Native Cursor

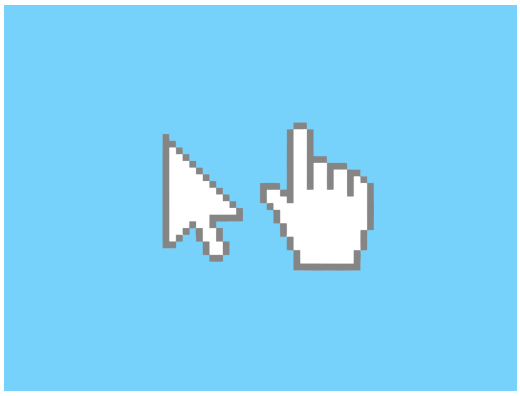
None

None

None

Table of contents

1. Getting Started



1.1 Installation

Download from the Asset Store.

1.2 What is Native Cursors?

Native Cursors is cross-platform package that allows you to change the cursor to any of the available cursors on the OS. This is useful for games that want to use the OS's cursor instead of a custom one.

Currently, it supports **WebGL**, **Windows**, **MacOS** and **Linux**.

1.3 How does it work?

It relies on P/Invoke to call the native APIs of each platform.

2. NativeCursor API

This is the native cursor API. It allows you to change the cursor to any of the available cursors. We recommend you use the `CursorStack` API instead for most use cases, it's more flexible, easier and serves as a wrapper for this raw API.

```
namespace Riten.Native.Cursors;

public static class NativeCursor
{
    /// <summary>
    /// Changes the OS's cursor to the specified cursor.
    /// Implementation varies based on environment.
    /// </summary>
    public static bool SetCursor(NTCursors cursor);

    /// <summary>
    /// Resets cursor to default state.
    /// Certain platforms may include extra cleanup.
    /// Prefer this over SetCursor(NTCursors.Arrow);
    /// </summary>
    public static void ResetCursor();
}
```

2.1 Cursor Types

⚠ Editor doesn't reflect the actual cursor, it should be considered as a placeholder for testing purposes. It's best to test in a build.

The `Busy` and `Invalid` cursors are known to be unreliable on MacOS.

```
namespace Riten.Native.Cursors;

public enum NTCursors
{
    Default,
    Arrow,
    IBeam,
    Crosshair,
    Link,
    Busy,
    Invalid,
    ResizeVertical,
    ResizeHorizontal,
    ResizeDiagonalLeft,
    ResizeDiagonalRight,
    ResizeAll,
    OpenHand,
    ClosedHand
}
```

3. Cursor Stack API

This is the cursor stack API. It allows you to change the cursor to any of the available cursors. The difference between this and the native cursor API is that this API allows you to push and pop cursors to a stack. This is useful for when you want to change the cursor to a specific cursor for a specific part of your code, and then revert it back to the previous cursor.

