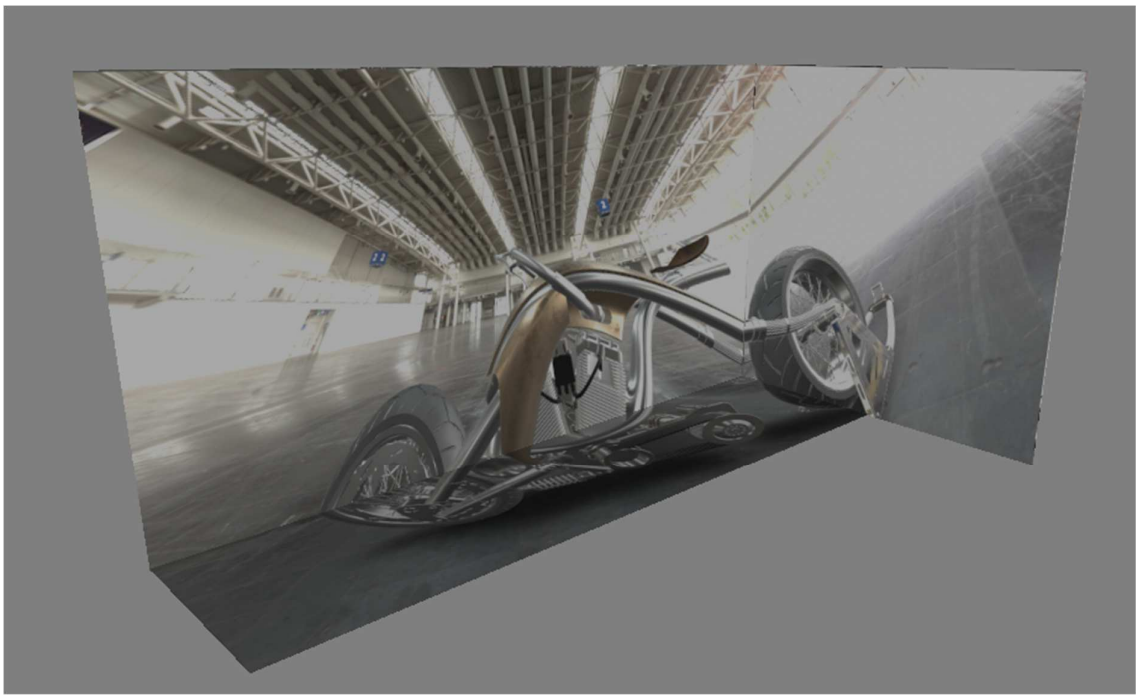
IHD Virtual Corner-CAVE

Author: Michael Adrian
Created: 07.05.2019
Modified: 07.05.2019
Release: 2019x FD01



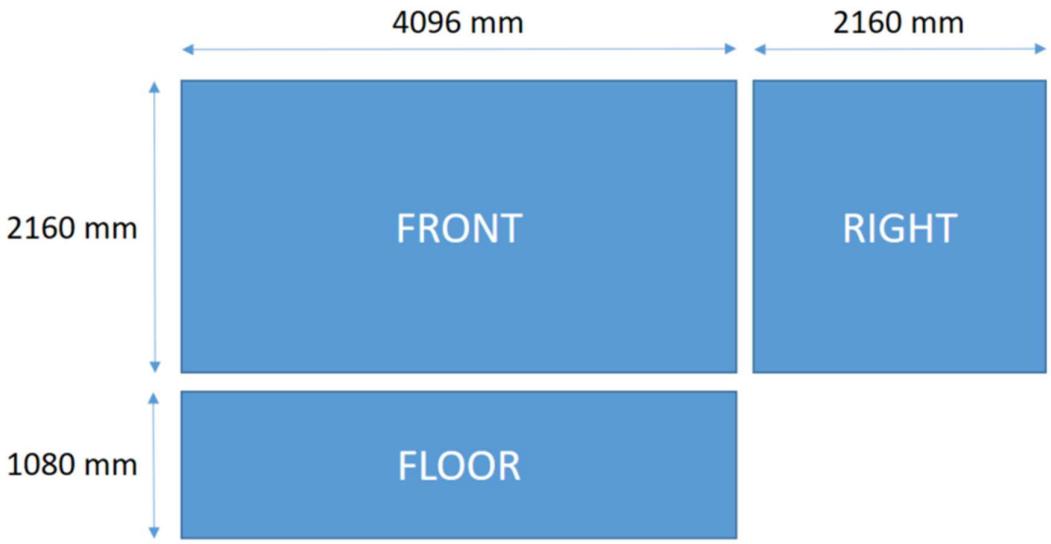
This is a configuration to simulate a typical Corner-CAVE like the LiVes on a two screen desktop. A game-pad is used to experience the immersive navigation mode. To try this set-up right away you need:

- 22 IHD tokens (if not available, a watermark will be shown)
- A dual screen desktop with a resolution of 1920x1080 per screen (If your resolution differs, you need to adjust the coordinates accordingly.)
- A game-pad like a Microsoft Xbox controller
- An XML-Editor like *NOTEPAD++*.

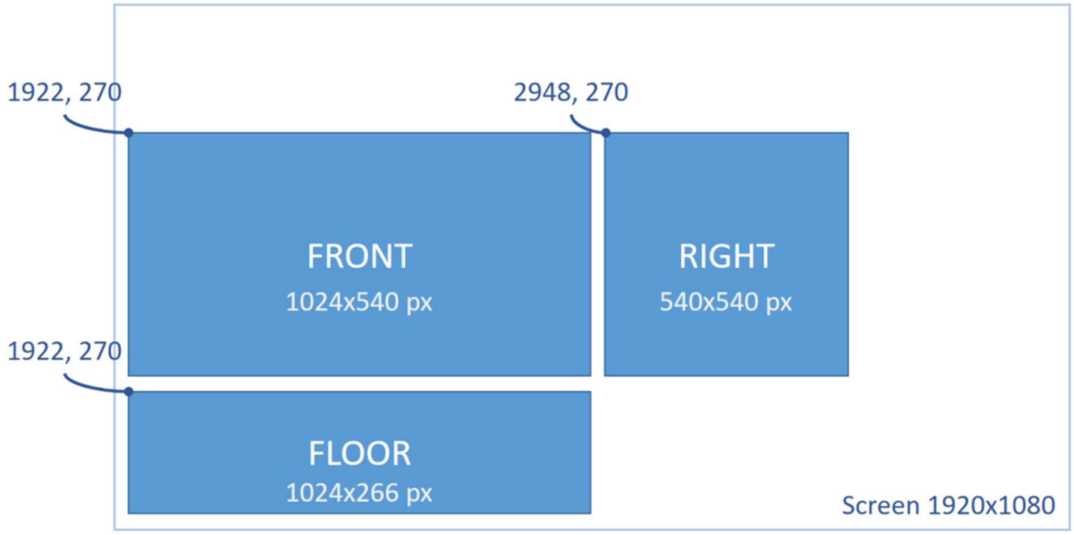
The screens of the virtual CAVE will be displayed on the rightmost screen of your windows desktop.

