"Splash Image"

Touchless.Design SDK

[Website] [Leap Motion SDK]

In light of the coronavirus crisis, touchless interaction has become the focus of many designers and developers who create kiosks and screen-based exhibits for public spaces. Using the Leap Motion Controller and V4 SDK, our Integrated Touchless System allows users to interact with our tables and exhibits with no physical touch. This system tracks the visitor's hand and watches for specific gestures, such as opening and closing, that give complete control over the mouse. The Integrated Touchless System is a mouse emulation system with multiple methods for onboarding and real-time feedback.

Alongside this system are three additional applications: a dynamic cursor overlay, a secondary instructional screen, and an LED system add-on. Download the latest release here. These applications are meant to provide both feedback and onboarding information to the visitor through color changes, gesture icons, and interaction information when hovering over buttons or drag areas. This repository also includes a Unity asset package that allows users to integrate support for the system into their own applications and a demo application for reference. The core system is built in .Net Framework 4.6.1 and the peripheral applications are built in the Unity Game Engine.

Important Notes:

- The Leap Motion V4 Orion SDK must be installed with a leap motion device connected for this application to function.
- This repository is meant to be built and run using the Touchless Add-on with a Leap Motion device, 3.5" display, and LED lights mounted on an Ideum Drafting Touch Table or an Ideum Touchless Pedestal.
- The keyboard shortcut **Ctrl+Alt+I** will toggle mouse emulation on/off when the service is running.

Preview Video:

Repository Layout

All of the source code for the Integrated Touchless System is in the src directory, as well as the peripheral applications and dependencies with the exception of the Leap Motion V4 Orion SDK. The Leap Motion V4 Orion SDK should be downloaded and installed before running the Touchless Design applications.

Service

This is the core of the Integrated Touchless System. It operates as a Windows service and is responsible for integrating the Leap Motion SDK, controlling the mouse, managing the Overlay and Add-on applications, setting the color of the attached LEDs, and communicating with any client applications. Certain settings such as gesture toggles can be configured via a system tray icon as shown below. Other options can be configured by editing the configuration JSON files in the Integrated Touchless System's root directory. This is explained in greater depth here.

"System Tray Icon"

Overlay

The Overlay application is a replacement for the Windows cursor and provides feedback to the user as they interact with the Integrated Touchless System. When enabled, the System will automatically manage and communicate with the Overlay application.

"Overlay Cursor"

When interacting with a client application that is connected to the Integrated Touchless System, the Overlay will have the following behaviors:

- When hovering over a button, it will show an animated select gesture.
- When hovering over a drag area, it will show a closed fist with arrows.
- When a selection is made, it will turn green and change size.
- If the screen is touched, it will turn red.

AddOn

The Add-on application is designed to display information on the secondary monitor and control the LED lights available on the Touchless Add-on. This application provides further feedback and onboarding information to supplement the Overlay cursor as users interact with the Integrated Touchless System. When enabled, the System will automatically manage and communicate with the Add-on application. In addition to the animated hand icon, the Add-on application also shows text feedback.

"AddOn"

Asset Package

The Unity asset package, "TouchlessDesign.unitypackage", can be imported into a Unity Engine project and used to integrate that project application with the Integrated Touchless System. More detailed instructions for this can be found here.

Example

The Example application is a demo client application that uses the Unity asset package to integrate with the Integrated Touchless System.

Ideum.Logging

This is a logging abstraction we use in Unity and is included in all of the Unity projects, but we've included the source code here for posterity.

Deployment

Service

Building:

In order to build the core of the Integrated Touchless System, open the TouchlessDesign.sln file in Visual Studio, right-click the Solution and select "Build Solution." We have included instructions that will build the solution at the following path on your system:

```
%appdata%/Ideum
```

Running:

Navigate to the build directory (see above), open `TouchlessDesign/bin/Service/`, and run the `TouchlessDesign.exe`. If the AddOn and/or Overlay applications have been built and configured, they will also be launched and the mouse should immediately begin reacting to the Leap Motion controller. An icon will appear in the system tray and by left clicking on the icon, the service UI will be displayed.

Configuration:

Configuration is primarily handled through the UI, but all configurable values are stored in JSON files that can be found in the touchless design directory.

- display.json: Allows you to adjust functionality related to the overall user experience such as toggling the onboarding on and off or adjusting values relating to the FadeCandy LED control.
- input.json: Allows you to adjust options related to mouse emulation.
- network.json: Configures network settings for the Service.

