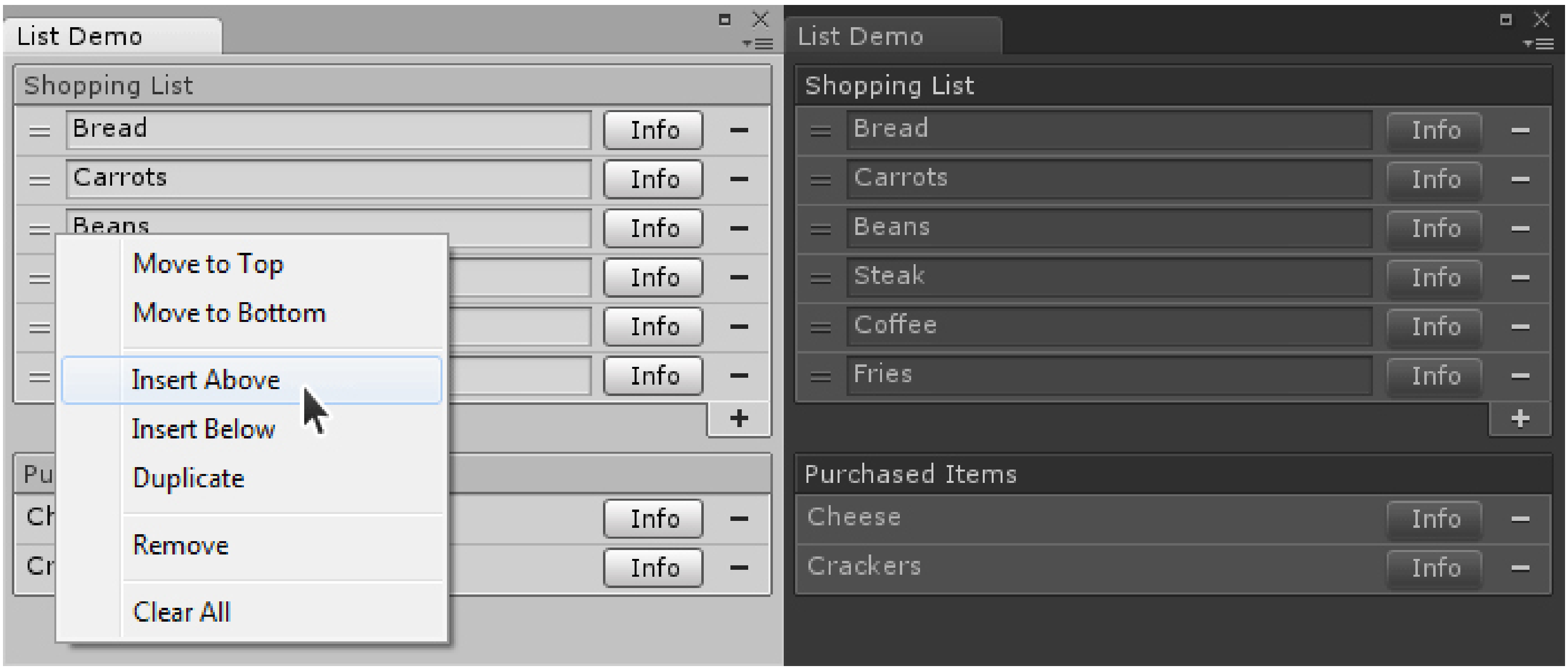
Rotorz Reorderable List Field

Editor Library for Unity



User Guide

2014-11-20

MIT License (Open Source)

Contribution Agreement

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.

Table of Contents

Chapter 1. Getting Started 1
Key Features 1
Updating Library 2
Submission to the Unity Asset Store 2
Chapter 2. Control Interface 3
Chapter 3. FAQ 4

Chapter 1. Getting Started

Rotorz Reorderable List Field is an open source library allowing Unity developers to add reorderable list fields to their custom editor GUIs. This library includes support for `IList<T>` and `SerializedProperty` though custom adaptors can be implemented if needed.

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. **DO NOT** contribute to this project unless you accept the terms of the contribution agreement.

Important

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.

Key Features

- 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.

Updating Library

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

Tip

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 capibilities 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.

Submission to the Unity Asset Store

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.

Important

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

To include a modified version within your package:

- 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 **Rotorz.ReorderableList** to something unique and DO NOT use the name "Rotorz". For example **YourName.ReorderableList** or **YourName.Internal.ReorderableList**.
- Place files somewhere within your own asset folder to avoid causing conflicts with other assets which make use of this project.

Chapter 2. Control Interface

The reorderable list field interface is intentionally simple with an optional title above. Support is provided for absolute positioning (`GUI`) and also for automatic layout (`GUILayout`).

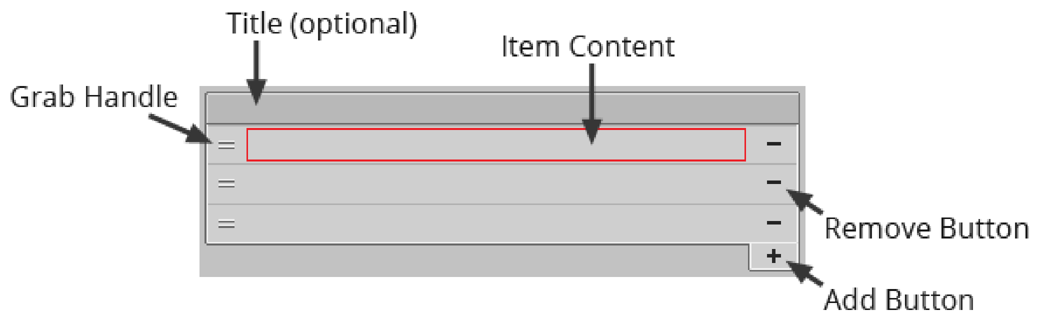


Figure 2-1. List and title field interface with default flags

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

Grab Handle
: 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.

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

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

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

Context Menu
: 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.

Chapter 3. Frequently Asked Questions

What is this for?

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

Where can I use list field controls?

They can be used in almost any editor interface:

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

> **CAUTION**
>
> 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!

Can this library be used with UnityScript?

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.

Can this library be used with C#?

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

Can reordering be disabled?

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.

Can list items have varying heights?

List items are presented via the `IReorderableListAdaptor` 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.

Can this list control work with other collection types?

Custom adaptors can be created by implementing the `IReorderableListAdaptor` interface. Like mentioned above, list items can be of varying heights if desired.

Is undo and redo supported?

The serialized property version of this control provides automatic support for undo and redo. However, undo and redo support must be custom implemented if required for all other versions of this control.

Can the context menu be customized?

Yes, you can subclass the `ReorderableListControl` class and override the method `AddItemsToMenu` to add or replace menu items.

Can list items be selected?

Currently there isn't an included API for this, however in many cases it is possible to implement some form of selection using custom item drawers.

Can this library be included within asset store packages?

Yes, please refer to Submission to the Unity Asset Store on page 2 for further information.