The CAVE Layout

The CAVE layout contains three screens: The main front-screen, a right and a floor screen. The total dimensions of our virtual CAVE are 4096x2160x2160 millimeter, whereas the floor screen covers only half of the bottom:



This layout is mapped onto three windows on the right screen of your desktop. A seam of two pixels is added to divide the windows for the sake of clarity:



The picture above shows the resolution and desktop offset in pixel for each window. Please note, that the resolution aspect ratio needs to match the physical dimensions, i.e. if your wall is 2000x1000mm, your resolution needs to have an aspect ratio of 2:1 (like 2048x1024).

In our example we simulate a 4kUHD front-wall with 4096x2160 pixel, thus we go for a 17:9 aspect ratio. The right screen has an aspect ratio of 1:1, the floor screen is 34:9.

The SUI Model

The underlying paradigm of an extended **IHD** configuration is the **SUI** model. **SUI** is an acronym for **S**ensorial **U**ser in **I**nteraction and defines a user and his physical environment. Within the scope of the document we will concentrate on a very straightforward use of this model.

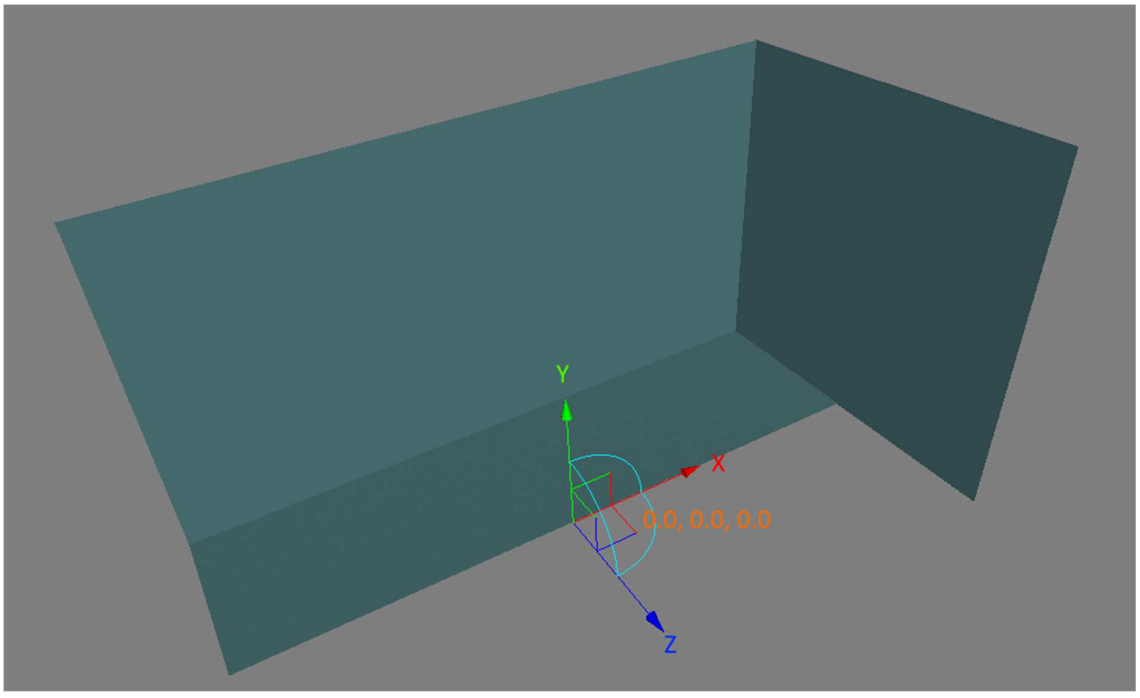
Simply spoken, the **SUI** model defines three levels, very much like the levels in a scene-tree:

- The **Interaction Context** – The topmost container and "root" origin for all subsequent transformations
- The **User** and the **Physical Environment** – Children of the **Interaction Context** defining the user and the displays
- The **Head** and **Hand** of the **User** – Children of the **User** defining its head and hand

All of these components can be transformed in relation to the origin of their parent.

Matching the virtual and the real

The most important task to perfectly match the virtual with the real world is a common reference point where all transformations and measurements in both worlds relate to. In the virtual world the world-reference-point is the position of the **Interaction Context**. To keep the example simple, we leave this point at its default position, which is 0,0,0. In the real world we can choose whatever position should match the virtual 0,0,0 position. Typically, it is the center of the CAVE floor or the center of the CAVE volume. In our example we choose the center of the CAVE floor:



The origin of our interaction context is mapped to the center of the CAVE floor. This "mapping" is not defined anywhere, it is a general determination we make in our mind to get things started.

The iV-Configuration XML

Any iV scenario like a CAVE or Immersive Wall needs a configuration file. This file defines

- The physical environment such as the screens
- The user together with its head and hands
- A variety of input devices such as trackers or in our case a game-pad

Sample configuration files can be found in your 3DEXPERIENCE installation folder under

```
..\Dassault Systemes\B420\win_b64\resources\xml
```

We use the file `CAVE_Desktop_2Screens_Gamepad.xml` as a starting point and modify it to fit our set-up described above. Please load this file into your XML editor **and save it under a different name in a location of your choice**.