AddOn and Overlay

1. Download Unity version 2019.3.2f1
2. Open the application in Unity, and go to File -> Build Settings.
3. Make sure the _App/Scenes/App Scene is checked and press Build.
4. In order for the Integrated Touchless System to find and run either of the peripheral applications, they should be built at the following respective paths:

- `%appdata%/Ideum/TouchlessDesign/bin/AddOn/`
- `%appdata%/Ideum/TouchlessDesign/bin/Overlay/`

5. In the TouchlessDesign directory, open the ui.json file and make sure the paths to the two applications are included in the ApplicationPaths field.

Now, when you run the Integrated Touchless System, it will automatically start up the applications specified in the ui.json file. Likewise, when the System is exited, it will terminate those same applications. Note: These applications will only provide gesture feedback when running alongside a client application that uses the Integrated Touchless System bindings, such as the Example application.

Touchless Multi-Device Support

This version of the Touchless SDK includes a Unity package that will run the core of the service inside of Unity, rather than through the normal touchless service application. This allows for easily integrating multiple realsense devices into Unity UI Event System, treating each touchless cursor similarly to a mouse cursor. A .unitypackage file will be provided containing all of the necessary code.

Setting up a Project

To set up the unity-based version of the Touchless service, download and import the provided package into a new Unity project. From there, you'll need to add the TouchlessDesign prefab into your scene and replace Unity's StandardInputModule with the TouchlessInputModule.

1. Open the Unity project.
2. Open Assets-> Import Package -> Custom Package...
3. This will open a dialog. Select the TouchlessDesign.unitypackage file in the src directory of this repository. When it prompts you to select which parts of the package to import, select Import All.
4. Add the TouchlessDesign prefab to the root of your scene, and assign the appropriate references.
5. With a Canvas and EventSystem in the scene, remove or disable Unity's StandaloneInputModule and add the TouchlessInputModule.
6. In a Monobehavior script, subscribe to the OnStarted and OnStopped events for Touchless Design. The `Examples` folder contains a script that does this, and also creates custom cursors on startup.

```
if (TouchlessDesign.Instance.isStarted) {
    HandleTouchlessDesignStarted();
}
else {
  TouchlessDesign.OnStarted += HandleTouchlessDesignStarted;
}
TouchlessDesign.OnStopped += HandleTouchlessDesignStop;
```

Once the Touchless service is started and it has loaded all users, you can start subscribing to user events as they come in.

```
  private void HandleTouchlessDesignStarted()
  {
    Debug.Log("Touchless Design Started.");
    foreach (TouchlessUser user in TouchlessDesign.Instance.Users.Values)
    {
      user.HoverStateChanged += HandleHoverStateChanged;
```

```
        user.CursorActivated += HandleCursorActivated;
        user.CursorDeactivated += HandleCursorDeactivated;
        user.PointerPressed += HandlePointerPressed;
        user.PointerReleased += HandlePointerReleased;
      }
    }
```

Examples

The `Examples` folder of the Touchless package contains some basic scripts and a scene that uses them. The Sample Scene can be used as a reference regarding scene and script setup in case there's any confusion.

Configuration:

The Unity version of Touchless is configured through directly editing json files in the `StreamingAssets>Touchless` directory. Each of the configuration files are the same as listed above, with the except of input.json, which will now allow you to add IP addresses and IDs for expected users.

```
"Players": [
  {
    "Id": 0,
    "IpAddress": "10.0.0.155",
    "BoundsWidth": 1920,
    "BoundsHeight": 1080
  },
  {
    "Id": 1,
    "IpAddress": "10.0.0.87",
    "BoundsWidth": 1920,
    "BoundsHeight": 1080
  }
]
```

Here we can set the ID, IP Address, and bounds settings for each user.

Note: The `Id` of a user should be no less than zero, and each user should have a unique ID assigned to them. This helps with making sure that it won't matter the order in which touchless pedestals or remote providers connect. These ID's are also used by Unity's event system to assign pointer ID's for GUI events.

Pedestal Setup Recommendations

It is recommended that each pedestal is assigned a static IP address so that they will not change in between daily reboots and become incompatible with the host `Players` configuration mentioned above.

It should also be ensured that each pedestal's input mode is set to `local` instead of `remote`. They should still have `Remote Provider Mode` enabled, however.