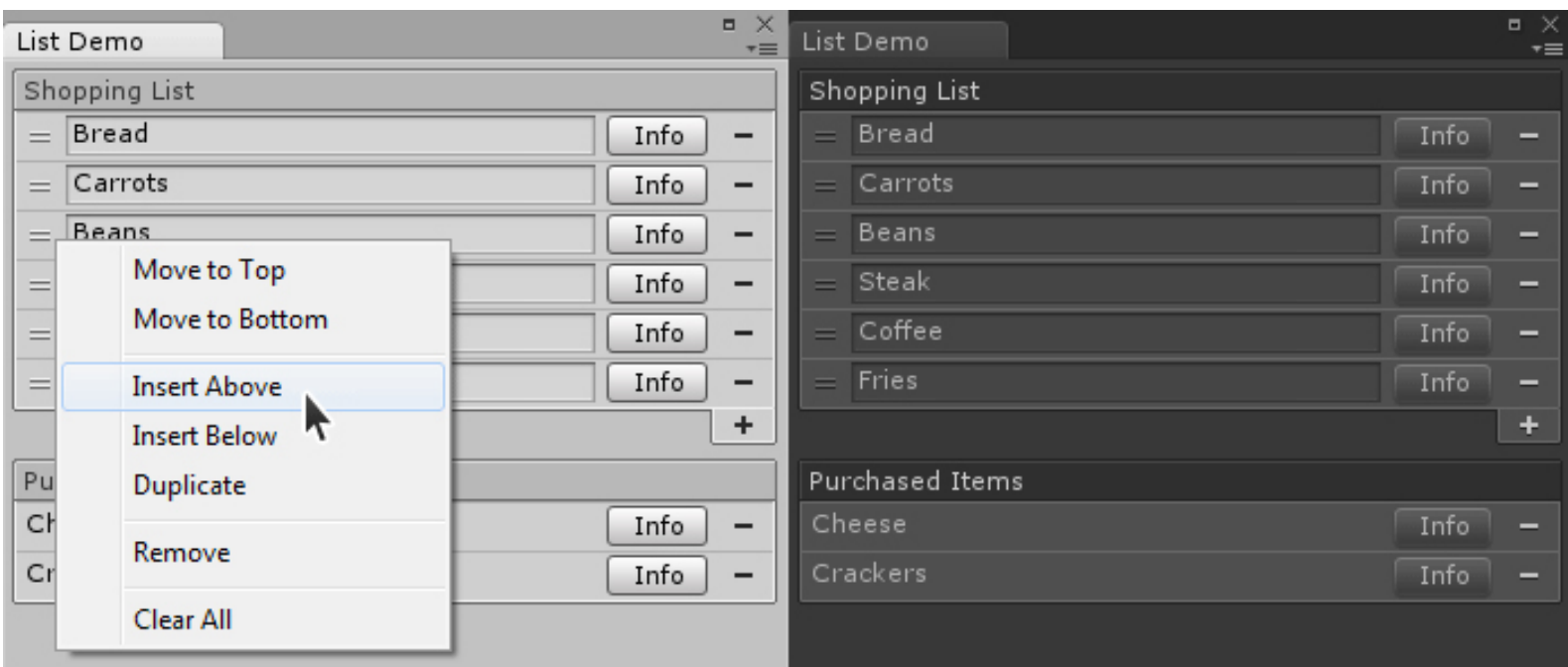


Editor Library for Unity



User Guide

The MIT License (MIT)

Copyright (c) 2013 Rotorz Limited

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

This project is licensed under the MIT license (see LICENSE). To be in the best position to enforce these licenses the copyright status of this project needs to be as simple as possible. To achieve this the following terms and conditions must be met:

- All contributed content (including but not limited to source code, text, image, videos, bug reports, suggestions, ideas, etc.) must be the contributors own work.
- The contributor disclaims all copyright and accepts that their contributed content will be released to the public domain.
- The act of submitting a contribution indicates that the contributor agrees with this agreement. This includes (but is not limited to) pull requests, issues, tickets, e-mails, newsgroups, blogs, forums, etc.

Key Features 1

Updating Library 2

Submission to the Unity Asset Store 2

`IList<T>`**`SerializedProperty`**

This guide provides some information regarding this library. Refer to the included API reference for documentation and usage examples (see "`API Reference.chm`").

The source code for this library can be found in the BitBucket repository (<https://bitbucket.org/rotorz/reorderable-list-editor-field-for-unity/overview>). Please use the issue tracker to report bugs and feature requests. contribute to this project unless you accept the terms of the contribution agreement.



Keep regular backups of your files regardless of which extensions you are using. Always be sure to review release notes and backup all relevant files before installing or updating any assets including Rotorz Reorderable List Field.

-
- Drag and drop reordering of list items!
 - Easily customized using flags.
 - Subscribe to add/remove item events.
 - Supports mixed item heights.
 - Disable drag and/or removal on per-item basis.
 - Styles can be overridden on per-list basis if desired.
 - Subclass list control to override context menu.

Always backup your files and read through release notes before updating to the latest version!



Watch [Reorderable List Editor Field for Unity](#) on Bitbucket to keep up to date with the latest!

Each update may contain new features and of course bug fixes. No software is ever perfect and Rotorz Reorderable List Field is no exception. Rotorz Limited strives to zap bugs as quickly as possible.

Rotorz Reorderable List Field is an editor library for the Unity software and as such is limited by the capabilities of Unity itself. Occasionally updates to Rotorz Reorderable List Field are necessary to workaround changes that have been made to the API exposed by Unity; often to take advantage of yummy new features!

We strive to keep the update process as simple as possible, though manual steps are sometimes necessary to take full advantage of the latest update.

The contents of this package (Rotorz Reorderable List Field) can be included in custom packages which are submitted to the asset store. We ask that asset publishers use the latest version of the package to avoid conflicting with other assets which make use of this library.



It is important that the license and documentation files are included and remain intact.

- Ensure that license and documentation files are included and remain intact. It should be clear that these relate to the reorderable list field library.
- Copyright and license information must remain intact in source files.
- Change the namespace to something unique and DO NOT use the name "Rotorz". For example or .
- Place files somewhere within your own asset folder to avoid causing conflicts with other assets which make use of this project.

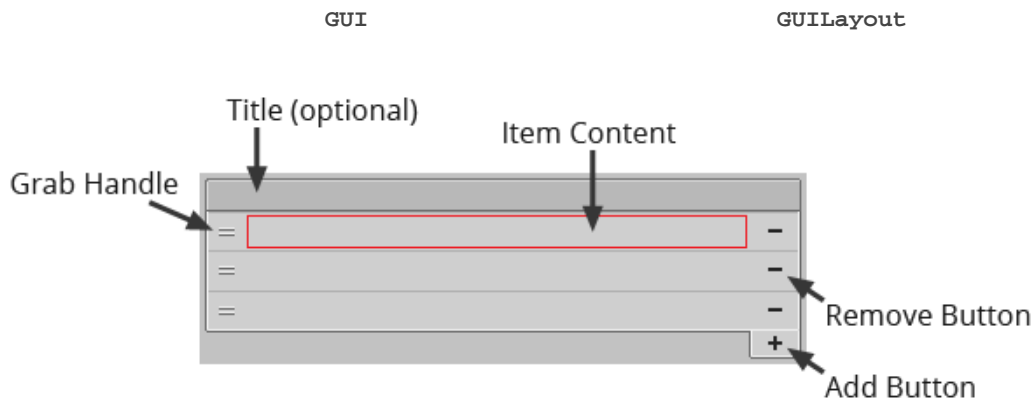


Figure 2-1. List and title field interface with default flags

Can be optionally added using `ReorderableListGUI.Title` accepting either a string or `GUIContent` instance.

Use left mouse button to begin dragging item. Drag operation can be cancelled by pressing the escape key or the right mouse button. Grab handle is primarily for decoration since users can also drag non-interactive areas of an item.

Presented using an item drawer or custom property drawer. Content position is inset slightly providing a comfortable gap between items.

Click to remove list item. Item removal can be disabled for the entire control or on a per item basis.

Append blank element to end of list. Item insertion can be disabled.

Additional commands can be shown by right-clicking on non-interactive areas of a list item. Context menu can be disabled or customized if needed.

Refer to `ReorderableListFlags` to learn about customizations which can be applied easily.

For adding reorderable list controls to your custom editor interfaces with buttons for adding and removing items.

They can be used in almost any editor interface:

- Editor windows.
- Custom inspectors.
- Custom property drawers (see note below).
- User preferences window.



Due to a bug in Unity (Case 568929), `GUI.DrawTexture` does not work properly for custom property drawers of the default inspector. There are no known issues with custom property drawers when nested within custom inspectors or editor windows however!

Yes, absolutely! The packaged version of this library can be consumed from custom UnityScript source code easily.

You can also use the non-compiled source code version of this library if desired. Since this library is implemented in C# it is necessary to place the non-compiled version into the path "Assets/Plugins/ReorderableList" to resolve ordering of compilation.

Yes, and since the library is itself implemented in C#, there are no issues with the ordering of compilation.

Yes, in fact there are a number of flags which can be used to enable and disable things. Refer to the `ReorderableListFlags` enumeration for available flags.

List items are presented via the `IReorderableListAdapter` interface which allows list items to have varying heights.

The serialized property version of this control supports this since custom property drawers provide a provision for height calculation. Though a fixed item height can be explicitly specified for improved drawing performance.

The provided `IList<T>` adaptor only supports fixed item heights. See API reference for example implementation of a custom adaptor with custom height calculation.