Creating the configuration

Let's go through the XML file step by step and modify each section according to our needs.

```
<?xml version="1.0" encoding="utf-8"?>

<IVConfiguration name="myIVConfigFile" version="0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="IVConfigurationV1.xsd">
```

These first lines can be ignored, leave them as they are.

```
<InputDevices>
    <Device
        type="Gamepad"
        name="Gamepad_Alias" />
  </InputDevices>
```

The next section defines the Gamepad as an input device. It'll be used later by its name "`Gamepad_Alias`". **NOTE: If you want to try this exercise without having a gamepad attached, you need to delete the complete section above, otherwise the scenario will not launch.**

```
<SUI>
  <SUIScenario name="CAVEScenario" type="CAVE">
```

This is the main section. The **`SUI`** section defines a name and the type of scenario. For a multi-pipe scenario, the type is "**CAVE**".

```
<InteractionContext name="InteractionContext"
   virtualPosition="float3(0.0,0.0,0.0)"
  virtualRotation="euler_zxy(0.0,90.0,0.0)">
```

The next section describes the **`Interaction Context`** with its name, its position and orientation. As our interaction context doesn't need to be rotated, we can set the x rotation value to 0:

```
  virtualRotation="euler_zxy(0.0,0.0,0.0)">
```

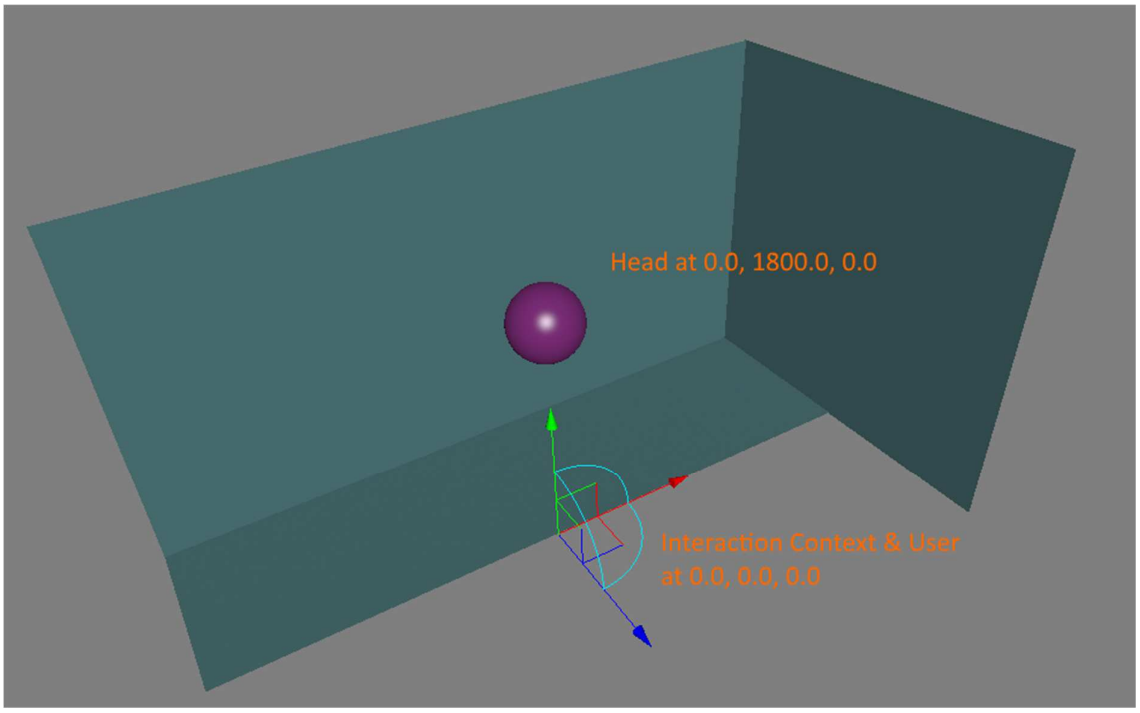
Configuring the User

```
<User name="User" physicalPosition="float3(0.0,0.0,0.0)" >
   <Head name ="Head" physicalPosition="float3(0.0,0.0,500.0)"/>
</User>
```

In the user section the **User** is defined with its position (relative to the **Interaction Context**) and its **Head**, which can be positioned relative to the **User**. We want to place the user at the room origin thus leaving its position at **0,0,0**. The head should be 1.8 meter above the ground, so we need to change the **Head** entry:

```
<User name="User" physicalPosition="float3(0.0,0.0,0.0)" >
   <Head name ="Head" physicalPosition="float3(0.0,1800.0,0.0)"/>
</User>
```

Our scenario now looks like this:



Configuring the Physical Environment

The most demanding part is the description of the physical environment, which in our case consists of three screens. To create a perspective correct, 3-dimensional, 1:1 scale picture, the system needs to know the size and transformation of each screen.

The first line…

```
<PhysicalEnvironment
 physicalPosition="float3(0.0,0.0,0.0)"
 physicalOrientation="euler_zxy(0.0,0.0,0.0)">
```

…can be left as it is. In specific cases it can be used to transform the complete physical set up relative to the **`Interaction Context`**. In our configuration are two **`Screen`** entries that we don't need. *Please delete them:*

```
<Screen
 name="LeftMainScreen"          delete!
 .
 .
 .
/>

<Screen
 name="RightMainScreen"
 .
 .
 .
/>
```

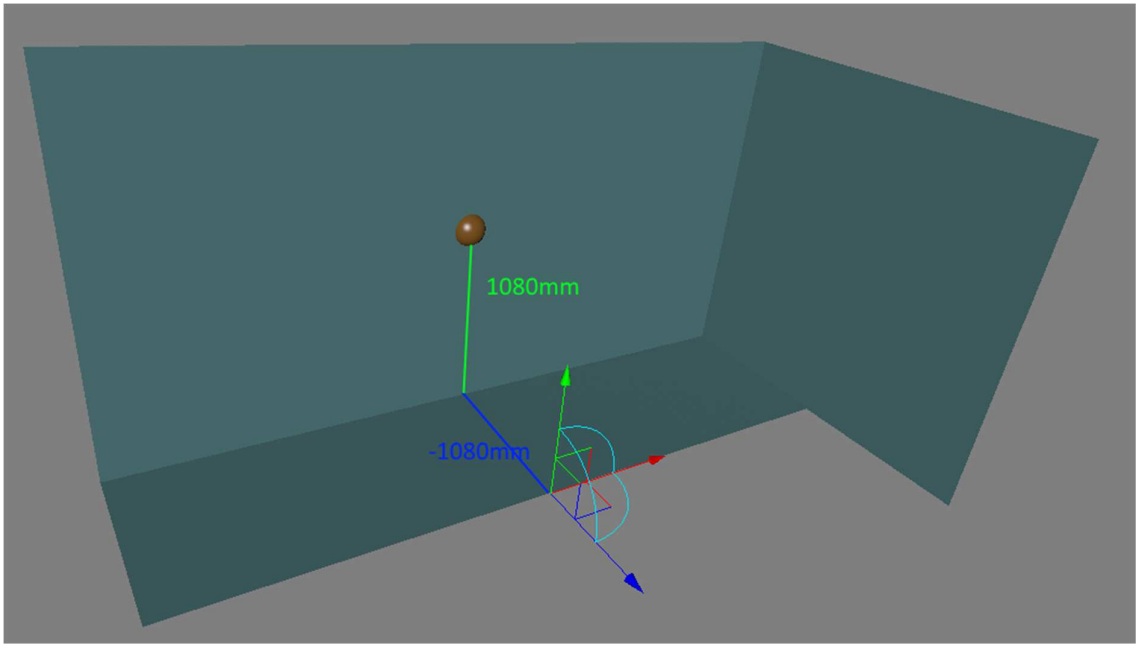
Next we fill in the screen description according to our parameters given in the first section "*The CAVE layout*". We will define three screens: A front- a right- and a floor-screen. The following entry is common to all three descriptions:

```
<Screen
  name="screenname"
  mainScreen="true"
. activeStereoscopy="false"
…
/>
```

