

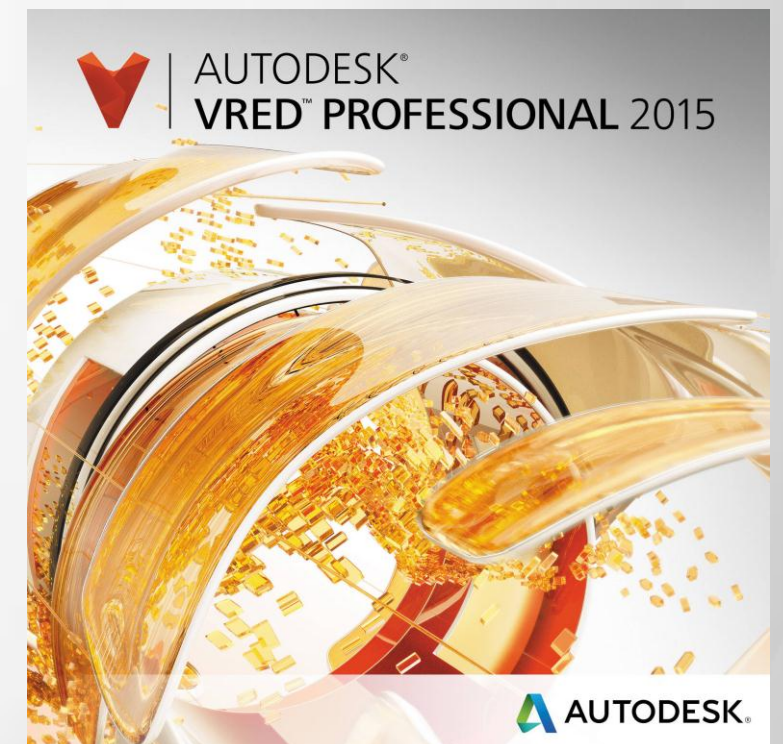
VRED Professional: An API Overview with Focus on Creating Custom User Interfaces

Bill Diack

Software architect, Autodesk Consulting Visualization Group
AT10281

Class summary

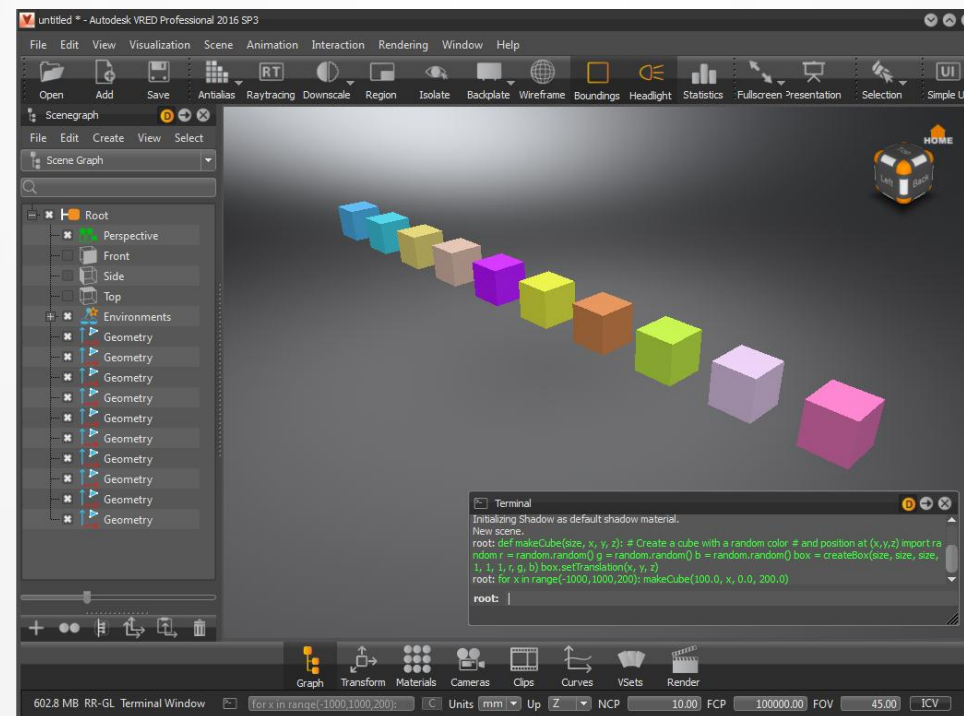
This class will provide an overview of Autodesk® VRED Professional's API with a focus on how to create **custom user interfaces**. I will provide an overview of the various versions of VRED and the API capabilities of each, followed by an **introduction to the API**, discuss techniques for accessing your API code from the application, what's new in 2016, and how to create custom user interfaces. Along the way we'll demonstrate some **practical custom examples**.



Key learning objectives

At the end of this class, you will be able to:

- Understand the various versions of VRED and the API capabilities of each
- Understand the different ways to extend VRED
- Write a simple extension to VRED using the API
- Create custom user interfaces for VRED



About the presenter

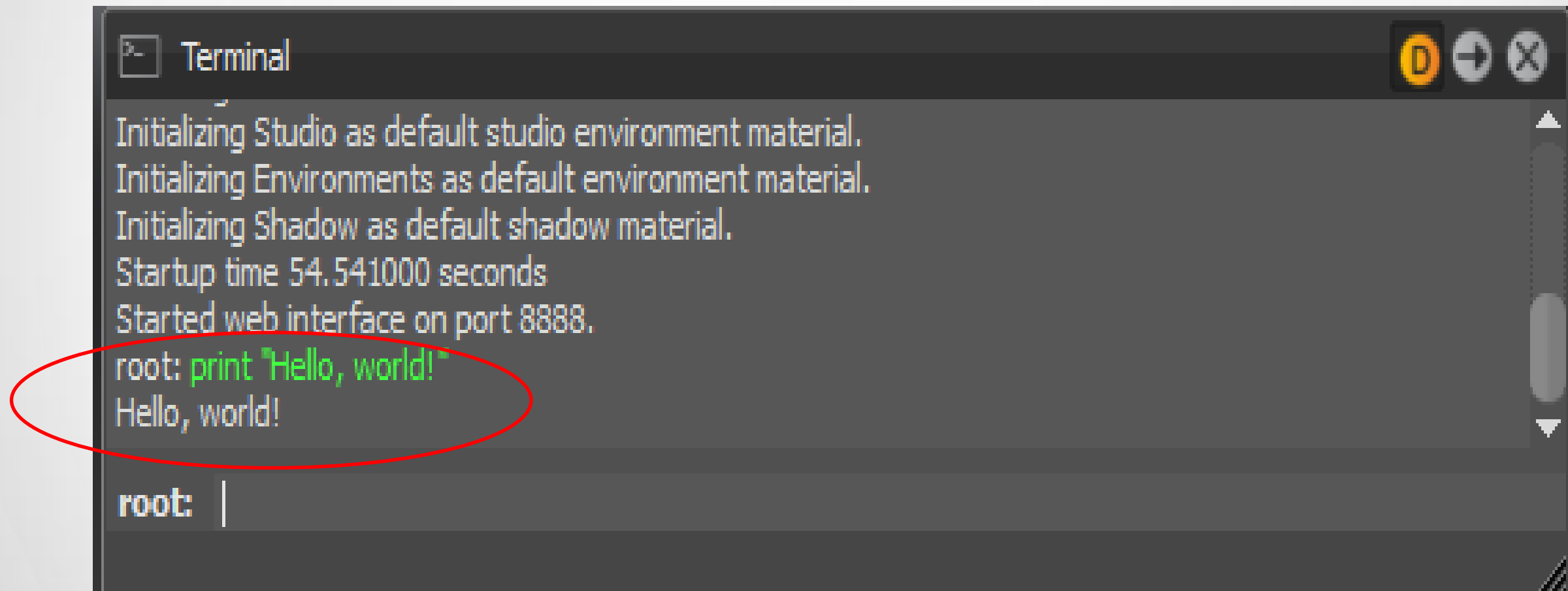
Bill Diack

- Software developer in Autodesk Consulting's Visualization Group
- Focus on VRED, Maya, custom applications
- Solutions for automotive, aerospace, manufacturing clients
- Multi-platform experience: Windows, Linux, MAC, iOS, Web
- Formerly development manager for the Maya API team

An overview of the examples I'll be showing...

An overview of the examples I'll be showing...

1) Print a simple message on the VRED terminal window



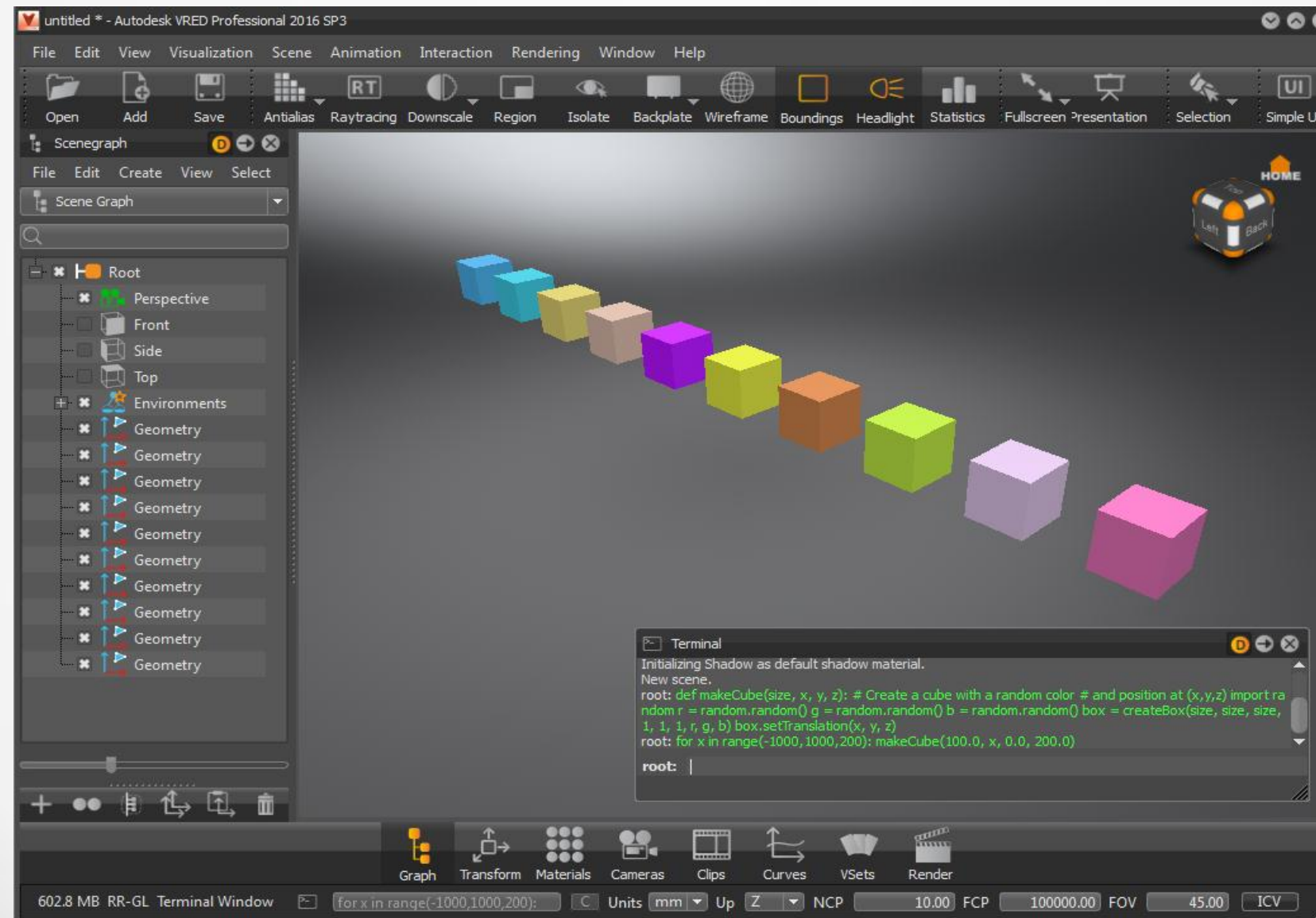
The image shows a terminal window titled "Terminal" with a dark background. The window contains the following text:

```
Initializing Studio as default studio environment material.  
Initializing Environments as default environment material.  
Initializing Shadow as default shadow material.  
Startup time 54.541000 seconds  
Started web interface on port 8888.  
root: print "Hello, world!"  
Hello, world!
```

The command `root: print "Hello, world!"` and its output `Hello, world!` are circled in red. Below the output, the prompt `root:` is visible with a cursor.

