wish to save the layout for later use, click the 'Save Layout To Game Data' button on the WordConnectGridManager. The WordConnectConfigurationData object will now contain the layout data.
    a. See chapter 4. *Editor* for more information on its properties.
5. Create a Button or other equivalent element, and have it call the StartGame method in the WordConnectGameManager, passing in a WordConnectConfigurationData object.
6. Play your scene and click the Button or other equivalent element to begin the game.

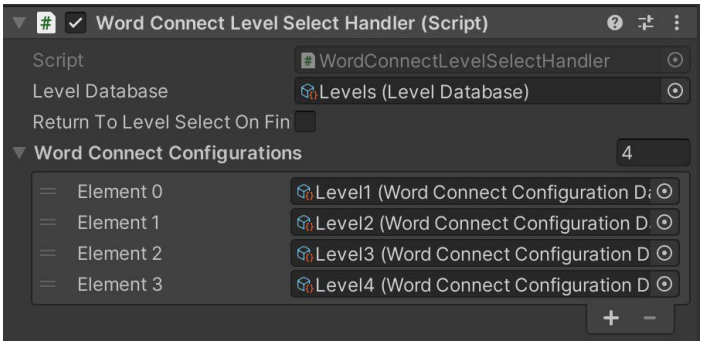
4. Editor

Core components that are essential to the functioning of the minigame are the WordConnectLevelSelectHandler, the WordConnectGameManager, the WordConnectGridManager, and the LetterInputUI components.

Then there are essential ScriptableObjects called WordHintPair, WordHintDictionary and WordConnectConfigurationData.

Components

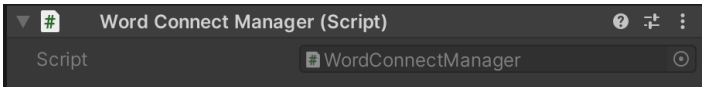
The WordConnectLevelSelectHandler component allows you to modify your levels, which are saved as configuration ScriptableObjects.



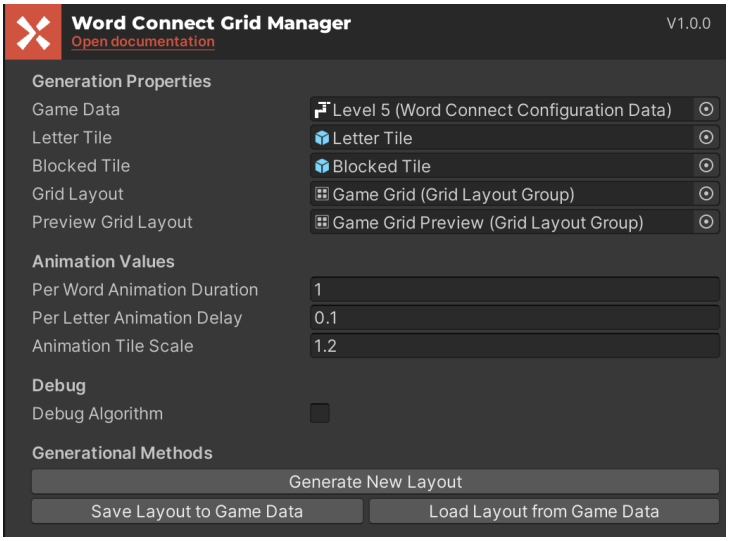
The properties in this component are defined and used as follows:

- **Level Database:** The database of levels, their score and whether they are unlocked.
- **Return to Level Select On Finish:** Whether the game should automatically return to the level selection screen upon the Finish event being called.
- **Word Connect Configurations:** List of data for each level in order. This should be used in combination with the LevelDatabase scriptable object.

The WordConnectGameManager component does not contain any variables which can be changed. Its sole purpose is to control the overall flow of the game. It does however have functions which can be called from a UI button.



The WordConnectGridManager component allows you to customize the game grid generation and UI elements used in the game, and connects other relevant game objects and components, like the prefabs for the grid tiles.