The only difference is the **mainScreen** switch. This one is set to **true** for the front-screen and **false** for the other two. Name, position, orientation and resolution is individual for each screen. Here is the description in detail:

The front-screen

To set the screen position, we need to know the offset from its center to the room-origin, which is the center of the **`Interaction-Context`**. For the front-screen the offset is the height of the floor-screen on the negative Z-Axis and the half of the height of the front screen on the positive Y-Axis:



For the screen description, this gives us the following lines:

```
physicalPosition="float3(0.0,1080.0,-1080.0)"
physicalOrientation="euler_zxy(0.0,0.0,0.0)"
```

The next entry describes the physical size of the screen. Our front screen is 4096-millimeter-wide and 2160-millimeter-high, which gives us this entry:

```
physicalSize="float2(4096,2160)"
```

Now we need to describe the position and size of the screen on our desktop. In a full-blown multi-pipe system each screen will correspond with one or more projectors but in principle it is a large desktop with a number of screen- (or window-) definitions. To simulate such a layout, we will open three windows on the right screen of our 2xHD desktop as shown in section “The CAVE Layout”.

In the screen definition we need to enter the screen coordinates of the top-left window corner and the resolution of the window. For the front-screen we insert

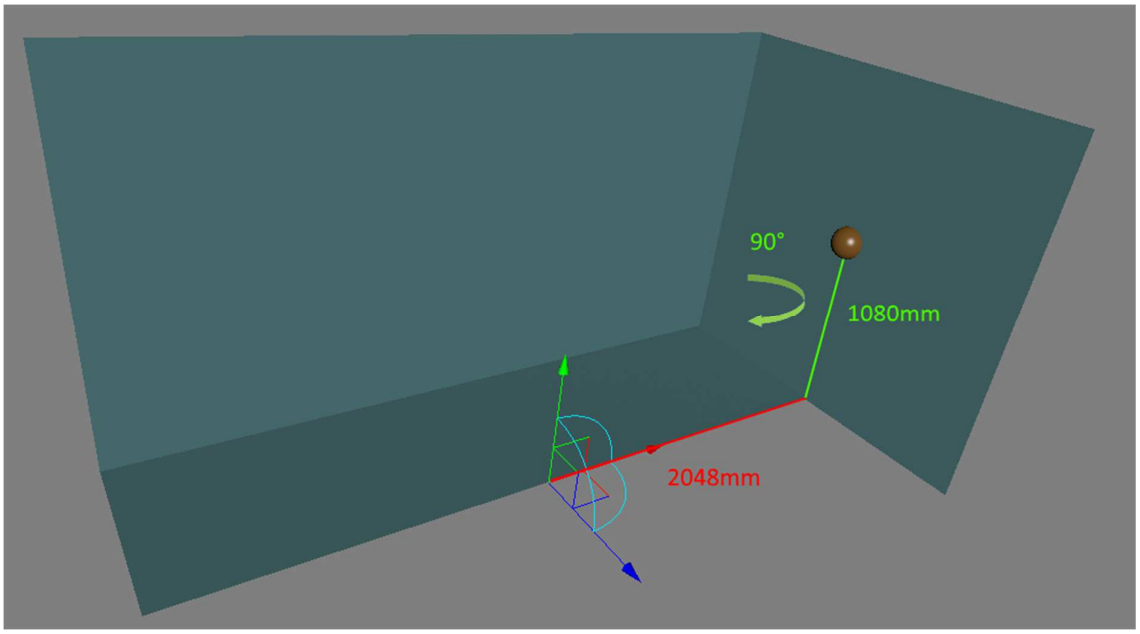
```
displayPosition="int2(1922,270)"
displayResolution="int2(1024,540)"
```

Here is the complete entry for the front screen:

```
<PhysicalEnvironment
 physicalPosition="float3(0.0,0.0,0.0)"
 physicalOrientation="euler_zxy(0.0,0.0,0.0)">

 <Screen
   name="Front"
   mainScreen="true"
   activeStereoscopy="false"
   physicalPosition="float3(0.0,1080.0,-1080.0)"
   physicalOrientation="euler_zxy(0.0,0.0,0.0)"
   physicalSize="float2(4096,2160)"
   displayPosition="int2(1922,270)"
```

```
   displayResolution="int2(1024,540)"
 />

</PhysicalEnvironment>
```

The right-screen



The right-screen is rotated 90 degrees clockwise around its Y-Axis. We need to define this in the physical parameters:

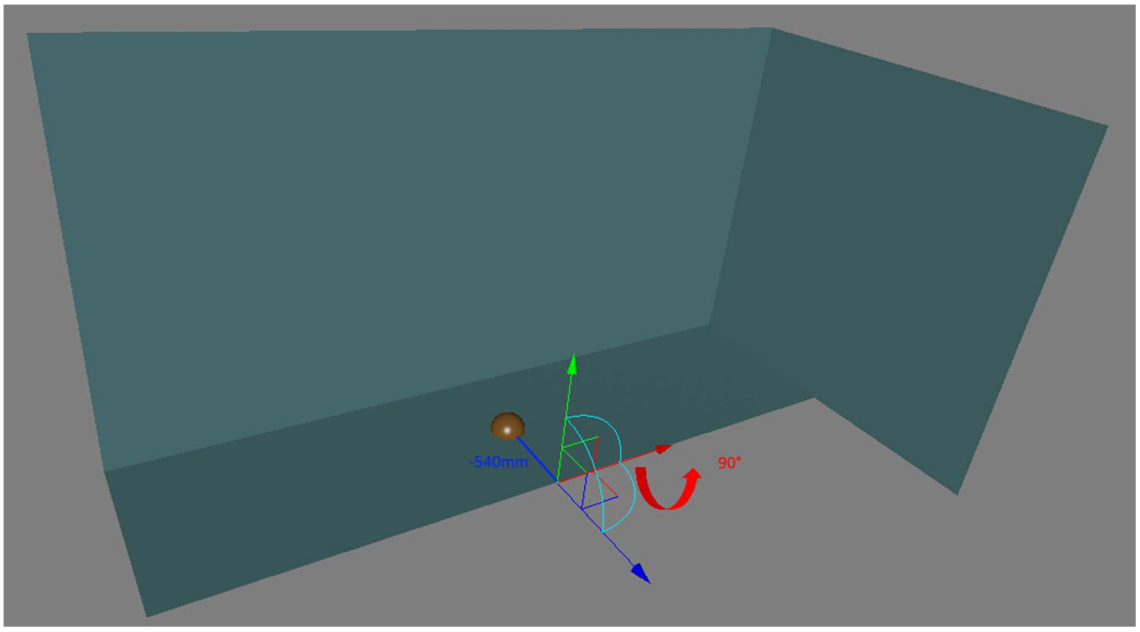
```
   physicalPosition="float3(2048.0,1080.0, 0.0)"
   physicalOrientation="euler_zxy(0.0,0.0,-90.0)"
```

