This lecture

- My research (not in your slides)
- Theory
- VRT
  - Basic structures
  - Useful functions
  - Simple demo

Implementation levels

- Application
- Scene Graph lib.
- Hardware

Hardware

- E.g. ATI, nVIDIA
  - A driver is used to communicate with the graphic card

Computer Graphic libraries

- E.g. OpenGL, DirectX
  - Provides a low-level programming interface
  - Functions for drawing graphics, e.g., triangles and lines, etc.

What kind of functionality do we want?

- Easier to understand than OpenGL
- More abstract interface
- Specialized for "Interactive Graphical Systems"
- Easy object loaders
- …

Scene-Graph libraries

- Also known as 3D engines
- Provides functions for
  - loading, rendering, manipulating geometry
  - material, texture and light source management
  - collision detection, level of detail (LOD) rendering, …
- Scene is represented using a graph, usually a directed acyclic graph or a tree
Application programming

Your task, during the lab sessions!

Scene-graph libraries (cont.)

- Models often hierarchical
  - Animals
  - Planet systems
  - Vehicles
- Each node has
  - Orientation relative to parent
  - Geometry
  - Material
  - Color
  - Etc...

Scene-graph design issues (cont.)

- Stored files could represent:
  - Simple geometry
    - Wavefront (.obj)
  - Whole scene
    - VRML
    - 3D studio
  - Standard or native file format?
  - Which programming language to use?
  - Function and data-type naming conventions

Scene-graph pros & cons

+ Provides better abstraction
  - Enables more rapid application development
- Less flexible and lower performance
  - Slower propagation of new hardware inventions

Other scene-graph libraries

- World Toolkit
  - Developed by Sense8
  - Different file formats: Wavefront, VRML, etc...
- Java3D
  - Java
  - Developed by Sun Microsystems
- Dive
  - Developed by SICS
  - Scene graph distributed transparency over network
Virtual Reality Toolkit (VRT)
- A scene-graph graphics library
- Developed by Stefan Seipel in 1997
- UU-local toolkit, but similar to other scene graph libs
- Used for 3D modeling and simulation
- Callback-based
- Still being developed
- Mainly used at the department

Virtual Reality Toolkit (cont.)
- Built as an interface on top of OpenGL
- VRT allows use of native OpenGL functions
- VRT uses OpenGL’s hardware assisted graphics acceleration
- Application programmers interface (API) in C
- Not object oriented
- Works with Windows, Linux, Solaris, Mac

Time to get to know VRT !!!
A quick walkthrough of:
- 8 data structures
- 70 functions
- implementation of a simple “robot”

VRT Basic structures
- **VRT_Context**
  - Holds information about the current VRT “universe”.
- **VRT_Note**
  - A placeholder in the scene graph that contains a transformation matrix relative to its parent and can also hold a geometry and links to a number of children nodes.
- **VRT_Geometry**
  - A set of polygons that build up an object in the scene. Must be linked to a node to be visible. Can also have color and other attributes.
- **VRT_Polygon**
  - A set of vertices and a normal vector, that build up a surface in a geometry. Can also have color on polygon level.
- **VRT_Vertex**
  - A point in the local coordinate system of a geometry. Can also have a normal vector on vertex level (for smooth shading) and texture coordinates.
- **VRT_Htx**
  - Handle to a texture object.
- **VRT_Camera**
  - Representing the observers view of the scene. Contains position, direction and field of view (fov). There can be several cameras in VRT, but only one can be active.
- **VRT_Light**
  - Represents a light source in the scene. Contains position, direction, colors, intensity and other attributes.

VRT General functions
- **VRT initialization / close functions:**
  - void VRT_Init (int argc, char** argv)
  - void VRT_Close (int displayMode)
  - void VRT_SetDisplay (int displayMode)
  - void VRT_SimulateLoop (VRT_HookPtr hook)
- **Start / exit simulation loop:**
  - void VRT_SimulateLoop (int displayMode)
  - void VRT_ExitSimulationLoop ()

VRT Basic structures (cont.)
- **VRT_Context**
  - Holds information about the current VRT “universe”.
- **VRT_Note**
  - A placeholder in the scene graph that contains a transformation matrix relative to its parent and can also hold a geometry and links to a number of children nodes.
- **VRT_Geometry**
  - A set of polygons that build up an object in the scene. Must be linked to a node to be visible. Can also have color and other attributes.
- **VRT_Polygon**
  - A set of vertices and a normal vector, that build up a surface in a geometry. Can also have color on polygon level.

VRT General functions (cont.)