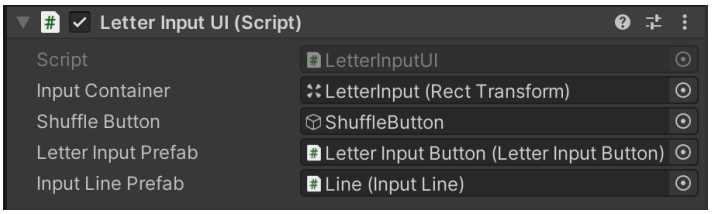


The header properties in this component are defined and used as follows:

- Generation Properties: This section covers prefab data that is to be used as well as active scene objects for the crossword puzzle to generate in.
    - Game Data: The data object for the game that is actively being played or modified. Should be set here when editing and generating, but handled by the WordConnectGameManager when playing.
    - Letter Tile: The prefab being used for letter tiles of the crossword.
    - Blocked Tile: The prefab being used for blocked tiles of the crossword.
    - Grid Layout: The scene grid layout where letter tiles will be generated and automatically formatted for the interactable game puzzle.
    - Preview Grid Layout: The scene grid layout where letter tiles will be generated and automatically formatted as a crossword puzzle preview

- **Animation Values**: These values dictate the values of animation when a tile has been found or hinted at.
    - **Per Word Animation Duration**: The duration of an entire word animating.
    - **Per Letter Animation Delay:** The delay between each individual letter animation within an entire word animation.
    - **Animation Tile Scale:** The scale a letter tile should animate to in the animation.
- **Generational Methods**: This section covers functionality related to crossword data modification.
    - **Debug Algorithm:** Toggle that allows users to see the order of words used or skipped in the creation of the generated crossword.
    - **Generate New Layout**: Will generate a preview layout based on the currently selected WordConnectConfigurationData, which contains a dictionary.
    - **Save Layout to Game Data:** Will save the currently generated and previewed layout to the WordConnectConfigurationData object, allowing it to be loaded and played.
    - **Load Layout from Game Data:** Will load the game layout from the currently selected WordConnectConfigurationData object and display it in the preview area of the scene.

The **LetterInputUI** component allows you to customize the input UI elements.

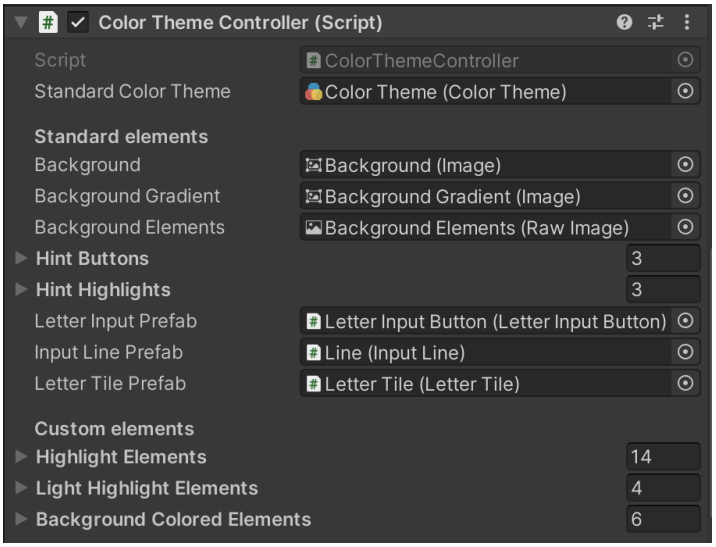


The properties in this component are defined and used as follows:

- **Input Container:** RectTransform object which will hold the letter input buttons.
- **Shuffle Button:** GameObject in the scene which will shuffle the letter input positions (optional)
- **Letter Input Prefab:** The prefab used to generate the letter input buttons.

- **Input Line Prefab:** The prefab used to generate lines between active letter input buttons.

The **ColorThemeController** component allows you to customize the colors of UI elements.