NOTE: Please be careful, the order for a euler-rotation is given in the order ZXY, not XYZ.

All other parameters are given like in the front-screen description. Here's the complete entry for the right screen:

```
<Screen
   name="Right"
   mainScreen="false"
   activeStereoscopy="false"
   physicalPosition="float3(2048.0,1080.0,0.0)"
   physicalOrientation="euler_zxy(0.0,0.0,-90.0)"
   physicalSize="float2(2160,2160)"
   displayPosition="int2(2948,270)"
   displayResolution="int2(540,540)"
/>
```

The floor-screen



Finally, he floor-screen is rotated 90 degrees clockwise around its X-axis. It's physical parameter entries are:

```
physicalPosition="float3(0.0,0.0,-540.0)"
physicalOrientation="euler_zxy(0.0,-90.0, 0.0)"
```

Here's the complete entry for the floor-screen:

```
<Screen
   name="Floor"
   mainScreen="false"
   activeStereoscopy="false"
   physicalPosition="float3(0.0, 0.0,-540.0)"
   physicalOrientation="euler_zxy(0.0,-90.0, 0.0)"
   physicalSize="float2(4096,1080)"
   displayPosition="int2(1922,812)"
   displayResolution="int2(1024,266)"
/>
```

That's it. We're done with the screens, now let's add some interaction device.

Configuring the interaction-device

If you scroll down in the configuration file, you'll see two sections, which rely to the configuration of your input devices and the navigation commands. These sections are

```
<DeviceMapping>

   mapping a device-axis to a transformation-action
   mapping a button to an event-action

</DeviceMapping>
```

and

```
<Navigation>

   applying a mapped transformation to an object (aka the user)
   applying a mapped event to a command

</Navigation>
```

First of all, let's clean this sections as we will fill them from scratch. Delete all lines in the <Navigation> and <DeviceMapping> sections:

```
  <Navigation>
    <NavigationMode name="NoNav">
      <Transform …
…
  </Navigation>

  <DeviceMapping>
    <AnalogMapping …
…
  </DeviceMapping>
```

In the <Navigation> section we now create the section for our one and only navigation-mode (NOTE: There can be multiple navigation modes but for the sake of this example we only use one.). Please insert the following lines:

```
<Navigation>

  <NavigationMode name="Fly" default="true">

  </NavigationMode>

</Navigation>
```

This entry creates a navigation-mode named "Fly" and defines it to be our default mode.

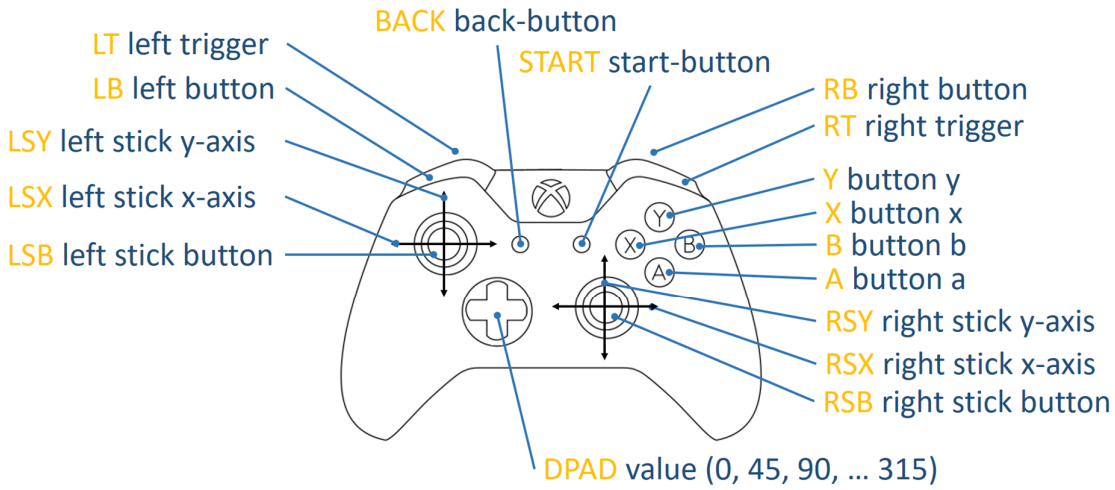
Now let's have a look on the paradigm behind these two sections:

In the <DeviceMapping> section we can assign the features of our input-device to actions we will use in our scenario. The input-device needs to be declared in the <InputDevices> section (also see above):

```
<InputDevices>
    <Device
        type="Gamepad"
        name="Gamepad_Alias" />
  </InputDevices>
```

In our scenario we use a game-pad, which is defined by the "type" parameter "Gamepad" and we will refer to it using the defined alias "Gamepad_Alias" (you can choose whatever alias you like).

The features of our game-pad we can use in our scenario are:



NOTE: The STICK and TRIGGER features deliver analog values, while the BUTTON and DPAD features deliver fixed values.

