SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

16IT AUGMENTED REALITY AND VIRTUAL REALITY

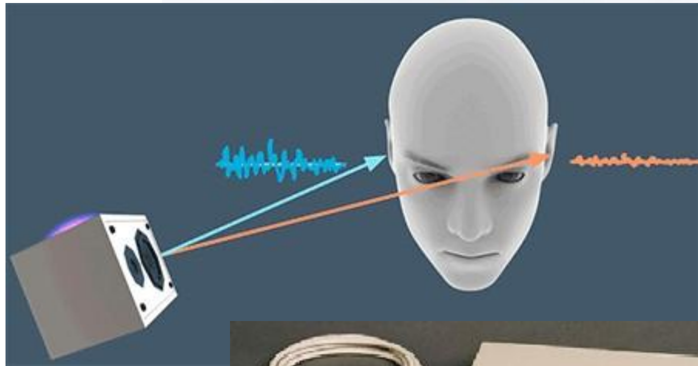
III YEAR – V SEM

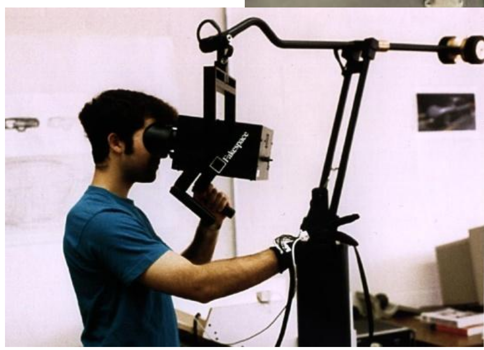
UNIT 5 – VR PROGRAMMING

TOPIC 1 –Toolkits and Scene Graphs

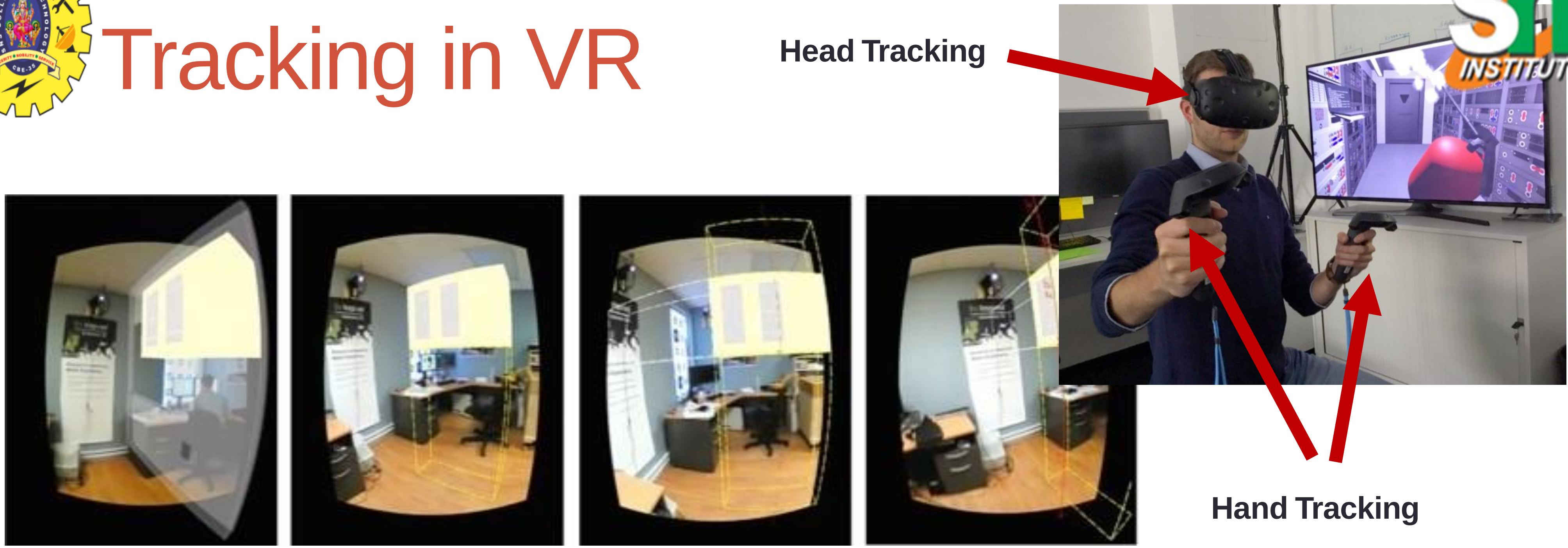
Recap – Last Week

- Survey of VR technologies
  - Tracking
  - Haptic/Tactile Displays
  - Audio Displays
  - Input Devices





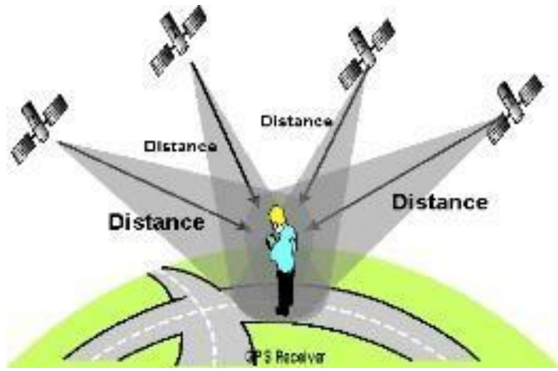
Tracking in VR



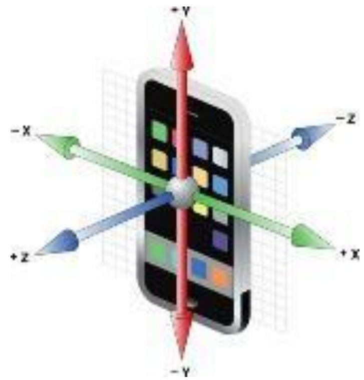
- Need for Tracking
  - User turns their head and the VR graphics scene changes
  - User wants to walking through a virtual scene
  - User reaches out and grab a virtual object
  - The user wants to use a real prop in VR
- All of these require technology to track the user or object
  - Continuously provide information about position and orientation

Tracking Technologies

- Active (device sends out signal)
  - Mechanical, Magnetic, Ultrasonic
  - GPS, Wifi, cell location



- Passive (device senses world)
  - Inertial sensors (compass, accelerometer, gyro)
  - Computer Vision
    - Marker based, Natural feature tracking
- Hybrid Tracking
  - Combined sensors (eg Vision + Inertial)



Haptic Feedback

- Greatly improves realism
- Hands and wrist are most important
  - High density of touch receptors
- Two kinds of feedback:
  - Touch Feedback
    - information on texture, temperature, etc.
    - Does not resist user contact
  - Force Feedback
    - information on weight, and inertia.
    - Actively resists contact motion

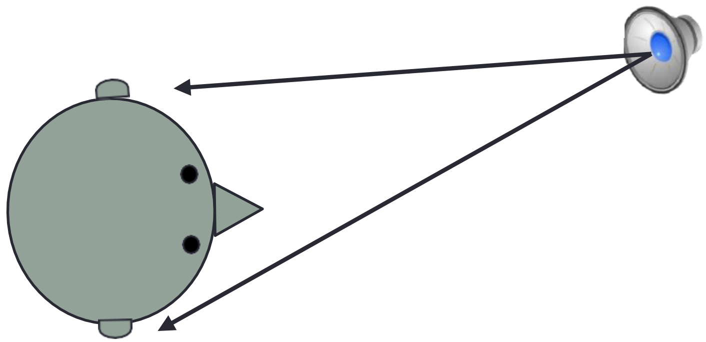
Active vs. Passive Haptics

- Active Haptics
  - Actively resists motion
    - Key properties
      - Force resistance, DOF, latency

- Passive Haptics
  - Not controlled by system
    - Use real props (e.g. styrofoam for walls)

Audio Displays

- Spatialization vs. Localization
- *Spatialization* is the processing of sound signals to make them emanate from a point in space
  - This is a *technical* topic
- *Localization* is the ability of people to identify the source position of a sound
  - This is a *human* topic, i.e., some people are better at it.
- Head-Related Transfer Function (HRTF)
  - Models how sound from a source reaches the eardrum
  - Needs to be measured for each individual

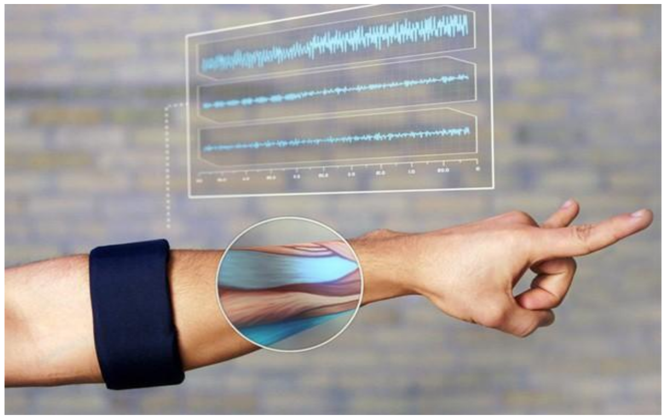


VR Input Devices

- Physical devices that convey information into the application and support interaction in the Virtual Environment