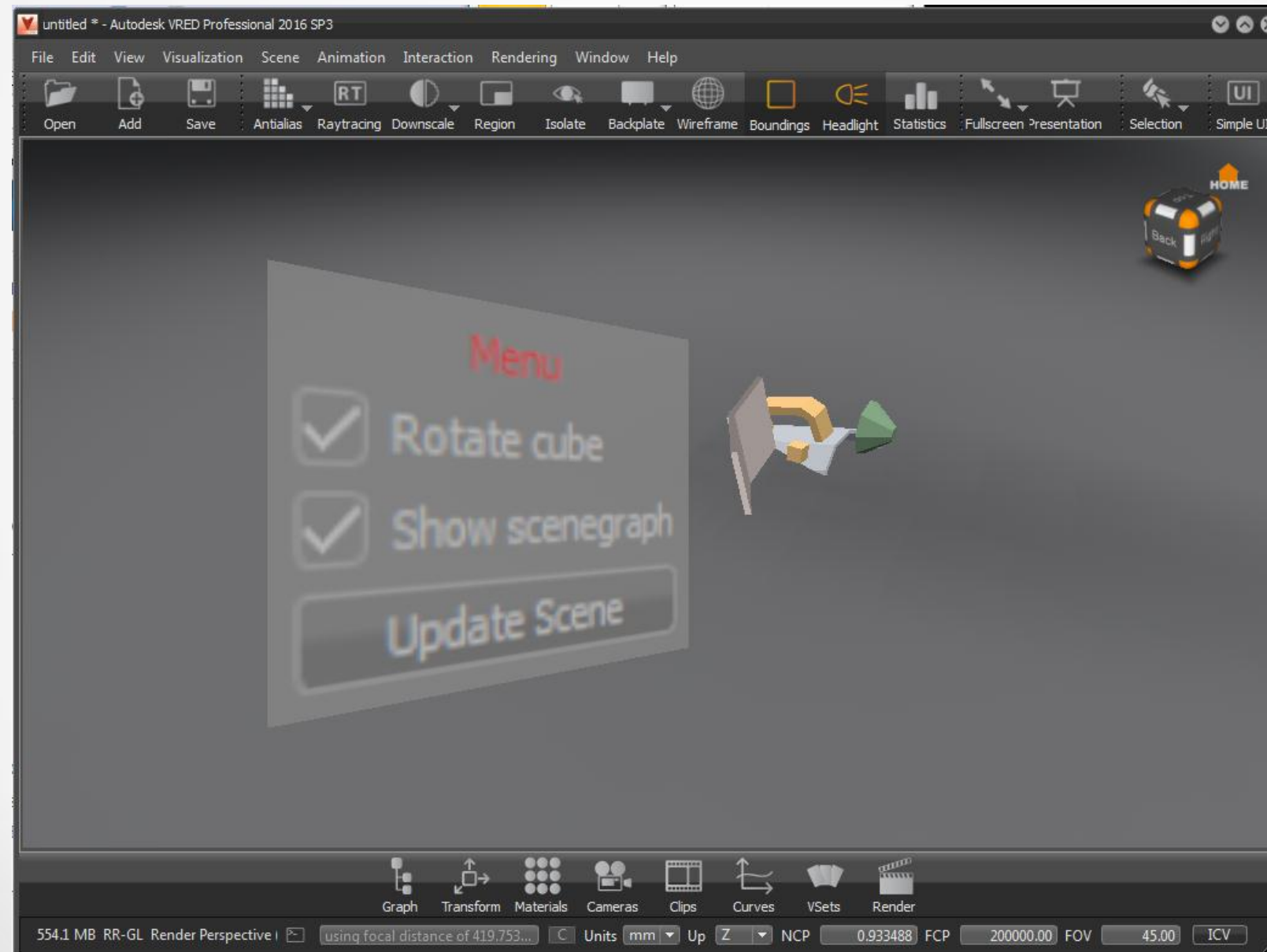
An overview of what I'll be showing in this class...

2) Create a python function that uses the API to create geometry



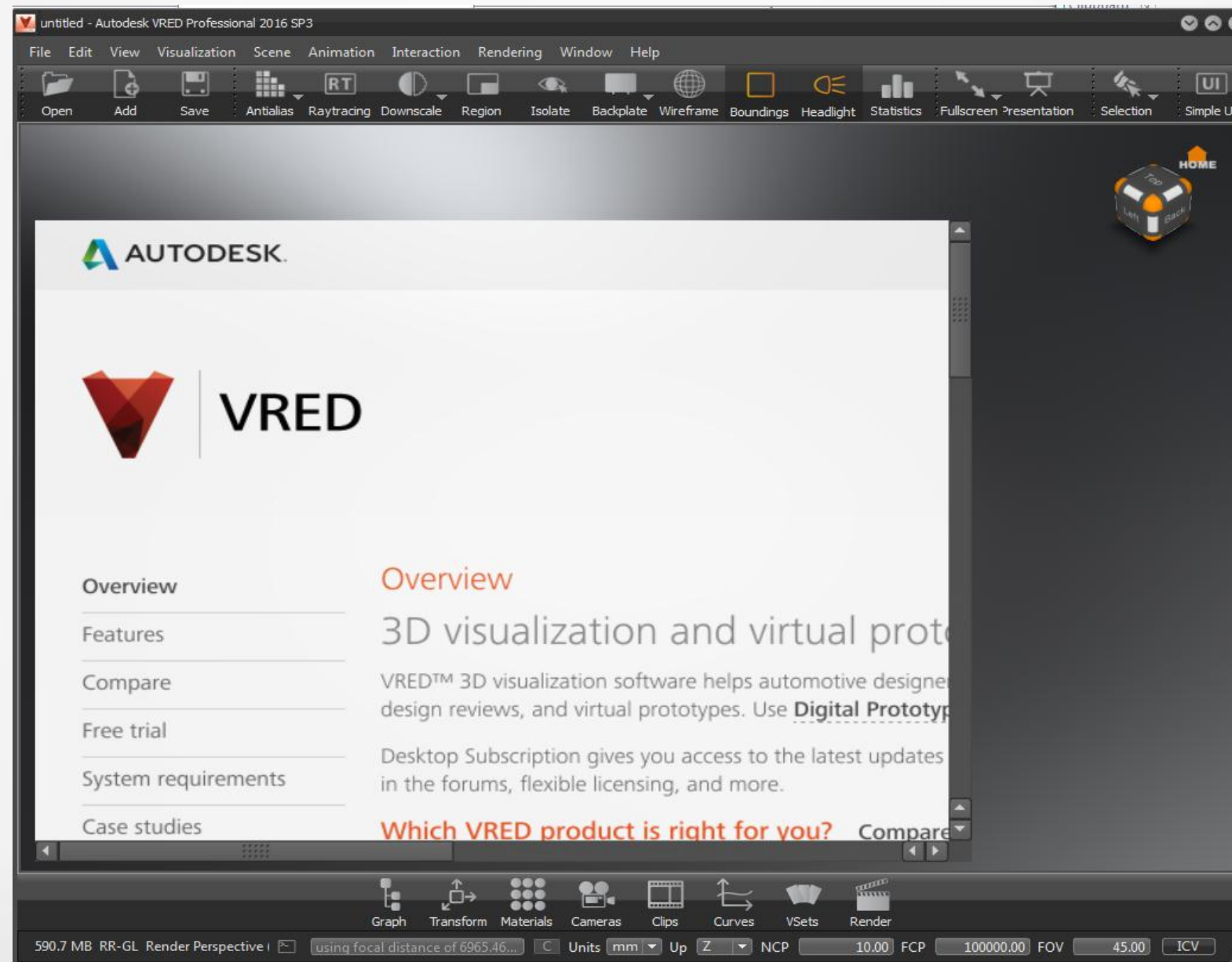
An overview of what I'll be showing in this class...

3) Creating UI within the viewport



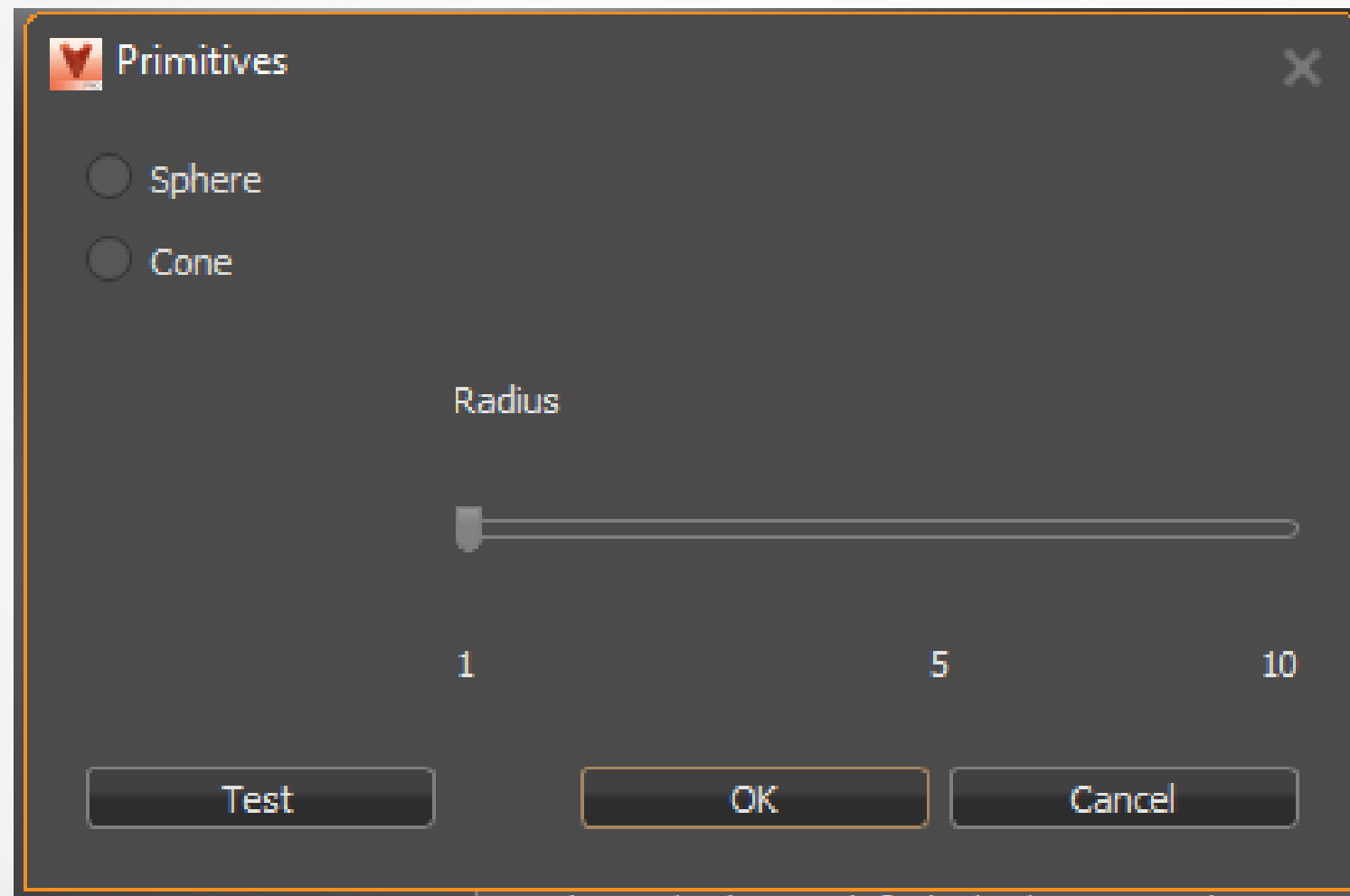
An overview of what I'll be showing in this class...

4) Placing web content within the viewport



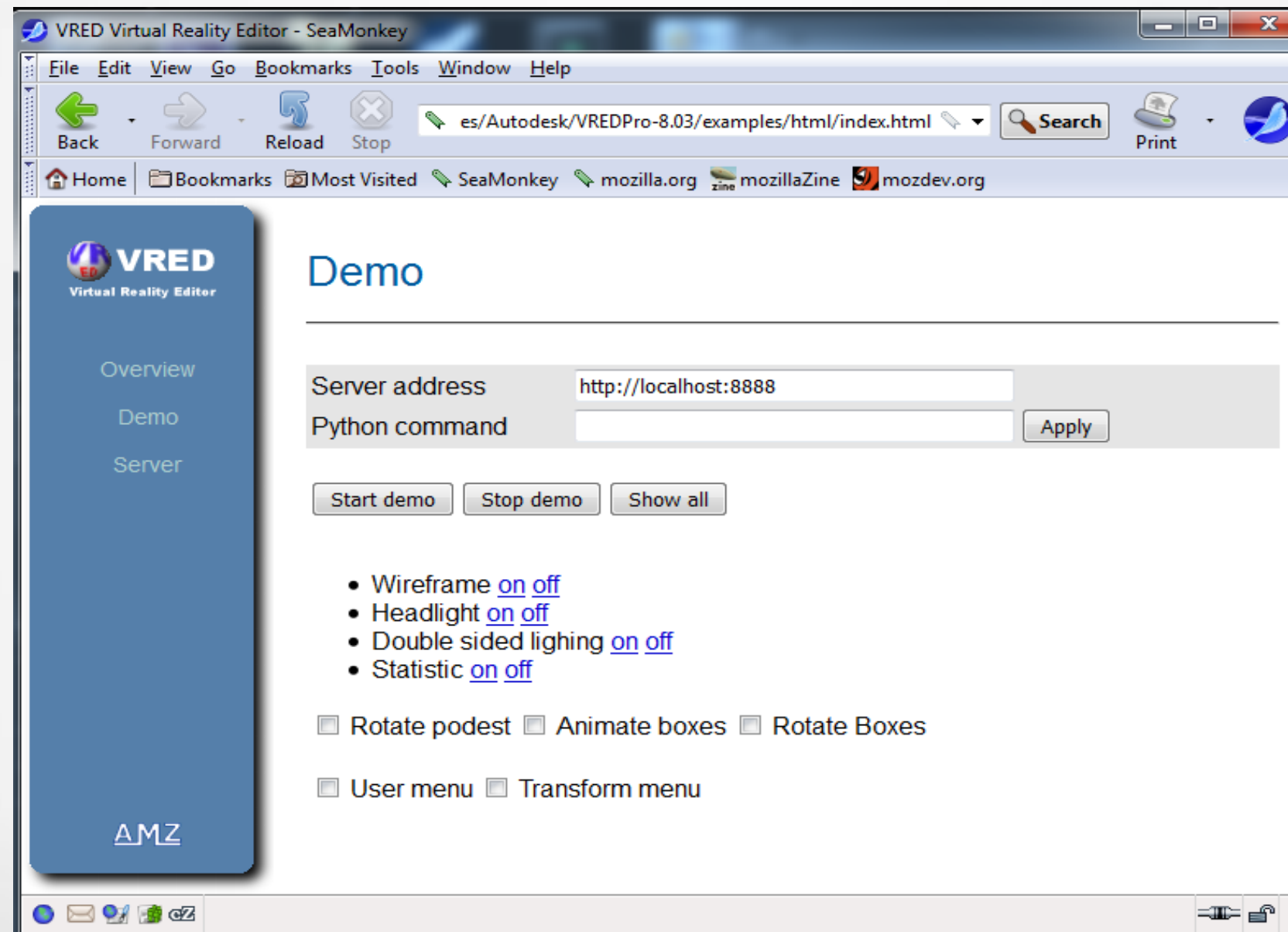
An overview of what I'll be showing in this class