To understand how a feature of the device can be used to navigate the environment, let's have a look at an example. Say we want to use the y-axis of the left control-stick to move the user forward and backward in the scenario. First, we create an entry in the `<DeviceMapping>` section to map the axis to a transformation action:

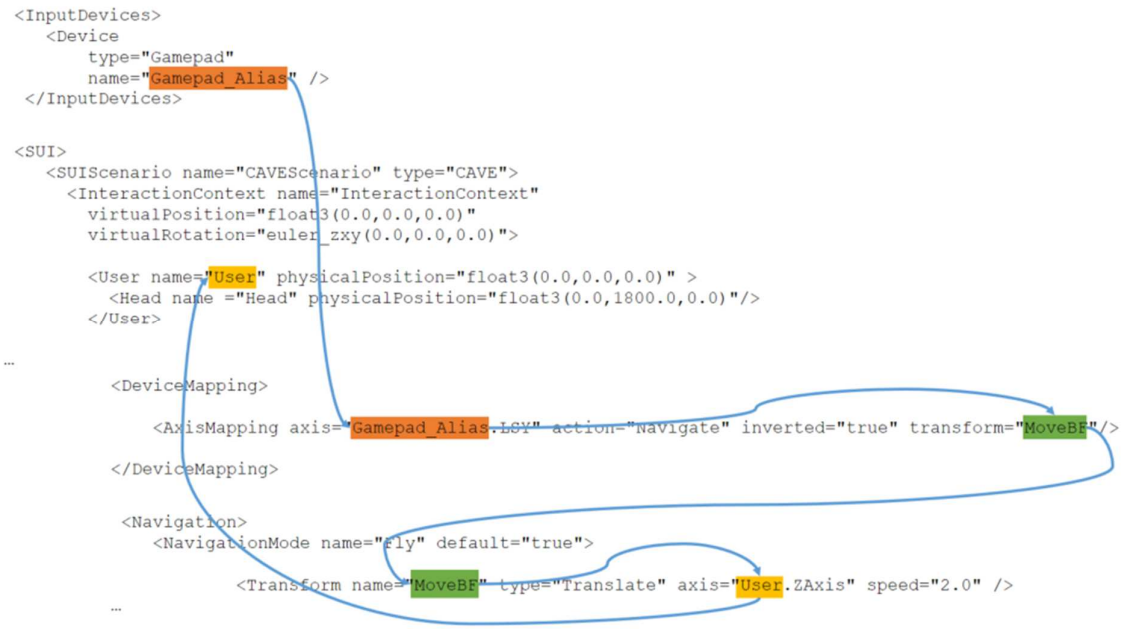
```
<DeviceMapping>

  <AxisMapping axis="Gamepad_Alias.LSY" action="Navigate"
  inverted="true" transform="MoveBF"/>

</DeviceMapping>
```

We use the "`AxisMapping`" entry to map the axis "`LSY`" of the device "`Gamepad_Alias`" to the transformation feature "`MoveBF`" within the action "`Navigate`". Since our Z-Axis is negative towards the screen, we need to invert the value.

Now we need to apply this feature to the user in the SUI model. In the `<NavigationMode>` section, we create a corresponding entry:

```
<NavigationMode name="Fly" default="true">

  <Transform name="MoveBF" type="Translate" axis="User.ZAxis"
  speed="2.0" />

<\NavigationMode>
```

The "`Transform`" entry now applies the mapping as a translation to the z-axis of the user. The "`speed`" parameter acts as a factor to the input device value. Here's an illustration trying to summarize the dependencies in our device-configuration:



NOTE: You can apply a transformation not only to the user but to any SUI element.

Now let's define the other mappings and navigation actions in a similar way. The yellow entries go into the <NavigationMode> section, the blue ones into the <DeviceMapping> section:

```
<AxisMapping axis="Gamepad_Alias.LSX" action="Navigate"
transform="MoveLR"/>

<Transform name="MoveLR" type="Translate" axis="User.XAxis"
speed="2.0" />
```

These two lines map the left sticks x-axis to the x-axis of the user, thus allowing for left-right movement (strafe).

```
<AxisMapping axis="Gamepad_Alias.LT" action="Navigate"
inverted="true" transform="MoveD"/>
<AxisMapping axis="Gamepad_Alias.RT" action="Navigate"
transform="MoveU"/>

<Transform name="MoveU" type="Translate" axis="Head.YAxis"
speed="2.0" />
<Transform name="MoveD" type="Translate" axis="Head.YAxis"
speed="2.0" />
```

This are the entries for the up-down movement. We map the left and right trigger to the heads y-axis in a way that LT moves the head up while RT moves it down.

```
<AxisMapping axis="Gamepad_Alias.RSY" action="Navigate"
inverted="true" transform="Pitch"/>
<AxisMapping axis="Gamepad_Alias.RSX" action="Navigate"
inverted="true" transform="Yaw"/>
```

```
<Transform name="Pitch" type="Rotate" axis="Head.XAxis"
center="Head.Origin" speed="1.0" />

<Transform name="Yaw" type="Rotate" axis="float3(0.0,0.0,1.0)"
center="Head.Origin" speed="1.0" />
```

These entries allow the rotation of the user's head. The first pair of entries do a pitch-action. It is mapped to the right sticks x-axis and results in a rotation of the user's head around the x-axis.

The second pair turns the user around. To define this turn, we rotate the head of the user around the worlds y-axis, so instead of an object axis, we use a fixed vector, given as a float3.

The last two entries are a bit special. Since rotation around two axes can easily mess up the view, we need a reset-button to level the view. Here is the definition:

```
<ButtonMapping action="Navigate" transform="ResetHorizontal"
button="Gamepad_Alias.BACK"/>

<Transform name="ResetHorizontal" type="ResetHorizontal"/>
```

With these entries we map the back-button to the "`ResetHorizontal`" action.

The complete navigation-and device-section:

```
<Navigation>
  <NavigationMode name="Fly" default="true">

    <Transform name="MoveBF" type="Translate" axis="User.ZAxis"
speed="2.0" />
    <Transform name="MoveLR" type="Translate" axis="User.XAxis"
speed="2.0" />
    <Transform name="Pitch" type="Rotate" axis="Head.XAxis"
center="Head.Origin" speed="1.0" />
    <Transform name="Yaw" type="Rotate" axis="float3(0.0,0.0,1.0)"
center="Head.Origin" speed="1.0" />
    <Transform name="MoveU" type="Translate" axis="Head.YAxis"
speed="2.0" />
    <Transform name="MoveD" type="Translate" axis="Head.YAxis"
speed="2.0" />
    <Transform name="ResetHorizontal" type="ResetHorizontal"/>

  </NavigationMode>
</Navigation>

<DeviceMapping>

  <AxisMapping axis="Gamepad_Alias.LSY" action="Navigate"
inverted="true" transform="MoveBF"/>
  <AxisMapping axis="Gamepad_Alias.LSX" action="Navigate"
transform="MoveLR"/>
  <AxisMapping axis="Gamepad_Alias.RSY" action="Navigate"
inverted="true" transform="Pitch"/>
  <AxisMapping axis="Gamepad_Alias.RSX" action="Navigate"
inverted="true" transform="Yaw"/>
  <AxisMapping axis="Gamepad_Alias.LT" action="Navigate"
inverted="true" transform="MoveD"/>
  <AxisMapping axis="Gamepad_Alias.RT" action="Navigate"
transform="MoveU"/>

  <ButtonMapping action="Navigate" transform="ResetHorizontal"
button="Gamepad_Alias.BACK"/>

</DeviceMapping>
```

And finally the complete configuration file.