Multiple Input Devices

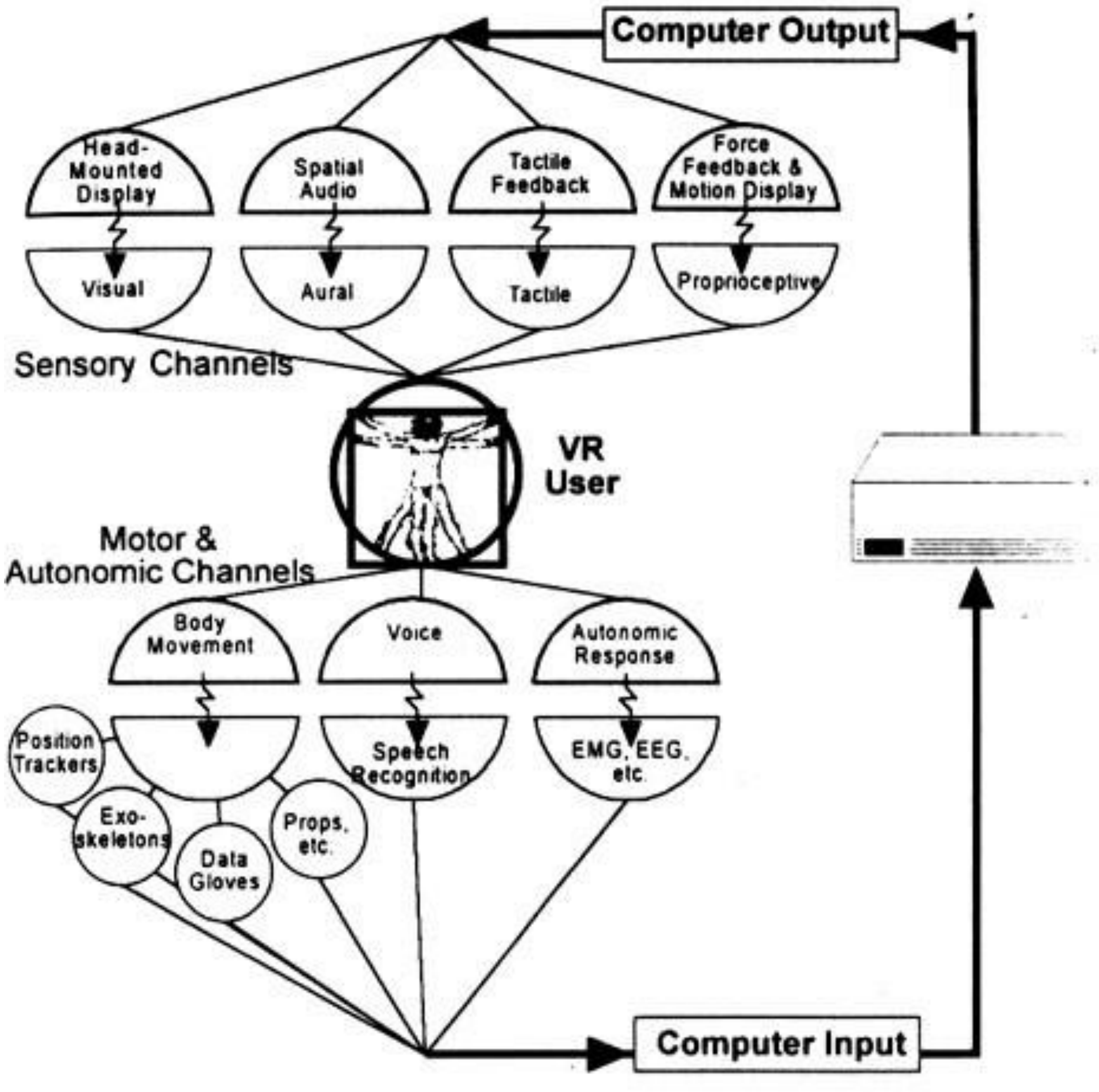
- Natural
  - Eye, gaze, full body tracking
- Handheld devices
  - Controllers, gloves
- Body worn
  - Myo armband
- Pedestrian devices
  - Treadmill, ball



Mapping Between Input and Output

Input

Output



Comparison Between Devices

From Jerald (2015)

Comparing between hand and non-hand input

	Proprioception	Consistent	Usable in Lap or the Side	Haptics Capable	Unencumbered	Physical Buttons	Hands Free to Interact with Real World	General Purpose
Hand Input Device Class								
World-Grounded Devices	✓	✓		✓	✓	✓	✓	
Non-Tracked Hand-Held Controllers		✓	✓	✓		✓		
Bare Hands	✓				✓		✓	✓
Tracked Hand-Held Controllers	✓	✓	✓	✓		✓		✓
Hand Worn	✓	✓	✓	✓		✓	✓	✓
Non-Hand Input Device Class								
Head Tracking	✓	✓					✓	✓
Eye Tracking							✓	
Microphone			✓		✓		✓	✓
Full-Body Tracking	✓	✓	✓	✓			✓	✓
Treadmills	✓	✓			✓		✓	

VR SYSTEMS

Creating a Good VR Experience

- Creating a good experience requires good system design
  - Integrating multiple hardware, software, interaction, content elements

Example: Shard VR Slide

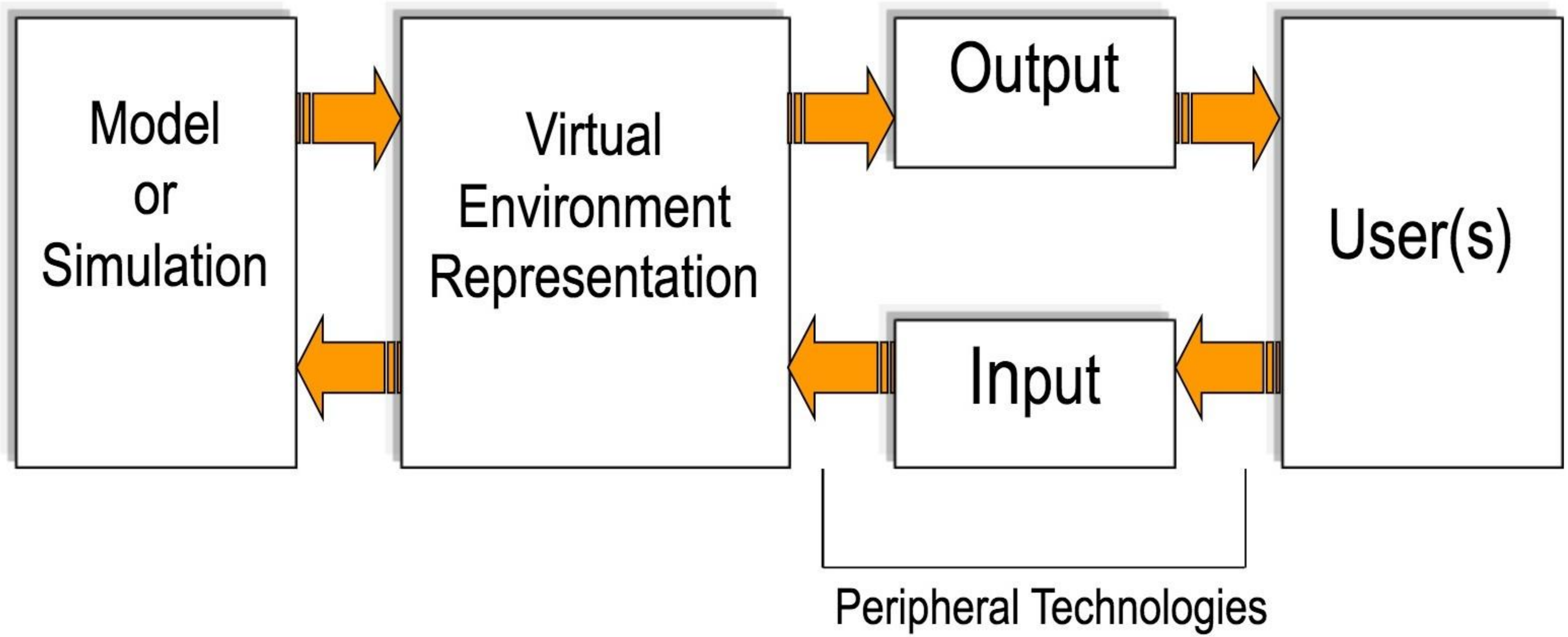
- Ride down the Shard at 100 mph - Multi-sensory VR https://www.youtube.com/watch?v=HNXYoEdBtoU

Key Components to Consider

- Five key components:
  - Inputs
  - Outputs
  - Computation/Simulation
  - Content/World database
  - User interaction

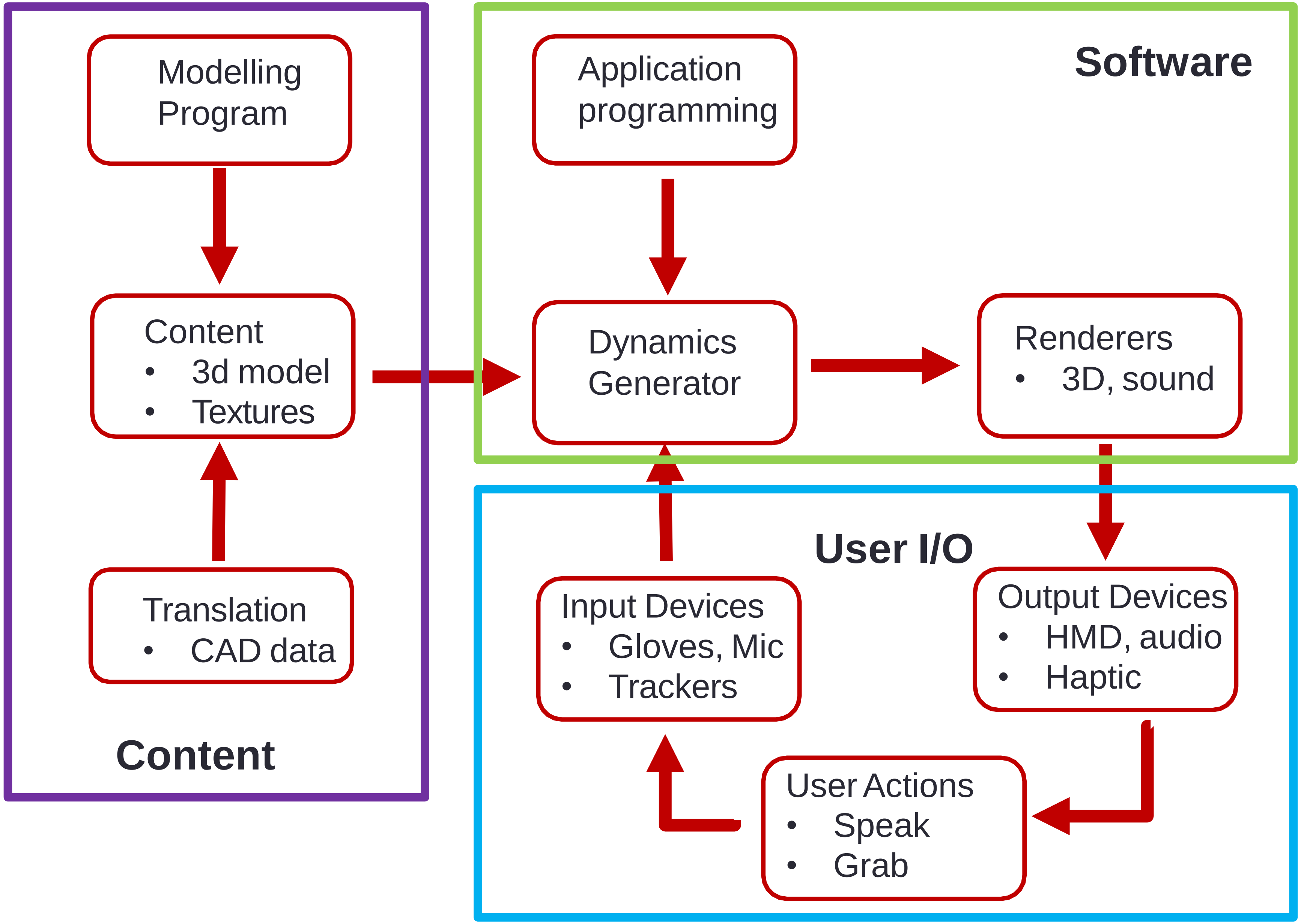
From: Sherman, W. R., & Craig, A. B. (2018). *Understanding virtual reality: Interface, application, and design*. Morgan Kaufmann.

Typical VR System



- Combining multiple technology elements for good user experience
  - Input devices, output modality, content databases, networking, etc.