5) Using the *vrWidget* class to create a window



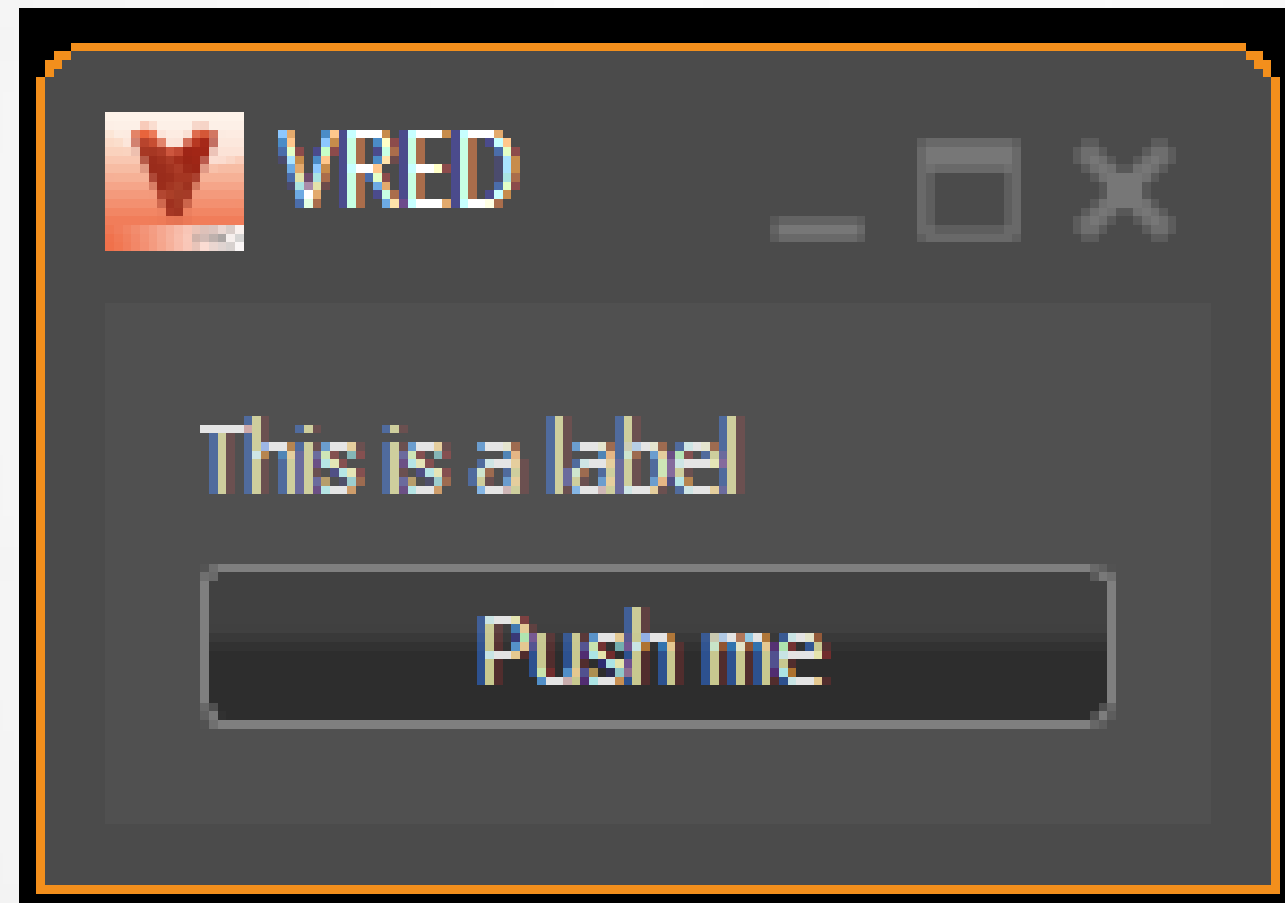
An overview of what I'll be showing in this class

6) Using a standalone application



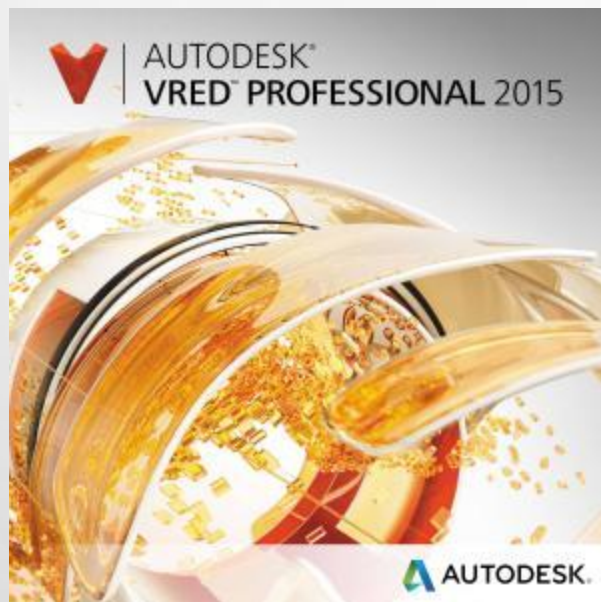
An overview of what I'll be showing in this class...

7) Using *PythonQt* to create a window



The different versions of VRED

- ***VRED Professional:*** Targeted at automotive designers. Ability to create high-end visualizations and virtual prototypes
- ***VRED Design:*** For designers to review and evaluate design ideas
- ***VRED:*** For product designers to visualize 3D models on the fly



Autodesk® VRED and API capabilities

Version	Web interface	API and command terminal
Autodesk® VRED Professional	✓	✓
Autodesk® VRED Design	Read only	Read only
Autodesk® VRED	✗	✗

About the VRED API

- Uses the industry-standard “python” language
- 854+ built-in functions
- 52+ classes
- Built-in support for many popular python libraries (“modules”)
- Easy to add additional modules, or even create your own from C++

What is python?

- Interpreted language, industry standard, open source
- Object-oriented programming (classes, inheritance, etc.)
- You “import” a module to access a common set of functionality
- You can issue commands interactively, or source a script file (.py)
- Lots of great documentation: e.g. <https://docs.python.org/2.7/> or <http://www.tutorialspoint.com/python/index.htm> or numerous books
- Here's a simple example to print the integers 0 through 10

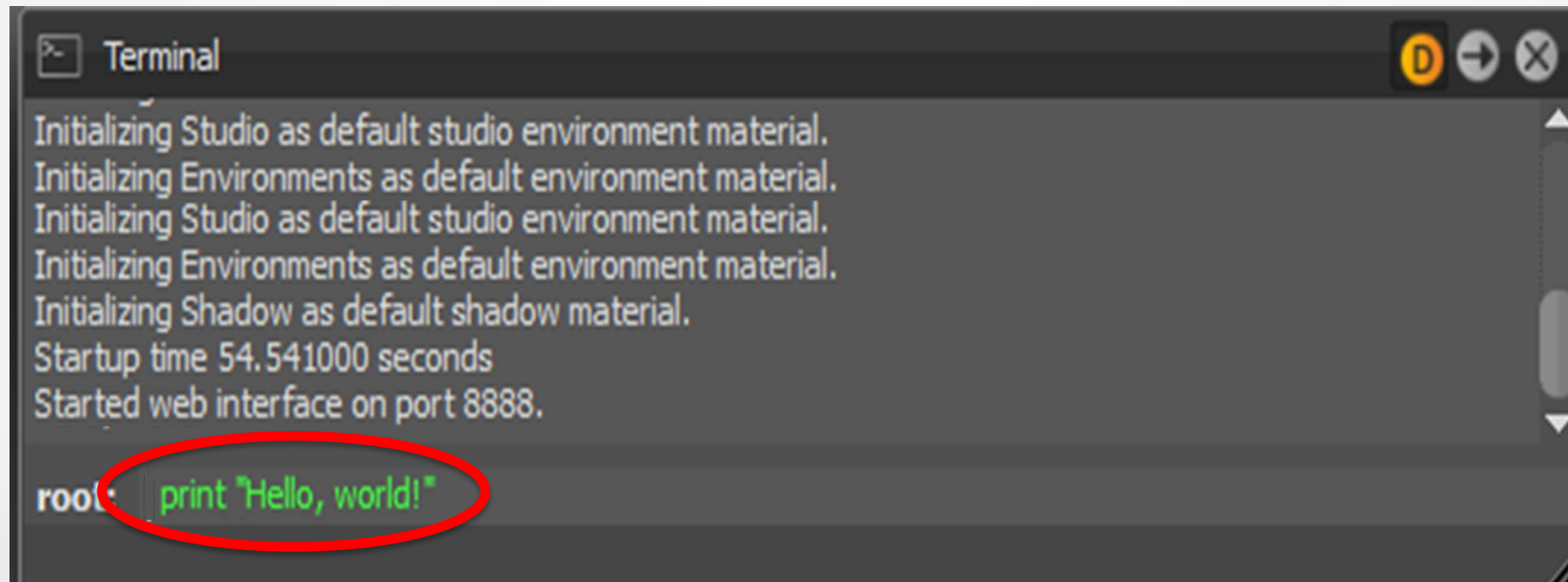
```
for count in range(10):  
    print "count=", count
```

What's new in the VRED 2016 API

- Updated python support to version 2.7.8
- Updated API examples
- Added PythonQt to allow access to the Qt UI framework(2016 SR1)
- Added interface to the Surface Analysis tool
- vrWidget now supported on Macintosh
- Custom functions now work through WebServer on Macintosh
- Added specific functions such as handling configuration files, flush transforms, script editor control, variant access

The different ways to extend VRED

- ***Terminal:*** Enter python commands

A screenshot of a terminal window titled "Terminal". The window has a dark background and standard window controls (minimize, maximize, close) in the top right corner. The terminal displays several lines of text: "Initializing Studio as default studio environment material.", "Initializing Environments as default environment material.", "Initializing Studio as default studio environment material.", "Initializing Environments as default environment material.", "Initializing Shadow as default shadow material.", "Startup time 54.541000 seconds", and "Started web interface on port 8888.". At the bottom, a prompt "root:" is followed by the command "print 'Hello, world!'" which is highlighted with a red oval.

```
Terminal
Initializing Studio as default studio environment material.
Initializing Environments as default environment material.
Initializing Studio as default studio environment material.
Initializing Environments as default environment material.
Initializing Shadow as default shadow material.
Startup time 54.541000 seconds
Started web interface on port 8888.
root: print "Hello, world!"
```

The different ways to extend VRED

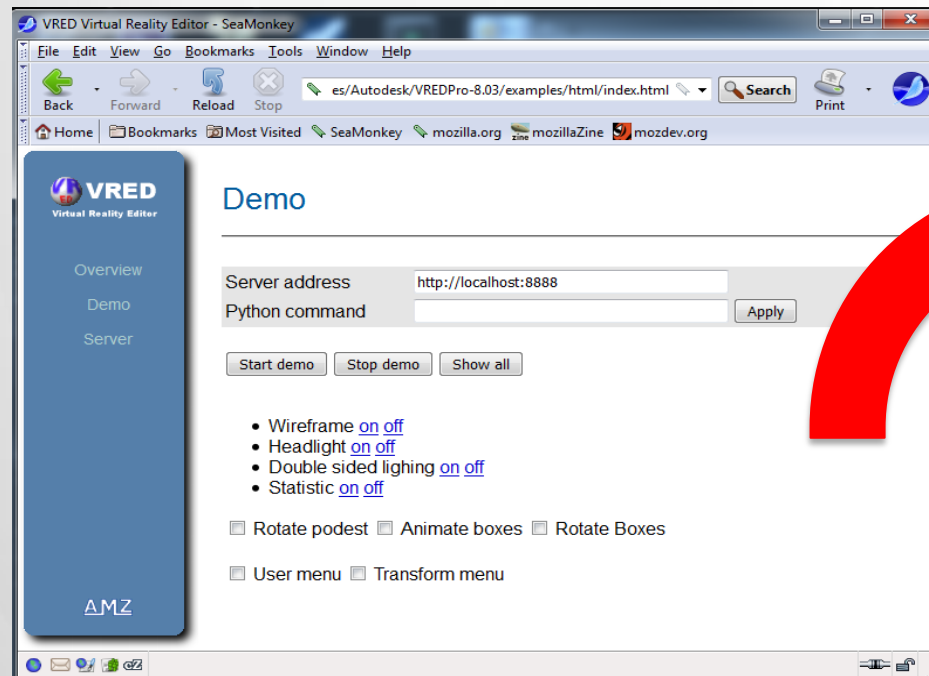
- ***Terminal:*** Enter python commands or source a python file

```
execfile( "C:/Users/billy/example5.py" )
```

