Enabling Closed-Source Applications for Virtual Reality via OpenGL Intercept Techniques

Co-located with IEEE VR 2014

David J. Zielinski (Duke University)
Ryan P. McMahan (The University of Texas at Dallas)
Solaiman Shokur and Edgard Morya (International Institute for Neuroscience of Natal)
Regis Kopper (Duke University)
Motivation - Part 1

- VR experiences often utilize special hardware.
- So we need to use software that can utilize the special hardware.
Motivation - Part 2

● To utilize VR, a user often needs to learn new language/application
● Barrier for adoption of VR.
● Could we VR-enable the desktop application the user is already familiar with?
  ○ even if it’s commercial/closed source?

Answer: OpenGL intercept-based techniques.
What is OpenGL?

"OpenGL is an API (Application Programming Interface) for rendering 2D and 3D computer graphics. The API is typically used to interact with a GPU, to achieve hardware-accelerated rendering. OpenGL was developed by Silicon Graphics Inc. in 1992." -Wikipedia

<table>
<thead>
<tr>
<th>MATLAB</th>
<th>OpenGL</th>
<th>Screen</th>
</tr>
</thead>
<tbody>
<tr>
<td>fill3([0 0 0],[1 0 0],[1 0 1],</td>
<td>glBegin(GL_TRIANGLES)</td>
<td>vertex = corner</td>
</tr>
<tr>
<td></td>
<td>glVertex3f(0,0,0); glVertex3f(1,0,0);</td>
<td></td>
</tr>
<tr>
<td></td>
<td>glVertex3f(1,0,1); glVertex3f(1,0,1);</td>
<td></td>
</tr>
<tr>
<td></td>
<td>glEnd();</td>
<td></td>
</tr>
</tbody>
</table>
Intercept Technique Example

**Standard Case**

- Drawing Command (e.g. “draw a triangle”)
- Graphics Driver (opengl32.dll)
- Monitor

**Intercept Case**

- Drawing Command (e.g. “draw a triangle”)
- Intercept Driver (new opengl32.dll)
- Graphics Driver (opengl32.dll)
- Monitor

[place new dll in directory of executable]
//inside our custom opengl32.dll

glColor3f(float r, float g, float b)
{
    //can do some modification to color here

    r=0.0;
    g=0.0;
    b=1.0; // make all colors blue!

    real_glColor3f(r,g,b); //call real glColor3f
}
Technique #1: In-And-Out

Desktop Application

Plugin or Built-in Scripting

graphics out

custom opengl32.dll

real opengl32.dll

input device data sender

input device data in
Technique #2: Intercept Tags

- Tags tell renderer to do extended functionality
- Tags are actually specific geometry calls
- Interpreted, not rendered

Conceptually

--- scene ---
draw_cube

<intercept_tag>
draw_sphere
<intercept_tag>

Host Renders

--- scene ---
draw_cube

draw_triangle(0,0..)

draw_sphere
draw_triangle(0,0..)

VR Application Processes

--- scene ---
draw_cube

load special transform
draw_sphere
unload transform
Intercept Techniques Uses

Hand-Off Techniques

- Hand-Off Manipulation
- Hand-Off Slice Plane

Visual Enhancements

- Interpolated Animations
- Shader Insertion

Display Techniques

- Level Of Detail
- Display Lists
Technique #3: Driver Mediated Head Tracking

- Desktop Application
- Graphics
- custom opengl32.dll
- real opengl32.dll
- Input device data
- Input device data sender
Case Study #1: MATLAB Technique: In-And-Out

Minimal additions to access data from VR devices

```matlab
vr = vr_interface(ServerIP, ServerPort);
while 1
    [event,btn,pos]=vr.get_button_event();
    if event==1 && btn==0
        % modify surface data here
    end
    clf; % clear screen
    surf(dataX,dataY,dataZ); % render surface
    drawnow; % flush
end
```
Results:

- Artificial potential function used to plan the motion of the robot [1]
- Potential function recalculates the best path to the target while adapting to the dynamic addition and removal of obstacles.
- We added in ML2VR functions to make the simulation interactive.
- We can now explore how the algorithm reacts in immersive VR!

Case Study #2: MATLAB + Intercept Tags

Hand-off Dragging

1. `draw_sphere`
   - `user presses and holds button`

2. `<intercept tag>` `draw_sphere` `<intercept tag>`
   - `Desktop App adds Tags`
   - `intercept driver`

3. `<intercept tag>` `draw_sphere` `<intercept tag>`
   - `intercept driver`

4. `draw_sphere`
   - `Desktop App removes Tags`
   - `user releases button`

VR Application/Renderer see the tags and allows Virtual Hand interaction at high FPS
Hand-Off Manipulation User Study

- place a solid cube completely inside a wireframe cube
- varying sizes of wireframe cube
- within subjects design. hand-off manipulation vs original virtual hand
- 14 subjects

Results: Handoff faster, less clutches. Self reported less sickness, more presence, higher usability.

Case Study #3:

- Using MotionBuilder for BCI Experiments.
- Can we VR-Enable MotionBuilder for the Oculus Rift?
Background: Brain Computer Interface (BCI)

- Real-time decoding of brain activity
- Invasive or Non-Invasive

We believe that VR can be used to train subjects with reduced mobility to utilize a BCI to control physical devices (e.g., wheel chairs and exoskeletons) in a safe environment.
Background: MotionBuilder

- Commercial motion capture and animation software by Autodesk.
- Normally used as part of larger workflow.

- Has high quality real-time inverse kinematics, to modify character animations.
- However, can create interactive experiences via its constraints relations system, python scripting, and a C++ software development kit (SDK).
MotionBuilder + Oculus: In-And-Out

- Existing MotionBuilder view mode Freeview (Parallel), aka side by side is close to what we need.
- In profiler disable frustrum culling.
- MotionBuilder user must add in camera control.

orientation data goes from rift to udp sender into custom plugin into constraints system.
Issues Overriding Projection Matrix

- When anti-aliasing is enabled in MB, need to extract the x and y shift, and put them into our new projection matrix.
- We can also extract the existing clip planes by looking at values A and B
  - near clip = \( \frac{B}{A-1} \)
  - far clip = \( \frac{B}{A+1} \)

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Y</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MB + Oculus: Driver Mediated Head Tracking

- No extra plugins or constraints needed for MotionBuilder.
- Latency seemed less. How can we test this?

```
MotionBuilder 2013

Our OpenGL32.dll

glViewport_{6,7}
Larger Viewport
Activate Shaders

 glGetMatrixd_{1,3}
New Projection_{4x4}
New Model_{4x4}

real OpenGL32.dll

Oculus Rift

Oculus SDK
```

Modifications to Rendering (viewport and projection matrix values changed, shaders activated) plus driver mediated headtracking (override model view matrix with orientation from oculus).
Issues Overriding Model Matrix

- The MotionBuilder “Live Shadow”, no longer works.

- Solution: override glMultMatrixd
  - examine matrix being multiplied
  - does it match the original (non-modified) model view matrix?
  - if yes - override it.
New: Position Tracking

MotionBuilder 2013

Our Opengl32.dll

- glViewport
- Larger Viewport
- Activate Shaders

- glLoadMatrixd
- New Projection
- New Model

Oculus SDK

real Opengl32.dll

Oculus Rift

Precision Position Tracker (PPT) - IR

PPT Studio

VRPN Server

VRPN to UDP
Results: Success!

- MotionBuilder
- Oculus Rift
- Orientation from Rift
- Position from PPT
Thank You!
Questions?

http://virtualreality.duke.edu
http://utdallas.edu/~rpm120130/fivelab/