- Some other useful functions:
  
  void VRT_PrintSceneGraph()
  float VRT_GetFrameRate()
  float VRT_GetSimulationTime()
  VRT_Node *VRT_CreateWorldReferenceFrame();

VRT Node functions

- Create / delete node:
  
  VRT_Node * VRT_NodeNew (VRT_Node *parent, char *name)
  int VRT_NodeDelete (VRT_Node *anode)

- Show / hide node:
  
  int VRT_NodeSwitchOff (VRT_Node *anode)
  int VRT_NodeSwitchOn (VRT_Node *anode)

- Connect geometry to node:
  
  int VRT_NodeSetGeometry (VRT_Node *anode, VRT_Geometry *geom)

VRT Node functions (cont.)

- Transform a node:
  
  • relative to the parent node
    int VRT_NodeTranslate (VRT_Node *anode, float tx, float ty, float tz)
    int VRT_NodeRotate (VRT_Node *anode, float rx, float ry, float rz)
    int VRT_NodeScale (VRT_Node *anode, float sx, float sy, float sz)

  • absolute transformation
    int VRT_NodeSetTranslation (VRT_Node *anode, float tx, float ty, float tz)
    int VRT_NodeSetRotation (VRTX_Node *anode, float rx, float ry, float rz)
    int VRT_NodeSetScale (VRTX_Node *anode, float sx, float sy, float sz)

VRT Geometry functions

- Create / delete geometry:
  
  VRT_Geometry * VRT_GeometryNew ()
  int VRT_GeometryDelete (VRTX_Geometry *geom)

- Add vertex / polygons to geometry:
  
  int VRT_GeometryAddVertex (VRTX_Geometry *geom, float x, float y, float z, float nx, float ny, float nz, float u, float v)
  VRT_HPlg VRT_GeometryNewTriangle (VRTX_Geometry *geom, int v1, int v2, int v3)
  VRT_HPlg VRT_GeometryNewQuad (VRTX_Geometry *geom, int v1, int v2, int v3, int v4)
  VRT_HPlg VRT_GeometryNewTriangleStrip (VRTX_Geometry *geom, int v1, int v2, int v3)

- Set geometry attributes:
  
  int VRT_GeometrySetColor (VRTX_Geometry *geom, int r, int g, int b, int alpha)
  int VRT_GeometrySetMaterial (VRTX_Geometry *geom, float *material)
  int VRT_GeometrySetTexture (VRTX_Geometry *geom, VRT_Htx texture)
  int VRT_GeometrySetShadingModel (VRTX_Geometry *geom, int shadingmodel)
  int VRT_GeometryFlipNormals (VRTX_Geometry *geom)
  int VRT_GeometryCalcVertexNormals (VRTX_Geometry *geom)

VRT Geometry functions (cont.)

- Some predefined VRT_Geometry objects:
  
  VRT_Geometry *VRT_Pyramid ()
  VRT_Geometry *VRT_Cube (float size)
  VRT_Geometry *VRT_Box (float dx, float dy, float dz)
  VRT_Geometry *VRT_Sphere (float radius, int resolution)
  VRT_Geometry *VRT_Cone (float base, float height, int resolution)
  VRT_Geometry *VRT_Plate (float size, char *text)
  VRT_Geometry *VRT_ModulatedPipe (int resolution, float *controls, float *radius)
VRT Geometry functions (cont.)

- Load geometry from object file:
  - VRT_Geometry *VRT_LoadSLG (char *file_name)
  - VRT_Geometry *VRT_LoadOBJ (char *file_name)

VRT Polygon functions

- Set attributes for individual polygons:
  - int VRT_PolygonSetShadingModel (VRT_HPlg poly, int rendermode)
  - int VRT_PolygonGetShadingModel (VRT_HPlg poly, int *rendermode)
  - int VRT_PolygonSetColor (VRT_HPlg poly, int r, int g, int b, int a)
  - int VRT_PolygonGetColor (VRT_HPlg poly, int *r, int *g, int *b, int *a)
  - int VRT_PolygonSetTexture (VRT_HPlg poly, VRT_Htx texture)
  - VRT_Htx VRT_PolygonGetTexture (VRT_HPlg poly)

VRT Camera functions

- Handle the default camera:
  - void VRT_ResetDefaultCamera ()
  - void VRT_SetDefaultCamera (float ex, float ey, float ez, float cx, float cy, float cz, float ux, float uy, float uz)
  - void VRT_GetDefaultCamera (float *ex, float *ey, float *ez, float *cx, float *cy, float *cz, float *ux, float *uy, float *uz)
  - int VRT_AttachCameraToNode (VRT_Node *node)
  - void VRT_SetCameraFieldOfView (float fov)

- There are also functions for handling general, user defined cameras.

VRT Light functions

- Create / delete lights:
  - VRT_Light *VRT_LightNew (int light_type)
  - int VRT_LightDelete (VRT_Light *light)

- Switch lights on / off:
  - int VRT_LightSwitchOn (VRT_Light *light)
  - int VRT_LightSwitchOff (VRT_Light *light)

- Set light attributes:
  - int VRT_LightSetColors (VRT_Light *light, float *colors)
  - int VRT_LightSetAttenuation (VRT_Light *light, float *attenuation)
  - int VRT_LightSetType (VRT_Light *light, int light_type)

- Light positioning functions:
  - int VRT_LightSetPosition (VRT_Light *light, float x, float y, float z)
  - int VRT_LightSetDirection (VRT_Light *light, float dx, float dy, float dz)

VRT Light functions (cont.)

- Replace texture:
  - int VRT_ReplaceRGBATexture (VRT_Htx *texture, int sx, int sy, char *texels)

- Set texture attributes:
  - void VRT_SetTextureModulationMode (int texture_modulation)

VRT Texture functions

- Create / load / delete texture:
  - VRT_Htx VRT_LoadTexture (char *filename)
  - VRT_Htx VRT_CreateRGBATexture (int sx, int sy, char *texels)
  - int VRT_DeleteTexture (VRT_Htx *texture)

- Replace texture:
  - int VRT_ReplaceRGBATexture (VRT_Htx *texture, int sx, int sy, char *texels)

- Set texture attributes:
  - void VRT_SetTextureModulationMode (int texture_modulation)
Hierarchical modeling

Example: (extremely) simple robot

- Simply 3 objects connected to each other hierarchically

```
Nodes connected hierarchically in scene-graph

Geometry (shared by all nodes)
```

```
robot = VRT_NodeNew(VRT_RootNode(), "Robot");
arm1 = VRT_NodeNew(robot, "Lower arm");
arm2 = VRT_NodeNew(arm1, "Upper arm");
gem = VRT_Box(0.05f, 0.3f, 0.05f);
VRT_GeometrySetColor(gem, 255, 0, 255, 255);
VRT_GeometrySetShadingModel(gem, VRT_SM_SHADE);
VRT_NoderGeometrySetGeometry(robot, gem);
VRT_NoderGeometrySetGeometry(arm1, gem);
VRT_NoderGeometrySetGeometry(arm2, gem);
VRT_NoderScale(arm1, 0.8f, 1.0f, 0.8f);
VRT_NoderScale(arm2, 0.8f, 1.0f, 0.8f);
VRT_NoderTranslate(arm1, 0.0f, 0.28f, 0.0f);
VRT_NoderTranslate(arm2, 0.0f, 0.28f, 0.0f);
```

Creating the Model Hierarchy

```
Robot
  \- Arm1
    \- Arm2
```

```
int main(int argc, char *argv[])
{
    VRT_Init(&argc, argv); /* initialize VRT Toolkit */
    VRT_SetDisplay(0); /* choose a display type, usually default */
    build_scene(); /* build up your own scenery */
    /* install a user-define callback function */
    VRT_SetCallback(VRT_HookPtrSimulationLoop);
    /* decide for background color */
    VRT_SetClearColor(0.7f,0.7f,8.0f,0.5f);
    /* install and set a virtual camera */
    VRT_SetDefaultCamera(6.0, 6.0, 6.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
    VRT_SetCameraFieldOfView(15.0f);
    /* start the simulation loop */
    VRT_SimulationLoop();
    /* shut down simulation server */
    VRT_Close();
    return 0;
}
```

```
static VRT_HookPtr SimulationLoop(VRT_CB_MSG *msg) /* Create the main (infinite) loop */
{
    switch (msg->id) /* Handle keyboard events */
    {
        case VRT_MSG_KEYBOARD_DOWN:
        {
            // Switch camera settings
            if (msg->arg1 == 'x')
            {   VRT_SetDefaultCamera(6.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);   break;
            }
            if (msg->arg1 == 'y')
            {   VRT_SetDefaultCamera(0.0, 6.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -1.0);  break;
            }
            if (msg->arg1 == 'z')
            {   VRT_SetDefaultCamera(0.0, 0.0, 6.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);   break;
            }
        } break;
    }
    default: break;
    return 0;
}
```