Heathen Systems

UIX Keyboard 2017.3

Overview

Heathen Systems Core is the foundation of Heathen Engineering's coded assets for the Unity 3D game engine. While the asset has been developed with Unity 2017.3 and later in mind the code should work with Unity 5.5 and later with little or no modification. Full source code is provided with the asset from the Unity Asset Store; Unity Asset Store is the only authorized distributor of Heathen Systems or any of its related components. Standard Unity Asset Store licensing conditions apply Heathen Systems and its related components, additional provisions will be considered on a case by case bases.

Support

Heathen Engineering Limited provides full support for Heathen Systems and related components as described in this document, on the Unity Asset Store page, the Heathen Engineering corporate site and limited to the versions of the Unity 3D game engine the asset package has been published under unless otherwise noted. You can contact support via Support@HeathenEngineering.com or the Heathen Engineering Support desk (visit HeathenEngineering.com for details).

Table of Contents

Key Concepts

Heathen Systems Core framework overview.

Keyboard

Heathen Systems UIX Keyboard is a mono behaviour which collects and manages the Keyboard Key objects below it in the Game Object hierarchy directing press events in a manner to simulate a keyboard, key pad or any other button array based input device.

Note that Keyboard is language and layout agnostic; that is the number, arrangement and values of the keys it manages are of no relevance to the keyboard its self. Keyboard only represents the keys as a singular object and directs their press events.

Keyboard cannot emulate hardware input; that is pressing a key on UIX Keyboard is no different than pressing any other Unity UI element such as a button. It does not drive events on the operating system nor does it set values in Unity's Input Manager.

Keyboard Output Manager

Heathen Systems UIX Keyboard Output Manager works in tandem with UIX Keyboard and Unity's Event System.

The output manager can direct string output from the keyboard to event calls, Unity Input Fields or any configured string field on any referenced mono behaviour (aka Components).

The output manager can be set to automatically direct output from the keyboard to the last focused Input Field and includes facilities for tracking insert point and selected text values for Input Fields.

Ligature

Ligature is a specialized Library Variable which defines character combinations and the ligature they represent. E.g. (c) becomes © similar to most modern word processors. This is an important facility for multi-language support as many keyboard systems use key combinations to form language characters having to many characters to be fully represented by single key press.

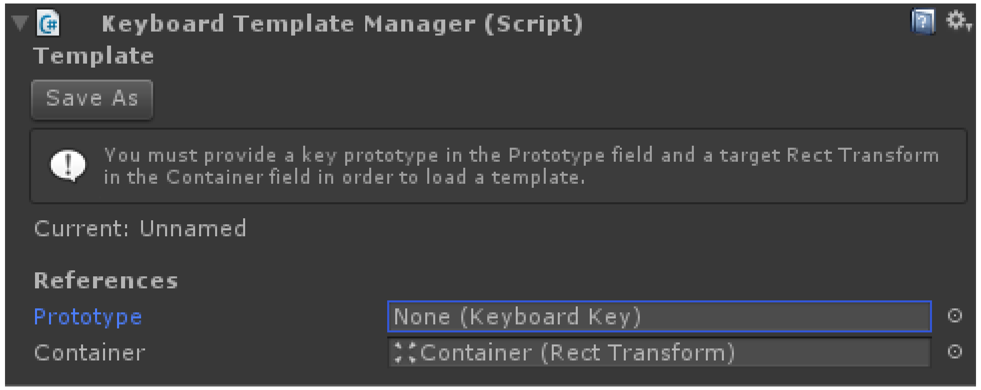
The Ligature Library Variable can be used to generate common ligature sets such as Korean characters, fraction characters and Katakana. To generate these standard sets, click the gear icon in the inspector window (upper right corner).

Creating a Keyboard

Prefab keyboards are available in the SystemUIX/Framework/Keyboard/Keyboard Prototypes folder however creating a keyboard from scratch can be accomplished in just a few steps.

To start with a template

1) Right click in your Hierarchy window or use the GameObject menu to create a new Keyboard (UIX > Composite Controls > UIX Keyboard)
   This will create a new VirtualKeyboard game object in your scene configured with an output manager and a template manager
2) Assign a Keyboard Key to the Prototype field of the Template manager.
   You can select a Key Prototype from the SystemUIX/Framework/Keyboard/Keyboard Prototypes folder or make your own. This is the basic style that will be used for each generated key.



You must provide a key prototype in the Prototype field and a target Rect Transform in the Container field in order to load a template.