From Content to User

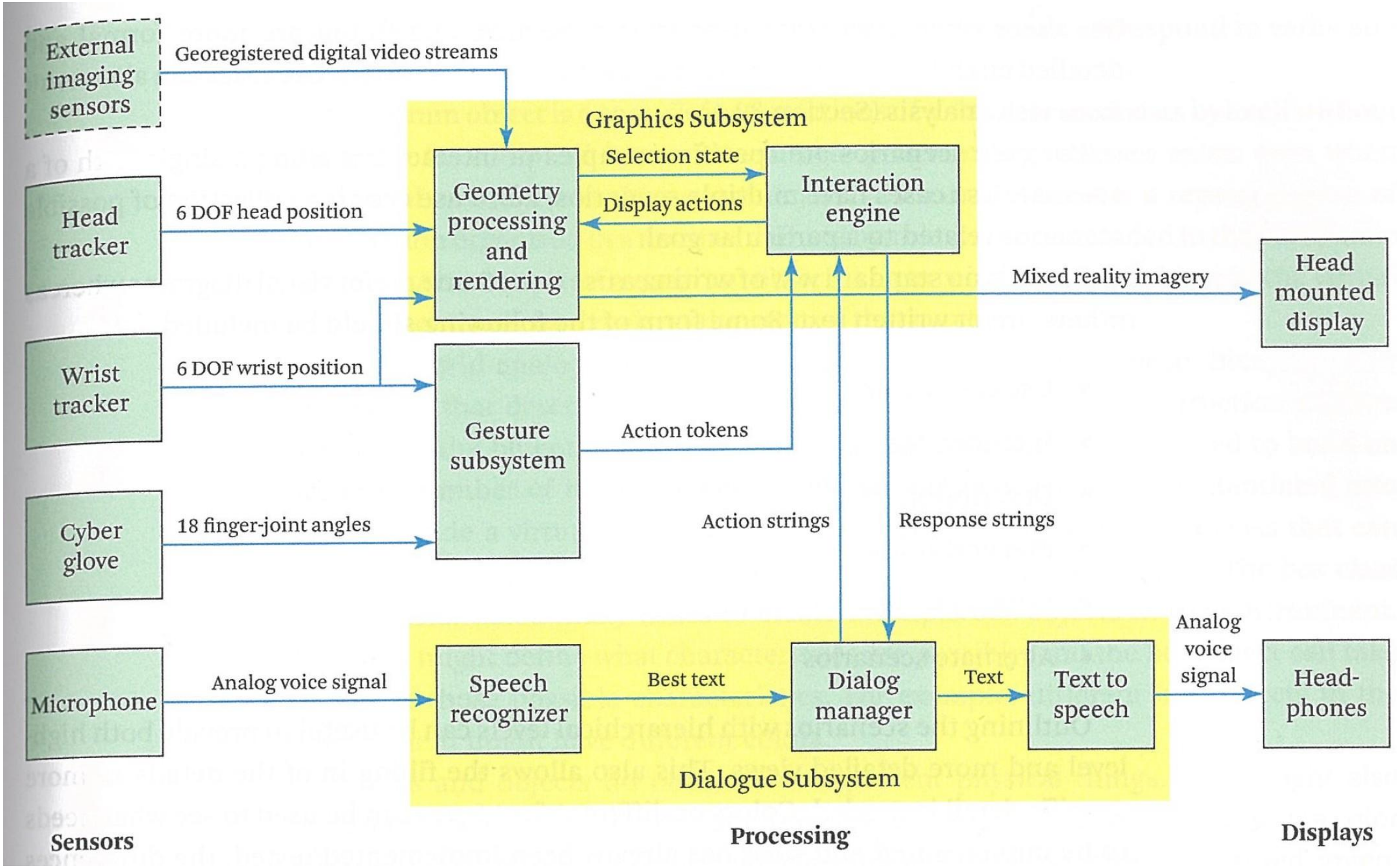


Case Study: Multimodal VR System

- US Army project
  - Simulate control of an unmanned vehicle
- Sensors (input)
  - Head/hand tracking
  - Gesture, Speech (Multimodal)
- Displays (output)
  - HMD, Audio
- Processing
  - Graphics: Virtual vehicles on battlefield
  - Speech processing/understanding

Neely, H. E., Belvin, R. S., Fox, J. R., & Daily, M. J. (2004, March). Multimodal interaction techniques for situational awareness and command of robotic combat entities. In *Aerospace Conference, 2004. Proceedings. 2004 IEEE* (Vol. 5, pp. 3297-3305). IEEE.

System Diagram



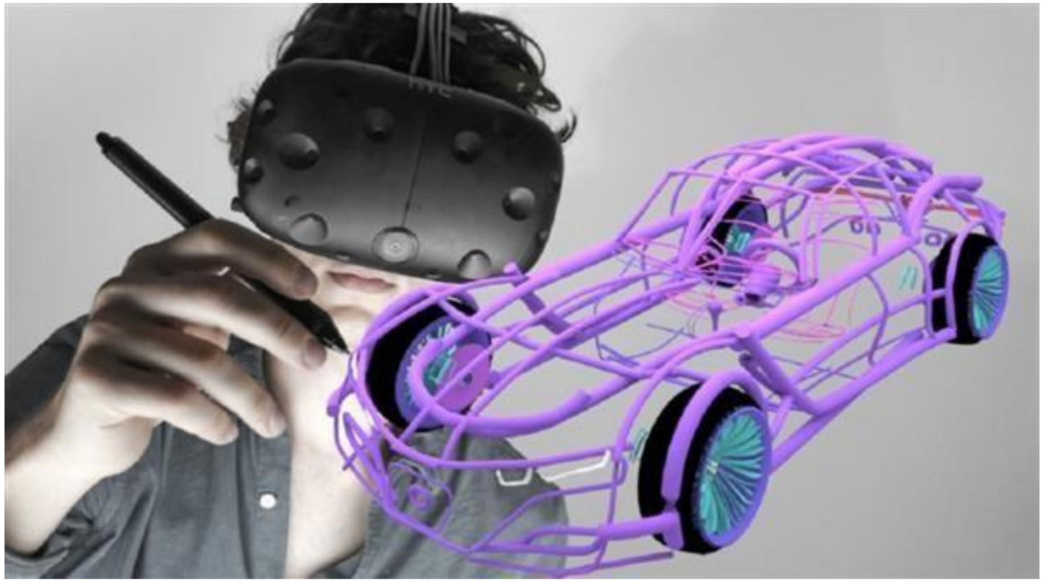
VR CONTENT

Types of VR Experiences

- Immersive Spaces
  - 360 Panorama's/Movies
  - High visual quality
  - Limited interactivity
    - Changing viewpoint orientation
- Immersive Experiences
  - 3D graphics
    - Lower visual quality
  - High interactivity
    - Movement in space
    - Interact with objects

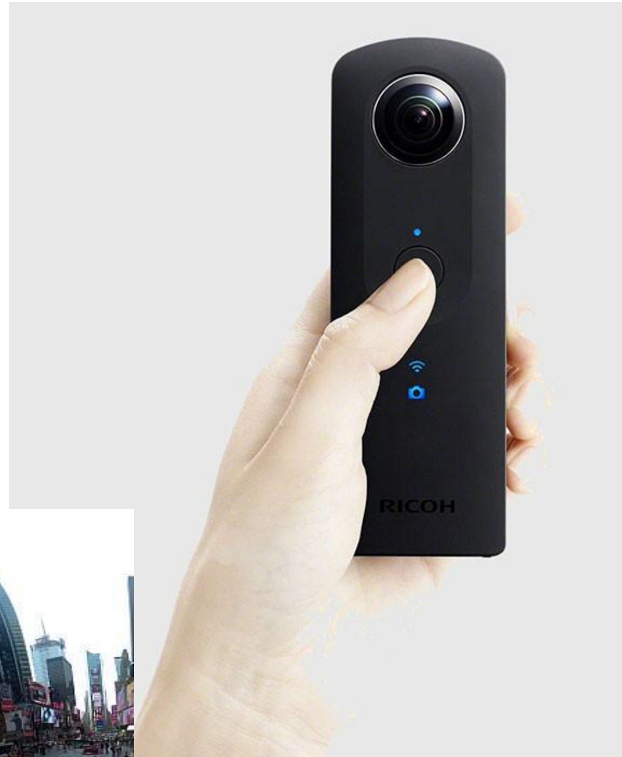
Types of VR Graphics Content

- Panoramas
  - 360 images/video
- Captured 3D content
  - Scanned objects/spaces
- Modelled Content
  - Hand created 3D models
  - Existing 3D assets



Capturing Panoramas

- Stitching individual photos together
  - Image Composite Editor (Microsoft)
  - AutoPano (Kolor)
- Using 360 camera
  - Ricoh Theta-S
  - Fly360



Consumer 360 Capture Devices

Kodac 360

Fly 360

Gear 360

Theta S

Nikon

LG 360

Pointgrey Ladybug

Panono 360

Bublcam

Example: Cardboard Camera

- Capture 360 panoramas
- Stitch together images on phone
- View in VR on Google Cardboard Viewer

Cardboard Camera

- https://www.youtube.com/watch?v=d5lUXZhWaZY

Stereo Video Capture

Vuze

Samsung

- Use camera pairs to capture stereo 360 video
- Samsung 360 round
  - 17 lenses, 4K 3D images, live video streaming, $10K USD
- Vuze+ VR camera
  - 8 lenses, 4K Stereoscopic 3D 360° video and photo, $999 USD

Samsung 360 Round

- https://www.youtube.com/watch?v=X_ytJJOmVF0

3D Scanning

- A range of products support 3D scanning
  - Create point cloud or mesh model
- Typically combine RGB cameras with depth sensing
  - Captures texture plus geometry
- Multi-scale
  - Object Scanners
    - Handheld, Desktop
  - Body Scanners
    - Rotating platform, multi-camera
  - Room scale
    - Mobile, tripod mounted

Example: Matterport

- Matterport Pro2 3D scanner
  - Room scale scanner, panorama and 3D model
  - 360° (left-right) x 300° (vertical) field of view
  - Structured light (infared) 3D sensor
  - 15 ft (4.5 m) maximum range
  - 4K HDR images

