Introduction to VRT

Interactive Graphical Systems

HT2004

Mikael Erlandsson
mikael.erlandsson@hci.uu.se

Libraries for Computer Graphics

- Application
- CG-Library
  - What functionality should be provided?
  - Which data structures should be used?
- OpenGL
- Hardware

Libraries for computer graphics

Example: OpenGL, DirectX
- Functions for drawing graphics, e.g. triangles and lines, immediate mode.

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
- Why is a graph or tree useful?

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

- Cameras, lights and animation do not cleanly fit in

(OpenGL & VRT)
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 transparently 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_Node**
  - 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 Basic structures (cont.)

- **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 observer's 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, color, intensity and other attributes.

### VRT General functions

- **VRT initialization / close functions:**
  - `void VRT_Init (argc, argv)`
  - `void VRT_Close ()`
  - `int VRT_SetDisplay (int displayMode)`
  - `void VRT_SetCallback (VRT_HookPtr hook)`

- **Start / exit simulation loop:**
  - `void VRT_SimulationLoop ()`
  - `void VRT_ExitSimulationLoop ()`

### 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 (VRT_Node *anode, float rx, float ry, float rz)`
    - `int VRT_NodeSetScale (VRT_Node *anode, float sx, float sy, float sz)`
### VRT Geometry functions

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

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

### VRT Geometry functions (cont.)

- **Set geometry attributes:**
  ```c
  int VRT_GeometrySetColor (VRT_Geometry *geom, int r, int g, int b, int alpha)
  int VRT_GeometrySetMaterial (VRT_Geometry *geom, float *material)
  int VRT_GeometrySetTexture (VRTX_Htx texture)
  int VRT_GeometrySetShadingModel (VRTX_ShadingModel)
  int VRT_GeometryFlipNormals (VRTX_Geometry *geom)
  ```

- **Load geometry from object file:**
  ```c
  VRT_Geometry * VRT_LoadPLG (char *file_name)
  VRT_Geometry * VRT_LoadOBJ (char *file_name)
  ```

### VRT Polygon functions

- **Set attributes for individual polygons:**
  ```c
  int VRT_PolygonSetShadingModel (VRTX_HPlg poly, int rendermode)
  int VRT_PolygonSetColor (VRTX_HPlg poly, int r, int g, int b, int a)
  int VRT_PolygonSetTexture (VRTX_HPlg poly, VRT_Htx texture)
  ```

### VRT Camera functions

- **Handle the default camera:**
  ```c
  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)
  ```

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

- Create / delete lights:
  \[ \text{VRT\_Light\_New} \text{(int light\_type)} \]
  \[ \text{VRT\_Light\_Delete} \text{(VRT\_Light \*light)} \]

- Switch lights on / off:
  \[ \text{VRT\_Light\_Switch\_On} \text{(VRT\_Light \*light)} \]
  \[ \text{VRT\_Light\_Switch\_Off} \text{(VRT\_Light \*light)} \]

- Set light attributes:
  \[ \text{VRT\_Light\_Set\_Colors} \text{(VRT\_Light \*light, float \*colors)} \]
  \[ \text{VRT\_Light\_Set\_Attenuation} \text{(VRT\_Light \*light, float \*attenuation)} \]
  \[ \text{VRT\_Light\_Set\_Type} \text{(VRT\_Light \*light, int light\_type)} \]

VRT Light functions (cont.)

- Light positioning functions:
  \[ \text{VRT\_Light\_Set\_Position} \text{(VRT\_Light \*light, float x, float y, float z)} \]
  \[ \text{VRT\_Light\_Set\_Direction} \text{(VRT\_Light \*light, float dx, float dy, float dz)} \]
  \[ \text{VRT\_Light\_Set\_Position\_Node} \text{(VRT\_Light \*light, VRT\_Node \*nposition)} \]
  \[ \text{VRT\_Light\_Set\_Target\_Node} \text{(VRT\_Light \*light, VRT\_Node \*ntarget)} \]

VRT Texture functions

- Create / load / delete texture:
  \[ \text{VRT\_Htx \_Load\_Texture} \text{(char \*filename)} \]
  \[ \text{VRT\_Htx \_Create\_RGB\_ATexture} \text{(int sx, int sy, char \*texels)} \]
  \[ \text{VRT\_Delete\_Texture} \text{(VRT\_Htx \*texture)} \]

- Replace texture:
  \[ \text{VRT\_Replace\_RGB\_ATexture} \text{(VRT\_Htx \*texture, int sx, int sy, char \*texels)} \]

- Set texture attributes:
  \[ \text{void \_VRT\_Set\_Texture\_Modulation\_Mode} \text{(int texture\_modulation)} \]

Creating the Model Hierarchy

```c
robot = VRT_NodeNew(VRT_RootNode(), "Robot");
arm1 = VRT_NodeNew(robot, "Lower arm");
arm2 = VRT_NodeNew(arm1, "Upper arm");
geom = VRT_Box(0.05f, 0.3f, 0.05f);
VRT_GeometrySetColor(geom, 255, 10, 255, 255);
VRT_GeometrySetShadingModel(geom, VRT_SM_SHADE);
VRT_NodeSetGeometry(robot, geom);
VRT_NodeSetGeometry(arm1, geom);
VRT_NodeSetGeometry(arm2, geom);
VRT_NodeScale(arm1, 0.8f, 1.0f, 0.8f);
VRT_NodeScale(arm2, 0.8f, 1.0f, 0.8f);
VRT_NodeTranslate(arm1, 0.0f, 0.28f, 0.0f);
VRT_NodeTranslate(arm2, 0.0f, 0.28f, 0.0f);
```

Hierarchical modeling

Example: (extremely) simple robot

- Simply 3 objects connected to each other hierarchically

```c
int main(int argc, char *argv[])
{
    VRT_Init(&argc, argv); /* initialize VRT Toolkit */
    // VRT\_Set\_Display(); /* choose a display type, usually default */
    build_scene(); /* build up your own scenery */
    /* install a user-define callback function */
    VRT\_Set\_Callback(VRT\_Hook\_Ptr\_Simulation\_Loop);
    /* decide for background color */
    VRT\_Set\_ClearColor(0.7f, 0.7f, 0.7f, 0.5f);
    /* install and set a virtual camera */
    VRT\_Set\_Default\_Camera(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0);
    VRT\_Set\_Camera\_Field\_Of\_View(15.0f);
    /* start the simulation loop */
    VRT\_Simulation\_Loop();
    /* 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;
            }
        }
        default: break;
    }
    return 0;
}