DTT
APPS WEB GAMES

SINCE 2010

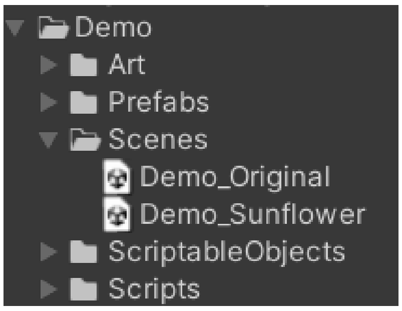
Hangman

Documentation | 13-07-22

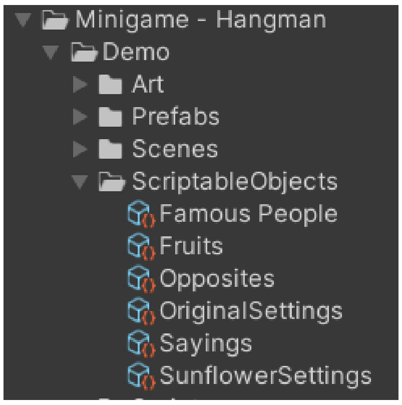
Table of Contents

1. Get started quickly

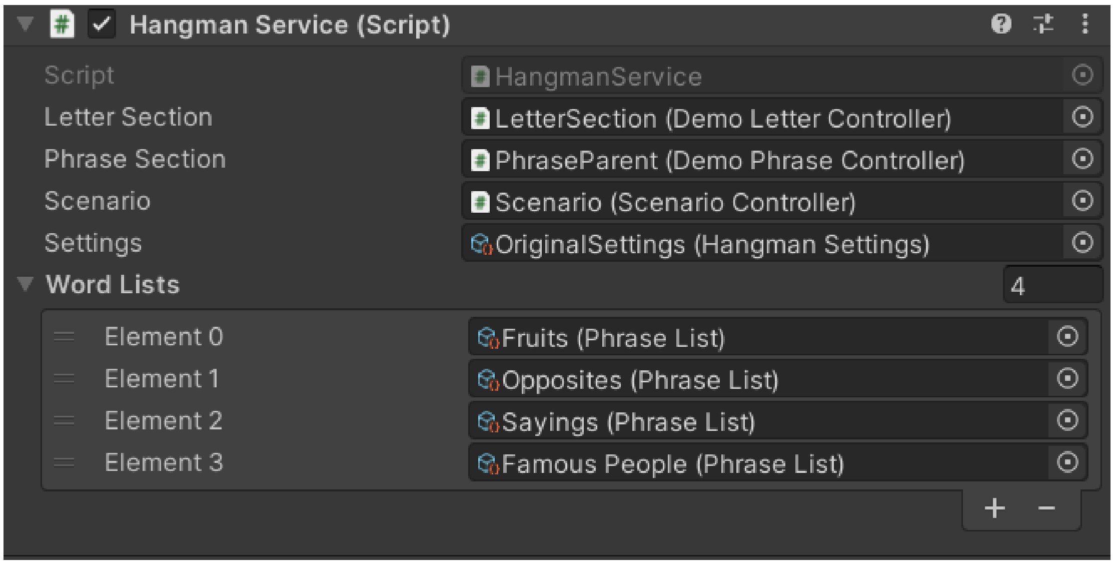
To get started, open either the “Demo_Original” or “Demo_Sunflower” scene in the Demo folder and press play.



This same folder contains assets for the respective scenes. You can edit the settings for the game here:



Part of the demo scene is a HangmanService object. This object holds a HangmanService component. On it you can change things like section references, settings and the phrase


Hangman Service (Script)

Script	HangmanService
Letter Section	LetterSection (Demo Letter Controller)
Phrase Section	PhraseParent (Demo Phrase Controller)
Scenario	Scenario (Scenario Controller)
Settings	OriginalSettings (Hangman Settings)
Word Lists	4
Element 0	Fruits (Phrase List)
Element 1	Opposites (Phrase List)
Element 2	Sayings (Phrase List)
Element 3	Famous People (Phrase List)

