Lab 5: Advanced camera handling and interaction
Learning goals:
1. Understanding motion tracking and interaction using Augmented Reality Toolkit
2. Develop methods for 3D interaction.
3. Understanding table projection using Window on World metaphor

Tasks: Land elevation of Scandinavia
1. Calculate the field of view (FOV) of the webcam and set the same value to the FOV of the virtual camera. Also calculate the distance from the virtual camera to the backdrop. The kanji marker pattern seen in the backdrop should at all positions and angles be covered by the virtual marker.
2. Create an interaction method using the supplied terrain patch to show land elevation for the past 1000 years. You shall keep the webcam in a static position and may only move the kanji marker pattern to interact. The \texttt{land\_patch\_node} shall keep the same orientation as the backdrop but scale relatively to the size and position of the marker. The land elevation should be presented in a visually pleasing way and not according to scientific facts.
3. Use the Window on World interaction metaphor with the marker pattern to do head tracking. First setup the Window on World to comply with your physical screen. Then use the ARToolkit and the marker pattern to control the WoW \texttt{eye\_node}. Fold a Post-it note in such a way that it can attach to your display, see Figure 2. The post-it note should have a square perpendicular to the screen. Model an equivalent virtual object (a quad) in such a way that it maps to the physical object. When the wow eye node is translated the virtual object should map to the real object.
   Hint: Place the webcam closely below your screen. To correct the difference between the center of the webcam and the center of the WoW \texttt{window\_node} apply a translation.
Prerequisites

You are given sample code that uses a toolkit for Augmented Reality. Augmented Reality is a way of adding computer generated imagery to a view of the real world.

Def: Augmented Reality is commonly defined by the following characteristics:
1. Combines real and virtual
2. Interactive in real time
3. Registered in 3-D

Azuma, 1997

The toolkit, called ARToolkit, uses a camera to identify a marker pattern printed on an ordinary paper. It calculates the position and rotation of the marker pattern in relation to the webcam. The ARToolkit provides the transformation matrix \( \text{target\_trans} \) of the marker pattern, with the webcam as its origin. The webcam is looking at a backdrop. The backdrop is modelled as a quad that visualise the image data from the webcam. To co-locate the marker pattern in the backdrop with the virtual marker pattern the field of view of the virtual camera must correspond to the field of view of the webcam, see Figure 1.

The transformation matrix \( \text{target\_trans} \) is applied to the VRT_Node \( \text{marker\_node} \). The \( \text{marker\_node} \) corresponds to the orientation of the paper marker pattern, if:

- The FOV of the virtual camera matches the FOV of the webcam.
- The size and distance between the backdrop

Figure 1 - left: Overview of the scene, right: the scene viewed from origin looking at the camera backdrop

Read about ARToolkit here: [http://www.hitl.washington.edu/artoolkit/](http://www.hitl.washington.edu/artoolkit/)

There might be questions asked on how ARToolkit works when you present the lab. So read the introductory material on the link above.

If you loose your kanji marker pattern you can print a new one using the file pattKanji.pdf

The source code and documentation for ARToolkit is available on the link above. In this VRT lab, source code version 2.65 is used. Though all you need for the lab is include in the sample source code.
The sample program lab5_elevation.exe provides you with the following scene graph. When you solve the first two parts of the lab you can press keyboard **p** to print the scene graph.

```
0 VRT_Nodename : vrtctx->root
  1 VRT_Nodename : default camera node
    1 VRT_Nodename : world coordinate reference frame
      2 VRT_Nodename : x-axis
      3 VRT_Nodename : head
    2 VRT_Nodename : y-axis
      3 VRT_Nodename : head
    2 VRT_Nodename : z-axis
      3 VRT_Nodename : head
  1 VRT_Nodename : backdrop
  1 VRT_Nodename : the scene
    2 VRT_Nodename : marker center
      3 VRT_Nodename : scales earth geometry
        4 VRT_Nodename : rotates earth geometry
          5 VRT_Nodename : translates earth geometry
            6 VRT_Nodename : land patch node
            6 VRT_Nodename : water patch node
      3 VRT_Nodename : world coordinate reference frame
        4 VRT_Nodename : x-axis
          5 VRT_Nodename : head
        4 VRT_Nodename : y-axis
          5 VRT_Nodename : head
        4 VRT_Nodename : z-axis
          5 VRT_Nodename : head
```

Table 1 – The scene graph of the sample program lab5_elevation.exe

**Key interaction with the sample program**

<table>
<thead>
<tr>
<th>Left / right arrow</th>
<th>Control FOV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up / down array</td>
<td>Control distance to backdrop</td>
</tr>
<tr>
<td>S</td>
<td>Change to scene view</td>
</tr>
<tr>
<td>C</td>
<td>Change to camera view</td>
</tr>
<tr>
<td>W</td>
<td>Turn on and off reference frames</td>
</tr>
<tr>
<td>M</td>
<td>Switch between virtual marker and land patch</td>
</tr>
</tbody>
</table>
**Window on World**

To create projections on head tracked table top displays the traditional central projected camera metaphor will not work very well. The window on world metaphor solves some of the problems. When configured correctly the windows on world metaphor will for all perspectives correctly project imagery in such a way that a real object will map correctly to a virtual object of the same size.

The example code in Table 2 shows how you can setup a Window on World. The eye node is positioned in the origin looking at the window node at -2 along the z axis. The code is configured for a screen with the width of 30 cm.

```cpp
void setup_camera_wow_projection()
{
    eye_node = VRT_NodeNew(VRT_RootNode(), "eye node");
    VRT_NodeSetTranslation(eye_node, 0, 0, 0);

    wow_node = VRT_NodeNew(VRT_RootNode(), "wow window");
    VRT_NodeSetTranslation(wow_node, 0, 0, -2.0);

    projector_wow = VRT_WoWNew(eye_node, wow_node);

    VRT_WoWSetWidthAndHeight(projector_wow, 0.30f, (768/1024)*0.30f); // the size of your monitor in meters
    VRT_WoWSetClipDistances(projector_wow, 1.9f, 4.0f);

    VRT_PipeSetProjector(VRT_PipeGetByIndex(0), projector_wow);
}
```

Table 2 – Example source code to setup a window on world projection

![Post-it note protruding as a square perpendicular to the screen](image)

![Virtual Post-it note (a quad)](image)

Figure 2 – Example of Virtual object mapped to a real object