NOTE: If you don't have a game-pad connected, please remove the entry marked blue. Otherwise you will not be able to start the session

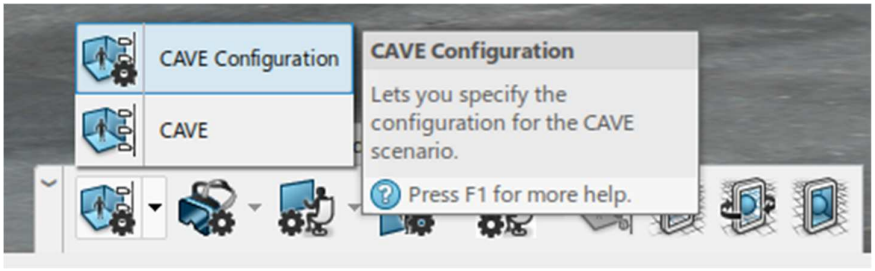
```
<?xml version="1.0" encoding="utf-8"?>

<IVConfiguration name="myIVConfigFile" version="0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="IVConfiguration.xsd">

  <InputDevices>
    <Device
      type="Gamepad"
      name="Gamepad_Alias" />
  </InputDevices>


  <SUI>
    <SUIScenario name="CAVEScenario" type="CAVE">
      <InteractionContext name="InteractionContext"
        virtualPosition="float3(0.0,0.0,0.0)"
        virtualRotation="euler_zxy(0.0,0.0,0.0)">

        <User name="User" physicalPosition="float3(0.0,0.0,0.0)" >
          <Head name ="Head" physicalPosition="float3(0.0,1800.0,0.0)"/>
        </User>

        <PhysicalEnvironment physicalPosition="float3(0.0,0.0,0.0)"
physicalOrientation="euler_zxy(0.0,0.0,0.0)">

          <Screen
            name="Front"
            mainScreen="true"
            activeStereoscopy="false"
            physicalPosition="float3(0.0,1080.0,-1080.0)"
            physicalOrientation="euler_zxy(0.0,0.0,0.0)"
            physicalSize="float2(4096,2160)"
            displayPosition="int2(1922,270)"
            displayResolution="int2(1024,540)"
          />

          <Screen
            name="Right"
            mainScreen="false"
            activeStereoscopy="false"
            physicalPosition="float3(2048.0,1080.0,0.0)"
            physicalOrientation="euler_zxy(0.0,0.0,-90.0)"
            physicalSize="float2(2160,2160)"
            displayPosition="int2(2948,270)"
            displayResolution="int2(540,540)"
          />
```

```
          <Screen
            name="Floor"
            mainScreen="false"
            activeStereoscopy="false"
            physicalPosition="float3(0.0,0.0,-540.0)"
            physicalOrientation="euler_zxy(0.0,-90.0,0.0)"
            physicalSize="float2(4096,1080)"
            displayPosition="int2(1922,812)"
            displayResolution="int2(1024,266)"
          />
        </PhysicalEnvironment>
      </InteractionContext>
    </SUIScenario>
  </SUI>

  <Navigation>
    <NavigationMode name="Fly" default="true">
      <Transform name="MoveBF" type="Translate" axis="User.ZAxis"
speed="2.0" />
      <Transform name="MoveLR" type="Translate" axis="User.XAxis"
speed="2.0" />
      <Transform name="Pitch" type="Rotate" axis="Head.XAxis"
center="Head.Origin" speed="1.0" />
      <Transform name="Yaw" type="Rotate" axis="float3(0.0,0.0,1.0)"
center="Head.Origin" speed="1.0" />
      <Transform name="MoveU" type="Translate" axis="Head.YAxis"
speed="2.0" />
      <Transform name="MoveD" type="Translate" axis="Head.YAxis"
speed="2.0" />
      <Transform name="ResetHorizontal" type="ResetHorizontal"/>
    </NavigationMode>
  </Navigation>

  <DeviceMapping>
    <AxisMapping axis="Gamepad_Alias.LSY" action="Navigate" inverted="true"
transform="MoveBF"/>
    <AxisMapping axis="Gamepad_Alias.LSX" action="Navigate"
transform="MoveLR"/>
    <AxisMapping axis="Gamepad_Alias.RSY" action="Navigate" inverted="true"
transform="Pitch"/>
```

```
    <AxisMapping axis="Gamepad_Alias.RSX" action="Navigate" inverted="true"
transform="Yaw"/>
    <AxisMapping axis="Gamepad_Alias.LT" action="Navigate" inverted="true"
transform="MoveD"/>
    <AxisMapping axis="Gamepad_Alias.RT" action="Navigate"
transform="MoveU"/>

    <ButtonMapping action="Navigate" transform="ResetHorizontal"
button="Gamepad_Alias.BACK"/>

  </DeviceMapping>


</IVConfiguration>
```

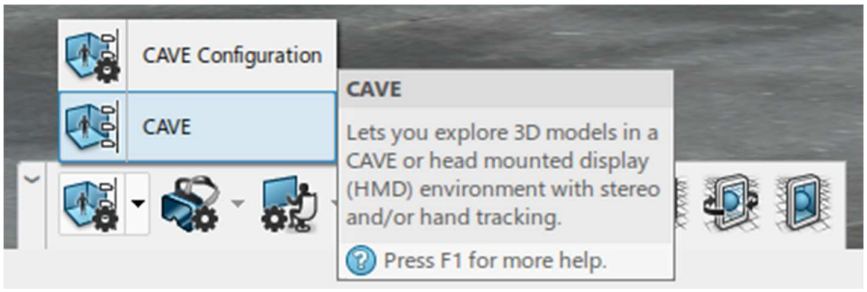
Loading and testing the scenario

Now let's see the result of our configuration in 3DEXPERIENCE. Open a scene in any CATIA 3D-App like "CATIA Live Rendering". From the VR/AR panel select "CAVE Configuration".