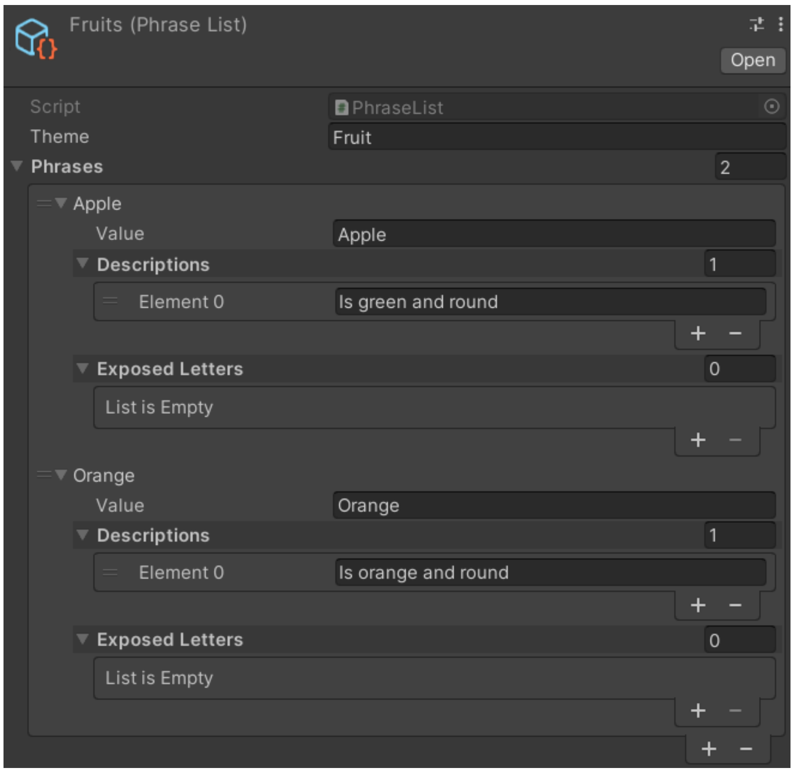
2. Introduction

The hangman minigame template is a minigame template that makes it possible to implement a hangman minigame as part of a larger project. The template is set up in a way that you are able to easily create and implement your own degenerative scenario like the hangman scenario.

The hangman scene is split up in three sections by default: The scenario section, the phrase section and the letters section.

The scenario section paints the scenario which has a degenerative theme to it. This might be a hanging man slowly revealing itself or a flower slowly losing its petals. The phrase section displays the words to be guessed by the user and the letters section holds all the letters the user can pick from to guess the phrase.

A settings object can be created, using the **Create/DTT/Hangman/Settings** menu option, which holds some options for your hangman minigame in addition to an editable list of phrase lists. Phrase lists are scriptable objects that can hold multiple phrases and can be created using the **Create/DTT/Hangman/PhraseList** menu option.



```
Script                  PhraseList
Theme                   Fruit
Phrases                                         2
  Apple
    Value               Apple
    Descriptions                                1
      Element 0         Is green and round
    Exposed Letters                             0
      List is Empty
  Orange
    Value               Orange
    Descriptions                                1
      Element 0         Is orange and round
    Exposed Letters                             0
      List is Empty
```

A phrase can have descriptions usable as hints and exposed letters which are used internally but can also be used to mix up gameplay by exposing the **&** character for example to have a phrase be opposites (e.g. Fire & Ice).

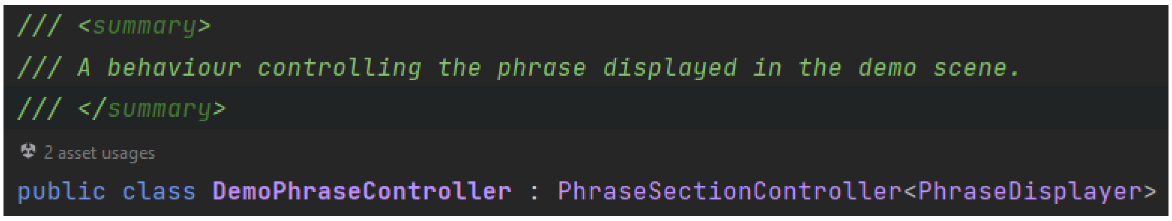
3. Set-Up

Sections

To get started you need to add the three default game sections to your project. Each one will be described here.

