**UGUI无限滚动列表**

绑定在Scroll View上面

using UnityEngine; using System.Collections; using UnityEngine.UI; using UnityEngine.EventSystems; using System; public class InfiniteScroll : MonoBehaviour, IDropHandler { public enum Direction { Top, Bottom }; public event Action<int, GameObject> FillItem = delegate {}; public event Action<Direction> PullLoad = delegate {}; [Header("Item settings")] public GameObject prefab; public int height = 110; [Header("Padding")] public int top = 10; public int bottom = 10; public int spacing = 2; [Header("Labels")] public string topPullLabel = "上拉刷新"; public string topReleaseLabel = "释放加载"; public string bottomPullLabel = "上拉刷新"; public string bottomReleaseLabel = "释放加载"; [Header("Directions")] public bool isPullTop = true; public bool isPullBottom = true; [Header("Pull coefficient")] [Range (0.01f, 0.1f)] public float pullValue = 0.05f; [HideInInspector] public Text topLabel; [HideInInspector] public Text bottomLabel; private ScrollRect \_scroll; private RectTransform \_content; private RectTransform[] \_rects; private GameObject[] \_views; private bool \_isCanLoadUp; private bool \_isCanLoadDown; private int \_previousPosition; private int \_count; void Awake () { \_scroll = GetComponent <ScrollRect> (); \_scroll.onValueChanged.AddListener(OnScrollChange); \_content = \_scroll.viewport.transform.GetChild(0).GetComponent <RectTransform> (); CreateViews (); CreateLabels (); } void Update () { if (\_count == 0) return; float \_topPosition = \_content.anchoredPosition.y - spacing; if (\_topPosition <= 0f && \_rects[0].anchoredPosition.y < -top-10f) { InitData (\_count); return; } if (\_topPosition < 0f) return; int position = Mathf.FloorToInt (\_topPosition / (height + spacing)); if (\_previousPosition == position) return; if (position > \_previousPosition) { if (position - \_previousPosition > 1) position = \_previousPosition + 1; int newPosition = position % \_views.Length; newPosition--; if (newPosition < 0) newPosition = \_views.Length - 1; int index = position + \_views.Length - 1; if (index < \_count) { Vector2 pos = \_rects[newPosition].anchoredPosition; pos.y = -(top + index \* spacing + index \* height); \_rects[newPosition].anchoredPosition = pos; FillItem (index, \_views[newPosition]); } } else { if (\_previousPosition - position > 1) position = \_previousPosition - 1; int newIndex = position % \_views.Length; Vector2 pos = \_rects[newIndex].anchoredPosition; pos.y = -(top + position \* spacing + position \* height); \_rects[newIndex].anchoredPosition = pos; FillItem (position, \_views[newIndex]); } \_previousPosition = position; } void OnScrollChange (Vector2 vector) { float coef = \_count / \_views.Length; float y = 0f; \_isCanLoadUp = false; \_isCanLoadDown = false; if (vector.y > 1f) y = (vector.y - 1f) \* coef; else if (vector.y < 0f) y = vector.y \* coef; if (y > pullValue && isPullTop) { topLabel.gameObject.SetActive (true); topLabel.text = topPullLabel; if (y > pullValue\*2) { topLabel.text = topReleaseLabel; \_isCanLoadUp = true; } } else topLabel.gameObject.SetActive (false); if (y < -pullValue && isPullBottom) { bottomLabel.gameObject.SetActive (true); bottomLabel.text = bottomPullLabel; if (y < -pullValue\*2) { bottomLabel.text = bottomReleaseLabel; \_isCanLoadDown = true; } } else bottomLabel.gameObject.SetActive (false); } public void OnDrop (PointerEventData eventData) { if (\_isCanLoadUp) PullLoad (Direction.Top); else if (\_isCanLoadDown) PullLoad (Direction.Bottom); \_isCanLoadUp = false; \_isCanLoadDown = false; } public void InitData (int count) { \_previousPosition = 0; \_count = count; float h = height \* count \* 1f + top + bottom + (count == 0 ? 0 : ((count - 1) \* spacing)); \_content.sizeDelta = new Vector2 (\_content.sizeDelta.x, h); Vector2 pos = \_content.anchoredPosition; pos.y = 0f; \_content.anchoredPosition = pos; int y = top; bool showed = false; for (int i = 0; i < \_views.Length; i++) { showed = i < count; \_views [i].