Unity Module for Deig

Henrik Engström

Version 1.0, 2024-08-27

First of all, the Unity module for Deig is highly customized to support a production process where the game logic is modeled in Deig. Unity is used to create production quality assets with animations, graphics, audio etc. The game logic is not modeled in Unity. Moreover, in the typical case, scenes are not created in Unity at all. All editing is done on prefabs that are loaded when the game is started.

The association between Deig entities and Unity prefabs is handled by *Scriptable Objects*. If an entity has not been assigned a prefab, the system will create a prototyping asset that will use the same texture/audio that is used in the Deig editor. This approach makes it possible to gradually replace the prototyping assets with proper prefabs. Changes to the game logic can be made and tested in parallel with the work with prefabs in Unity. Figure 1 shows the production pipeline that has been used for Deig games.

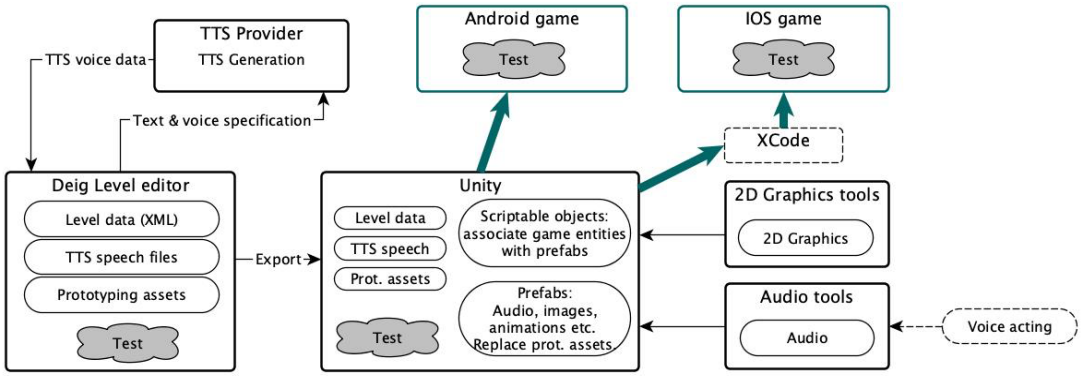

Figure 1. Deig game production pipeline.

Setup

Follow these steps to connect a Deig game with a unity project:

- Install the Deig editor from `http://deig.se`.
- Create a new Unity project. Make sure it is configured as 2D (if not, the importing of textures will fail)
- Install the DeigCore package.
- Open the Deig editor and select the menu item: `Utils->Select Unity catalog`.
- Browse to the location of your unity project and select the root folder of it.
- Export the game from Deig to Unity by selecting the menu item: `Utils->Export whole game to unity`.
- Switch to unity and open the *Deig assignment window*: `Deig->Assignment window`
- Click the "Deig Settings" button (top left in Figure 2). This will open an inspector where you can select the active game.
- Click "Run" (or chose the menu item `Deig->Run`). You will now be able to play the game using the prototyping assets (the same used in the Deig editor). To improve the game you can replace the prototyping assets with Unity prefabs.

Note that the development of the game can happen concurrently in the Deig editor and in Unity. If changes are made to the dialog or game logic in Deig they can be exported to Unity. The associations to prefab elements are stored in scriptable objects in Unity and will be maintained when the game levels change. If new elements are introduced, prefabs will obviously have to be added.

Running the game

A game can be started by pressing the "run"-button, in the Deig assignment window (Figure 2). This will load the scene `Load` (without saving any open scenes so beware). The `Load` scene will then load the scene `Game` which contains all components needed to run a Deig game. The entities of the game will be loaded according to the specifications from the Deig editor (stored in XML files). The prefabs are loaded based on the assignments made in scriptable objects. If an assignment is missing it will load a "prototyping" version of the prefab which more or less is the asset used in the Deig editor.

It is possible to run and build a Deig game without any additional editing in Unity.

Deig Assignment Window

The Deig assignment window is the hub for making associations between Deig entities (locations, characters, audio etc.) and prefabs (Figure 2). Associations are made on a game, chapter or location level.

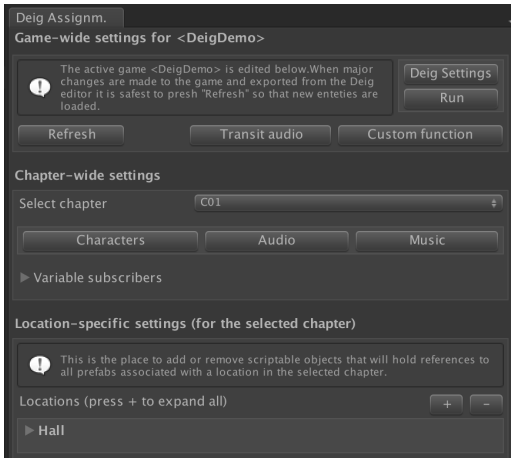

Figure 2. The upper part of the Deig assignment window in Unity.

The most complex case is the associations made for locations. A location has a prefab for the background environment (the graphics, ambience audio etc.) and prefabs for each interactable (one for the audio and one for the graphics). Position of interactables can be adjusted in Unity to enable a fine-grained control of its position (e.g. pixel-perfect alignment with the background). The location prefab will store the offset of images, audio and the interaction zone – for each interactable. This means that the interactable should not be moved in the Deig editor once the Unity editing has started.

Using scriptable objects to create prefabs

Configuring Locations

It is possible to access all scriptable objects from the Deig assignment window. Each location has a section (Figure 3), which shows the elements involved. To edit this location, the button to the left can be clicked (to generate or select the associated scriptable object).

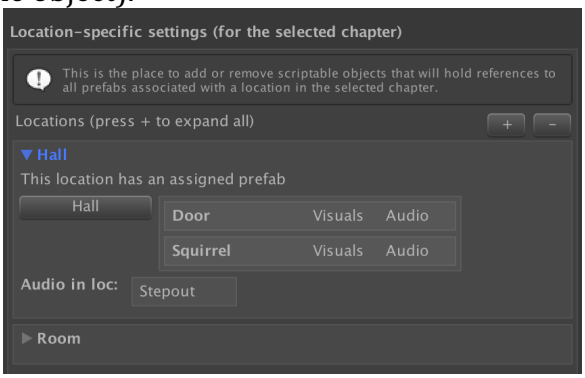

Figure 3. A section of the Deig assignment window where a location is presented.

Location

When the scriptable object has been created (or selected) the inspector shows a panel with a reference to the location prefab (Figure 4) and all involved interactables (Figure 6). It is possible to create a preview scene of the location by clicking the button at the bottom of Figure 4.

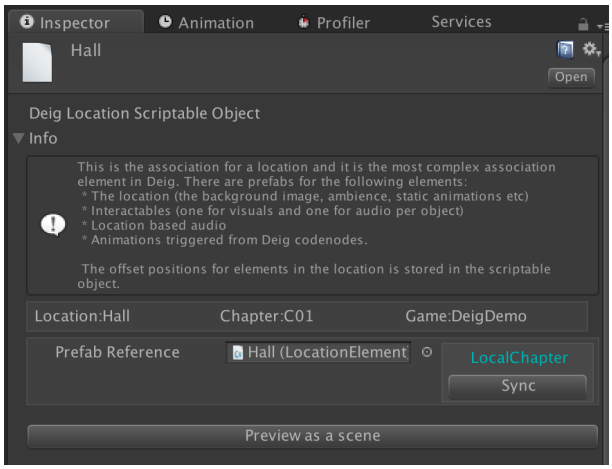

Figure 4. The upper part of the inspector for a location scriptable object.

A location prefab (Figure 5) contains the background image (`Location_ENV`), ambience audio (`Roomtone_AMB`) and other non-interactable properties.

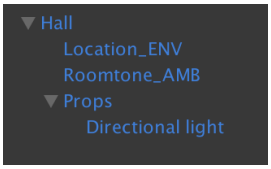

Figure 5. The elements of the location prefab.

Interactables

The interactables in a location will have their own prefabs that can be accessed from the scriptable object for the location (Figure 6).

Figure 6. The middle part of the location scriptable object where interactables are assigned.

For each interactable, there will be one prefab for the audio and one prefab for the visual representation. The scale and position of these prefabs can be adjusted in the inspector or it can be adjusted in the preview scene. The interaction zone has an offset that can be changed in the same way. Note that the interaction zone prefab is shared between all interactables. For this reason there is no `Prefab Reference` for the zone. It can also not be rescaled (the activation zone should be the same for all interactables).

Animations and Location audio

Animations and positioned audio are assigned in the bottom part of the location scriptable object (Figure 7).

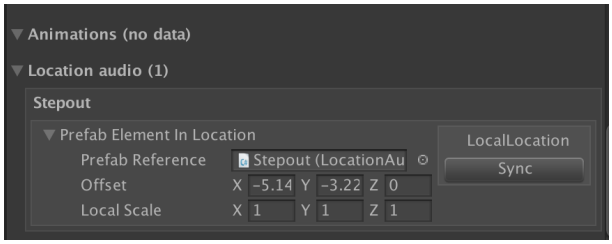

Figure 7. The bottom part of the location scriptable object where location audio and animations are assigned.

Preview Scene

When the preview button in Figure 4 is pressed, a scene with that location is created (Figure 8). As can be seen, all prefabs are instantiated and placed in the hierarchy. The Dummy object is used to enable testing the location with some dummy functions. By pressing the play button it is possible to inspect the location and listen to ambience etc. The logic of the game will however not be run.

Figure 8. The preview scene for a location (Hall)

The prefabs for interactables, location audio and animations can be scaled and repositioned. Note that these changes should not be applied directly to the prefab. Instead you should use the save button that appears in the inspector (Figure 9).

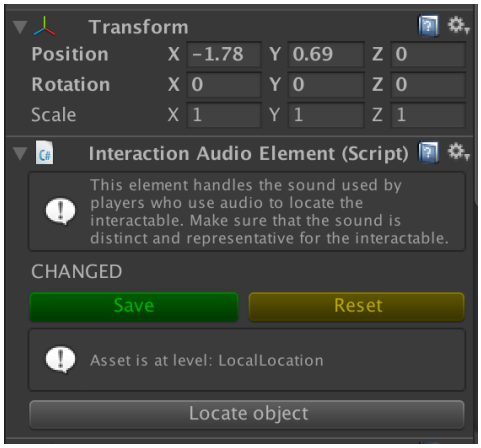

Figure 9. The inspector for a relocated prefab. Note that the save button should be used instead of applying changes to a prefab.

If changes are applied to the prefab, this will cause problems when prefabs are shared (se below).

Note that when an entity has not yet got an assigned prefab, the preview will load a prototyping version. This will have "(autoassigned)" added to its name in the scene hierarchy. It is not possible to adjust positions of such prototyping prefabs.

Configuring other entities

The locations are the most complex entity to configure. In addition, there are a number of entities that has a more straightforward association from their Deig name to a prefab. These are presented below.

Characters

The characters that are shown when a dialog is presented are assigned in the window shown in figure 10.

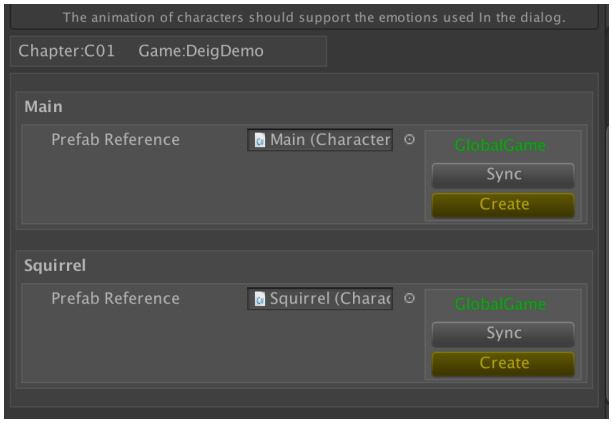

Figure 10. Character association for a chapter.

A character prefab has an animation controller which handles the different emotions that are flagged in the dialog editor (in Deig).

Audio

There is a difference between *location audio* (has a position) and *chapter audio* (is location independent). Chapter audio is assigned by clicking on the middle button in Figure 2 (below chapter selection). Location audio is assigned in the Location scriptable object (Figure 7).

Transit audio is assigned for each door. This is done for all transits in the game. Note that it is possible to have different sounds for each direction between locations (one sound when you go from A to B and another when you go from B to A).

Note 1: The Deig Unity package has a component `AudioRandom` that is used to play audio. This means audio files are not assigned directly to an audio source but are added to the list in the `AudioRandom` class. This class randomly selects sounds to be played which enables a more rich experience. If only one file is added the behavior will be the same as a traditional audio source.

Note 2: Prefabs for audio has an animation that is played. This can be replaced to visualize the sound to players who cannot hear the sound or has disabled it.

Music

Music is associated for the chapter. A music prefab contains audio clip and has some controls for fade time etc.

Custom Functions

Custom functions are handled by a prefab that has a custom function script component. The Deig package contains an example script that illustrates how this can be handled (`TestFunction.cs`).

Variable Subscribers

Variable subscribers is a way to add behavior to Unity objects that is controlled by changes to Deig Boolean variables. By adding a variable subscriber component to a game object it will receive notifications from the Deig engine when that variable is changed. Two types of subscribers are pre-defined that will enable or disable a game object if the variable is true or false respectively. Other types of subscribers can be added.

The variable subscriber inspector (Figure 11) can be used to find and remove subscribers. Note that this panel is looking at all prefabs in the project – not only for the selected game.

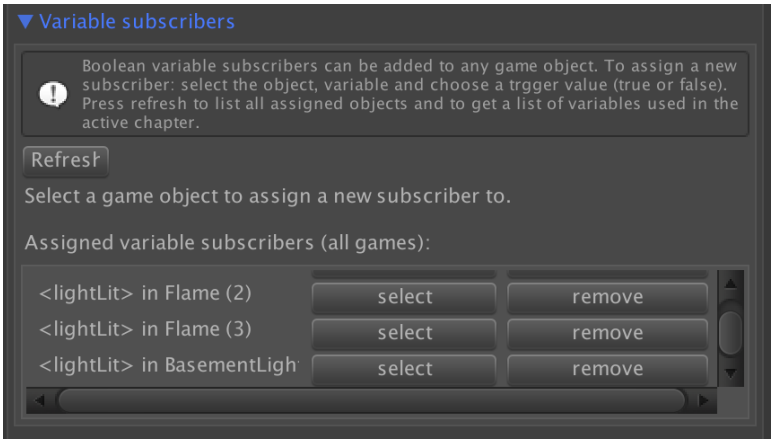

Figure 11. The variable subscriber inspector.

Dialog

Dialog lines are not associated with scriptable objects. They are always loaded from a Resource folder. The prototyping dialog is taken from:
`Assets/Deig/Resources/PrototypingAssets/<gamename>/Audio/Voice`

The final dialog audio is taken from:
`Assets/Deig/Resources/Games Prefab Association/<gamename>/Audio/Voice`

When voice acting has been recorded it should be placed in the latter folder.

What is loaded from a Resource-folder?

All prototyping assets are loaded from the resource folder. This should only be used during development.

All scriptable objects are loaded from a resource folder. They contain references to prefabs. These prefabs do not have to be located in a resource folder.

Note: if you want to use the prototyping (TTS) files in the final game, you can move the Voice folder from `PrototypingAssets` to `Games Prefab Association`. When the final game is built the remaining files in the prototyping folder should be deleted (in order to reduce the size of the built game).