The properties in this component are defined and used as follows:

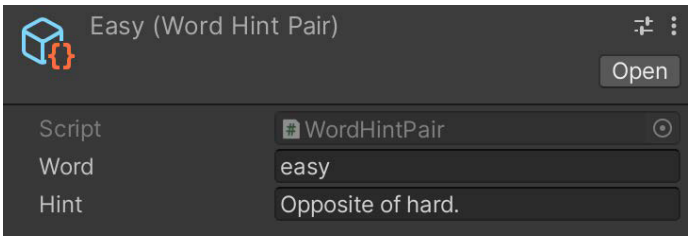
- **Standard Color Theme**: The color theme to use if none has been specified in the WordConnectConfigurationData
- **Standard Elements**: This section covers the standard UI elements which will be colored at runtime.
    - **Background:** The background image.
    - **Background Gradient**: The gradient image applied over the background
    - **Background Elements:** The small moving details applied over the background.
    - **Hint Buttons:** List of button backgrounds.
    - **Hint Highlights:** List of secondary button backgrounds which contain the buttons content.
    - **Letter Input Prefab:** Prefab to change the colors of the letter inputs.
    - **Input Line Prefab:** Prefab to change the colors of the input lines.
    - **Letter Tile Prefab:** Prefab to change the colors of the letter tiles in the grid
    - **Letter Input Prefab:** Prefab to change the colors of the letter inputs.

- **Custom Elements**: This section covers the standard UI elements which will be colored at runtime.
    - **Highlight Elements:** List of image components which will be colored with the set highlight color.
    - **Light Highlight Elements**: List of image components which will be colored with the set light highlight color.
    - **Background Colored Elements:** List of image components which will be colored with the set background color.

This can change the feel of each level. Two ColorTheme Scriptable Objects are included in the Demo:

Scriptable Objects

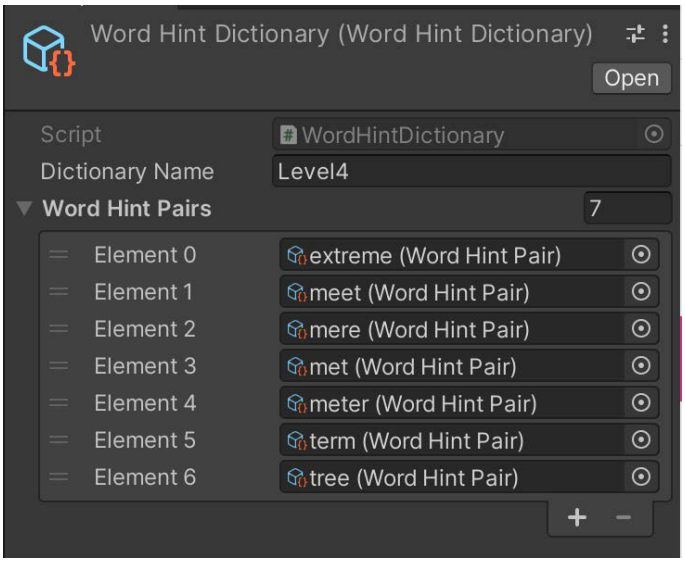
The **WordHintPair** scriptable object allows you to customize individual words and their associated hints.



The header properties in this scriptable object are defined and used as follows:

- **Word**: The individual word associated with the object's hint.
- **Hint (Optional):** The individual hint associated with the object's word. Only necessary if descriptive hints are desired in-game.

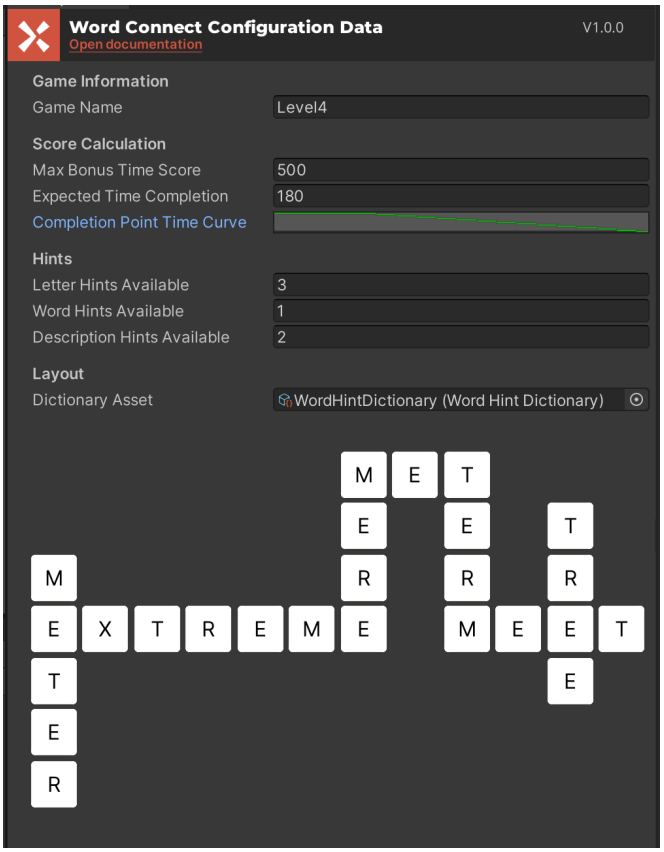
The **WordHintDictionary** scriptable object functions as a collection of individual words and their associated hints, allowing for easy use and reuse of a dictionary.



The header properties in this scriptable object are defined and used as follows:

- **Dictionary Name:** The name of the dictionary object.
- **Word Hint Pairs:** The collection of WordHintPair objects that make up the dictionary.

The **WordConnectConfigurationData** scriptable object allows you to customize specific puzzle data and store game layouts for future loading.

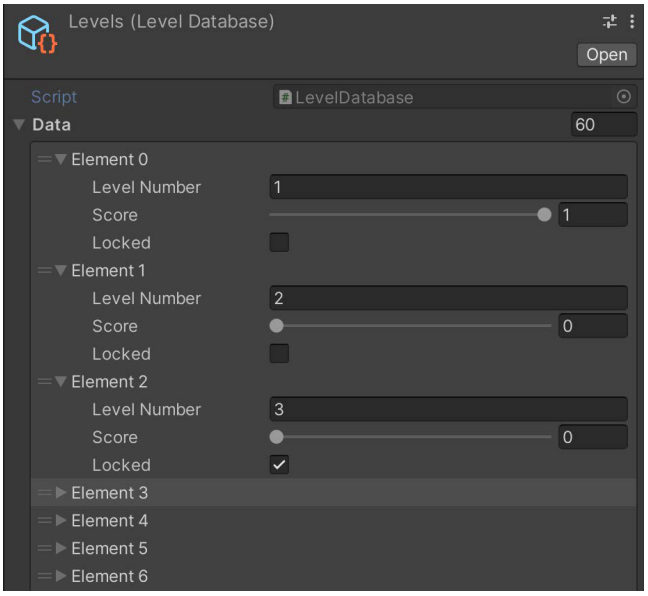


The header properties in this scriptable object are defined and used as follows:

- **Game Information:**
  - **Game Name:** The name of the puzzle itself.
- **Score Calculation:**
  - **Max Bonus Time Score:** The maximum amount of bonus points the player can get based on the completion time curve.
  - **Expected Time Completion:** How long in seconds the puzzle is expected to take.
  - **Completion Point Time Curve:** An AnimationCurve which allows for an advanced way of modifying how the bonus time score decreases over time.
- **Hints**
  - **Word Hints Available:** the number of word hints available in this game.
  - **Letter Hints Available:** The number of letter hints available in this game.
  - **Description Hints Available:** The number of descriptive hints available in this game.
- **Layout**

- **Dictionary Asset:** The WordHintDictionary associated with the puzzle and to be used in layout generation.
- **Layout preview:** If a layout has been generated and saved by the GridManager a preview of the game layout will be displayed here.

The **LevelDatabase** scriptable object holds the global data of a level.



The header properties in this scriptable object are defined and used as follows:

- **Data:** Array that defines individual level data.
  - **Level Number:** The number associated with the saved level.
  - **Score:** A user's score achieved for this level on a scale of 0 to 1.
  - **Locked:** If the level is considered locked and therefore not playable.

5. API

WordConnectManager

This class acts as the entry point for the Word Connect minigame.

Property Name	Type	Description
Configuration	WordConnectConfigurationData	The configuration file for a level that will be used in the game.
WordConnectState	WordConnectState	Class which holds all data of the currently active game.
AvailableLetters	List<LetterInput>	Whether the game is active.
gameTimer	Stopwatch	Is used to track the current amount of time spent on a game.
IsPaused	bool	Is true when the game is paused.
IsGameActive	bool	Is true when the game has started and isn't finished.

