VR Graphics Programming for, well, you know

Some (relatively) easy steps to virtual reality, Part 4

Tom Sgouros

Center for Computation and Visualization
Brown University
thomas_sgouros@brown.edu

Spring 2018

OSCAR cluster

- Lots of CPUs in lots of nodes.
- Giant shared storage array: /gpfs
- Heterogeneous array. Some have graphics cards, some don't, old, new.
- Identical OS image.
- Separate authorization / authentication from rest of campus.
- Mostly batch jobs through SLURM.

Modules on OSCAR

- Support heterogeneous software.
- Can't load all the specialized software onto images.
- Controls environment variables like PATH, CPATH, LIBPATH, MANPATH, etc.

```
$ module load minvr
```

OSCAR accounts

- Quotas, myquota
- data and scratch
- ssh configuration
- .modules file

-

YURT structure

- Opti-trak tracker for head and hands
- VRPN Button presses
- 19 Linux machines, cave001, cave002, etc.
- 69 video projectors
- Scalable software applied to output.

MinVR Display Graph in practice

```
<YURTGraph>
  <RootNode displaynodeType="VRGraphicsWindowNode" windowtoolkitType="VRFre
    <LookAtNode displaynodeType="VRHeadTrackingNode">
      <StereoNode displaynodeType="VRStereoNode">
        <ScalableProjectionNode displaynodeType="VRScalableNode">
          <NearClip>0.01</NearClip>
          <FarClip>100.0</FarClip>
        </ScalableProjectionNode>
      </StereoNode>
    </LookAtNode>
  </RootNode>
</YURTGraph>
```

Compiling for YURT

- Use login003 or login004
- ssh3.ccv.brown.edu
- VNC client (CCV page, Computing/Software, look left), use desktop3 or desktop4.
- Don’t compile on dev nodes or on login001 or login002.

Controlling the YURT

- `pjcontrol` command.
- Projectors numbered, list on paper near kiosk, or ops guide. http://github.com/tsgouros/yurt-ops
- cave-utils module
- `pjcontrol 0-68 on` also `off`, `error`
- `pjcontrol 27 mono; sleep 10; pjcontrol 27 stereo`

Oops

- cavesupport@ccv.brown.edu
- support@ccv.brown.edu