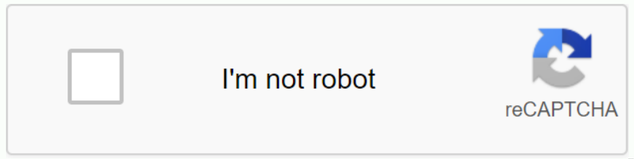

Continue

Unity reorderable list

Unity is a wrapper feature that makes it easy to build a rearranged list of array serailized properties. The order list is an undocumented editor class that visualizes the list in UnityEditor. For more information, see this post. Note this is the array editor that has an array editor start editing example editor assets/examples/editors/ShopMenuEditor.cs: In the Unity5.3.5f1 example, the test sequence list var property = this.serializedObject.FindProperty ArrayEditor is a build visualized by ArrayProperty. var list = re-orderableListUtility.CreateAutolayout (property); Perform layouts // Foldable reorderableListUtiliti.DoLayoutListWithFoldout (list); Or the original list. DoLayoutList() Custom Header var list = ReorderList.CreateAutoLayout (Property, New String[] { {Column1, Column2, Column3}, or Custom Column Width var List = ReorderList.CreateAutoLayout (Property, null, New Float?] create a list of the sequences in which there is. { 100, 100, 100 },); Or combine themvar list = reorderListUtility.CreateAutoLayout (properties, new strings[] { column1, column 2, column3}, new float?[] { 100, 100, 100 },); Or specify the width of the last column and rearrange the other var list = reorderListUtility.CreateAutoLayout (property, new string[] { Column1, Column2, ColumnLast }, new float?[] { null, null, 100 },); License this project is licensed under the Apache license - see LICENSE.md file for more information 2 Global Material Settings Tool For Unity version editing rider API: 5.6 Is interested in creating on-demand listings with custom checkers custom 'events'. Each event has a scriptable object and a timestamp to trigger. Therefore, each item in the list must have a 'custom event' and a float. During my hour-long study, I found that even though they seemed very useful, there was an incredibly low amount of information about these orderly lists. In answer, there is as much information as you can do about this list and how to use it, to help yourself and others. =) Update: Like this article? Then check out other articles on how to create a drop-down filter for Unity Administrator. Over the past five months or so, me and my team have been involved in developing hybrid tower defense and RTS games as part of a school project (please try by clicking on the link!) while developing the game, and we quickly realized that in order to allow us the flexibility to experiment with our level design, we experimented with a class of Unity called order list. If you've read another article here, you've noticed that you typically link unity classes to pages in the Unity scripting reference. This article did not do so here because there was no official document at the time it was written. The reorder list class is submitted under the namespace inside UnityEditor and there is no official documentation. We are It's about, however, because of how useful it is, and because of how little information is currently online about it. The problem with basic folding is that it is typically dependent on one of the inspector's GameObjects to maintain a collection of objects - we declare an array or list, and then release it: // Wave is a strut that will be explored later in the document. Public waves [waves]; This will give us exactly the same results as the above administrators. Public Listings&lt&Wave;Wave; This provides a foldthat you can use to insert elements into the collection. Flimsy basic unity collection folding. However, this folding is a big problem because it cannot be reconfigured in order of elements. The WaveManager script made it easy to manage the spawning of mobs in the game, which caused a big problem. If you can't do something as simple as reordering wave elements, there's no point in this script! To do this, we explored and found other interface options... Solution: Reordering list reorganizing is now much easier. Elements can also be added and removed. ... This solves the team's problems, making the level design process faster and more efficient than ever before. Now i could see the entire array at a glance and adjust it much faster. You can share the way I got mine without more ado, so maybe here because you google about ReorderableList. The post-ad article continues: If you set it first, the data structure of each individual element in your wave array. Because it covers structural types (such as strucks) and enumerations (i.e., enumeration), you can do some reading if you are not familiar with them. By default, struts and enumerations are ways to declare new data types for