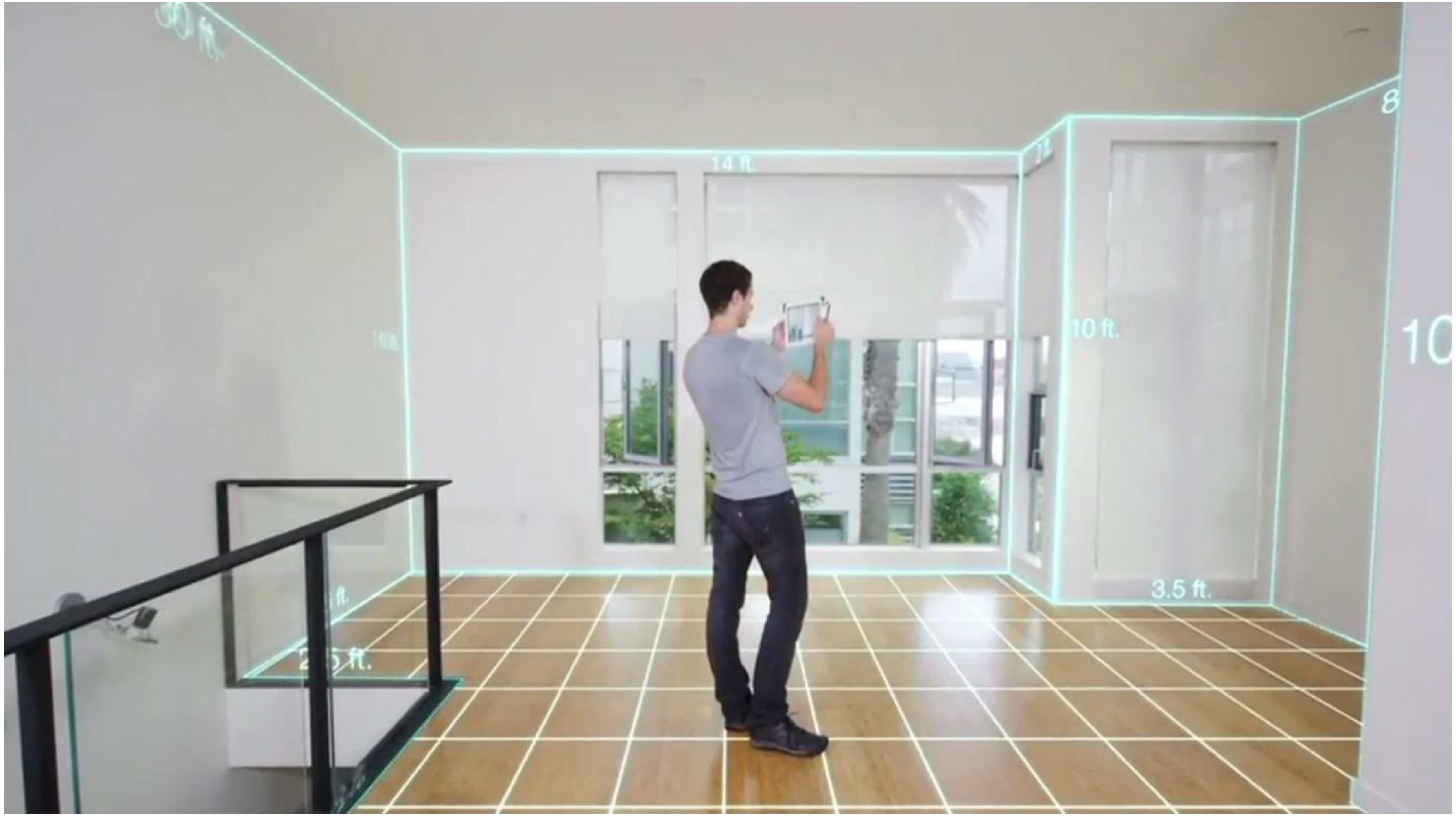
Matterport Pro2 Lite

- https://www.youtube.com/watch?v=SjHk0Th-j1I

Handheld/Desktop Scanners

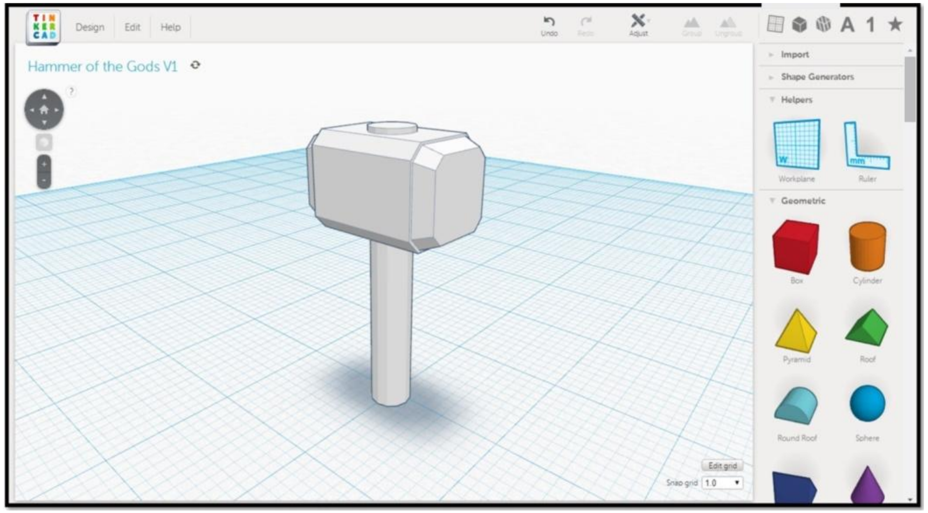
- Capture people/objects
- Sense 3D scanner
  - accuracy of 0.90 mm, colour resolution of 1920×1080 pixels
- Occipital Structure sensor
  - Add-on to iPad, mesh scanning, IR light projection, 60 Hz

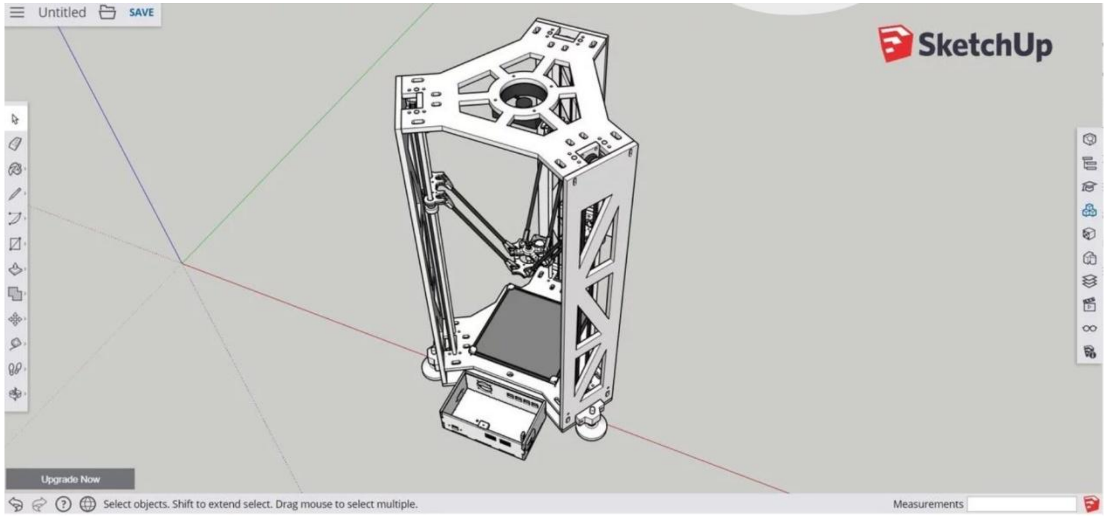
Structure Sensor



- https://www.youtube.com/watch?v=7j3HQxUGvq4

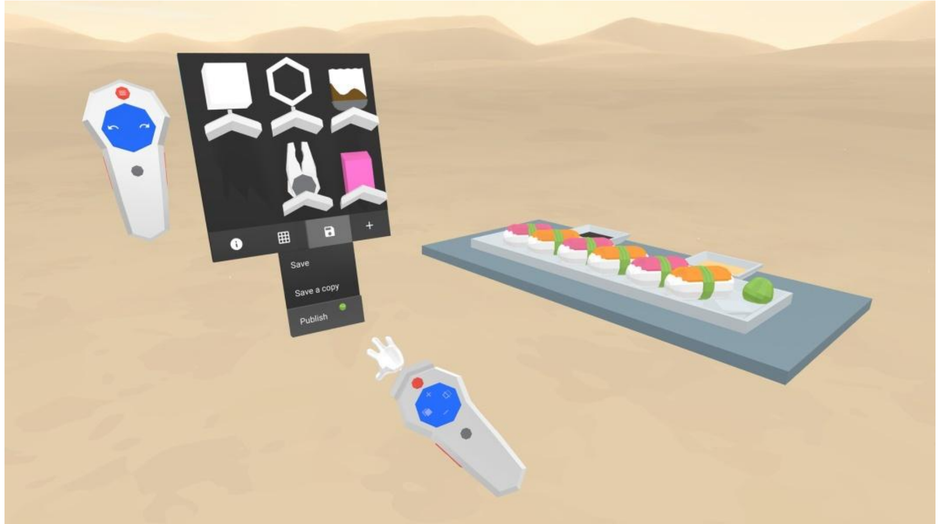
3D Modelling





- A variety of 3D modelling tools can be used
  - Export in VR compatible file format (.obj, .fbx, etc)
  - Especially useful for animation - difficult to create from scans
- Popular tools
  - Blender (free), 3DS max, Maya, etc.
- Easy to Use
  - Tinkercad, Sketchup Free, Meshmixer, Fusion 360, etc.

Modelling in VR

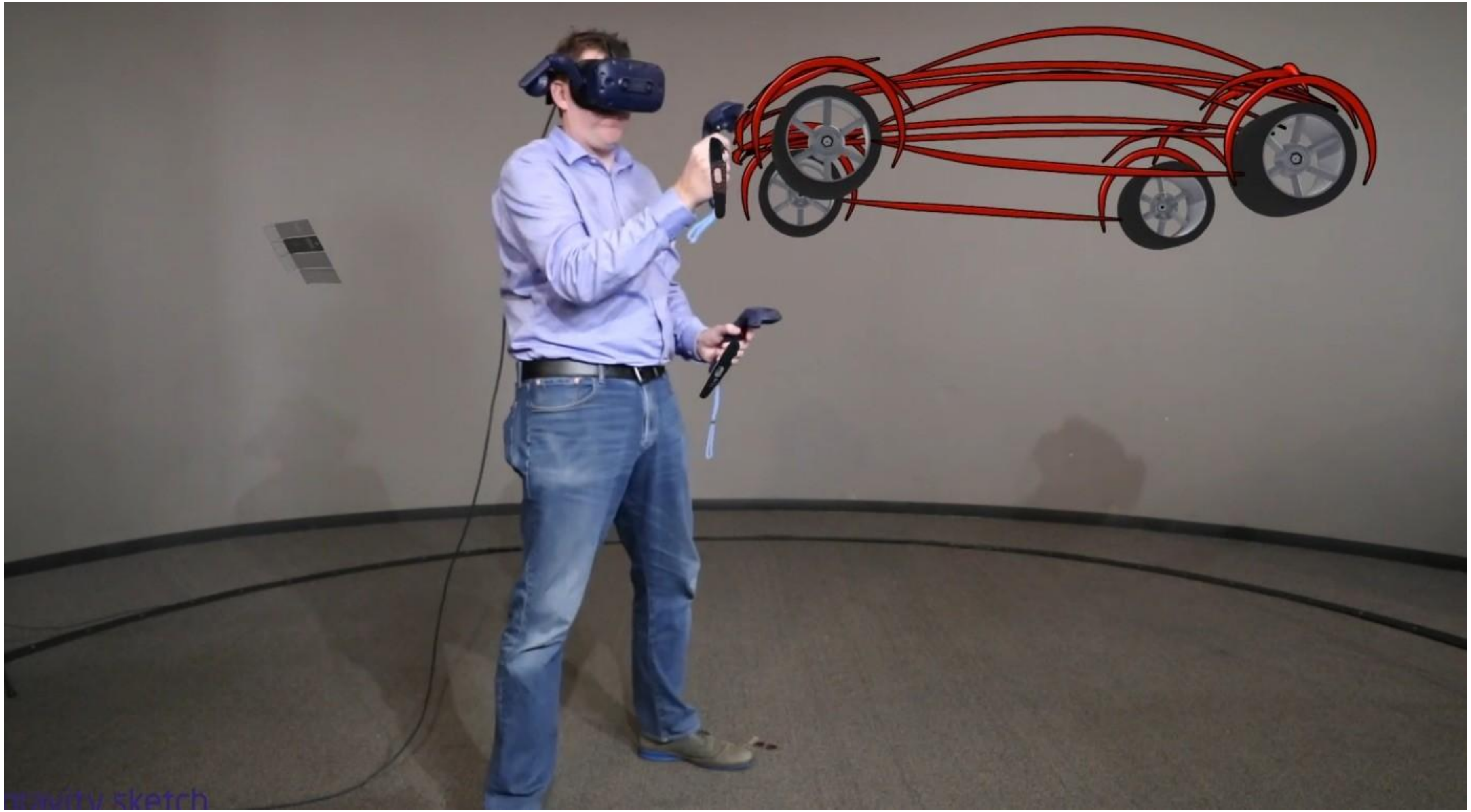


- Several tools for modelling in VR
  - Natural interaction, low polygon count, 3D object viewins
- Low end
  - Google Blocks
- High end
  - Quill, Tilt brush – 3D painting
  - Gravity Sketch – 3D CAD

Example: Google Blocks

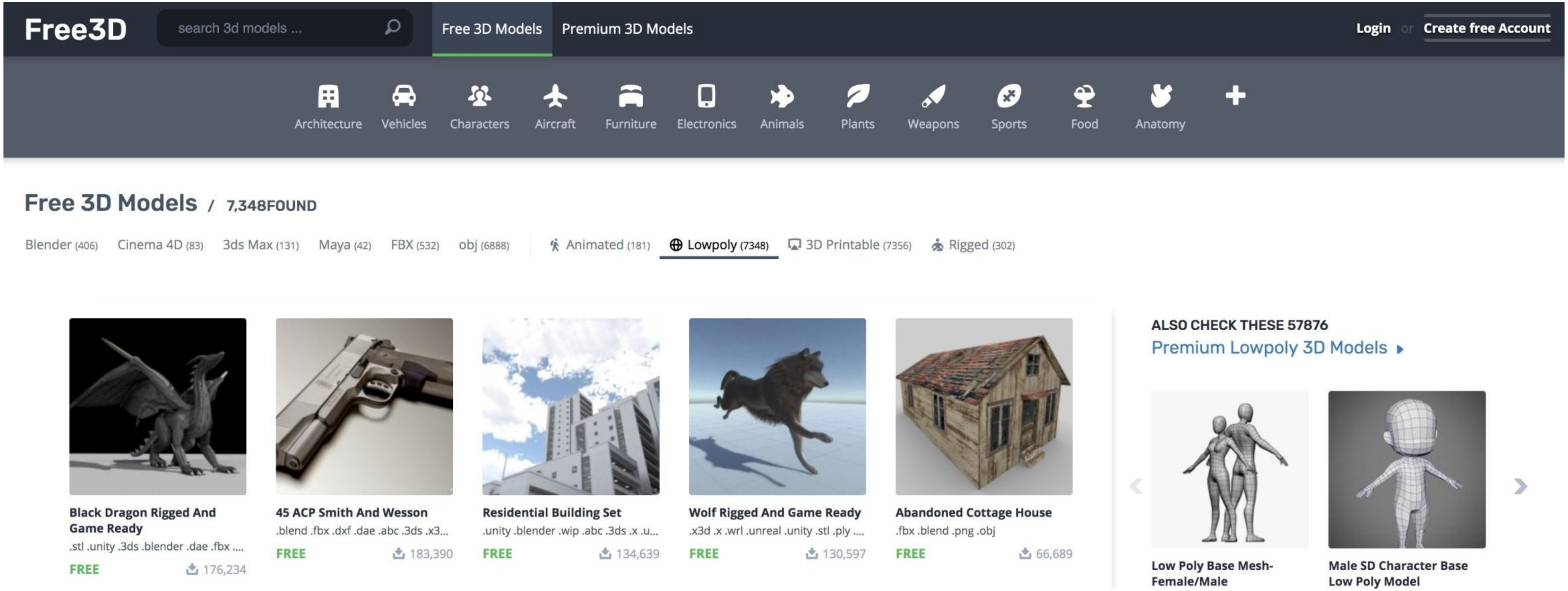
- https://www.youtube.com/watch?v=1TX81cRqfUU

Example: Gravity Sketch



- https://www.youtube.com/watch?v=VK2DDnT_3l0

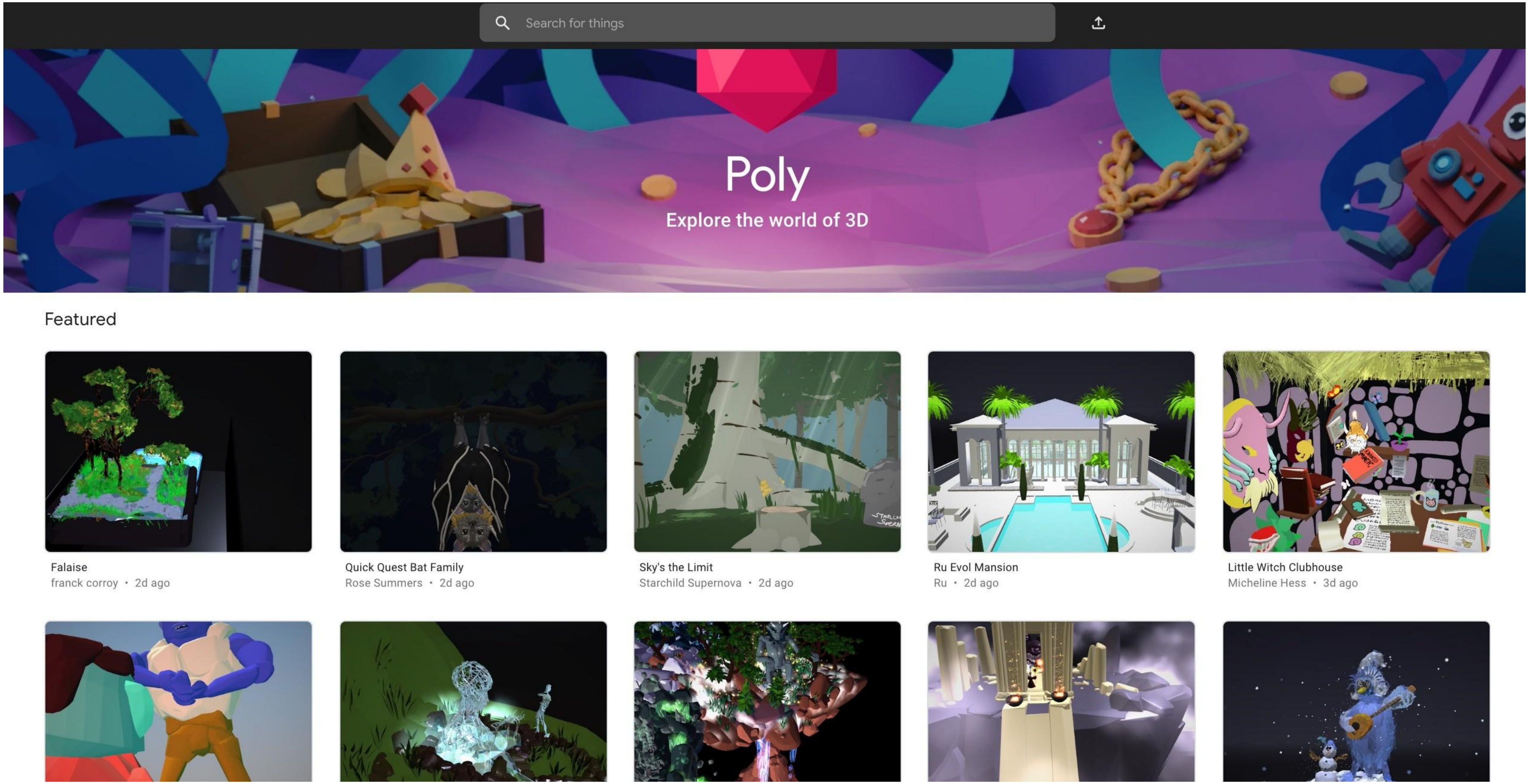
Download Existing VR Content



- Many locations for 3D objects, textures, etc.
  - Google Poly - Low polygon VR ready models
  - Sketchfab, Sketchup, Free3D (www.free3d.com), etc.
- Asset stores - Unity, Unreal
  - Provide 3D models, materials, code, etc..

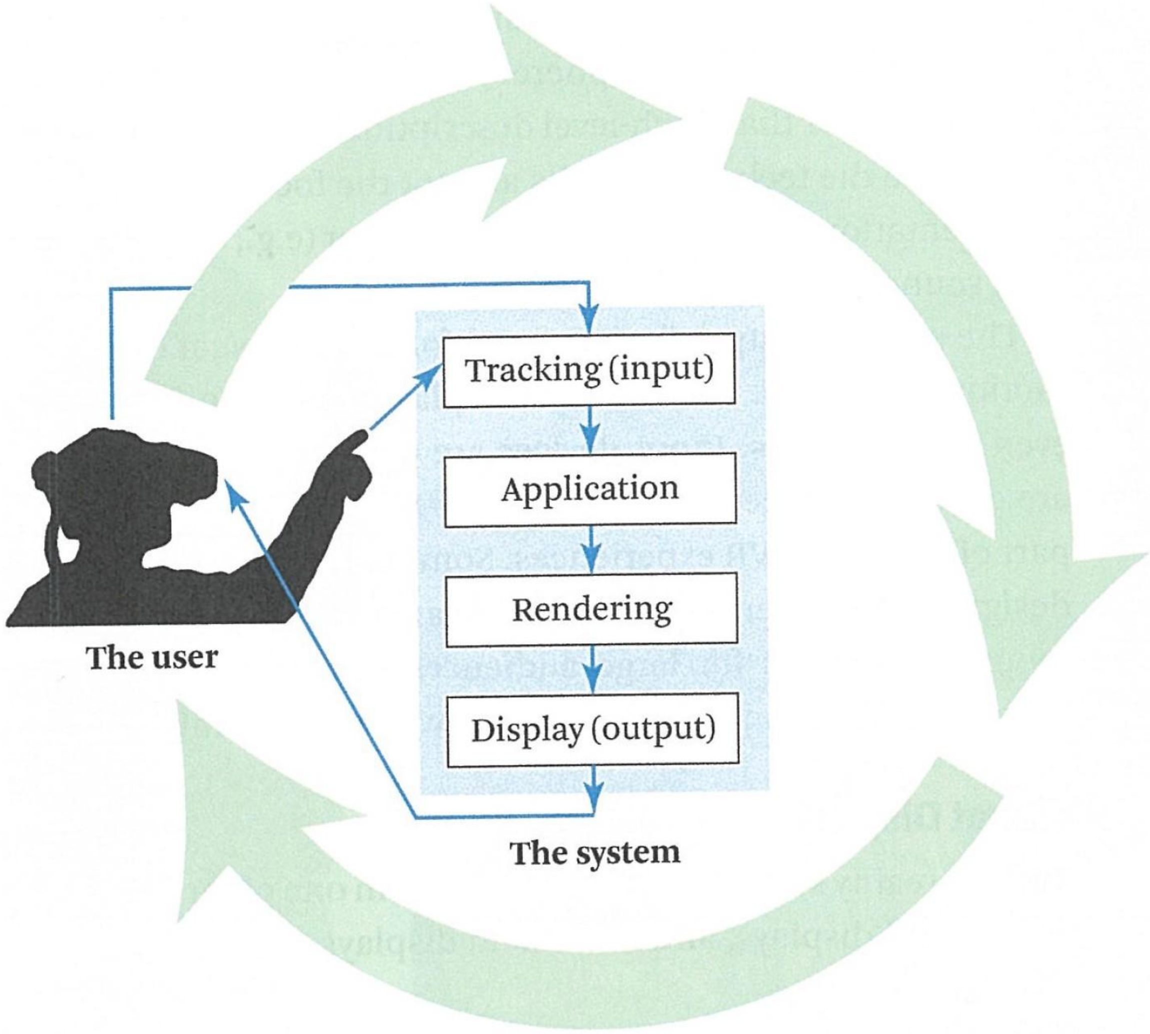
Google Poly

- https://poly.google.com/ - search for models you’d like



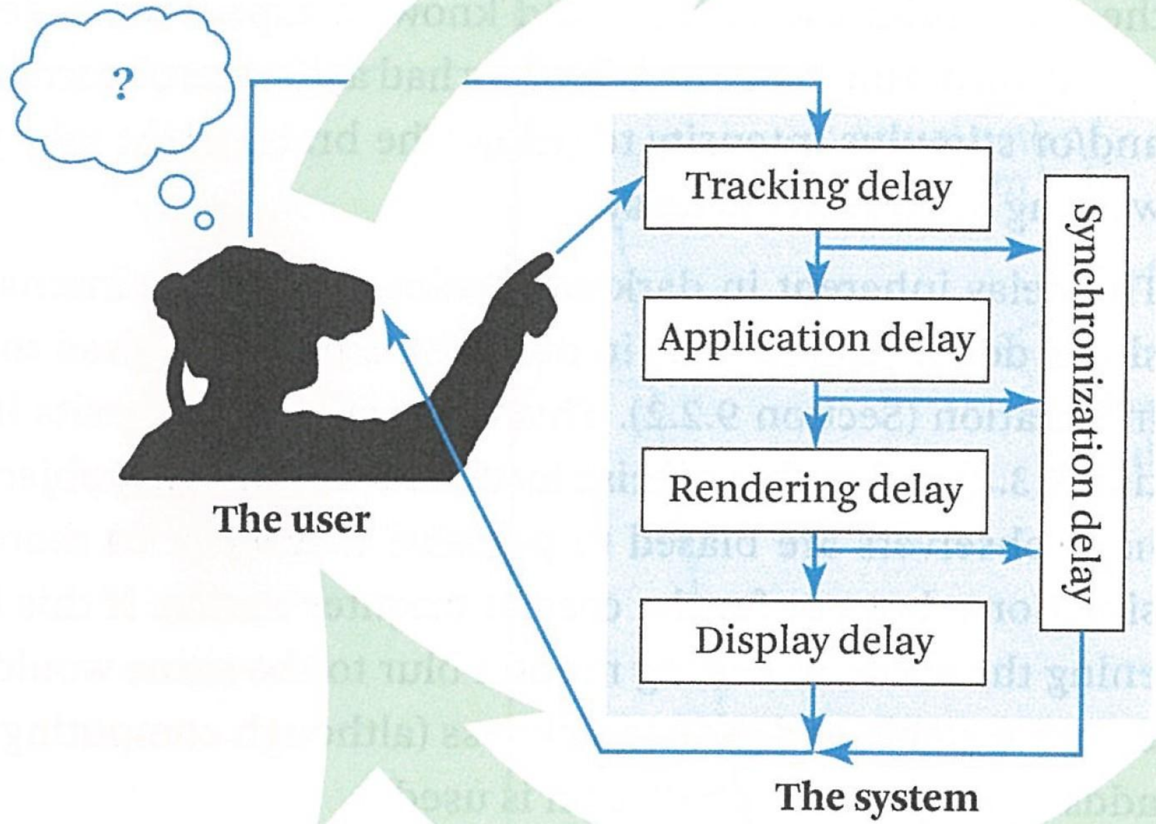
SIMULATION

Typical VR Simulation Loop



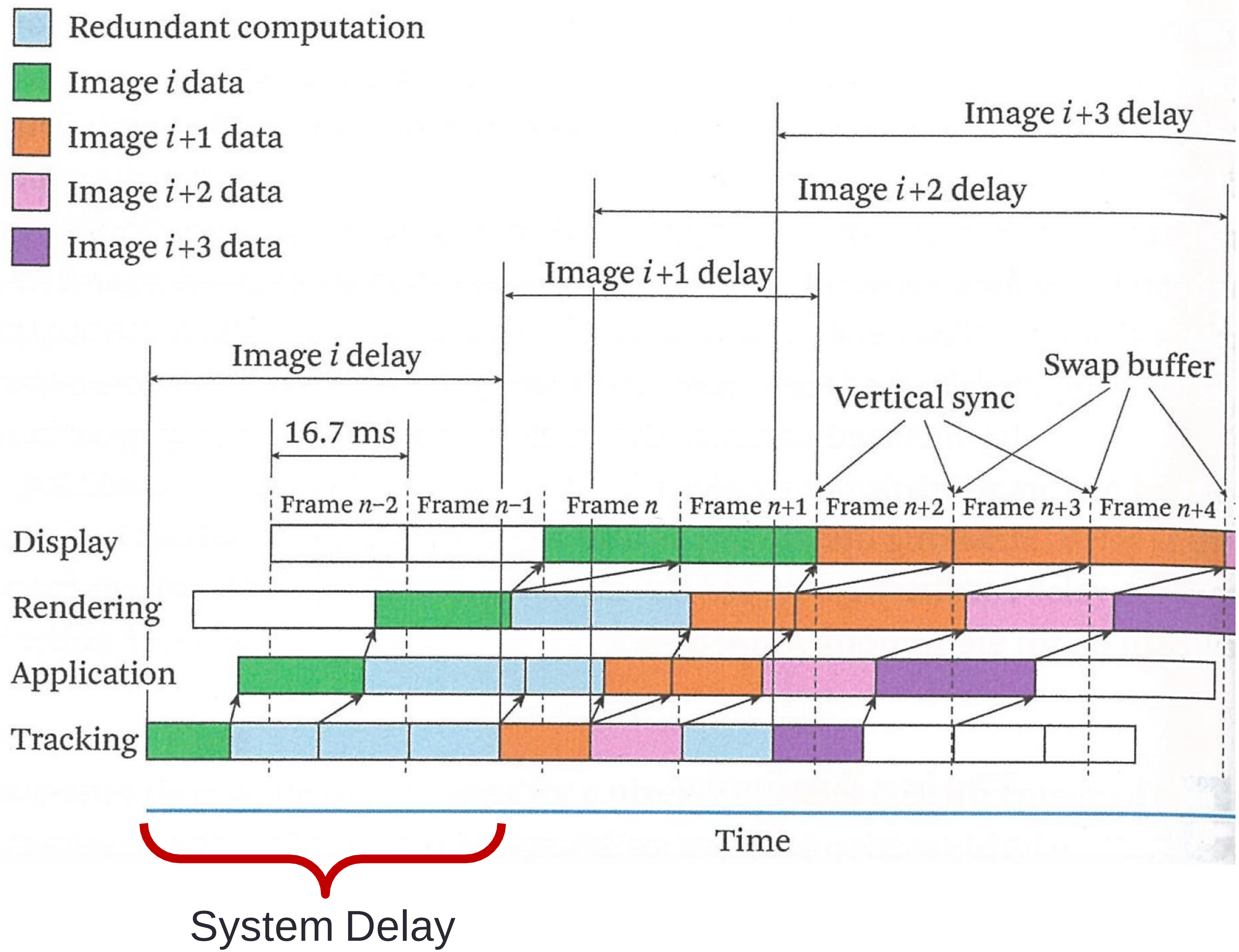
- User moves head, scene updates, displayed graphics change

