

# VR Graphics Programming for, well, you know

## Some (relatively) easy steps to virtual reality, Part 3

Tom Sgouros

Center for Computation and Visualization  
Brown University  
thomas\_sgouros@brown.edu

Spring 2018

# What is virtual reality?

Linking the world you made  
up to the world you live in.

- Linking head movements to the view matrix.
- Natural interaction modes: pointing, gestures.
- Stereo imagery.
- Immersion.

# Input

- Asynchronous input plus
- Continuously tracking equals
- LOTS of events.

# MinVR

**Flexible display structure** Configurable display node tree.

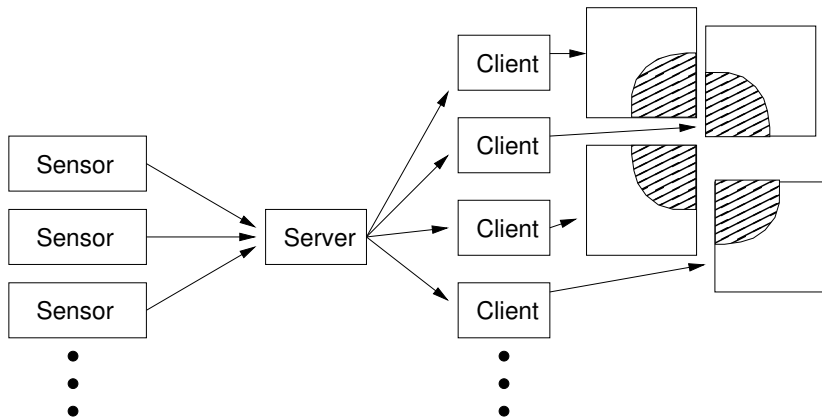
**Flexible events** You can look at them.

**Varying network geometry** Supports networked, not networked, etc.

**Extensible** Plugin architecture using dynamic libraries.

**Configurable** Most network/hardware/graphics changes can be accommodated at runtime through a configuration file.

# MinVR Server/Client Structure

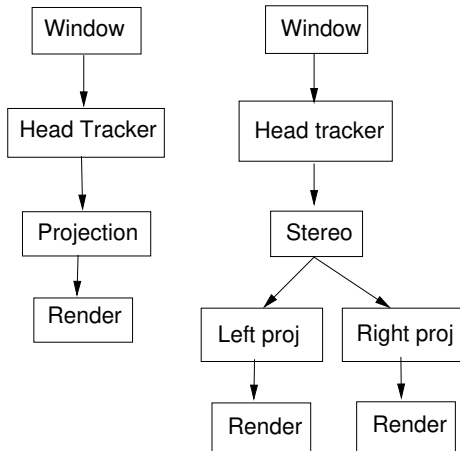


“data”  
“data”  
“data”

“Are you ready?”  
“Here’s the data”  
“Swap”

# MinVR Display Graph

- Simple desktop display on left
- Stereo YURT nodes on right
- “Render” refers to the user’s render function.



## MinVR Data Index

- Hierarchical collection of names and data.
- Supports VRFloat, VRInt, VRString, VRFloatArray, VRIntArray, VRStringArray.
- And VRContainer, which defines a “name space.”
- Use exists() and getValue().
- Has a “state” that can be pushed and popped.
- Used for events and render state. Also configuration files, using an XML parser.

# MinVR Display Graph in practice

```

<displayName:/MinVR/Desktop/RootNode>
|   Type: VRGraphicsWindowNode
|   Values Added:
|       WindowWidth
|       WindowHeight
|   Values Needed:
|       <none>
|   <displayName:/MinVR/Desktop/RootNode/HeadTrackingNode>
|   |   Type: VRHeadTrackingNode
|   |   Values Added:
|   |       HeadMatrix
|   |       CameraMatrix
|   |   Values Needed:
|   |       <none>
|   |   <displayName:/MinVR/D...RootNode/HeadTrackingNode/ProjectionNode>
|   |   |   Type: VRProjectionNode
|   |   |   Values Added:
|   |   |       ViewMatrix
|   |   |       ProjectionMatrix
|   |   |   Values Needed:
|   |   |       CameraMatrix (required)

```



## MinVR blah blah blah. How do I use it?

- Make something to implement the `VREventHandler` and `VRRenderHandler` interfaces.
- i.e. write an `onVREvent()` function, `onVRRenderContext()`, and `onVRRenderScene()`.
- Create a `MinVR::VRMain` object, use it to add the event and render handlers and initialize it.