[/i]gameObject.SetActive (showed); pos = \_rects.anchoredPosition; pos.y = -y; pos.x = 0f; \_rects.anchoredPosition = pos; y += spacing + height; if (i + 1 > \_count) continue; FillItem (i, \_views); } } public void ApplyDataTo (int count, int newCount, Direction direction) { \_count = count; float newHeight = height \* count \* 1f + top + bottom + (count == 0 ? 0 : ((count - 1) \* spacing)); \_content.sizeDelta = new Vector2 (\_content.sizeDelta.x, newHeight); Vector2 pos = \_content.anchoredPosition; if (direction == Direction.Top) { pos.y = (height + spacing) \* newCount; \_previousPosition = newCount; } else pos.y = newHeight - (height \* spacing) \* newCount - (float)Screen.currentResolution.height; \_content.anchoredPosition = pos; float \_topPosition = \_content.anchoredPosition.y - spacing; int index = Mathf.FloorToInt (\_topPosition / (height + spacing)); int all = top + index \* spacing + index \* height; for (int i = 0; i < \_views.Length; i++) { int newIndex = index % \_views.Length; FillItem (index, \_views [newIndex]); pos = \_rects [newIndex].anchoredPosition; pos.y = -all; \_rects [newIndex].anchoredPosition = pos; all += spacing + height; index++; if (index == \_count) break; } } void CreateViews () { GameObject clone; RectTransform rect; int fillCount = Mathf.RoundToInt((float)Screen.currentResolution.height / height) + 2; \_views = new GameObject[fillCount]; for (int i = 0; i < fillCount; i++) { clone = (GameObject)Instantiate (prefab, Vector3.zero, Quaternion.identity); clone.transform.SetParent (\_content); clone.transform.localScale = Vector3.one; clone.transform.localPosition = Vector3.zero; rect = clone.GetComponent<RectTransform> (); rect.pivot = new Vector2(0.5f, 1f); rect.anchorMin = new Vector2(0f, 1f); rect.anchorMax = new Vector2(1f, 1f); rect.offsetMax = new Vector2(0f, 0f); rect.offsetMin = new Vector2(0f, -height); \_views = clone; } \_rects = new RectTransform[\_views.Length]; for (int i = 0; i < \_views.Length; i++) \_rects = \_views.gameObject.GetComponent <RectTransform> (); } void CreateLabels () { GameObject topText = new GameObject ("TopLabel"); topText.transform.SetParent (\_scroll.viewport.transform); topLabel = topText.AddComponent<Text> (); topLabel.font = Resources.GetBuiltinResource<Font> ("Arial.ttf"); topLabel.fontSize = 24; topLabel.transform.localScale = Vector3.one; topLabel.alignment = TextAnchor.MiddleCenter; topLabel.text = topPullLabel; RectTransform rect = topLabel.GetComponent<RectTransform> (); rect.pivot = new Vector2(0.5f, 1f); rect.anchorMin = new Vector2(0f, 1f); rect.anchorMax = new Vector2(1f, 1f); rect.offsetMax = new Vector2(0f, 0f); rect.offsetMin = new Vector2(0f, -55f); rect.anchoredPosition3D = Vector3.zero; topText.SetActive (false); GameObject bottomText = new GameObject ("BottomLabel"); bottomText.transform.SetParent (\_scroll.viewport.transform); bottomLabel = bottomText.AddComponent<Text> (); bottomLabel.font = Resources.GetBuiltinResource<Font> ("Arial.ttf"); bottomLabel.fontSize = 24; bottomLabel.transform.localScale = Vector3.one; bottomLabel.alignment = TextAnchor.MiddleCenter; bottomLabel.text = bottomPullLabel; bottomLabel.transform.position = Vector3.zero; rect = bottomLabel.GetComponent<RectTransform> (); rect.pivot = new Vector2(0.5f, 0f); rect.anchorMin = new Vector2(0f, 0f); rect.anchorMax = new Vector2(1f, 0f); rect.offsetMax = new Vector2(0f, 55f); rect.offsetMin = new Vector2(0f, 0f); rect.anchoredPosition3D = Vector3.zero; bottomText.SetActive (false); } } [AppleScript] 纯文本查看 复制代码 using UnityEngine; using System.Collections; using UnityEngine.UI; public class Controller : MonoBehaviour { public InfiniteScroll scroll; private int count = 100; void Start () { scroll.FillItem += (int index, GameObject item) => { //这里我们可以填写和修改item prefab //改变文字，图像等 //通过索引，我们可以从JSON数组获取数据，例如 item.transform.GetChild (0).GetComponent<Text> ().text = "item #" + index; }; scroll.PullLoad += (InfiniteScroll.Direction obj) => { //这里我们监听拖拽刷新事件并处理它 //它可以将数据从服务器加载到JSON对象并附加到列表 //做到这一点，调用ApplyDataTo函数，其中arg1 =通用项追加后计数，arg2 = count追加，arg3 =追加方向（顶部或底部） count += 20; scroll.ApplyDataTo (count, 20, obj); }; //函数初始化无限滚动 scroll.InitData (count); } }