The different ways to extend VRED

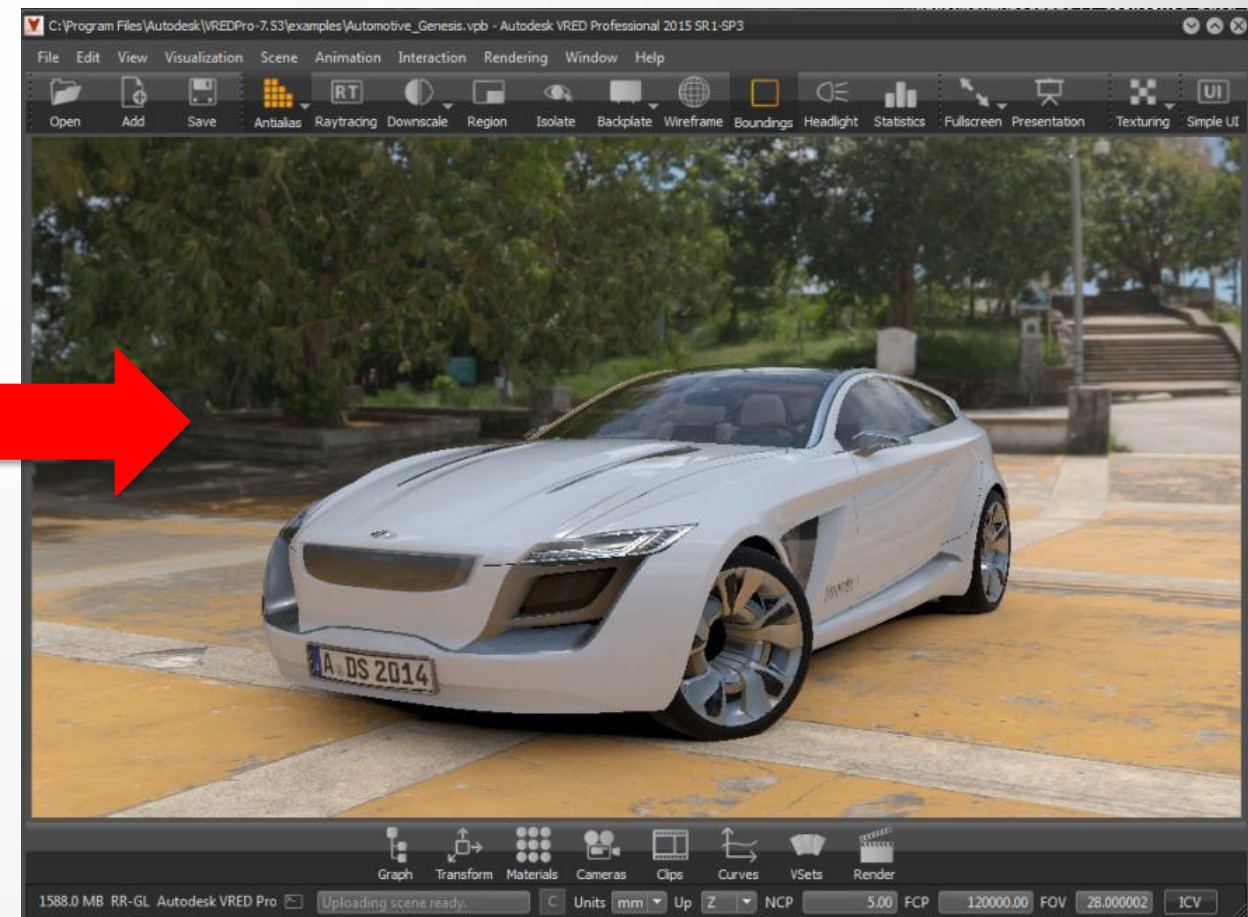
- ***Terminal:*** Enter python commands or source python files
- ***WebInterface:*** Send commands from an external application to VRED using HTTP

External application



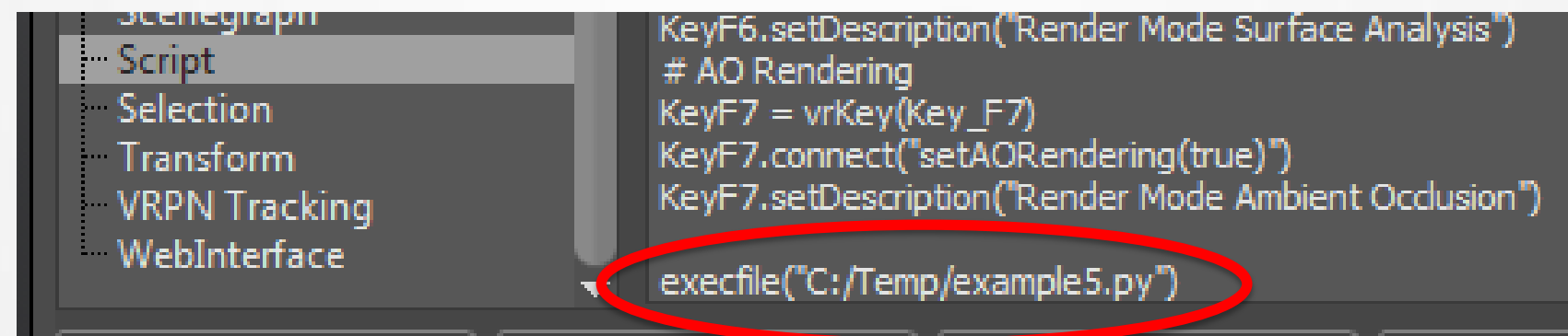
HTTP

VRED



The different ways to extend VRED

- ***Terminal:*** Enter python commands or source python files
- ***WebInterface:*** Send commands from an external application to VRED using HTTP
- ***Preferences:*** In the Script section of Preferences you can embed python commands that are invoked each time a new scene is created

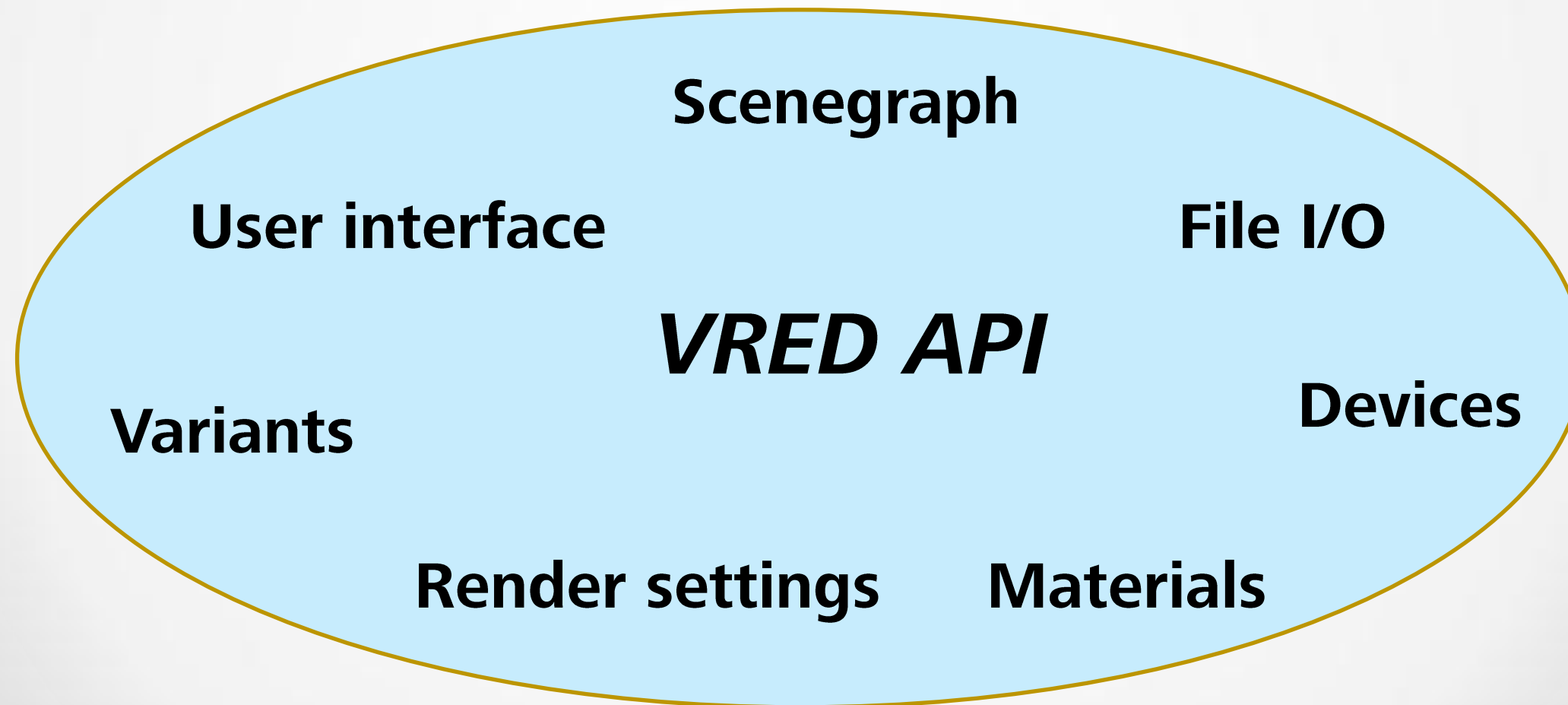


The different ways to extend VRED

- ***Terminal:*** Enter python commands or source python files
- ***WebInterface:*** Send commands from an external application to VRED using HTTP
- ***Preferences:*** In the Script section of Preferences you can embed python commands that are invoked each time a new scene is created
- ***Variant sets:*** Add python commands to the variant logic

What the API can do for you

Seven conceptual areas:



Functions and classes

- Functions perform an action and may return a value
- For example to import a file we call *loadGeometry*, e.g.
`loadGeometry("$VRED_ROOT/examples/geo/cup-test.osb")`
- Some functions return an “object” (instance of a class)
`sph = createSphere(1, 100, 1, 1, 1)`
- To get the type of an object, `print type(sph)`
- A class has methods to operate on the object. These are listed in the API docs, but you can also `print dir(sph)`

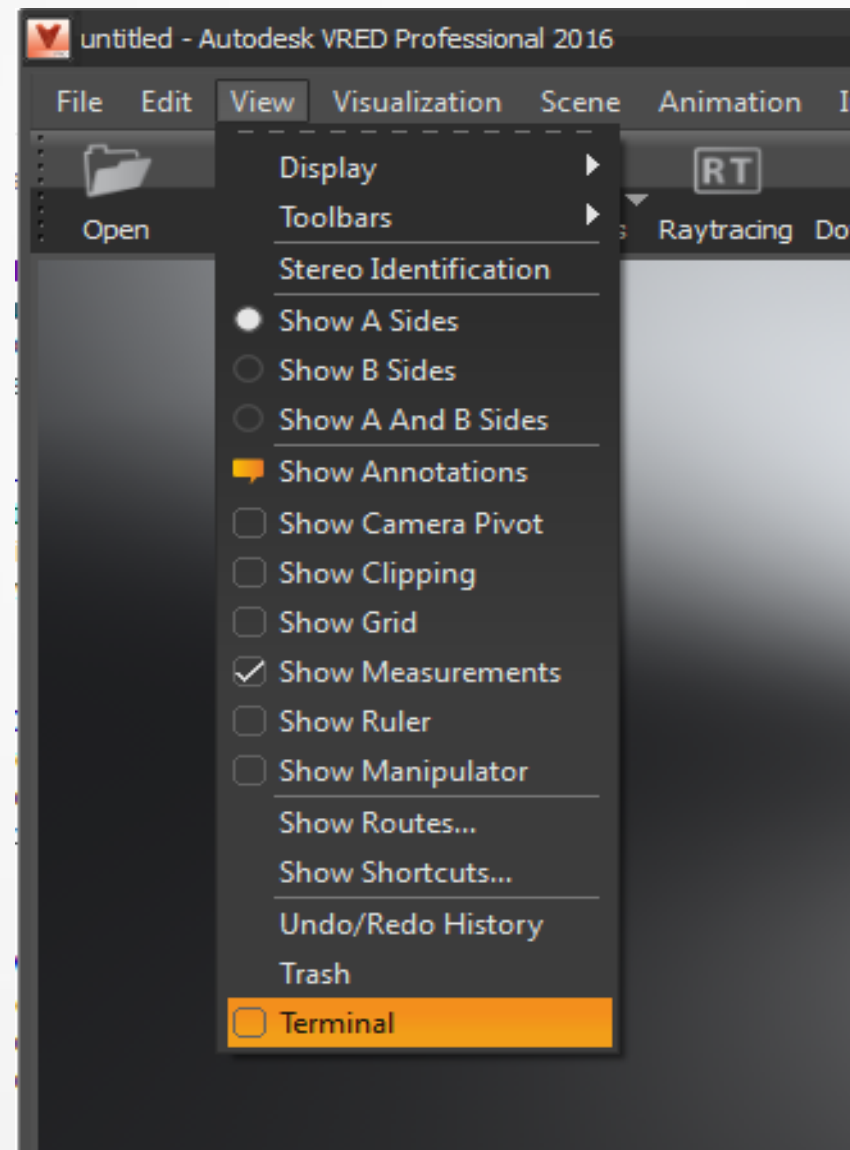
An example class: *vrNodePtr*

- For example, the *vrNodePtr* class represents a node within VRED's scenegraph
- The *vrNodePtr* class contains 94 methods
- Here are some of them:
 - `getName()` ← returns the node's name
 - `setTranslation(x,y,z)` ← positions the node
 - `copy()` ← creates a new geometry by duplicating the node
 - `setMaterial()` ← assigns a material to the node
- Example of calling a method on an object:
`name = sph.getName()`