Event Name	Argument(s)	Description
Initialized		Called when all the game settings and configuration have been initialized.
BuildGame		Called when the WordConnectGridManager can start building
Started		Called when the game has started
StateUpdated	WordConnectState	Called when a change occurs on the WordConnectState.
Paused		Called when the game is paused.
Finish	WordConnectResult	Called when the game has finished

Method name	Return Type	Parameters	Description
StartGame	void	WordConnectConfigurationData	Starts the game using the given level data.
Pause	void		Pauses the minigame.
Continue	void		Continues the minigame.
ForceFinish	void		Forces the game to finish.
ForceStop	void		Forces the game to stop.
ForceRestart	void		Forces the game to restart.
HandleLetterInput	void	LetterInput	Checks whether the letter input is valid and handles accordingly.
RevealLetterHint	void		Reveals a single letter in the game.
RevealWordHint	void		Reveals an entire word in the game.
RevealDescriptiveHint	void		Reveals a description of a word in the game.
SubmitWord	void		Submits the current Input and checks whether the input is a word in the game.
CheckGameComplete	void		Checks whether all words in the game have been found.
GetAllLettersFromWords	List<char>	List<string>	Returns all letters necessary to spell all words in the given parameter.
SetAvailableLetters	void	List<char>	Sets the available letters which can be used in this game to spell words.

WordConnectGridManager

Class that handles the logic behind generating and managing grid elements.

Method name	Return Type	Parameters	Description
GenerateNewVectorLayout	void		Generates one possible game layout and saves it for displaying.
ChangeGameConfiguration	void		Changes the active game configuration.
LoadLayoutFromGameData	void		Loads the current/active layout from the game data object.
SaveLayoutToGameData	void		Saves the current/active layout to the game data object for future use.
BuildActiveLayouts	void		Builds the active and preview layouts and manages the relevant listeners.
SetupLayoutSettings	void		Correctly sets the Grid Layouts Group to function with the current algorithm.

WordConnectState

Class which holds all the data of the currently active game.

Property Name	Type	Description
CurrentWordInput	List<LetterInput>	List of letter input that has been registered in order.
WordsInCrossword	List<string>	List of all words in the current. (found or not)
CorrectlyAddedWords	List<string>	List of all words that have been found.
HintOptions	List<HintOption>	List of all available hint options.
CompletelyHintedWords	List<HintOption>	List of fully hinted words.

CurrentRevealedDescriptiveHint	HintOption	The current active descriptive hint.
WordHintBalance	int	The number of hints the player has left which reveal an entire word.
LetterHintBalance	int	The number of hints the player has left which reveal a single letter.
LetterHintBalance	int	The number of hints the player has left which reveal a description of a word.

Method name	Return Type	Parameters	Description
AddLetterInput	void	LetterInput	Pushes the LetterInput onto the CurrentWordInput list.
RemoveLastLetter	void		Removes the last element of the CurrentWordInput list.
ClearInput	void		Clears the input sequence.
GetCurrentWordInput	string		Retrieves the LetterInput sequence in string format.
SetAvailableWords	void		Loads the current/active layout from the game data object.
SetHintBalance	void	int, int, int	Sets the balance of all hint options available in this game. (letter hint, word hint and descriptive hint)

6. Support and feedback

If you have any questions regarding the use of this asset, we are happy to help you out. Always feel free to contact us at:
unity-support@d-tt.nl

(We typically respond within 1-2 business days)

We are actively developing this asset, with many future updates and extensions already planned. We are eager to include feedback from our users in future updates, be they 'quality of life' improvements, new features, bug fixes or anything else that can help you improve your experience with this asset. You can reach us at the email above.

Reviews and ratings are very much appreciated as they help us raise awareness and to improve our assets.

DTT stands for Doing Things Together

DTT is an app, web and game development agency based in the center of Amsterdam. Established in 2010, DTT has over a decade of experience in mobile, game, and web based technology.

Our game department primarily works in Unity where we put significant emphasis on the development of internal packages, allowing us to efficiently reuse code between projects. To support the Unity community, we are publishing a selection of our internal packages on the Asset Store, including this one.

More information about DTT (including our clients, projects and vacancies) can be found here:

https://www.d-tt.nl/en/