Phrase Section

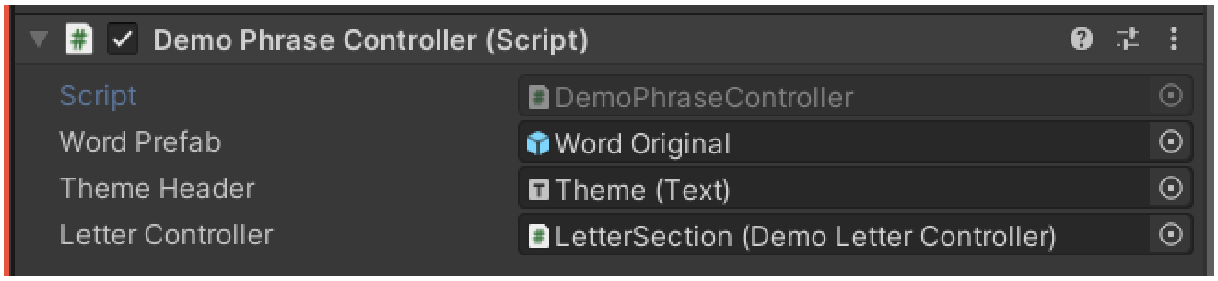
The phrase section will contain all words that form the phrase the user has to guess. You start by creating a class that derives from the generic PhraseSectionController<T> class. Where T is the type of phrase displayer you want to use. By default you can use the included PhraseDisplayer class but a custom implementation is also possible.



```
/// <summary>
/// A behaviour controlling the phrase displayed in the demo scene.
/// </summary>
2 asset usages
public class DemoPhraseController : PhraseSectionController<PhraseDisplayer>
```

An example implementation of the original hangman phrase section controller.

Add your custom phrase controller to a gameobject in your scene and make sure that all fields in the inspector are set.



Demo Phrase Controller (Script)

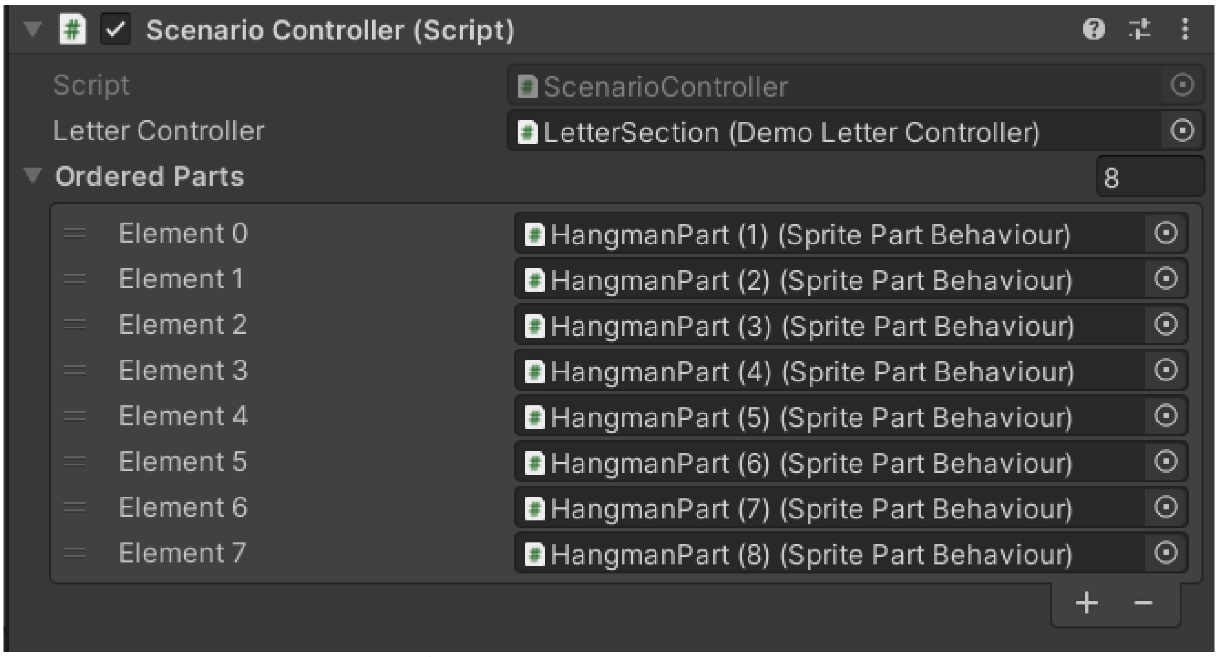
Script	DemoPhraseController
Word Prefab	Word Original
Theme Header	Theme (Text)
Letter Controller	LetterSection (Demo Letter Controller)

An example implementation of the original phrase section controller inspector.

Examples of a phrase section implementation can be found in both Demo_Original and Demo_Sunflower scenes.

Scenario Section

The scenario section will contain scenario parts that can be unlocked when the user has chosen an incorrect letter. Start by adding a ScenarioController component to a gameobject in your scene and make sure the inspector fields are set up correctly.


Scenario Controller (Script)

Script	ScenarioController
Letter Controller	LetterSection (Demo Letter Controller)
Ordered Parts	8
Element 0	HangmanPart (1) (Sprite Part Behaviour)
Element 1	HangmanPart (2) (Sprite Part Behaviour)
Element 2	HangmanPart (3) (Sprite Part Behaviour)
Element 3	HangmanPart (4) (Sprite Part Behaviour)
Element 4	HangmanPart (5) (Sprite Part Behaviour)
Element 5	HangmanPart (6) (Sprite Part Behaviour)
Element 6	HangmanPart (7) (Sprite Part Behaviour)
Element 7	HangmanPart (8) (Sprite Part Behaviour)

An example implementation of the original hangman scenario controller.

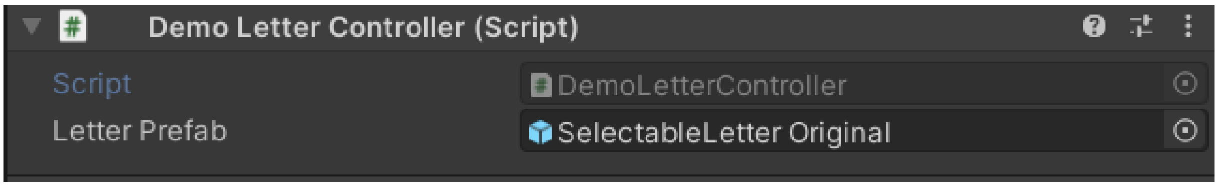
The scenario controller is made to be flexible for inheritance but it is not necessary.

An example of a scenario controller implementation can be found in the Demo_Sunflower scene.

Letter Section

The phrase section will contain the letters the user will be able to choose from when guessing the phrase.

Start by creating a class that derives from the abstract LetterSectionController class. You can now add it to a gameobject in the scene and start adding the required inspector fields.

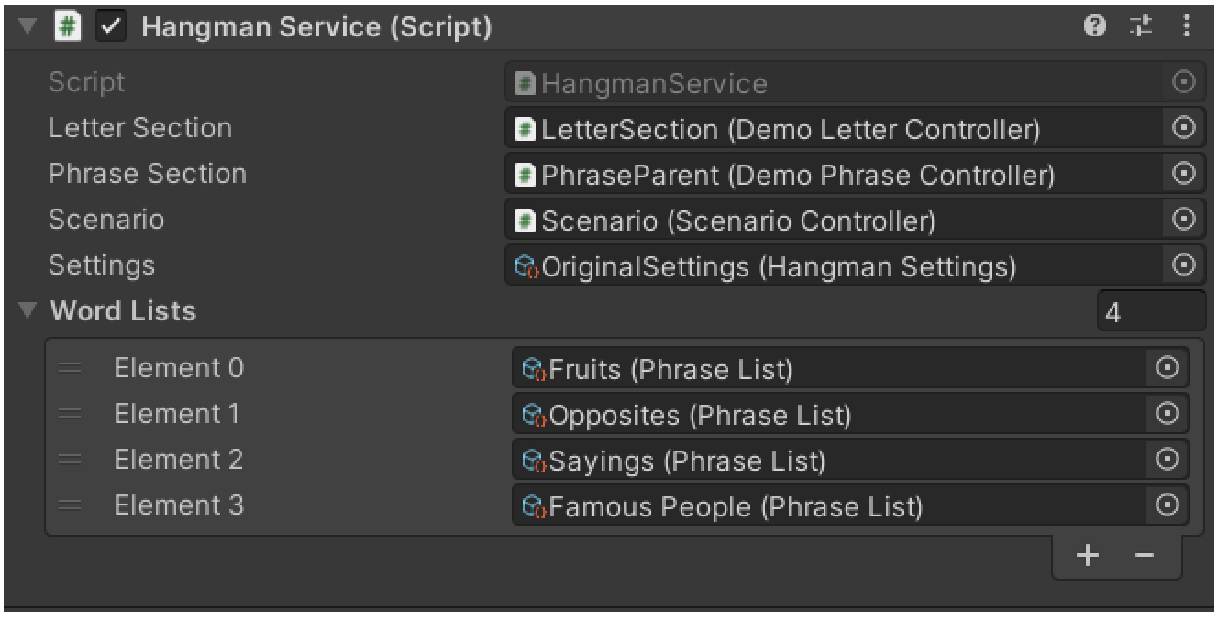

Demo Letter Controller (Script)

Script	DemoLetterController
Letter Prefab	SelectableLetter Original

An example of a letter section controller inspector in the original hangman demo scene.

The Hangman service

After creating the required sections for the game you can add the HangmanService component to a game object in the scene and start filling in the inspector fields.