Creating simple extensions to VRED

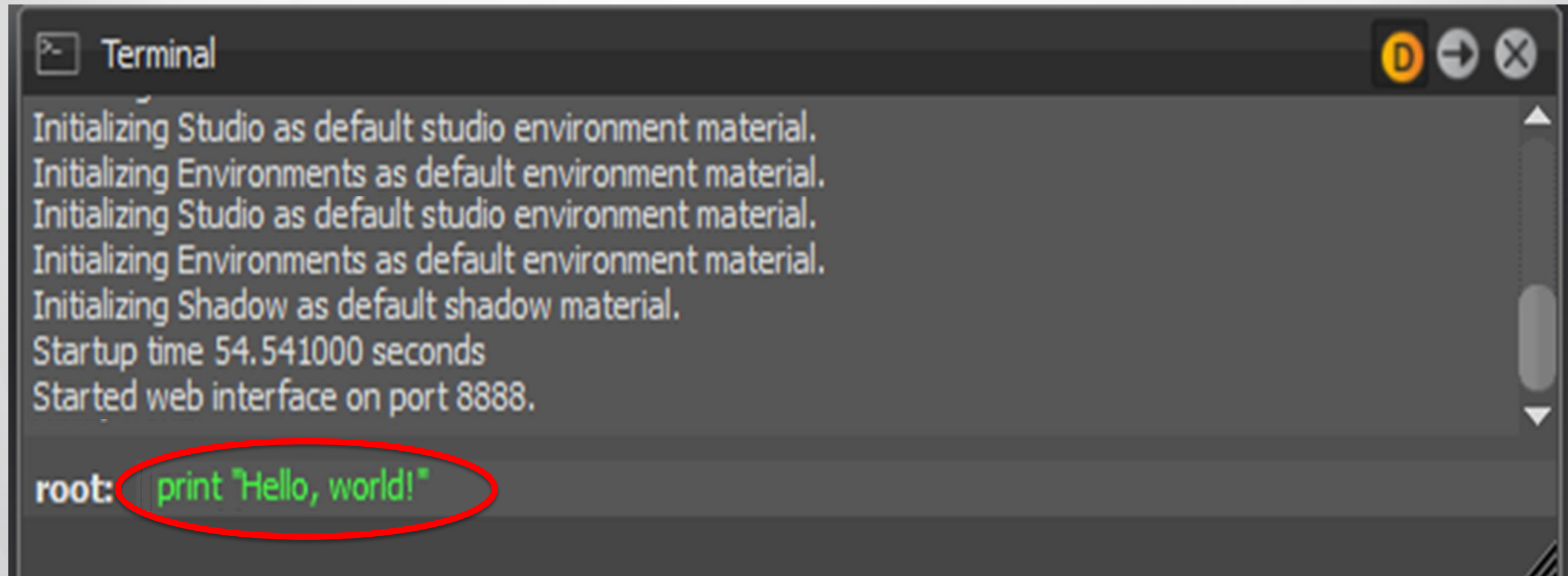
Example 1) Print a message on the Terminal Window

- Access the Terminal Window from the “View” menu on the top bar



Example 1) Print a message on the Terminal Window

- The Terminal window appears.
- Enter the command *print "Hello, world!"* at the root: prompt then press ENTER

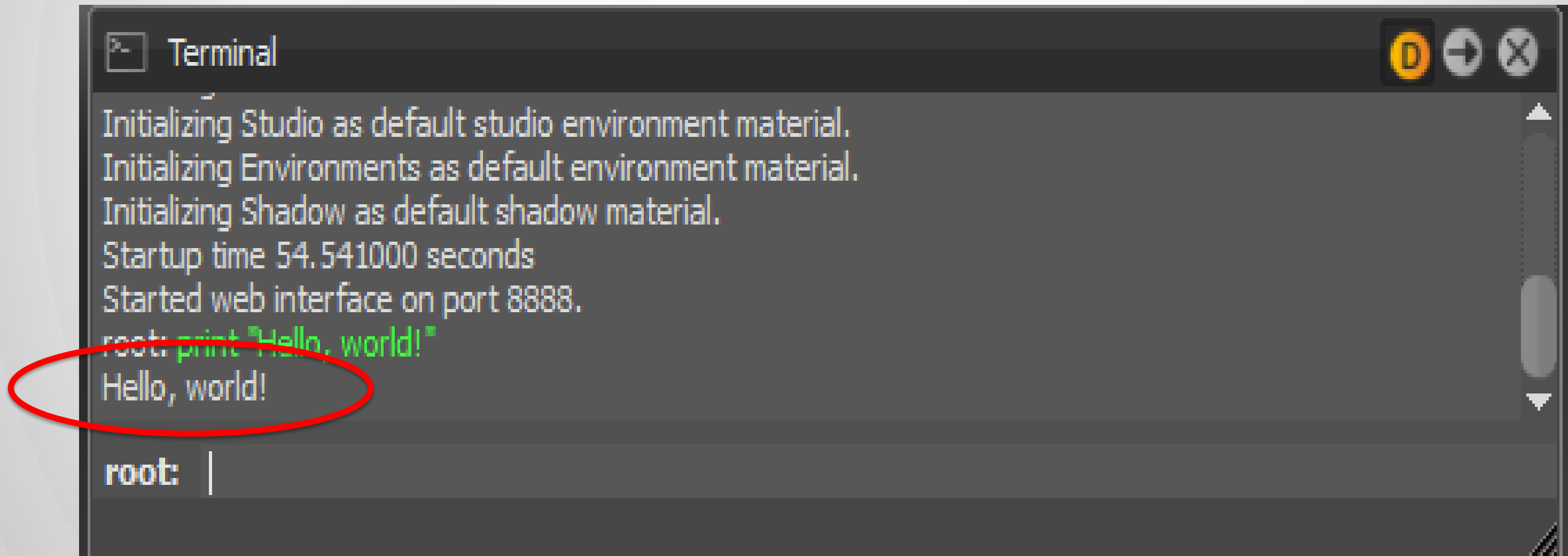


The screenshot shows a terminal window titled "Terminal" with standard window controls. The terminal output includes several initialization messages: "Initializing Studio as default studio environment material.", "Initializing Environments as default environment material.", "Initializing Studio as default studio environment material.", "Initializing Environments as default environment material.", and "Initializing Shadow as default shadow material." It also displays "Startup time 54.541000 seconds" and "Started web interface on port 8888." At the bottom, the root prompt "root:" is visible, followed by the command `print "Hello, world!"` which is highlighted with a red oval.

```
Terminal
Initializing Studio as default studio environment material.
Initializing Environments as default environment material.
Initializing Studio as default studio environment material.
Initializing Environments as default environment material.
Initializing Shadow as default shadow material.
Startup time 54.541000 seconds
Started web interface on port 8888.
root: print "Hello, world!"
```

Example 1) Print a message on the Terminal Window

- The Terminal window appears.
- Enter the command *print "Hello, world!"* at the root: prompt then press ENTER



The screenshot shows a terminal window titled "Terminal" with standard window controls. The terminal output includes initialization messages for Studio, Environments, and Shadow, followed by startup time and web interface status. The command `root: print "Hello, world!"` is entered and executed, with the output `Hello, world!` displayed below it. A red oval highlights the command and its output. The prompt `root: |` is visible at the bottom of the terminal.

```
Terminal
Initializing Studio as default studio environment material.
Initializing Environments as default environment material.
Initializing Shadow as default shadow material.
Startup time 54.541000 seconds
Started web interface on port 8888.
root: print "Hello, world!"
Hello, world!
root: |
```

Example 2) Create a python function that uses the API to create geometry

- I use the createBox command to create the cube
- This returns a vrNode class which is the “handle” to the new geometry

```
box = createBox( sx, sy, sz, divx, divy, divz, r, g, b )
```

- I then use the setTranslation method on the vrNode to position

```
box.setTranslation( tx, ty, tz )
```

- I put these python commands into a “function” which I can call multiple times
- Then I call the function from within a loop to generate 10 geometries

Example 2) Create a python function that uses the API to create geometry

- Here's the code you can paste into the Terminal Window...

```
# Function to make a cube at location (x,y,z) with random color:
```

```
def makeCube(size, x, y, z):
```

```
    import random
```

```
    r = random.random()
```

```
    g = random.random()
```

```
    b = random.random()
```

```
    box = createBox(size, size, size, 1, 1, 1, r, g, b)
```

```
    box.setTranslation(x, y, z)
```

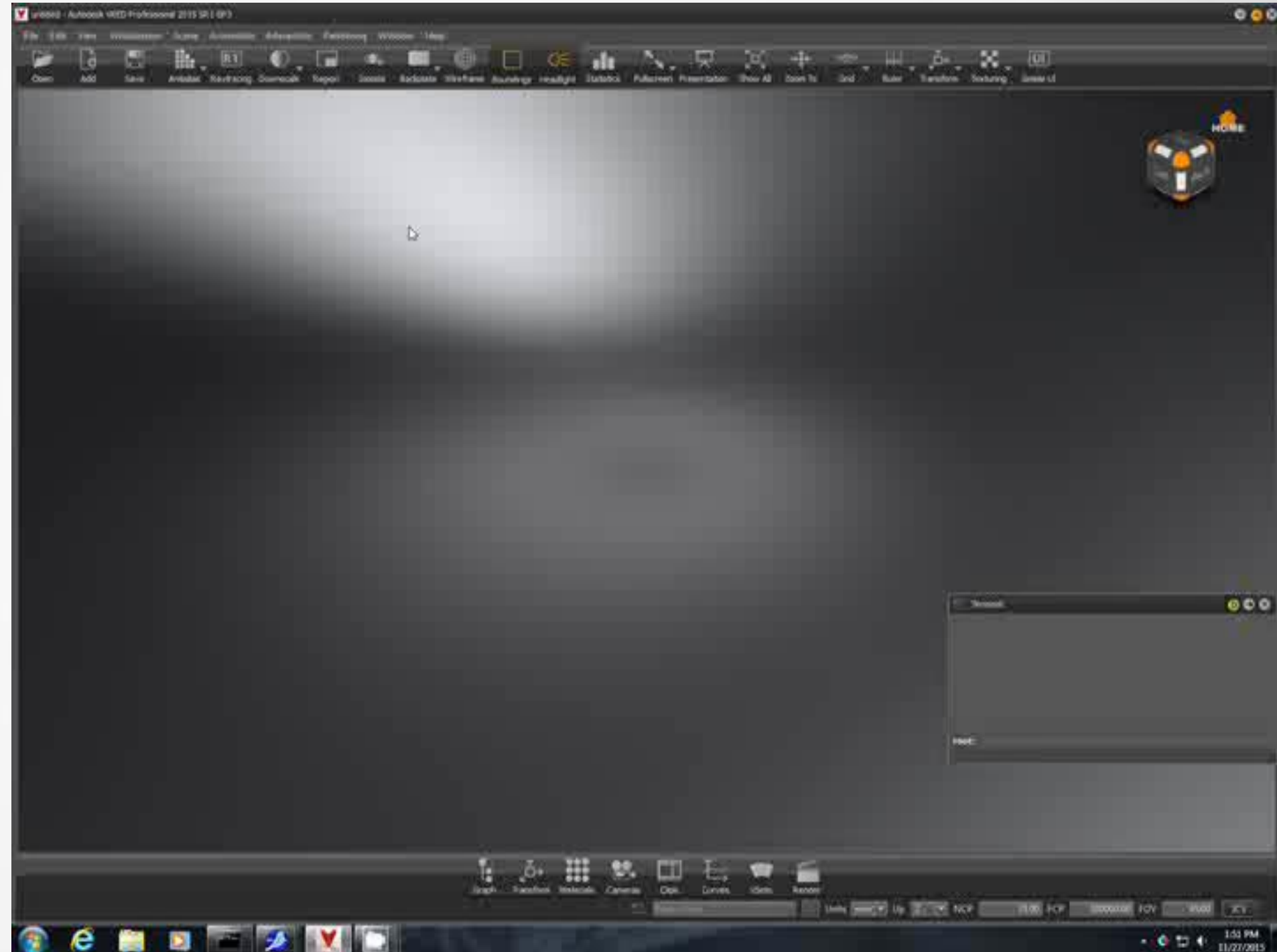
```
# Loop through the number -1000 to +1000 in steps of 200
```

```
for x in range(-1000,1000,200):
```

```
    makeCube(100.0, x, 0.0, 200.0)
```

Example 2) Create a python function that uses the API to create geometry

The result:10 cubes:



Example 3) Create a UI inside a 3D viewport

- Use the *vrMenu* class
- First, create the menu object

```
menu = vrMenu( 0.05, 1, 1 )
```
- Next, use methods on the menu object to add UI elements

```
menu.addLabel("Menu")  
menu.addPushButton("Update scene", "updateScene()")
```
- The second parameter on “clickable” elements is a python command
- Optionally transform the menu object using `setTransform()`
- Then show or hide the UI using `show()` or `hide()`

```
menu.show()
```

Example 3) Create a UI inside a 3D viewport

Complete source code is online as [example3.py](#)



Example 4) Adding web UI inside a 3D viewport

- Use the *vrMenu* class, then its *setUrl()* method
- Create the menu object

```
menu = vrMenu( 0.05, 1, 1 )
```
- Use the *setUrl* method to specify the web page

```
menu.setUrl("http://www.autodesk.com/products/vred/overview/")
```
- Now when you call *menu.show()*, the web page will appear
- You can use HTML elements for your custom UI

Example 4) Adding web UI inside a 3D viewport

Here is the example UI (source code is online as `example4.py` in the extras)



Creating UI in a separate window

- The following examples create UI in their own windows
- You can move, resize, minimize and maximize the windows independently of VRED
- I'll show three approaches:
 1. Using the *vrWidget* class
 2. A standalone application that uses the WebInterface
 3. A PyQt example

Example 5) Using the *vrWidget* class

- Call *vrWidget* to create the standalone window
- It takes either the name of a file containing the UI definition or a string

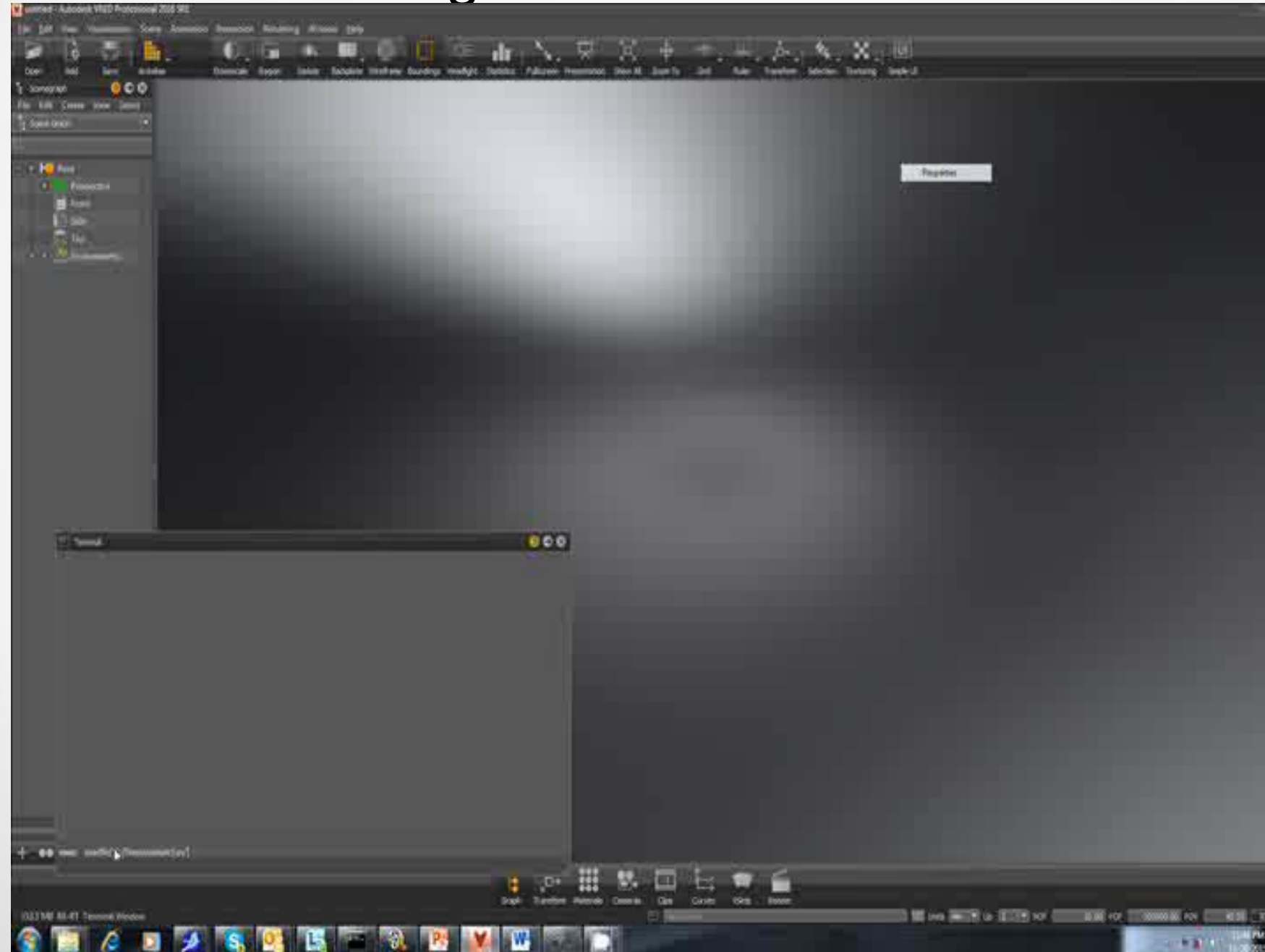
```
widget = vrWidget("gui/test.ui")
```
- The UI definition is an XML string you can create using QtDesigner. Search for “Qt Designer’s UI file format” in the online Qt documentation
- Next, create python functions to be called when buttons are clicked

```
def toggledCheckBox(s):  
    print 'Toggled checkbox to', s
```
- Finally, call *show()* to display the menu

```
widget.show()
```

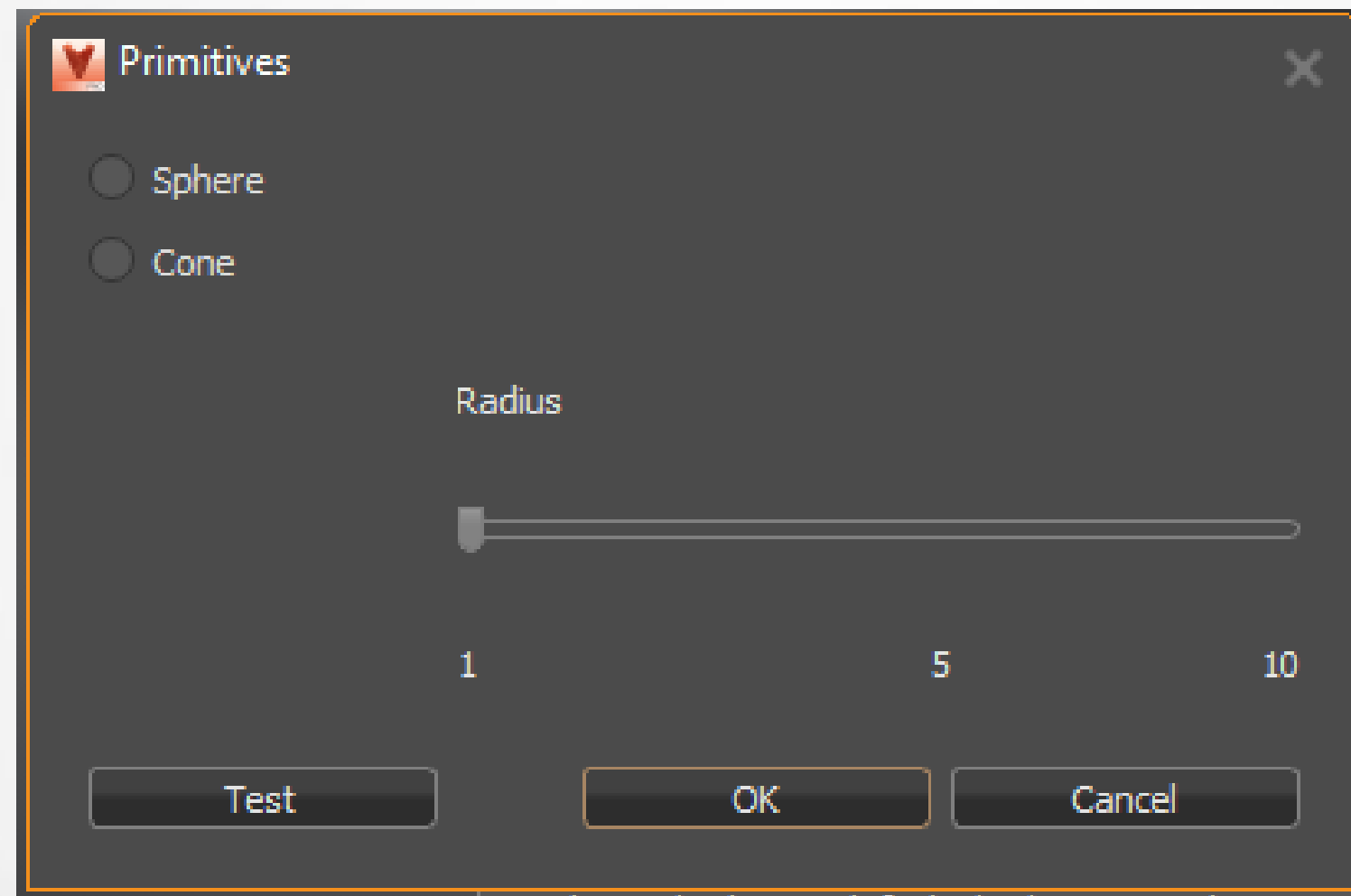
Example 5) Using the *vrWidget* class

And here is a video of the widget in action



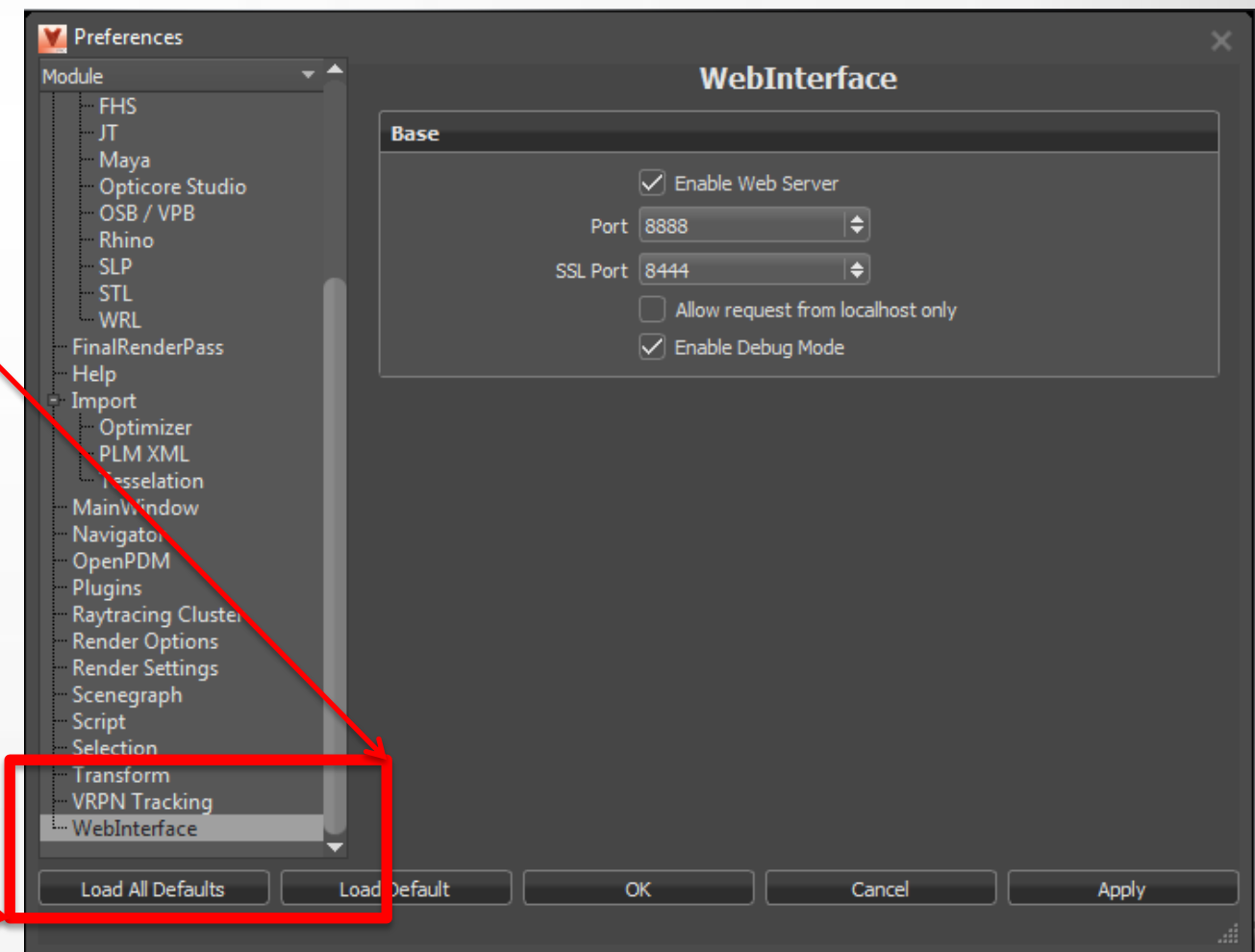
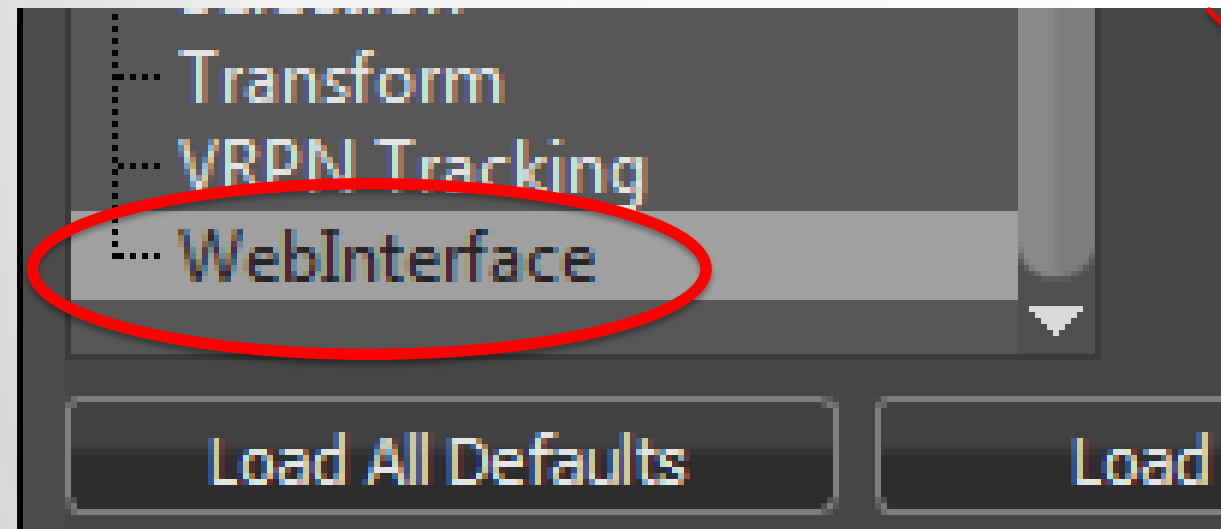
Example 5) Using the *vrWidget* class

Here is the example UI (source code is online as `example5.py` in the extras)