code. Wave.cs [System.Serializable] Declares the enumeration type of public structure wave {// mob. Public enumeration mob {goblins, mucus, bats} public mobs; // What kind of enemies should be generated from this wave? WaveManager.cs system.If you are using a collection; System. Collections.Use General; Use the Unity engine; Public Class Wave Manager: Mono Behavior {Public Wave [] Waves; } This gives the inspector's basic arrangement folding. I'm here again to take {//&wave&wave&wave&wave&wavemanager arrays are included; serialized property waves; sortable lists will work with a list of rearranged lists. [CustomEditor (WaveManager)]] As long as you specify the class you want to modify the view of the check view, it doesn't matter if the order list is set to do the actions of the settings list to do the actions of the // rearrange list to do its job. However, for organizations, people typically name classes that are modified by a custom editor as a designated editor (and therefore waveManagerEditor). Also, make sure that this class is inherited from the editor instead of the typical MonoBehavior. How about these methods? We will mainly work in two ways WaveManagerEditor.cs. OnEnable:This method runs every time a WaveManager object is loaded into the Unity editor. It is used to search the properties of the wavemanager by the WaveManager editor. OnInspectorGUI(): This method is responsible for drawing the inspector interface to the object. Default. OnInspectorGUI() pulls out the object's default checker interface, so you can get a flimsy folding of the default sit-down by leaving it in your code. After all, we remove it because we don't want a fold, but now it's better to keep the base. Inspector GUI () there while working on the order type list. This makes it easy to compare and reference both the foldlist and the reorder list. To start initializing a class property, getting a wave array from WaveManager.cs. To do this, OnEnable() must be added to the following line: WaveManagerEditor.cs private void OnEnable() { ///<wave> Importthe array into a serialized property form in The Fauma.// Note the attributes of the parent editor class: Serialized Object.Find (Wave); // Reorderable List Settings = Setting a new order list (continuous object, wave, true); ReorderableList has now been reset, but it is not enough to run and do it. This is done by defining the methods of the callback delegate provided by ReorderableList. These delegates allow you to customize specific sections of the reorder list. Here are some useful things: Callback Delegate DescriptionDraw ElementCallback determines how each element in the ElementCallback list is drawn. >/serializedObject>/wave>/wave&t;/wave&.../wave&t;When an element is added, it is called when the element is removed. There are more callback delegates available, which can only be found when searching through the official source code. In this article, you will draw the body of the header and the list, respectively, using only drawElementCallback and drawHeaderCallback. WaveManagerEditor.cs personal void OnEnable () {// We are creating a list of sequences from WaveAttributes. Delegates who can draw elements from the list.drawHeaderCallback = DrawHeader; Set displayHeader to 'false' in the resortable list constructor to skip this line. If you look at the delegates defined in the source code, you see the same parameter definition. To draw a header, use EditorGUI.LabelField() to draw text to create a header in the reorder list. WaveManagerEditor.cs void DrawHeader (rect rect) { string name = wave; EditorGUI.LabelField (rectification, name); This will give us a header of the list in order: how our order list looks like it is now. For each element in the list, you must draw the individual properties of each element (the elements defined Wave.cs the previous) for each element in the reordering list. In the code below, EditorGUI.PropertyField () draws an input field, while EditorGUI.LabelFields () draws text that describes the input field. WaveManagerEditor.cs Void DrawListItems (Rect Rect, int index, boolIsActive, boolisActive) { Serialproperty element = list.serializedProperty.GetArrayElementIndex (index); // Elements in the list // Create property fields and label fields for each property. FindPropertyRelative (mob), GUIContent.none); Level attributes // Level (width 100, height of one line) EditorGUI.LabelField (new Rect.x, rect.y, 100, EditorGUIUtility.singleLineHeight), level; The property field for the level. Because int does not require too much space, the width is set to 20, which is the height of a single