# MinVR Can you be more specific?

## Use MVRDemo.h

```
class MVRDemo : public MinVR::VREventHandler,
    public MinVR::VRRenderHandler {

    typedef MinVR::VRDataIndex VRState;

    MVRDemo(int argc, char** argv);
    virtual void onVREvent(const MinVR::VREvent &eventData);
    virtual void onRenderContext(const VRState& stateData);
    virtual void onRenderScene(const VRState& stateData);

    void run() { while (_main->mainloop()) {} };

    int getLeftoverArgc() { return _main->getLeftoverArgc(); };
    char** getLeftoverArgv() { return _main->getLeftoverArgv(); };
};
```

# MinVR: do it like this

```
MVRDemo(int argc, char** argv) {  
  
    _main = new MinVR::VRMain();  
  
    _main->addEventHandler(this);  
    _main->addRenderHandler(this);  
    _main->initialize(argc, argv);  
}
```

# MinVR event handler

```
void onVREvent(const MinVR::VREvent &event) {  
  
    // std::cout << "Hearing event:" << event << std::endl;  
  
    if (event.getName() == "KbdEsc_Down") {  
        shutdown();  
    } else if (event.getName() == "FrameStart") {  
        _oscillator = event.getValue("ElapsedSeconds");  
    }  
}
```

# MinVR Events

## Some typical events:

Hearing event:FrameStart

- | AnalogValue = 10.598754 (float)
- | ElapsedSeconds = 10.598754 (float)
- | EventType = AnalogUpdate (string)

Hearing event:Mouse\_Move

- | EventType = CursorMove (string)
- | NormalizedPosition = 0.901563,0.693750 (floatarray)
- | Position = 577.000000,444.000000 (floatarray)

Hearing event:Head\_Move

- | EventType = TrackerMove (string)
- | Transform = 0.550934,-0.557136,-0.621348,0.000000,0.727462,0.685475,0.03

# MinVR render context

```
void onVRRenderContext(const VRState &renderState) {  
  
    // std::cout << "entering graphics context:" << renderState << s  
  
    for (int i = 0; i < _lines.size(); i++) {  
        _lines[i].startAnim(plugPos(mrand.get(10),mrand.get(10)));  
        _lines[i].step();  
    }  
  
    if ((int)renderState.getValue("InitRender") == 1) {  
        _checkContext();  
        _initializeScene();  
        _scene.prepare();  
    }  
    _scene.load();  
}
```

# MinVR render scene

```
void onVRRenderScene(const VRState &renderState) {  
  
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT);  
  
    std::vector<float> pm =  
        renderState.getValue("ProjectionMatrix");  
    glm::mat4 proj = glm::mat4( pm[0], pm[1], pm[2], pm[3],  
                                pm[4], pm[5], pm[6], pm[7],  
                                pm[8], pm[9], pm[10], pm[11],  
                                pm[12], pm[13], pm[14], pm[15]);  
  
    ...  
    _scene.draw(view, proj);  
}
```

# Getting and building MinVR

```
$ git clone -b beta https://github.com/MinVR/MinVR
$ cd MinVR
$ mkdir build
$ cd build
$ cmake .. -DWITH_DOCUMENTATION=ON
           -DWITH_PLUGIN_FREEGLUT=ON
           -DWITH_PLUGIN_Glfw=ON
           -DWITH_PLUGIN_OPENGL=ON
$ make install
```



## Getting and building demo-graphic

Attach MinVR to your demo-graphic build by defining MINVR\_ROOT to point to the MinVR INSTALL directory. If you didn't set it with cmake, this will be in MinVR/build/install.

```
$ git clone https://github.com/tsgouros/demo-graphic
$ cd demo-graphic
$ mkdir build
$ cd build
$ cmake .. -DMINVR_INSTALL_DIR=${MINVR_ROOT}
           -DBUILD_DOCUMENTATION=ON
$ make
```

All built? Try these:

```
$ bin/treeDemo
$ bin/objDemoMinVR -c ../config/desktop-freeglut.xml
```