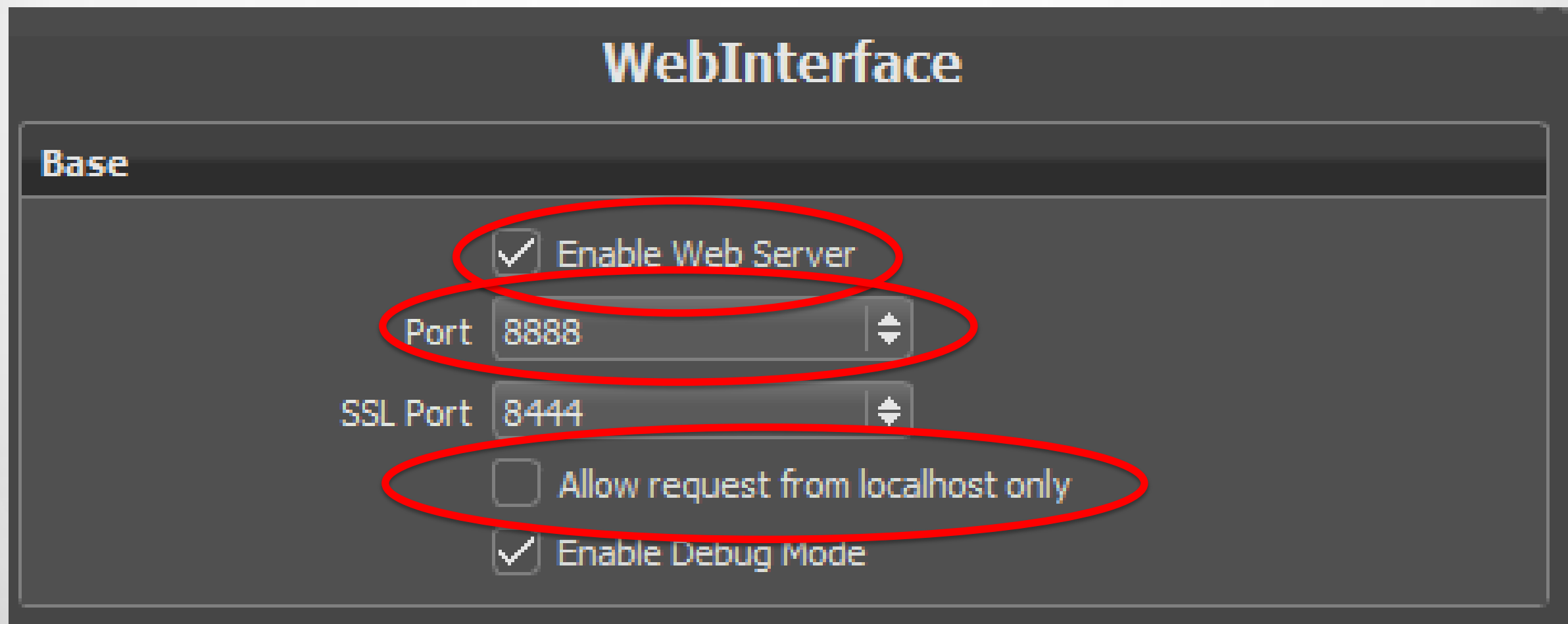
Example 6) A standalone application

- Create your UI in a separate application. Could be a web browser-based UI, a JAVA, C# or C++ app, or even a plug-in to another application such as Maya.
- In VRED, enable the WebInterface. It's under Preferences



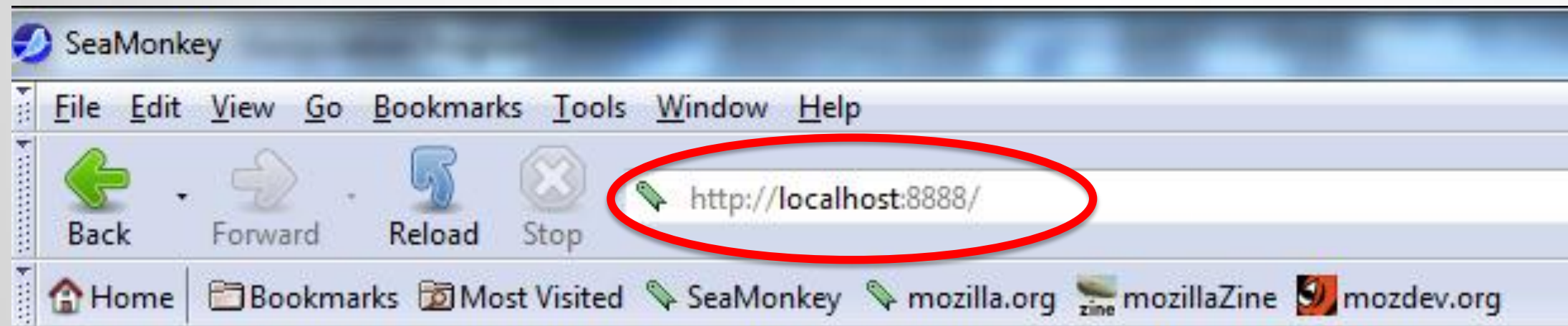
Example 6) A standalone application

- Enable the WebInterface by clicking the “Enable” toggle ON
- Set the port number to whatever your standalone app wants to use
- Turn on “localhost only” if you want requests only from your local machine



Example 6) A standalone application

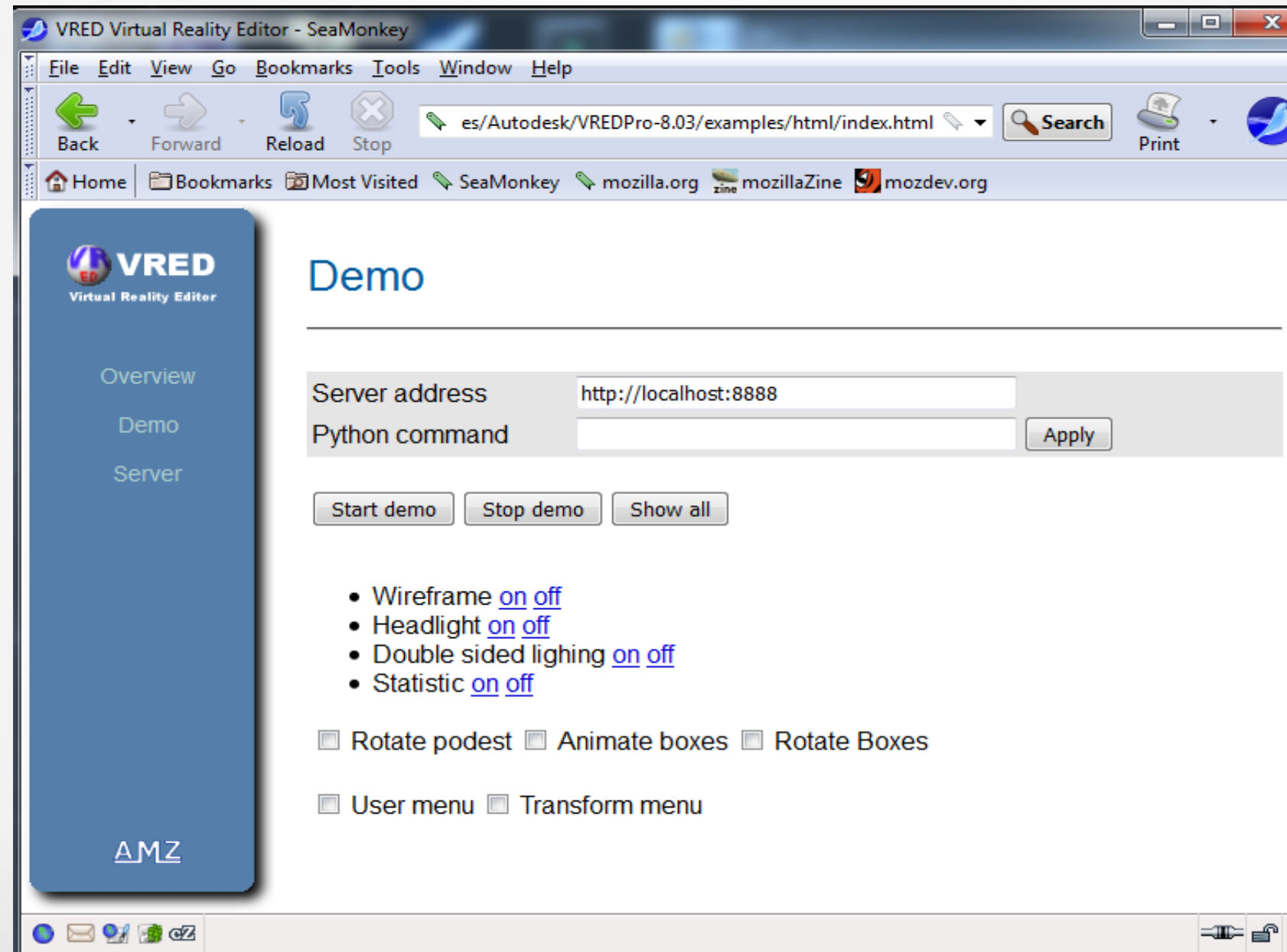
- The easiest example uses a web browser
- Navigate the URL tab to `http://localhost:8888/`



- Your browser will display a UI generated by VRED

Example 6) A standalone application

Here is the example UI



Example 6) A standalone application

- You can also create a custom UI, for example your own custom web page that communicates with VRED using the WebInterface
- Your code will have to send requests and receive replies from VRED
- This is done by sending python commands formatted as HTTP messages
- It's very easy to do. Here's an example...

Example 6) A standalone application

- The following python function send commands to the WebInterface

```
import urllib
import socket
def sendVredCmd(cmd, replyNeeded):
    reply = None
    cmd = urllib.quote(cmd)
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(("localhost", 8888))
    s.send("GET ")
    if replyNeeded:
        s.send("/pythoneval?value=")
    else:
        s.send("/python?value=")
    s.send(cmd)
    s.send(" HTTP/1.1\r\n\r\n")
    if replyNeeded:
        reply = s.recv(5000)
    s.close()
    return reply
```

Example 6) A standalone application

- Call the function like this...

```
sendVredCommand( "print 'Hello, world!'", False )
```

- When you call a VRED command that returns a value, the 2nd arg is True
- For example, to query the variant groups try this:

```
rslt= sendVredCmd( "vrVariantSetsService.getVariantGroups()", True )  
print rslt
```

- The result needs parsing:

```
HTTP/1.1 200 OK  
Server: VREDMicroWebInterface  
Access-Control-Allow-Origin: *  
Connection: close  
Expires: 0  
Content-Type: text/html
```

```
vred_python_result=(u'Animation', u'Roof', u'License Plate', u'Lights', u'Carpaints',  
                    u'Interior', u'Environment')
```

Key points for standalone application UIs

Pros:

- Ultimate choice in language: HTML, Java, C#, etc.
- Can extend applications such as Maya to talk to VRED
- UI application can be running on a different computer

Cons:

- Need to enable WebInterface
- Can only perform “textual” queries because VRED types are unknown
- Need to parse HTTP replies

Example 7) PythonQt

- With VRED 2016 SR1, PythonQt provides Qt access within VRED python
- Almost the entire repertoire of ~1000 Qt classes are available
- You can use Qt Designer to create an XML string for your UI as with *vrWidget*
- Benefit is the much greater variety of Qt functionality available
- You can also make direct Qt calls

Example 7) PythonQt

- To programmatically create your UI, first create the window:

```
win = QtGui.QWidget()
```

- Create a layout and assign to the window:

```
layout = QtGui.QVBoxLayout()
```

```
win.setLayout(layout)
```

- Add content such as labels, buttons, tabs, sliders, more layouts etc. to create your UI

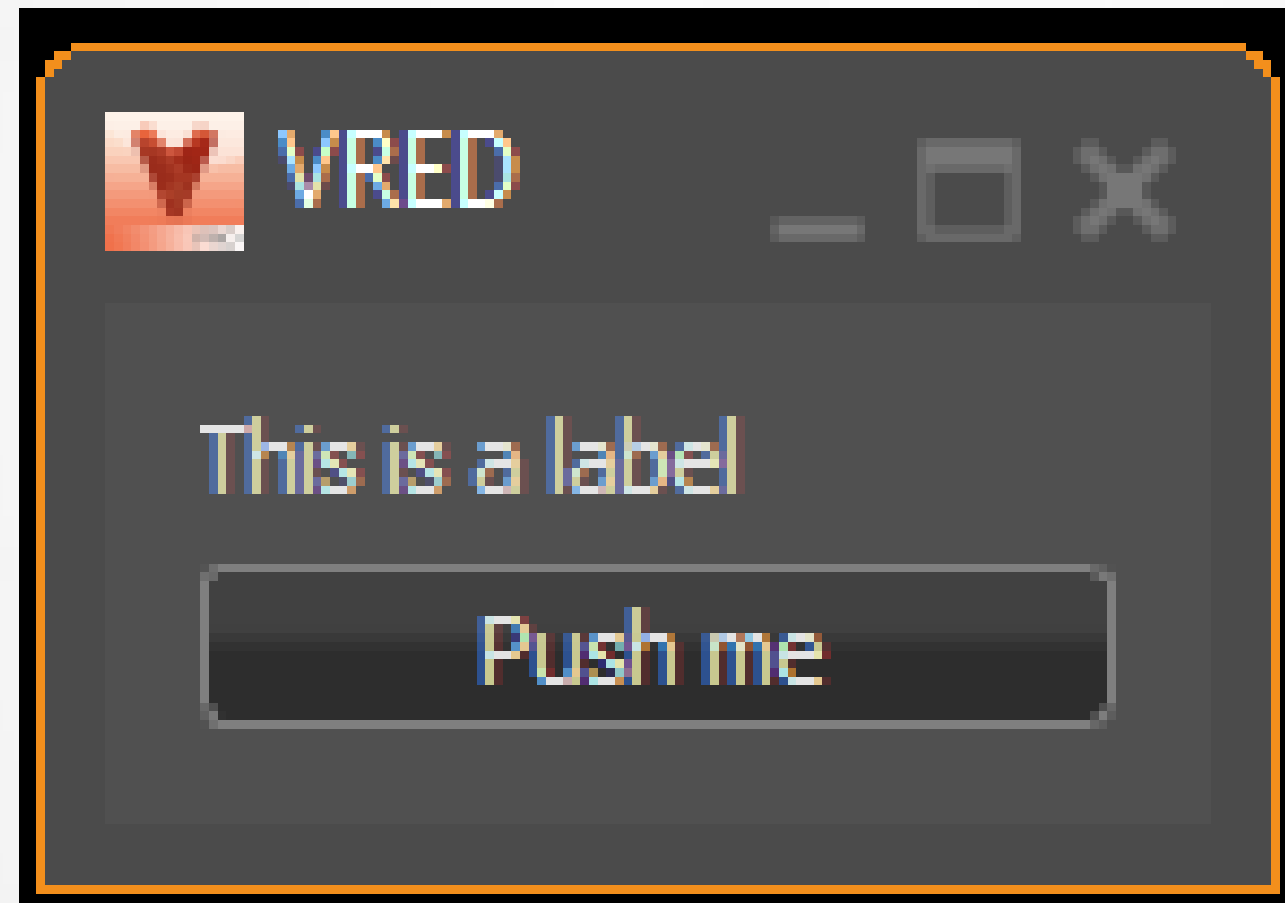
```
button = QtGui.QPushButton("Push me")
```

```
button.clicked.connect(buttonCallback)
```

```
layout.addWidget(button)
```

Example 7) PyQt

Here is the example UI (source code is online as `example7.py` in the extras)



Example 7) PyQt

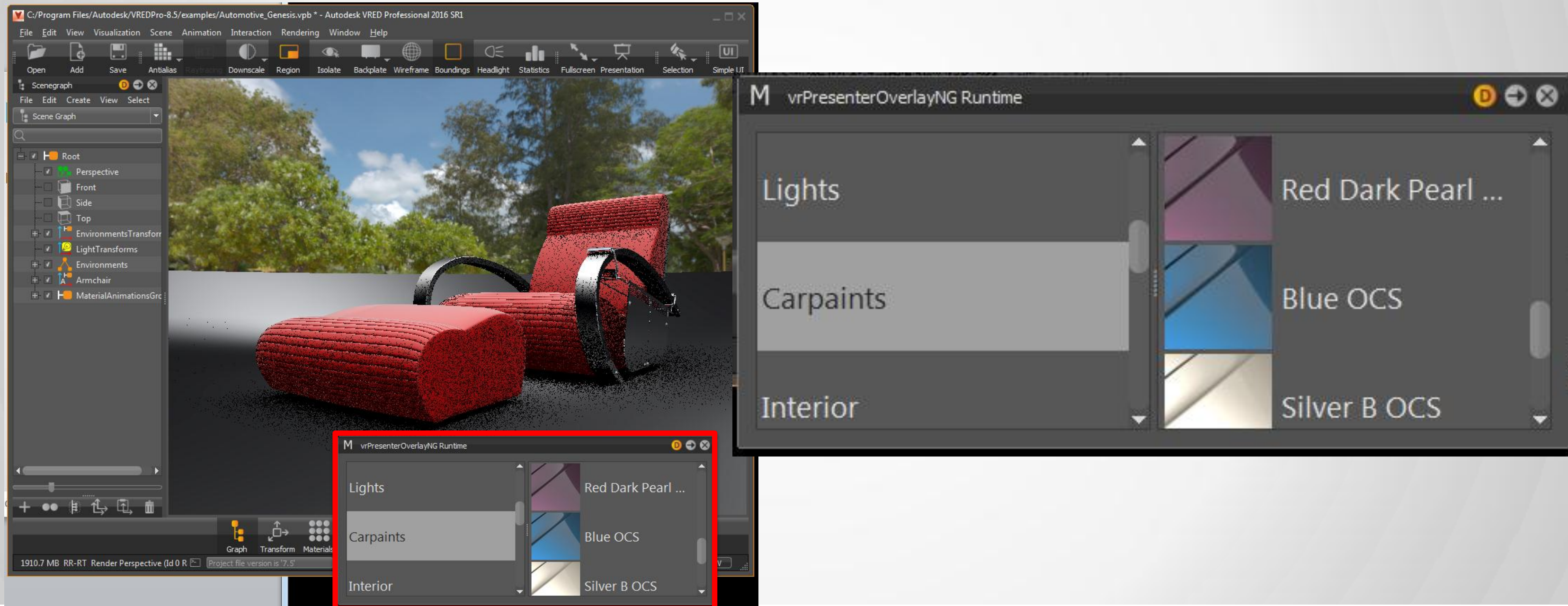
- If you prefer to define your UI as an XML string (as we saw with the vrWidget example 5), create the UI thusly:

```
from PyQt import QtCore
from PyQt import QtGui
uibuffer = QtCore.QBuffer(uixml)
loader = QtGui.QUiLoader()
ui = loader.load(uibuffer)
```

- Then you connect the UI elements to callbacks as was done in example 5.

Example 7) PythonQt

The example is *module.py* from the VRED 2016 distribution



What's the best approach for custom UI creation?

Approach	Example	Style	WebInterface	Notes
vrMenu with UI	example3	3D menu in viewport	No	Limited UI toolset
vrMenu with web content	example4	3D content in viewport	Required	Need to use web-based UI; slow
vrWidget with XML UI	example5	Separate 2D window	No	Somewhat limited in what you can do with UI
Standalone application	example6	Separate 2D window	Required	Very flexible, but lots of coding & sync issues
PythonQt	example7	Separate 2D window	No	Full Qt functionality; both XML & programic

A cool trick for launching your UI

- If you are using the WebInterface, create a desktop icon that launches a python script. The script sends an *execfile* command to VRED which causes a python file containing the UI to be executed.
- Voila... the UI window will be created each time the user double clicks the desktop icon.
- If NOT using the WebInterface, place the *execfile* command in the Script section of your Preferences.
- The UI window will be created when you do a newScene or open a scene.

Conclusion

- **Objective 1:** Understand the various versions of VRED and the API capabilities of each
- VRED Professional contains the complete API
- VRED Designer has read-only API support
- VRED does not support the API

Conclusion

- ***Objective 2:*** Understand the different ways to extend VRED
- Terminal Window
- WebInterface
- Script Preferences
- Variant scripting

Conclusion

- **Objective 3:** Write a simple extension to VRED with the API
- We showed how to display messages
- A more sophisticated example that used the API to create and manipulate nodes

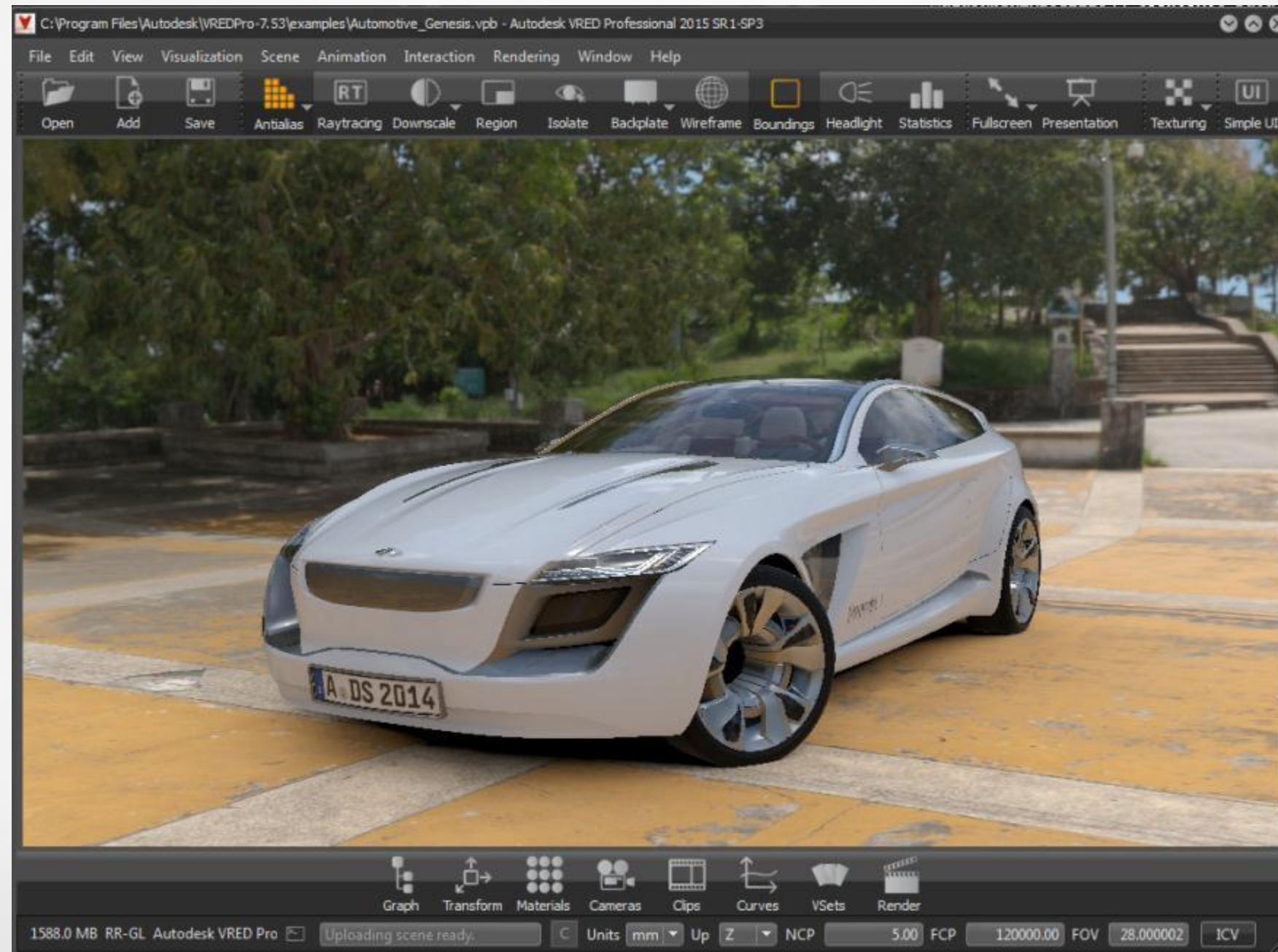
Conclusion

- **Objective 4:** Create custom user interfaces for VRED
- We showed UI and web content in a viewport
- vrWidget creating UI from an XML string
- Creating a standalone application for the UI and using the WebInterface to communicate with VRED
- Using PythonQt from within VRED to create a Qt-based UI

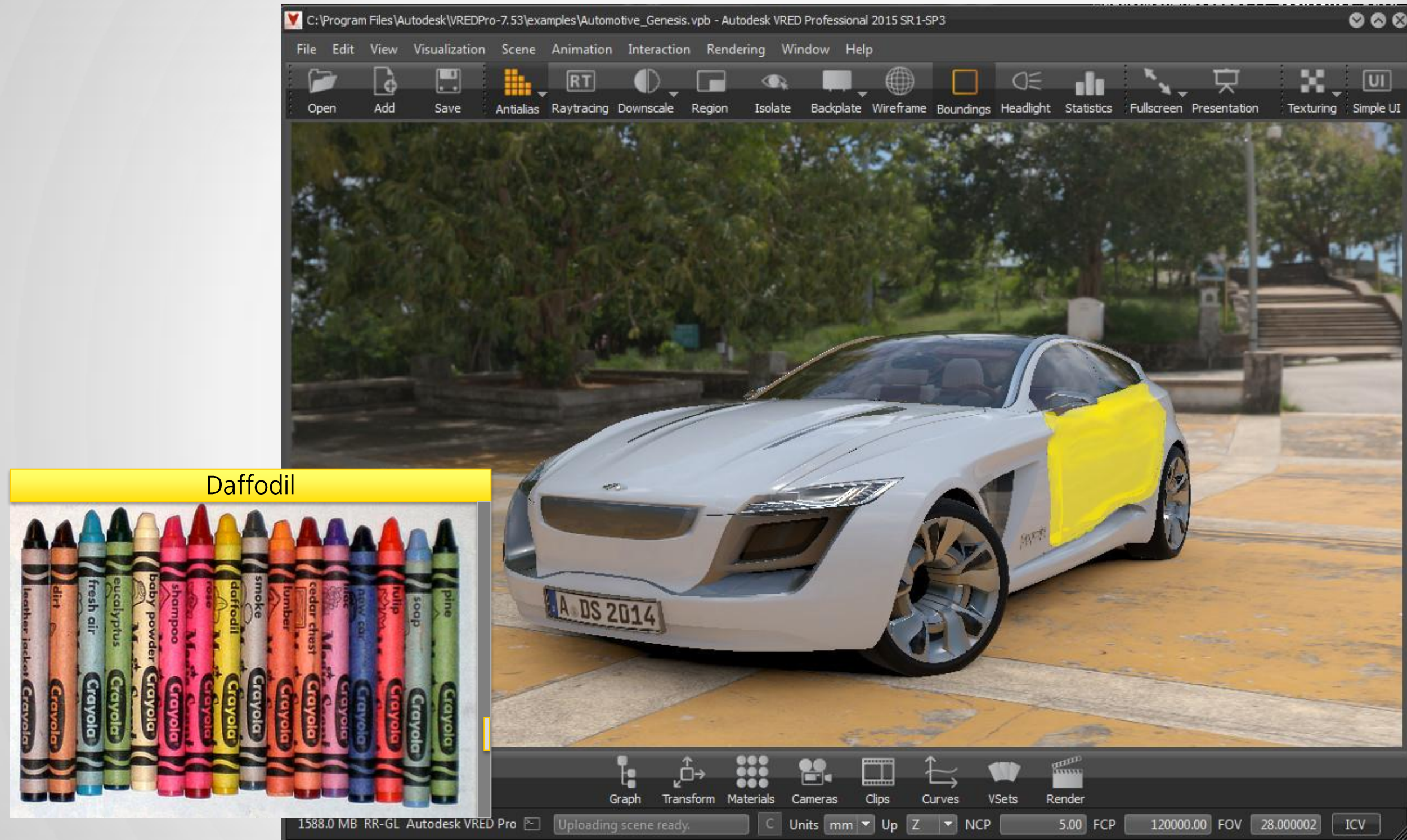
Questions?



An overview of Autodesk® VRED Professional



Creating custom user interfaces



An overview of what I'll be showing in this class...

6) Using *vrWidget* to create UI in a separate window