line. EditorGUI.PropertyField (new Rect.x, rect.y, 20, Element. FindPropertyRelative (level), GUIContent.none); Label field for 'quantity' properties //quantities (width 100, height of one line) EditorGUI.LabelField (new Rect.x, rect.y, 100, EditorGUIUtility.singleLineHeight), quantity); Property field for quantity (width 20, one line height) EditorGUI.PropertyField (new Rect.x, rect.y, 20, EditorGUIUtility.singleLineHeight) element. FindPropertyRelative (quantity), GUIContent.none); } Continue with the article after the ad: If you check the manager at this point in the list rendering, you will not see any order list. This is because you need to instruct OnInspector GUI () to draw itself in the reorderlist. WaveManagerEditor.cs //This is the ability to create a base for the custom editor to create a void invalid OnInspectorGUI () {base to work publicly with the onInspectorGUI(); Serialized Object.Update(); Update the representation of the array properties in the inspector list. Here are two problems: the old array folding still exists. We just need to get rid of the base. To solve this problem, the onInspectorGUI () fields for our elements are overlapping with each other, because we haven't set where their location should be. So make the next edit to the method below: WaveManagerEditor.cs//This is a function that allows the custom editor to work publicly. [OnInspectorGUI (); Serialized Object.Update(); Update the representation of array properties in the inspector list. WaveManagerEditor.cs Void DrawListItems (Rect Rect, int index, boolisActive, boolisActive) { serialproperty element = list.serializedProperty.GetArrayElementAtIndex (index); ///The elements in the list //List create property fields and label fields for each property. FindPropertyRelative (mob), GUIContent.none); 'Level' property // label field for level (width 100, height of one line) EditorGUI.LabelField (new Rect.x + 120, rect.y, 100, EditorGUIUtility.singleLineHeight), level); The property field for the level. Because int does not require too much space, the width is set to 20, which is the height of a single line. EditorGUI.PropertyField (new Rect.x + 160, rect.y, 20, EditorGUIUtility.singleLineHeight), elements. FindPropertyRelative (level), GUIContent.none); 'Quantity' property // Quantity (width 100, height of one line) EditorGUI.LabelField (new Rect.x + 200, rect.y, 100, EditorGUIUtility.singleLineHeight), quantity); Property field for the quantity (width 20, height of one line) EditorGUI.PropertyField (new Rect.x + 250, rect.y, 20, EditorGUIUtility.singleLineHeight) element. FindPropertyRelative (quantity), GUIContent.none); } When all this is done, we will end up with the following: our final product. Here's a complete script for WaveManagerEditor.cs: WaveManagerEditor.cs using UnityEngine; Use unity editors; Use the Unity Editor inside; WaveManager [WaveEditor)] Public Class WaveManager Editor: Editor {//Array properties will edit serialized property waves. //Rearrange list We work with a reorder list list; Personal Void OnEnable () {/// We are creating a list of order from Wave Properties. Invalid DrawListItems (Rect rect, int index, boolIsActive, boolisActive) { serialist element = list.serializedProperty.GetArrayElementAtIndex (index); //List elements // Property fields and label fields for each property Do not create a label field because the enumeration is self-exclaimed. FindPropertyRelative (mob), GUIContent.none); 'Level' property // label field for level (width 100, height of one line) EditorGUI.LabelField (new Rect.x + 120, rect.y, 100, EditorGUIUtility.singleLineHeight), level); The property field for the level. Because int does not require too much space, the width is set to 20, which is the height of a single line. EditorGUI.PropertyField (new Rect.x + 160, rect.y, 20, EditorGUIUtility.singleLineHeight), elements. FindPropertyRelative (level), GUIContent.none); 'Quantity' property // label field for quantity (width 100, height of one line) EditorGUI.LabelField (new Rect.x + 200, rect.y, 100, EditorGUIUtility.singleLineHeight), quantity); Property field for the quantity (width 20, height of one line) EditorGUI.PropertyField (new Rect.x + 250, rect.y, 20, EditorGUIUtility.singleLineHeight) element. FindPropertyRelative (quantity), GUIContent.none); Void DrawHeader (rect rect) { string name = wave; EditorGUI.LabelField (rectification, name); } //The ability to reveal the custom editor action. Invalid OnInspectorGUI () {Serialized Object.Update (); List. Dolayout List (); Serialized Object.AppliedModification Properties (); } Articles after advertising Continue: Ads: