VR Engines and Unity

Gordon Wetzstein
Stanford University

EE 267 Virtual Reality
Lecture 16

stanford.edu/class/ee267/
Overview

- VR engines: sound, physics, networking, IO, ...
- Unity
- course projects
VR Engine
(so far)

Graphics

IO
VR Engines - Audio

• middleware – between audio card and application (e.g. game)

• usually provides functionality for:
  • loading different types of sound files
  • mixing and mastering
  • 3D sound
  • occlusions, echoes, reverberation, …
VR Engines - Audio

• examples:
  • FMOD - www.fmod.org
  • OpenAL - “OpenGL for sound”
  • SDL – provides basic functionality
  • …
VR Engines – 3D Audio

- start with mono sound $x_A(t)$
- head-related impulse response (HRIR) model time delay and attenuation via convolution $\tilde{h}_L(t)$, $\tilde{h}_R(t)$
- basically different temporal shift for each ear
- but HRIR also includes other effects created by shape of ear and other factors
VR Engines – 3D Audio

• sound, 3D sound, coupling sound and physics, accurate HRIR or head-related transfer function gets much more complicated

• Prof. Doug James in CS is working on physics & sound, check out his recent SCIEN talk if you’re interested: “Physics-based Animation Sound: Progress and Challenges”

VR Engines - Physics

- framework to simulate:
  - rigid body dynamics (e.g. collision detection)
  - soft body dynamics (e.g. deformation, cloth, …)
  - fluid dynamics (water, smoke, fire, …)
VR Engines - Physics

- examples:
  - Open Dynamics Engine (http://www.ode.org/): free 😊 but limited to rigid body dynamics & collision
  - Bullet Physics (http://bulletphysics.org/): free 🙂, rigid & soft body dynamics, widely used
  - havok (owned by Microsoft) – not free 😞 but widely used, real-time rigid body dynamics
rendered in blender, bullet physics - https://www.youtube.com/watch?v=-6Sl5CCxp3Q

Early Tests
VR Engines – User Interface (UI)
VR Engines – User Interface (UI)

• concept is straightforward: widgets, menus, buttons, checkboxes, ...

• types of UIs:
  • non-diegetic – lives in screen space (e.g. player status); doesn’t work in VR (no screen space)
  • spatial UI – lives in the virtual world
  • diegetic – menus in world
VR Engines - IO

- support for interfaces: keyboard, mouse, 3D mouse, standard haptic devices, ...

- VR engine would provide functionality as well (e.g. Unity)
VR Engines – Content Creation

- 3D modeling programs / Computer-aided Design (CAD):
  - Maya (production)
  - 3ds Max (games)
  - Blender – free
  - SolidWorks – 3D printing & fabrication
  - Tinkercad: free & online
VR Engines – Content Creation

• what’s involved?
  • conceptual design
  • 3D modeling
  • animation and/or simulation
  • scripting behavior and artificial intelligence of characters
  • testing
  • … many different stages in application development …
VR Engines - Scripting

- core engine is usually designed for performance – C++
- developing applications should be easy! the user almost never wants to touch the C++ source but needs flexibility
- provide a script-based interface to allow user to change anything they need for their application
  - create & manipulate objects
  - script behavior
  - change shaders (e.g. change camera or fragment shader art)
VR Engines - Networking

• manage low-level communication protocols (TCP/IP, UDP, …)
• ensure that character states, graphics, sound, and everything else is synchronized
• connect to application that’s running as client
• network updates, messages, …
Popular VR/Game Engines

- **Unity**: cross-platform, Direct3D (Win), OpenGL (Mac & Linux), iOS & Android support, also came console APIs; personal license is free; seems to be the easiest to use so we’ll use it for Lab 6 and HW 6.

- **Unreal**: very popular, lots of awards! Unreal Engine 4 is free.

- **CryEngine**: popular game engine, just announced support for VR; free for non-commercial use.
Unity

live demo
Additional Information

• Unity game engine: https://unity3d.com/
• Unity tutorials: https://unity3d.com/learn/tutorials
Other Aspects of VR
Latency, Post-rendering Warp, Eye Tracking
Latency

- min acceptable: 20 ms
- interactive applications <20 ms (say target is 5 ms)

The latency between the physical movement of a user’s head and updated photons from a head mounted display reaching their eyes is one of the most critical factors in providing a high quality experience.

- John Carmack
Latency – where does it come from?

- IMU ~1 ms
- sensor fusion, data transfer
- rendering: depends on complexity of scene & GPU – a few ms
- data transfer again
- display: LCD ~60 Hz = 16 ms; OLED <1 ms
Latency – how bad is it really?

- example:
  - 16 ms (display) + 16 ms (rendering) + 4 ms (orientation tracking) = 36 ms latency total
  - head rotates at 60 degrees / sec (relatively slow)
  - 1Kx1K display over 100 degrees field of view

- in 36 ms, my head moved 1.92 deg ~ 19 pixels = size of thumb at arm’s length! too much
Display Pixel Updates

Raster Scan
(e.g. electron beam in CRT)

Rolling Update
(most LCDs)

Global Update
(some LCoS, DLP, other)
Display Pixel Switching - Persistence

- after the display pixel switched states, how long is it on?