System Delays

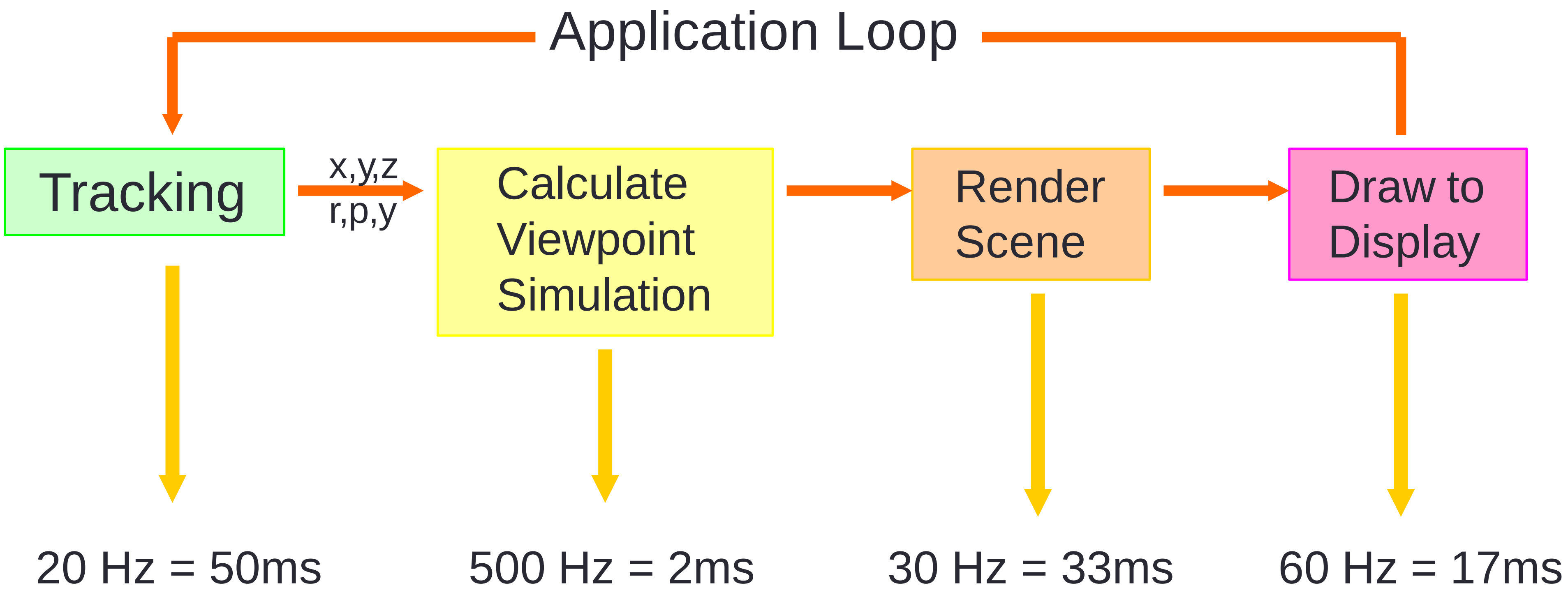


- Need to synchronize system to reduce delays

Typical Delay from Tracking to Rendering



Typical System Delays



- Total Delay = 50 + 2 + 33 + 17 = 102 ms
  - 1 ms delay = 1/3 mm error for object drawn at arms length
  - So total of 33mm error from when user begins moving to when object drawn

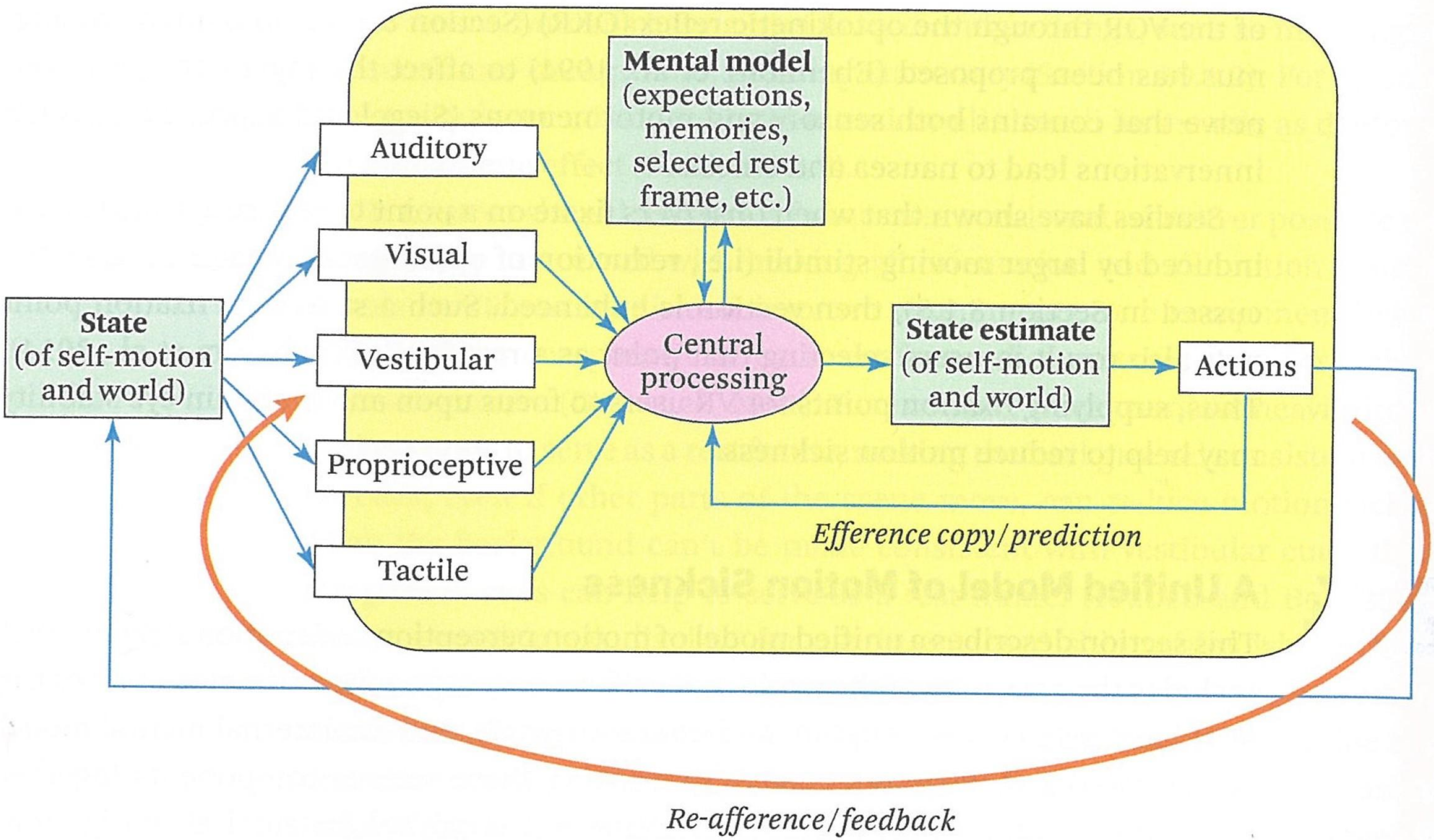
Living with High Latency (1/3 sec – 3 sec)

- https://www.youtube.com/watch?v=_fNp37zFn9Q

Effects of System Latency

- Degraded Visual Acuity
  - Scene still moving when head stops = motion blur
- Degraded Performance
  - As latency increases it’s difficult to select objects etc.
  - If latency > 120 ms, training doesn’t improve performance
- Breaks-in-Presence
  - If system delay high user doesn’t believe they are in VR
- Negative Training Effects
  - User train to operative in world with delay
- Simulator Sickness
  - Latency is greatest cause of simulator sickness

Simulator Sickness

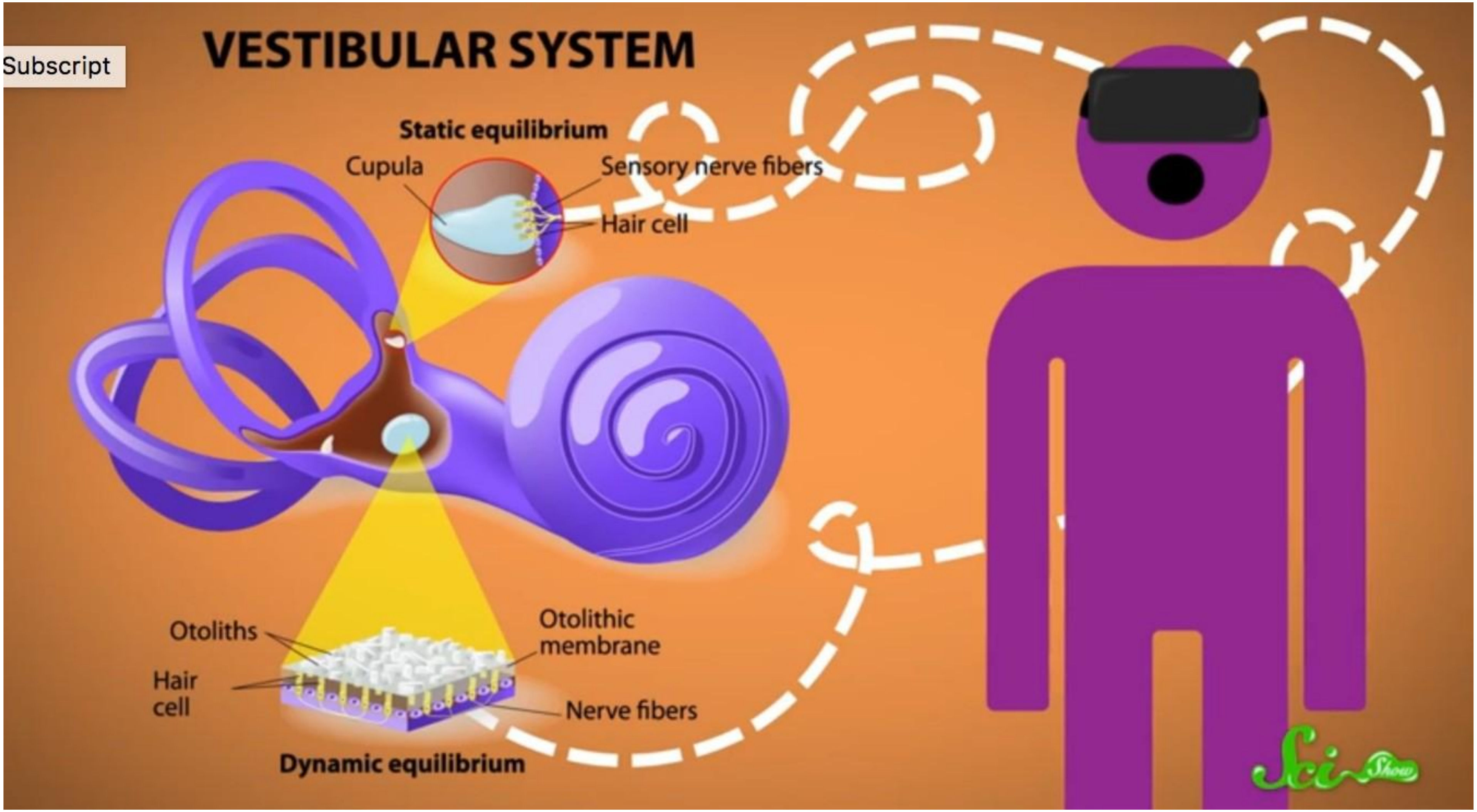


- Visual input conflicting with vestibular system

Many Causes of Simulator Sickness

- 25-40% of VR users get Simulator Sickness, due to:
- Latency
  - Major cause of simulator sickness
- Tracking accuracy/precision
  - Seeing world from incorrect position, viewpoint drift
- Field of View
  - Wide field of view creates more periphery vection = sickness
- Refresh Rate/Flicker
  - Flicker/low refresh rate creates eye fatigue
- Vergence/Accommodation Conflict
  - Creates eye strain over time
- Eye separation
  - If IPD not matching to inter-image distance then discomfort