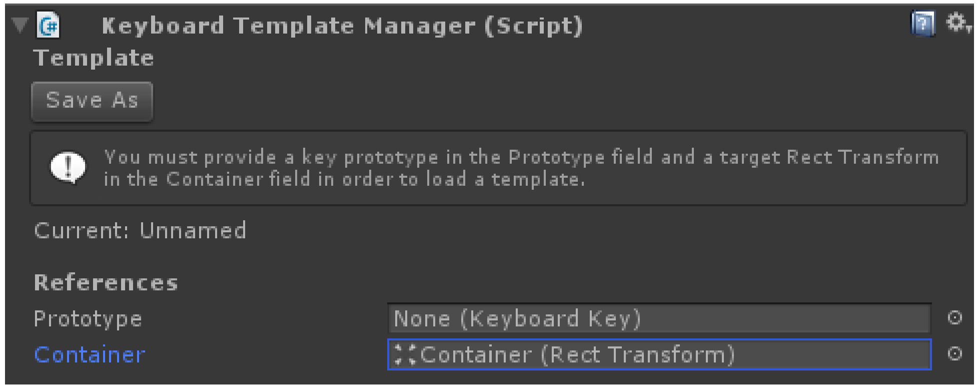
Current: Unnamed

References

Prototype: None (Keyboard Key)

Container: Container (Rect Transform)

3) Assign a transform at or under the Keyboard as the Container.
   This is where generated keys will be instantiated to and must be a child of the Keyboard's game object.



You must provide a key prototype in the Prototype field and a target Rect Transform in the Container field in order to load a template.

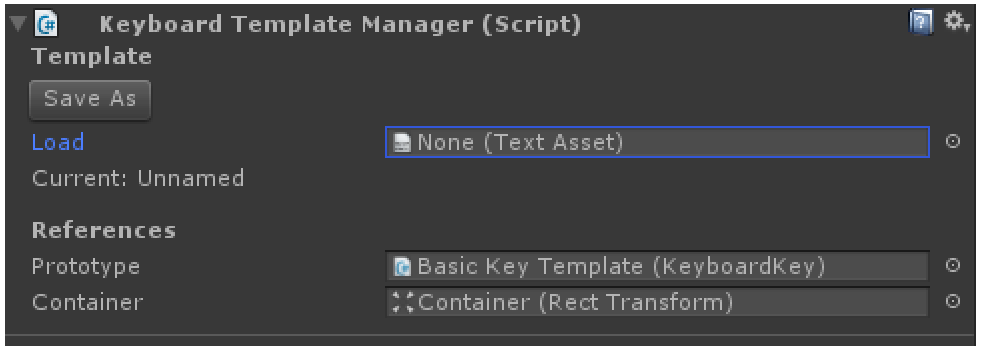
Current: Unnamed

References

Prototype: None (Keyboard Key)

Container: Container (Rect Transform)

4) Select the desired template and slot it in the Load field.
   Once you drop the file the tool will generate the keys and clear the load field.



Keyboard Template Manager (Script)

Template

Save As

Load: None (Text Asset)

Current: Unnamed

References

Prototype: Basic Key Template (KeyboardKey)

Container: Container (Rect Transform)

To start from scratch

1) Right click in your Hierarchy window or use the GameObject menu to create a new Keyboard (UIX > Composite Controls > UIX Keyboard)
   This will create a new VirtualKeyboard game object in your scene configured with an output manager and a template manager.
2) Create a new game object under the Keyboard to serve as your custom key template.
   a. Add the Keyboard Key component to your key template.
   b. Add child game objects to represent the Normal, On Shift, On AltGr and On Shift + AltGr displays
      These will be turned on an off by the keyboard based on the keyboard state. They can be empty transforms but cannot be null (un set)

   At this point it would be wise to save your key template as a Unity Prefab for later use.

3) Now duplicate position and arrange the keys as you see fit
   There are no requirements or restrictions on arrangement; anything supported by Unity is supported by the keyboard.
   For each key
   a. Set the desired Key type
   b. Populate the 'Code' value of the key
   c. populate the Keyboard Key return values for Normal, On Shift, On AltGr and On Shift + AltGr as required.
      Note you can instruct the key to parse return values from the key code in which case you do not need to manually populate the Return Values.
4) Once completed you can save the current configuration to a new template file. This is not required but can save time when creating similar keyboards by prepopulating your keys.