**Full Persistence**
- (most LCDs)

**Half Persistence**
- (strobe backlight)

**Low Persistence**
- (OLED, strobing)

example: switch from white to black to white to black as fast as possible
Rapid relative motion

head mounted display

20 degree rotation

head mounted display

slides from Michael Abrash 2013
Space-time diagrams (static)
Spatial movement over time

slides from Michael Abrash 2013
Spatial movement over time

slides from Michael Abrash 2013
Pixel-based movement

slides from Michael Abrash 2013
Sequential RGB display

One frame

Time
Time
Sequential RGB with eyes moving

one frame

time

slides from Michael Abrash 2013
Full persistence
Zero persistence

How photons end up on the retina

x

time

one frame

slides from Michael Abrash 2013
Full persistence + head rotation

slides from Michael Abrash 2013
Post-rendering Image Warp

• also called time-warp by John Carmack
• minimize end-to-end latency
• original paper from Mark et al. 1997, also Darsa et al. 1997
• overview:
  1. get orientation from IMU, perhaps also position
  2. render scene into off-screen buffer (larger than screen)
  3. read latest orientation from IMU
  4. warp rendered image with latest orientation
• 2D image translation v 2D image warp v 3D image warp
End-to-End Low-Latency Optical See-Through AR Pipeline

Tracker
30,000 Hz

3D rotation +
3D translation

slides from Zheng et al. 2014
End-to-End Low-Latency Optical See-Through AR Pipeline

Tracker
30,000 Hz

3D rotation +
3D translation

Renderer
30 Hz

Color + Depth
(> Disp. Res.)

Render resolution  Display resolution

slides from Zheng et al. 2014
End-to-End Low-Latency Optical See-Through AR Pipeline

Tracker
30,000 Hz

Renderer
30 Hz

Color + Depth
(> Disp. Res.)

Post-Rendering 3D Warp
300 Hz

3D rotation +
3D translation

Color
(> Disp. Res.)

Render resolution
Display resolution

slides from Zheng et al. 2014
End-to-End Low-Latency Optical See-Through AR Pipeline

Tracker
30,000 Hz

3D rotation +
3D translation

Post-Rendering 3D Warp
300 Hz

Color + Depth
(> Disp. Res.)

2D Warp
3,000 Hz

Color
(> Disp. Res.)

2D rotation +
2D translation

Renderer
30 Hz

Color + Depth
(> Disp. Res.)

3D rotation +
3D translation

Color
(> Disp. Res.)

render resolution
display resolution

slides from Zheng et al. 2014
End-to-End Low-Latency Optical See-Through AR Pipeline

Tracker  
30,000 Hz

3D rotation + 3D translation

Post-Rendering 3D Warp  
300 Hz

3D rotation + 3D translation

2D Warp  
3,000 Hz

2D rotation + 2D translation

Color + Depth  
(> Disp. Res.)

Color  
(> Disp. Res.)

Color  
(> Disp. Res.)

2D Offset  
30,000 Hz

2D translation

A cascade of successively simpler and faster renderers running in GPU

Render resolution  Display resolution

slides from Zheng et al. 2014
Summary: Latency, Persistence, etc.

- predictive tracking (e.g. LaValle ICRA 2014)
- post-rendering warp
- design and build really great hardware & algorithms
- use OLED displays or strobing backlights for low persistence
- design some type of a device to actually measure latency!
Eye Tracking

- necessary for gaze-contingency paradigm (foveated rendering, gaze-contingent rendering, gaze-contingent focus, ...)
- interaction
- eye contact
- ...
Eye Tracking

• many different techniques:
  • electro-oculography
  • contact lens tracking
  • video-oculography
  • pupil / corneal reflection tracking
  • dual Purkinje image
Eye Tracking

• some interesting properties one can exploit:
  • pupillary light reflex doesn’t work in near infrared (IR)
  • red-eye effect with co-axial camera - light source
  • purkinje images for off-axis illumination
Eye Tracking – Pupil / Corneal Reflection
Eye Tracking – Pupil / Corneal Reflection

- corneal reflection stays constant, pupil center moves relative!
Eye Tracking – Pupil / Corneal Reflection

- corneal reflection stays constant, pupil center moves relative!
Eye Tracking – Dual Purkinje

- track relative location of Purkinje images

http://ppw.kuleuven.be/home/english/research/lep/resources/purkinje
Eye Tracking

- where am I looking? what am I looking at?

http://www.getbusymedia.com/small-business-insights-eye-tracking/
References and Further Reading

• Google Project Tango: https://developers.google.com/project-tango/

• Post-rendering warp:
  • John Carmack “Time Warp”, 2013 (blogs)

• Latency: