Dual Scripting in a Virtual Reality Engine
Embedding Python in XVR

Emanuele Ruffaldi
pit@sssup.it
VR installations can be complex systems

Integration is hard: many aspects to tackle, many subtle details easy to overlook
The complexity is intrinsic

Real-Time Graphics
Haptic Interaction
Network communication
Positional sound
Physics Engine
Collision Detection
Virtual Reality Application
Writing VR applications is an hard task

- Often require good C++ skill and a deep knowledge of several HW technologies (video/audio/haptic)

- Plenty of tools available, but mixing them is no trivial. Also, hi-performance tools and libraries need to be properly handled (otherwise performances might suffer)

- Multidisciplinary: team-work is a necessity
3D @ PERCRO
What is XVR?

- A fully integrated development environment
- S3D a C-like programming language (but VR-oriented)
- The IDE integrates a very fast compiler
- Using precompiled byte code
- The Virtual Machine executed in a Web plugin
- Applications can be embedded inside web pages
- Data exchange with the Web page JavaScript, Flash
- Extensibility through external C++ modules (custom or a-la CType)
**XVR Workflow**

**Dedicated scripting language**

```c
// XVR snippet to update crowd positions, orientations and frames of avatars

for(i=0;i<NUMULATION;i++)
{
    orientations[i] = orientations[i] + 0.05;
    frameNumbers[i] += 0.5;
    positions[i][0] = positions[i][0] + 0.05;
    positions[i][1] = positions[i][1] + 0.05;
    positions[i][2] = positions[i][2] + 0.05;
}

CROWD SetOrientations(orientations);
CROWD SetFrameNumbers(frameNumbers);
CROWD SetPositions(positions);
Posx = VectorRotate(1.0, 0.0, 0.0, Posx);
Locx SetPosition(Posx);
CROWD SetTaskPosition(Posx);

// Rendering CODE
ScreenBegin();
DrawFloor();
CROWD Draw(GetCameraPosition());
ScreenEnd();
...
```

**Output**

- Binary
- ByteCode
- Compiler
- Interpreter (Virtual Machine)
A WEB-enabled technology...
Advanced VR Installations
Limits of S3D

- Yet another scripting language although with a small learning curve
- No debugging tools
- Compile only language, no dynamic scripting
- No multi-threading

... now enters PYXVR
Introducing PYXVR

- Scripting system for VR and 3D Web applications based on Python
- All the advantages of XVR and Python
- Python
  - Wonderful language
  - Debugging (e.g. Winpdb)
  - Existing libraries
  - Dynamic Execution
- XVR
  - Web Deployment with versioning
  - Core 3D/VR libraries
- PYXVR uses
  1. Extending an existing XVR application with Python modules
  2. Developing a full 3D/VR application in Python
The Python script accesses all the functions (e.g. glColor) and objects of the XVR VM. Also the functions defined in the S3D script.
PYXVR application structure

The XVR engine load the application and invokes Callbacks

- **OnDownload(param)** for getting resources
  - files are downloaded in a temporary directory and zip archives unpacked
- **OnInit(param)** for initialization
- **OnFrame()** at every rendering frame (~50Hz)
- **OnTimer()** about every 1ms
- **OnEvent(event)** for asynchronous events

The typical PYXVR application sends these events to Python
Minimal PYXVR Application

```python
#include <Script3d.h>

extern function PythonEngine;
var py;

function OnDownload(script)
{
    FileDownload("pyxvr.zip");
}

function OnInit(script)
{
    LoadModule( "pyxvr_0141.dll");
    py = PythonEngine();
    py.evalFile("pyxvrapp.py");
    py.call("OnInit");
}

function OnFrame()
{
    py.call("OnFrame");
}

function DrawGrid(x)
{
    // ...
}

from pyxvr import *

mesh = None
pos = 0.5

def OnInit():
    global mesh
    mesh = CVmNewMesh("box.aam")
    mesh.Normalize(1)
    SetCameraPosition([0,2,-10])
    CameraSetTarget(0,0,0)

    def DrawGridPY(n):
        glLineWidth(n)
        glDisable(GL_LIGHTING)
        glColor(1,0.5,0.5)
        glBegin(GL_LINES)
        for i in range(-100,100,10):
            glVertex(i, 0, 100)
            glVertex(i, 0, -100)
            glVertex(100, 0, i)
            glVertex(-100, 0, i)
        glEnd()

XVR.DrawGrid(3)
SceneBegin()
```

```python
def OnFrame():
    global mesh
glTranslate(0,pos,0)
    DrawGridPY(2)
    XVR.DrawGrid(3)
SceneEnd()
```
PYXVR Deployment

Core Components

- **Myscript.py**: My Python Script
- **pyxvr.zip**: PYXVR core and Python libraries
- **pyxvrmin.s3d.bin**: Stub XVR application that loads Myscript.py

**Web**
- **Myscript.htm**

**Offline**
- **pyxvr.exe**
Example - RSS

Python provides modules

Feedparser based parsing of RSS
Type Mapping

Type mapping is fundamental, and primitive types are directly mapped (int, bool, String)

From XVR to Python

- vector of float \([1,2,3]\) => array('f')
- array => List
- object => wrapper object of class XVRObject

From Python to XVR

- object => only of class XVRObject
- list, tuple => array (although could be vector)
Example – PYXVR particle

Python porting of the particle system
PYXVR - Particle Performance

A performance comparison only on the update of the particles, not *rendering*

<table>
<thead>
<tr>
<th>Number of Particles</th>
<th>S3D</th>
<th>PY</th>
<th>S3D Wine</th>
<th>PY Wine</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>23fps</td>
<td>19fps</td>
<td>20fps</td>
<td>15fps</td>
</tr>
<tr>
<td>20000</td>
<td>12fps</td>
<td>10fps</td>
<td>11fps</td>
<td>8fps</td>
</tr>
<tr>
<td>40000</td>
<td>6fps</td>
<td>5fps</td>
<td>5fps</td>
<td>4fps</td>
</tr>
</tbody>
</table>

S3D is little faster because of optimized code for vector operations, and Python version could be optimized

Linux version is running Wine so it is using Python for Windows
Example – file access

Simple 3D photo browser that uses Python file listing functions
Advanced use - On the fly coding

Modify the Python code and the 3D scene at run time

Should behave like a Python shell
Need improvements in error handling.
Advanced Uses - Stackless

• Virtual Reality applications with agents are pretty interesting
• Stackless provides a interesting way to change the programming model
• Just replace python24.dll with the one from Stackless

```python
def Agent(id):
    life = random.randint(1,1000)
    pos = [random.random()*5-2.5,random.random()*3,0]
    print ("Agent ",id," ",life," ",pos)
    vel = [random.random()*0.05-0.025,random.random()*0.04,0]
    for i in range(life):
        pos[0] = pos[0] + vel[0]
        XVR.SetAgentPosition(id, pos[0],pos[1],pos[2])
        schedule()
    print "death"
```
Open Issues

- Security of execution from Web pages
- Improving method invocation performance (by name)
- Access of Python objects from XVR
- Windows only (except Linux using WINE)
Conclusions

- PYXVR is a tool for writing VR solutions, 3D Web applications or Games
- Based on the great Python language and an advanced VR toolkit

Enjoy it, just Google “PYXVR”