The structure of prefabs

Prefabs can be created for each instance in the game (e.g. for each location in each chapter) but they can also be shared by placing them on a chapter- or game-level. If, for example, the same location is used in several chapters then it is possible to use the same prefab for all chapters. The idea is that it should be possible to create special versions of e.g. a location in one chapter but that other chapters can use one prefab that they share.

Note that the Unity inspector (for scriptable objects) have "sync"-buttons that associates a Deig objects with prefabs. These buttons starts at the most local level and looks for prefabs with the corresponding name. If you want to use another prefab, the association can be changed by dragging the prefab to the associated field (prefab reference).

The scriptable object inspectors will show which level an associated prefab is at. The following levels exists:

- `GlobalGame` - prefabs shared in all chapters of the game.
- `LocalChapter` - prefabs shared in a chapter. These are located in a folder that has the chapter name.
- `LocalLocation` - prefabs that are unique to a location (in a chapter). These are located in a folder that has the location name (which in turn is located in a folder with the chapter name)
- `Custom` - the scriptable object is associated with a prefab that is located outside the default hierarchy.
- `Missing` - no prefab is assigned or the assigned prefab cannot be located.

When the sync-button is pressed in an association (e.g. in Figure 10), the system will start to search at the most local level, and move upwards. If a fitting prefab is found, it will be used. It is possible to drag-and-drop prefabs that are located outside the defined hierarchy. In this case the prefab will be flagged as "Other" in the inspector.

Example

If the Main character prefab should be used for all chapters. You can move the folder:

```
Assets/Deig/Games/<gamename>/Characters/C01/Main
```

to:

```
Assets/Deig/Games/<gamename>/Characters/Main
```

Then you will only have to press "sync" in the character scriptable object in each chapter of the game.

Deig Settings Scriptable Objects

There are some scriptable objects that are global to the Deig environment and used for all games. They are accessible from the Deig Settings (Figure 12)

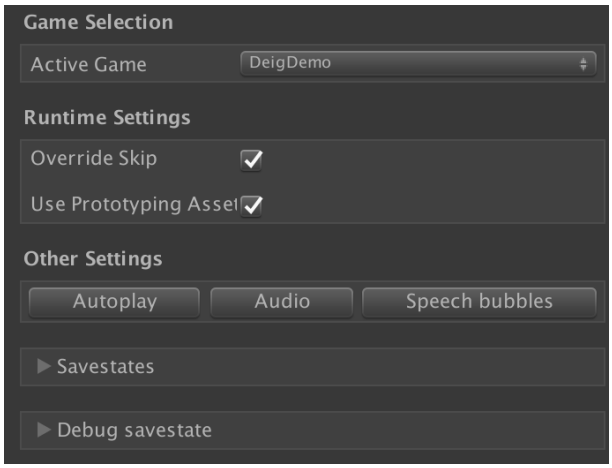

Figure 12. The global Deig settings scriptable object inspector.

The active game dropdown is used to select which Deig game that should be active. If the game is changed you should press "refresh" in the Deig assignment window (Figure 2).

There are two "runtime settings" that can be altered. "Override Skip" will make it possible to skip dialog (by pressing the space bar or by swiping) even if it is the first time it is played. The "Use prototyping Assets" can be unchecked when a game is ready to be shipped (and all elements have got proper prefabs). This will (slightly) increase the performance. Don't forget to delete the folder "`Deig/Resources/PrototypingAssets`". If not, it will be included in the build and waste space.

Autoplay

The autoplay settings are used for testing. By checking autoplay, the game will play automatically (simulating the dragging of a finger). The player is very stupid and performs random choices.

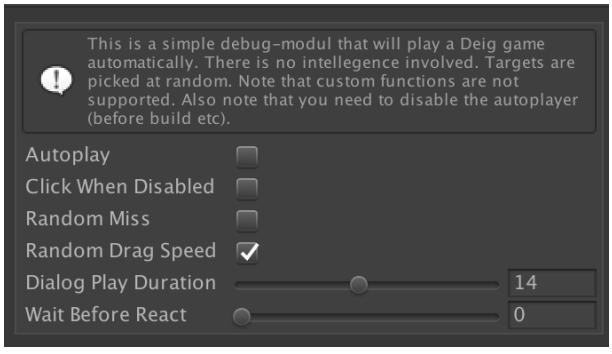

Figure 13. The autoplay settings scriptable object inspector.

