UIWidget继承于UIRect，实际上会重写OnInit、OnStart、OnUpdate与OnAnchor（前提是四个角任意一个角设置了锚点）函数

重要属性：

[System.NonSerialized] public UIPanel panel;

[System.NonSerialized] public UIGeometry geometry = new UIGeometry();

[System.NonSerialized] public UIDrawCall drawCall;

**OnInit函数：**

protected override void OnInit ()

{

base.OnInit();

RemoveFromPanel();

mMoved = true;

Update(); // 这里调用的实际上是UIRect的Update函数

}

**OnStart函数：**

protected override void OnStart ()

{

#if UNITY\_EDITOR

if (GetComponent<UIPanel>() != null) // UIWidget

{

Debug.LogError("Widgets and panels should not be on the same object! Widget must be a child of the panel.", this);

}

else if (!Application.isPlaying && GetComponents<UIWidget>().Length > 1)

{

Debug.LogError("You should not place more than one widget on the same object. Weird stuff will happen!", this);

}

#endif

**CreatePanel();**

}

**OnUpdate函数：**

protected override void OnUpdate ()

{

if (panel == null) CreatePanel();

#if UNITY\_EDITOR

else if (!mPlayMode) ParentHasChanged();

#endif

}

**CreatePanel函数：**

public UIPanel CreatePanel ()

{

if (mStarted && panel == null && enabled && NGUITools.GetActive(gameObject))

{

panel = UIPanel.Find(cachedTransform, true, cachedGameObject.layer);

if (panel != null)

{

mParentFound = false;

panel.AddWidget(this);

CheckLayer();

Invalidate(true);

}

}

return panel;

}

**更新锚点：**

protected override void OnAnchor ()

{

float lt, bt, rt, tt;

Transform trans = cachedTransform;

Transform parent = trans.parent;

Vector3 pos = trans.localPosition;

Vector2 pvt = pivotOffset;

// Attempt to fast-path if all anchors match

if (leftAnchor.target == bottomAnchor.target &&

leftAnchor.target == rightAnchor.target &&

leftAnchor.target == topAnchor.target) **// 四个锚点是同一个对象**

{

Vector3[] sides = leftAnchor.GetSides(parent); **// 获取锚点基于父节点的相对位置**

if (sides != null) **// 锚点有UIRect组件或者有Camera组件**

{

lt = NGUIMath.Lerp(sides[0].x, sides[2].x, leftAnchor.relative) + leftAnchor.absolute;

rt = NGUIMath.Lerp(sides[0].x, sides[2].x, rightAnchor.relative) + rightAnchor.absolute;

bt = NGUIMath.Lerp(sides[3].y, sides[1].y, bottomAnchor.relative) + bottomAnchor.absolute;

tt = NGUIMath.Lerp(sides[3].y, sides[1].y, topAnchor.relative) + topAnchor.absolute;

mIsInFront = true;

}

Else **// 锚点没有UIRect组件且没有Camera组件，则只取Transform组件**

{

// Anchored to a single transform

Vector3 lp = GetLocalPos(leftAnchor, parent); **// 计算锚点基于父节点的相对坐标**

lt = lp.x + leftAnchor.absolute;

bt = lp.y + bottomAnchor.absolute;

rt = lp.x + rightAnchor.absolute;

tt = lp.y + topAnchor.absolute;

mIsInFront = (!hideIfOffScreen || lp.z >= 0f);

}

}

else

{

mIsInFront = true;

// Left anchor point

if (leftAnchor.target) **// 计算左边位置**

{

Vector3[] sides = leftAnchor.GetSides(parent);

if (sides != null)

{

lt = NGUIMath.Lerp(sides[0].x, sides[2].x, leftAnchor.relative) + leftAnchor.absolute;

}

else

{

lt = GetLocalPos(leftAnchor, parent).x + leftAnchor.absolute;

}

}

else lt = pos.x - pvt.x \* mWidth;

// Right anchor point

if (rightAnchor.target) **// 计算右边位置**

{

Vector3[] sides = rightAnchor.GetSides(parent);

if (sides != null)

{

rt = NGUIMath.Lerp(sides[0].x, sides[2].x, rightAnchor.relative) + rightAnchor.absolute;

}

else

{

rt = GetLocalPos(rightAnchor, parent).x + rightAnchor.absolute;

}

}

else rt = pos.x - pvt.x \* mWidth + mWidth;

// Bottom anchor point

if (bottomAnchor.target) **// 计算下边位置**

{

Vector3[] sides = bottomAnchor.GetSides(parent);

if (sides != null)

{

bt = NGUIMath.Lerp(sides[3].y, sides[1].y, bottomAnchor.relative) + bottomAnchor.absolute;

}

else

{

bt = GetLocalPos(bottomAnchor, parent).y + bottomAnchor.absolute;

}

}

else bt = pos.y - pvt.y \* mHeight;

// Top anchor point

if (topAnchor.target) **// 计算上边位置**

{

Vector3[] sides = topAnchor.GetSides(parent);

if (sides != null)

{

tt = NGUIMath.Lerp(sides[3].y, sides[1].y, topAnchor.relative) + topAnchor.absolute;

}

else

{

tt = GetLocalPos(topAnchor, parent).y + topAnchor.absolute;

}

}

else tt = pos.y - pvt.y \* mHeight + mHeight;

}

// Calculate the new position, width and height

Vector3 newPos = new Vector3(Mathf.Lerp(lt, rt, pvt.x), Mathf.Lerp(bt, tt, pvt.y), pos.z); **// 计算新位置，同时考虑到了锚点位置**

newPos.x = Mathf.Round(newPos.x);

newPos.y = Mathf.Round(newPos.y);

int w = Mathf.FloorToInt(rt - lt + 0.5f);

int h = Mathf.FloorToInt(tt - bt + 0.5f);

// Maintain the aspect ratio if requested and possible

if (keepAspectRatio != AspectRatioSource.Free && aspectRatio != 0f)

{

if (keepAspectRatio == AspectRatioSource.BasedOnHeight)

{

w = Mathf.RoundToInt(h \* aspectRatio);

}

else h = Mathf.RoundToInt(w / aspectRatio);

}

// Don't let the width and height get too small

if (w < minWidth) w = minWidth;

if (h < minHeight) h = minHeight;

// Update the position if it has changed

if (Vector3.SqrMagnitude(pos - newPos) > 0.001f)

{

cachedTransform.localPosition = newPos; **// 更新位置**

if (mIsInFront) mChanged = true;

}

// Update the width and height if it has changed

if (mWidth != w || mHeight != h)

{

mWidth = w; **// 更新宽度与高度**

mHeight = h;

if (mIsInFront) mChanged = true;

if (autoResizeBoxCollider) ResizeCollider();

}

}