Hangman Service (Script)

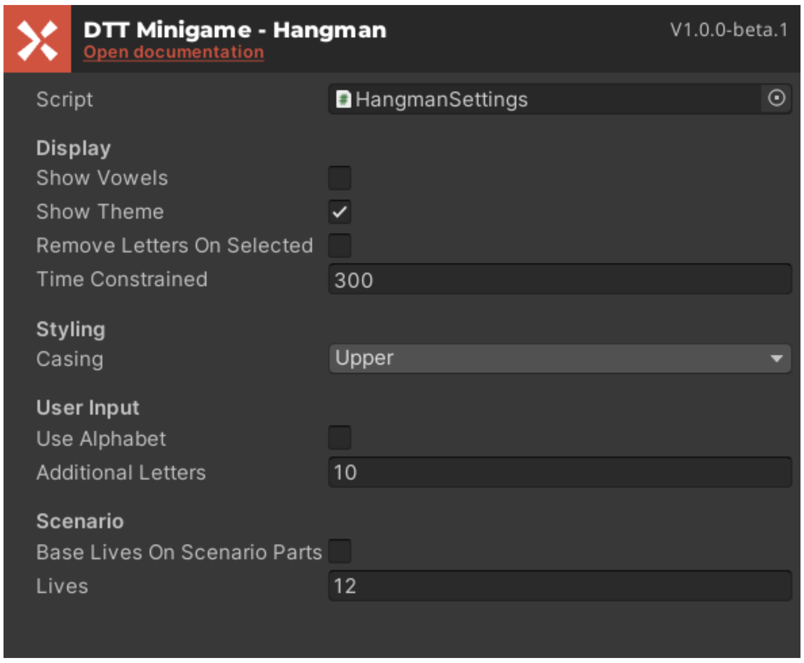
Script	HangmanService
Letter Section	LetterSection (Demo Letter Controller)
Phrase Section	PhraseParent (Demo Phrase Controller)
Scenario	Scenario (Scenario Controller)
Settings	OriginalSettings (Hangman Settings)
Word Lists	4
Element 0	Fruits (Phrase List)
Element 1	Opposites (Phrase List)
Element 2	Sayings (Phrase List)
Element 3	Famous People (Phrase List)

An example implementation of the hangman service inspector in the original hangman demo scene.

The HangmanService component is made to be flexible for inheritance so a custom implementation can also be used.

4. Editor

Hangman Settings


DTT Minigame - Hangman
Open documentation
V1.0.0-beta.1

Script	HangmanSettings
Display	
Show Vowels	☐
Show Theme	✓
Remove Letters On Selected	☐
Time Constrained	300
Styling	
Casing	Upper
User Input	
Use Alphabet	☐
Additional Letters	10
Scenario	
Base Lives On Scenario Parts	☐
Lives	12

1. **Show Vowels:** Whether to show the vowels at the start of the game.
2. **Show Theme:** Whether to show the theme the word belongs to.
3. **Remove Letters On Selected:** Whether letters should disappear after being clicked.
4. **Time Constrained:** The time constrained in seconds. Setting this to a value other than 0 will tell the game service to use a timer.
5. **Casing:** The casing to use for letters in the game.
6. **UseAlphabet:** Whether to use all letters of the alphabet to choose from.
7. **Additional Letters:** The amount of additional letters to use when not using the alphabet to choose from.
8. **Base Lives On Scenario Parts:** Determines whether the scenario parts should be generated based on the amount of lives set, or set the amount of lives to the scenario parts set up in the scene.
9. **Lives:** The amount of lives the user has visualized in the scenario degeneration process.

5. API

HangmanService

Provides the core gameplay services usable for a hangman game.

Property Name	Type	Description
Settings	HangmanSettings	The settings currently being used for the game.
IsPaused	bool	Whether the game is paused
IsGameActive	bool	Whether the game is active

Event Name	Argument(s)	Description
Finish	HangmanResult	Fired when the game has finished
Started		Fired when the game has started
Finished		Called when the game has finished
Paused		Called when the game is paused.

Method name	Return Type	Parameters	Description
StartGame	void		Starts the game using the default settings
StartGame	void	HangmanSettings	Starts the game using custom settings
RestartGame	void		Restarts the game using the currently active settings
RestartGame	void	HangmanSettings	Restarts the game using the currently active settings

AddSection	void	IGameplaySection Controller	Adds a section to the hangman service to receive callbacks.
Pause	void		Pauses the minigame.
Continue	void		Continues the minigame.
ForceFinish	void		Forces the game to finish.
Cleanup	void		Clears the game sections.

7. 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/