Quick Start Guide

1. Import ICAROS SDK Desktop.unitypackage in Unity.
2. Open Scenes/Examples/Import.scene
3. It is recommended to save your scene under a different name now. (File -> Save scene as...)
4. Import your VR Plugin of choice. Refer to the respective Documentation for help.
5. Link the included CameraRig in the hierarchy at UISystem -> VRCamera and attach it to the LocalPlayer example.
6. Link another Camera for Monitor only UI at UISystem -> MenuCamera and attach it to the LocalPlayer example.
7. Select ICAROS -> Input from the Unity menu bar and choose one of the predefined devices. You may also change their settings or define your own in the Input-settings of Unity. This does not change anything related to the ICAROS Controller.
8. [Optional] Put some geometry in your scene as orientation points.
9. Change your Architecture to x86_64 in the Unity Build Settings.
10. Change your API Compatibility Level to .Net 4.X in the Unity Player Settings
11. Congratulations! You should be able to do your first build!

Next Steps

1. Change the placeholder icon at UISystem -> CompanyLogoTemplate to your own.
2. Take a look at LocalPlayer. For Example increase the acceleration of the player. It is not recommended to carelessly change the rotation factors.

Modules and their functions

1. Input
   This module is used to get easily readable input from all input devices supported by the Icaros. The needed setup for use can be seen and copied from the 'Import' scene.

1.1. DeviceManager
   Singleton -> make sure to call functions through DeviceManager.Instance.

   Events:

   - NewDeviceRegistered
     Called for each new device detected. Parameter: the device as IInputDevice.
   - DeviceLost
     Called when the connection to a device is permanently lost. Parameter: the device as IInputDevice.
   - DeviceUsed
     Called when a device gets marked as used. Parameter: the device as IInputDevice.
   - DeviceReconnecting
     Called when a currently used device changes its reconnection state. Parameter: The current State as DeviceManager.reconnectingDeviceState.
   - OnDebugMessage
     Called when methods outside the unity framework provide debug information.

Methods:

- UseDevice
  Marks the device as used. Unused devices are inactive in delivering key input. Parameter: the device as IInputDevice.
- GetRegisteredDevices
  Returns a list of all registered IInputDevices.
- GetUsedDevices
  Returns a list of all used IInputDevices.

1.2. IInputDevice

Interface used to get Input without having to know what device is used in reality.
The following Events and Methods are supported:

```
event System.ActionFirstButtonPressed
event System.ActionSecondButtonPressed
event System.ActionThirdButtonPressed
event System.ActionFourthButtonPressed

bool FirstButtonDown()
bool SecondButtonDown()
bool ThirdButtonDown()
bool FourthButtonDown()

event System.ActionFirstButtonReleased
event System.ActionSecondButtonReleased
event System.ActionThirdButtonReleased
event System.ActionFourthButtonReleased

event System.Action<float> xAxisRotated
event System.Action<float> yAxisRotated
event System.Action<float> zAxisRotated
event System.Action<Quaternion> RotationChanged

string GetDeviceTypeID()
string GetDeviceName()
bool IsInUse()
```

2. Localization

Module used to easily implement different localizations for your project. You can find some predefined values in the file 'Localization/Localization.csv'. Please make sure to always correctly translate those values if you choose to add additional languages.

You can easily update changes made in external programs used for editing the csv file by selecting ICAROS -> Localization -> Load CSV File from the Unity menu bar.

2.1. LocalizationManager

Accessible in the UnityEditor:

LocalizationFileName -> Choose the path/name of your localization file.
DefaultLanguage -> This is the language the game starts at if the user did not choose a preferred language yet.

Easy Access Methods for your Scripts:

void SetLanguage (string languageID)
Change the currently used language to the language of the given ID. It is not recommended to use this to ignore the users settings.

string Get (string tokenID)
returns the text for the given token in the currently selected language.

2.2. LocalizedText

Convenience MonoBehaviour. Can be added to a Unity.UI.Text element via drag and drop and will automatically fill out the text element with the given tokenID using the LocalizationManager.

3. UI

This module is used for easy access to and integration of UI elements in the main menu chain. [This is not meant for or a replacement of in-game UI]

3.1. UISystem

Accessible in the UnityEditor:

MenuCamera -> Camera used for displaying monitor/screen only elements.
VRCamera -> Your VRCameraRig used for displaying elements on the hmd.
CompanyLogoTemplate -> Insert a Unity-GameObject containing your Companies Logo(s) here. It will be displayed after the Health and Safety Warning Screen.
CompanyLogoDisplayTime -> The time you're the above screen will be displayed for, in seconds.

Easy Access Methods for your Scripts:

void Restart()
Restart thewhole UI chain starting with Health and Safety Warning.

void BackToMainMenu()
Restart the UI chain at the Main menu.

void BackToExternalCameraView()
Restart the UI chain at the external camera view.

void AddLanguageToOptions(string languageID)
Add 'languageID' to the list of available languages. Make sure the chosen ID is already existent in your Localization file. (Example: 'EN' for english)

void RegisterOnPlayFunction(System.Action OnPlay)
Choose a method that listens for presses of the 'play'menu-item by the player.

void RegisterOnMenuItemSelectedFunction(System.Action<string> OnMenuItemSelected)
Choose a method that listens for other menu-item presses. The given string parameter contains the ID chosen for the menu-item when calling RegisterMenuItem and respective variants.

void RegisterMenuItem(string Id, string Title, string Parent)
Register a new menu-item as child element of the ID given in 'Parent'. 'Id' takes the ID of the new element. 'Title' takes the tokenID of the respective text in Localization.

void RegisterPlayMenuItem(string Id, string Title)
Same as the above for registering directly in the play menu.

void RegisterOptionsMenuItem(string Id, string Title)
Same as the above for registering directly in options menu.

void RegisterUnlistedMenuItem(string Id, string Title)
Same as the above for registering menu-items outside the default menu flow. Responsibility to correctly open the menu and direct menu flow lies with the caller of the method.

IMPORTANT(!): All OpenMenu methods may only be called while still in the menu screen. Otherwise refer to BackToMainMenu.

void OpenMenu(string Id)
Open the previously registered menu element with ID ‘Id‘. Make sure this element has children to display.

void OpenMainMenu()
Open the main menu.

void OpenOptionsMenu()
Directly skip to the options menu.

void CloseUI()
Close the UI System. May only be called once the player successfully chose a device (which includes accepting Health and Safety Warnings).

It is not recommended to change or touch any part of the SDK not mentioned in this document.