Audio
The audio settings scriptable object associate different mixer groups etc.

Speech Bubbles
The speech bubble scriptable object associate speech bubbles for different emotions (and inner dialog if it is the left side).

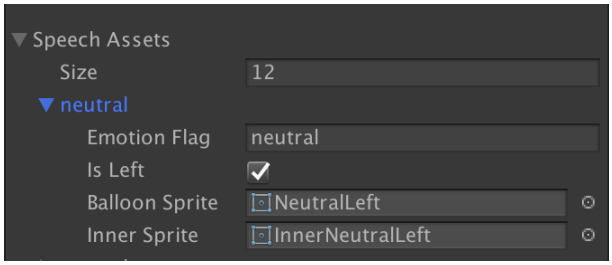

Figure 14. The speech bubbles settings scriptable object inspector.

It is possible to add new emotions and add extend the speech bubble array.

Savestates
A Deig game saves the state to a folder. The inspector in Figure 15 can be used to erase the savefiles or to print the location of savefiles.

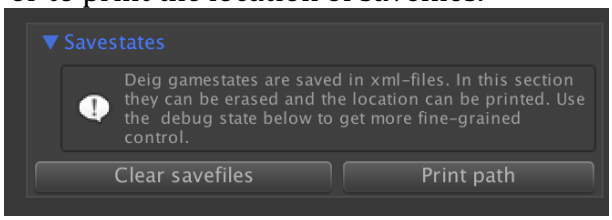

Figure 15. The inspector for savestate (in the Deig settings)

It is possible to launch the game from a specified location (Figure 16).

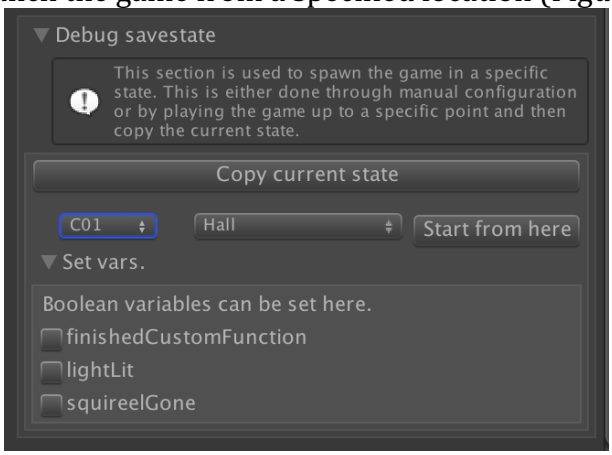

Figure 16. The inspector for configuring the saved state (in the Deig settings)

The button at the top of Figure 16 ("Copy current state") loads the last saved state to the settings. When "start from here" is pressed, the game will instantly launch in this state (without loading the menu chapter). It is also possible to select a chapter and location manually. Boolean variables can be toggled in the inspector.

Example of a fully implemented game

You can install the DeigDemo-package to see an example of a "game" that has been (almost) fully implemented in Unity with animations etc. The voice-acting is not present.

Figure 17. A location from the DeigDemo.

Disclaimers

Note that the usage of animations, custom functions and variable subscribers may need additional testing to make sure the behavior is as expected. In particular you should test to quit the game, restart and continue to make sure the state of e.g. the animation is as expected.

Acknowledgments

The following persons have developed the content in this package:

Arslan Tursic - 2D Art
Henrik Engström - Programming
Linus Nordgren - Programming
Tobias Karlsson - Animation
Per Anders Östblad - Audio & music

The development of Frequency Missing and Marvinter has been funded by the The Swedish Post and Telecom Authority, Sveriges Radio and the University of Skövde.

If you have any questions, please contact henrik@deig.se.

Visit https://deig.se for more information and to download the Deig editor.