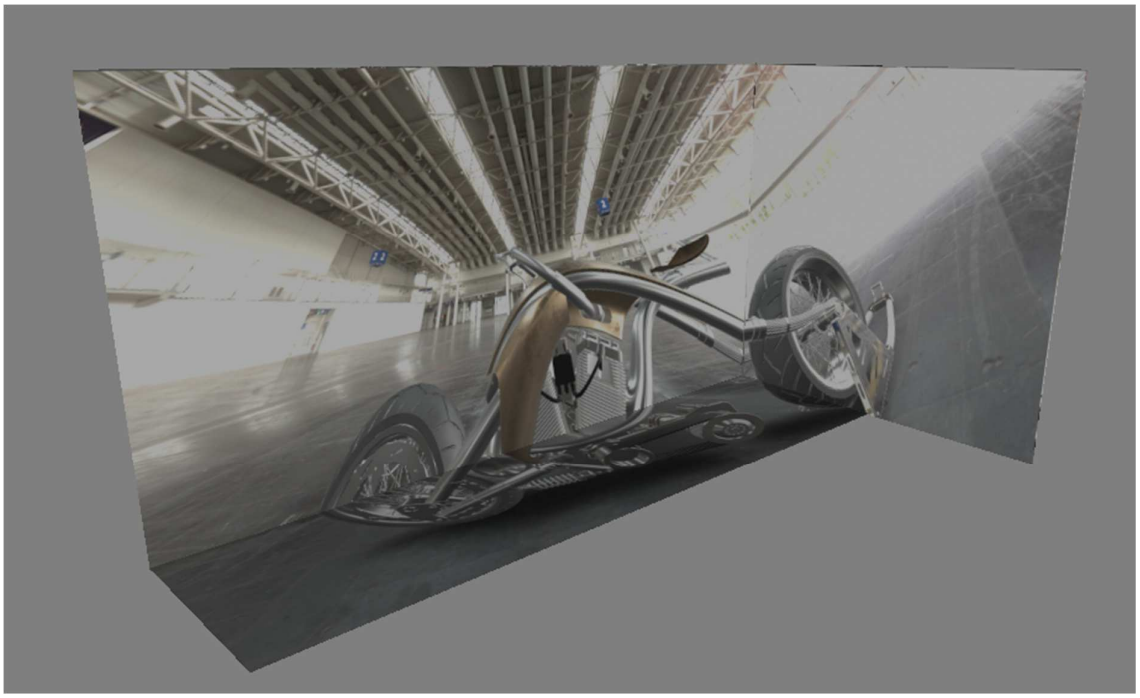
In the section "Immersive Environment Files" click the [+] button and pick the .XML file you've just created.

In the "Scenario" section select "Auditorium", in the "Model" section select "Custom Scale" and leave the factor at 1.0. Click [OK]. Now select CAVE to start the CAVE scenario:



After a short while the CAVE screens should show up on your rightmost screen (or elsewhere if configured different):

Don’t mind the perspective distortion. The result looks a bit wired on a flat screen but it makes sense as soon as we map it on a virtual model with the correct geometry:



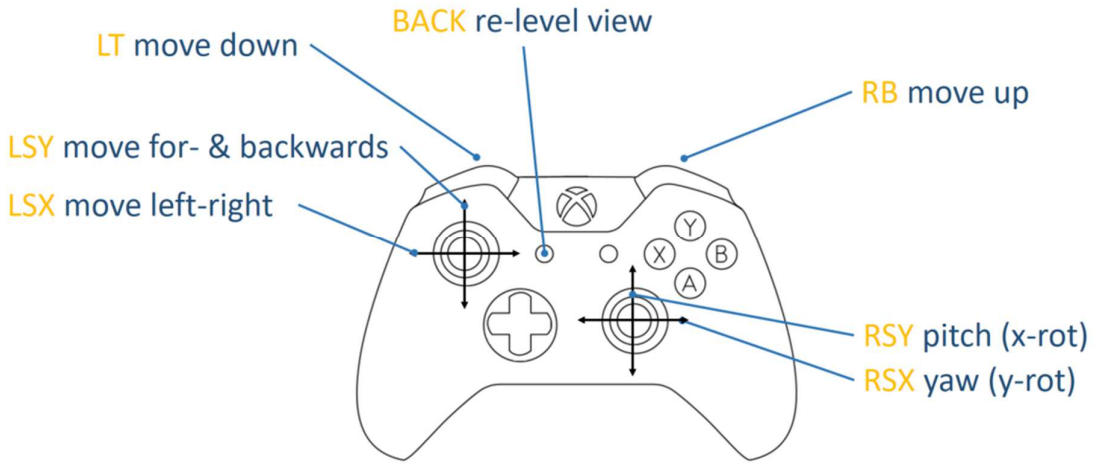
Auditorium & Immersive Scenario

You can now navigate the scenario from your desktop screen as you know it. The virtual CAVE screens will follow within the limits that emerge from the fact that the desktop screen uses another viewing geometry than the CAVE. This is a necessary mechanism to keep the two views as consistent as possible.

As you might see, the game-pad has no influence on the navigation here. This is a feature of the “Auditorium” mode where the navigation is done by the desktop-operator and the auditorium just watches. To allow for auditorium navigation, we need to switch to “Immersive” mode.

Stop the scenario by again clicking on the CAVE icon and from the CAVE-Configuration panel select “Immersive” in the “Scenario” section. Now start the CAVE scenario again.

In “Immersive” mode you are now able to navigate the environment using the game-pad. According to our mapping in the configuration file, we can use these controls:



That’s it, we’re done! We’ve created a non-trivial configuration of a three sided CAVE. In theory we could slightly modify the “Screen” configuration and use it with some real CAVE hardware.

Additional information

Here's some additional information if you want to know more about advanced IHD set-ups.

Configuring more than one Navigation mode

If you want to create your own navigation-mode in addition to the one described in this document, you can simply add it to the navigation section:

```
<NavigationMode name="Walk">
.
.
.
  <Constraint name="ZOffsetCst" type="LimitPositionOffset"
coordinate="Z" minValue="0" maxValue="0" />
.
.
.
</NavigationMode>
```

You should name your addition mode different from the first one like "Walk" in our example here. Now we need an action to switch between the modes. Here is the configuration entry for the `<DeviceMapping>` section. Since this action is switching the navigation mode itself, it has no corresponding entry there:

```
<ButtonMapping type="SwitchNavigationMode"
button="Gamepad_Alias.START"/>
```

Additional configuration settings

In the above example we used a "`Constraint`" setting to limit the user to only navigate on ground-level. If you want to know more about the configuration settings open your online user-assistance and navigate to

There'll you find some additional information about how to configure an advanced iV-scenario for your individual needs.

If you have a particular question, feel free to contact me: Michael.ADRIAN@3ds.com