This is used internally for the preset components, but you can and should use it for your own components as well.

```
namespace Riten.Native.Cursors;

public static class CursorStack
{
    /// <summary>
    /// Pushes a cursor to the stack.
    /// Higher priority means this cursor will override other cursors with lower priority.
    /// </summary>
    /// <returns>The id of the pushed cursor. Use this to remove the cursor later.</returns>
    public static int Push(NTCursors cursor, int priority = 0, int secondaryPriority = 0);

    /// <summary>
    /// Replace the cursor with the given id with the given cursor.
    /// This will not change the priority of the cursor.
    /// Use this if you want to change a cursor that is already in the stack.
    /// For example, if you want to change the cursor during a drag operation to indicate that the drag is invalid.
    /// </summary>
    public static bool Replace(int id, NTCursors cursor)

    /// <summary>
    /// Pops the cursor with the given id.
    /// You can get the id from the return of Push().
    /// </summary>
    /// <returns></returns>
    public static bool Pop(int id);

    /// <summary>
    /// Pops the cursor that is being rendered.
    /// Always prefer using Pop(int id) if you have the id.
    /// </summary>
    /// <returns>True if a cursor was removed, false otherwise.</returns>
    public static bool Pop();

    /// <summary>
    /// Removes all cursors from the stack.
    /// Resets the cursor to default.
    /// </summary>
    public static void Clear();

    /// <summary>
    /// Returns true if the stack is empty.
    /// </summary>
    public static bool IsEmpty { get; }

    /// <summary>
    /// Returns the cursor at the top of the stack.
    /// It will be the cursor with the highest priority.
    /// </summary>
    /// <returns>Default if the stack is empty. Otherwise, the cursor with highest priority.</returns>
    public static CursorStackItem Peek();

    /// <summary>
    /// A debug GUI to see the current stack.
    /// Uses GUILayout, so it's not very customizable.
    /// Just dumps the stack from top to bottom.
    /// </summary>
    public static void OnDebugGUI();

    /// <summary>
    /// If true, the cursor will not change until this is set to false.
    /// The stack will still be updated, but won't affect the cursor.
    /// Once unpaused, the cursor will be updated automatically.
    /// </summary>
    public static void PauseRendering(bool isPaused);

    /// <summary>
    /// Call this if you used NativeCursor.SetCursor() directly and want to reapply the stack's cursor.
    /// </summary>
    public static void ReApply()
    {
        OnStackChanged(true);
    }
}
```

I recommend you to look at the source code for the components implementations to see how they use this API if it's not clear enough.

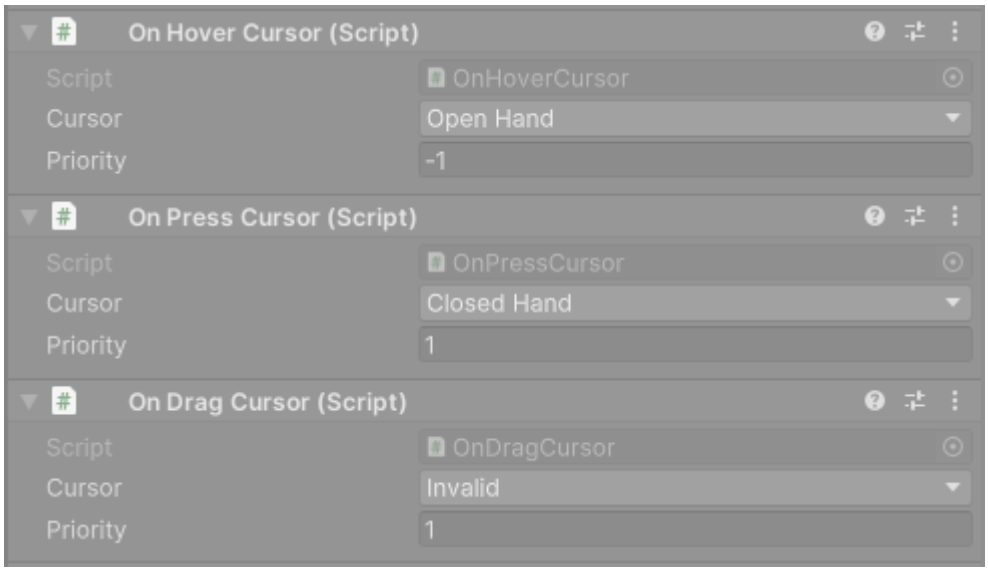
3.1 Example

```
// Push a cursor to the stack.
var cursorId = CursorStack.Push(NTCursors.Link);

// ... do stuff / wait for an event / whatever ...

// Pop the cursor from the stack.
CursorStack.Pop(cursorId);
```

4. UI Components



4.1 OnHoverCursor

Once the cursor is over the component, the cursor will change to the given cursor. If multiple components are hovered, the cursor will be the one with the highest priority.

4.2 OnPressCursor

Once the component is pressed, the cursor will change to the given cursor. If multiple components are pressed, the cursor will be the one with the highest priority.

4.3 OnDragCursor

Once the component is dragged, the cursor will change to the given cursor. If multiple components are dragged, the cursor will be the one with the highest priority.