Motion Sickness



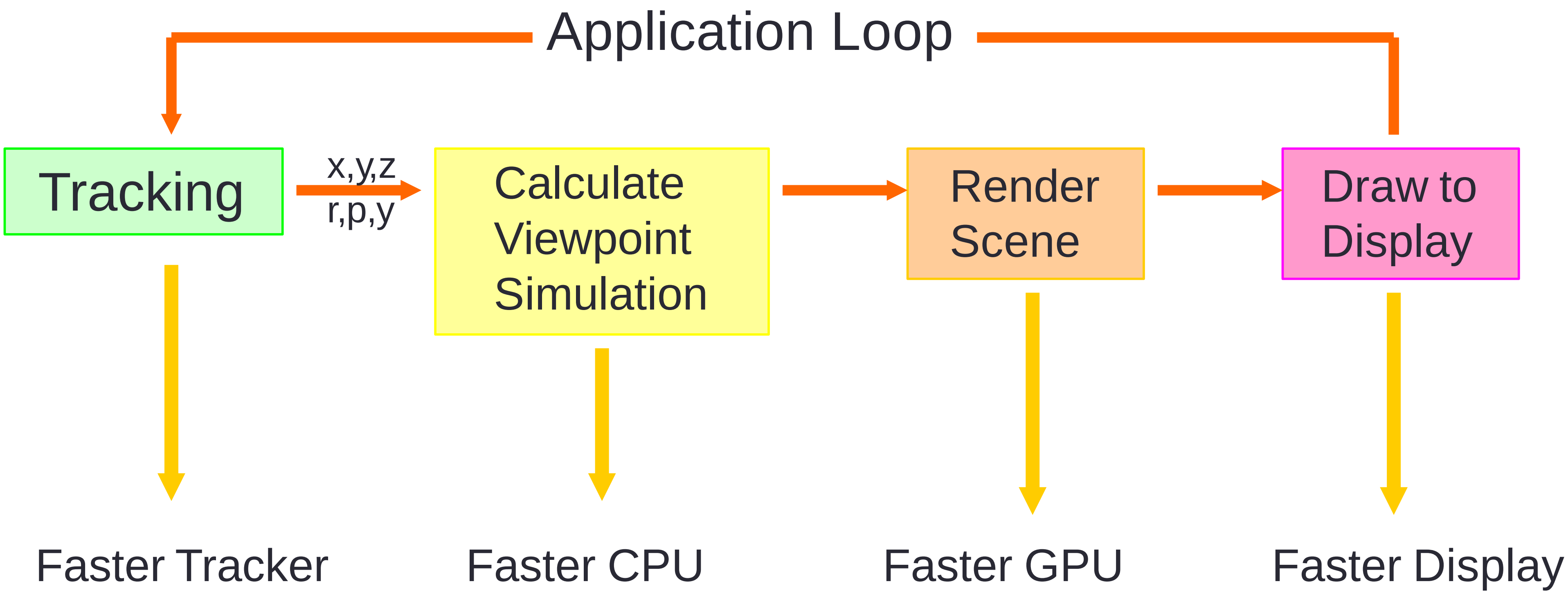
- https://www.youtube.com/watch?v=BznbIlW8iqE

How to Reduce System Delays

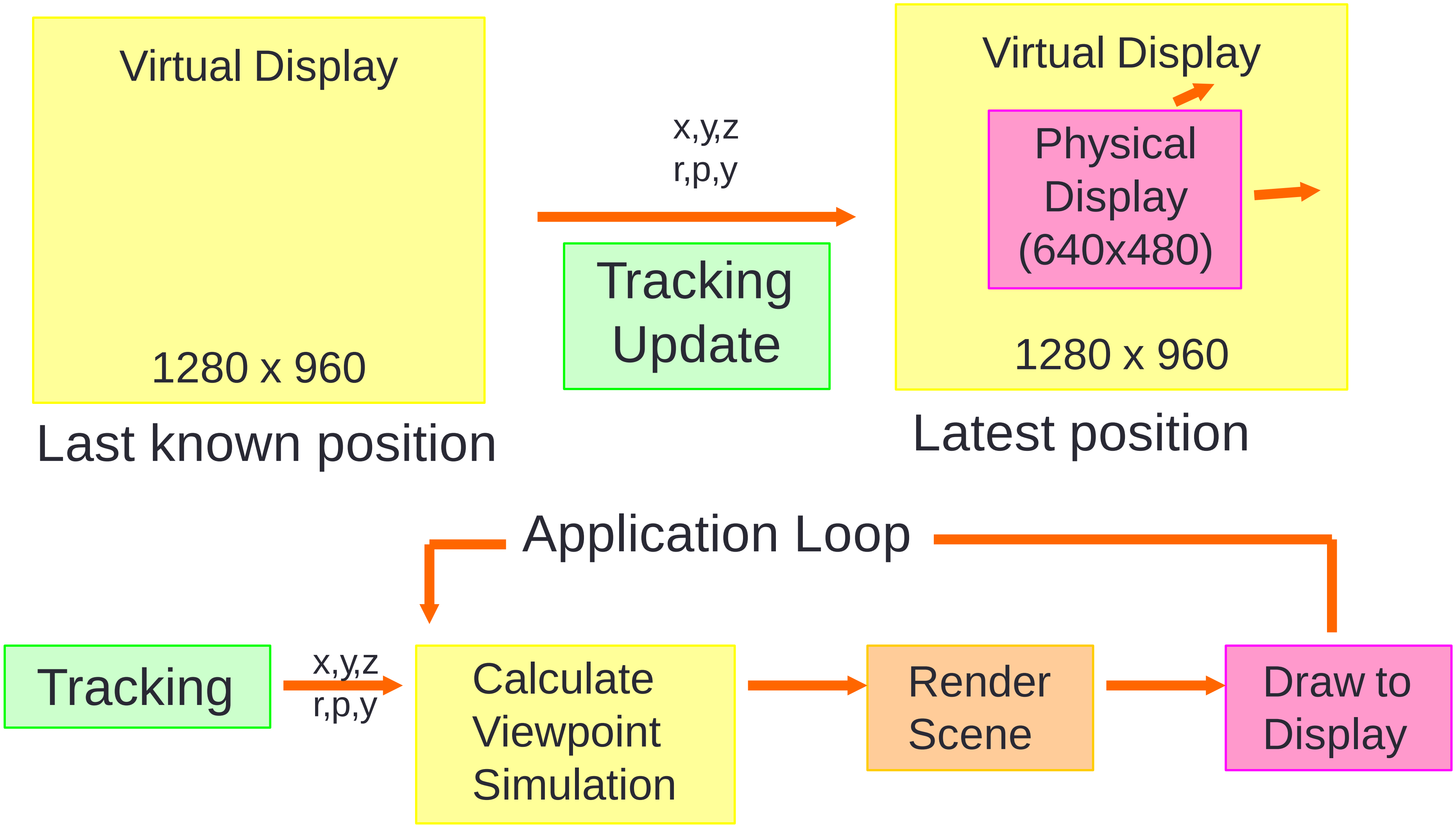
- Use faster components
  - Faster CPU, display, etc.
- Reduce the apparent lag (Time Warp)
  - Take tracking measurement just before rendering
  - Remove tracker from the loop
- Use predictive tracking
  - Use fast inertial sensors to predict where user will be looking
  - Difficult due to erratic head movements

Jerald, J. (2004). *Latency compensation for head-mounted virtual reality*. UNC Computer Science Technical Report.

Reducing System Lag

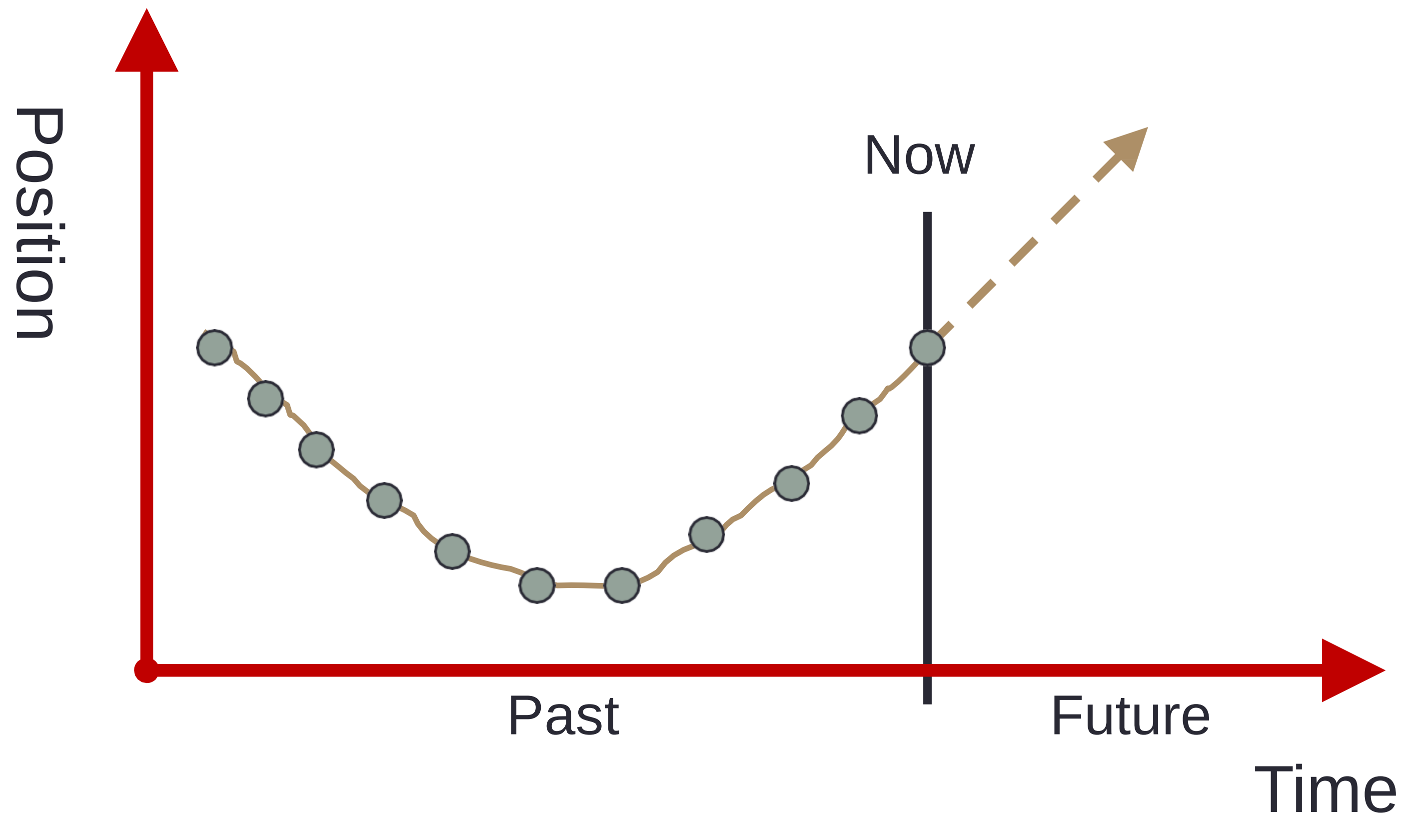


Reducing Apparent Lag (TimeWarp)



Create virtual display large than physical display and move at last minute

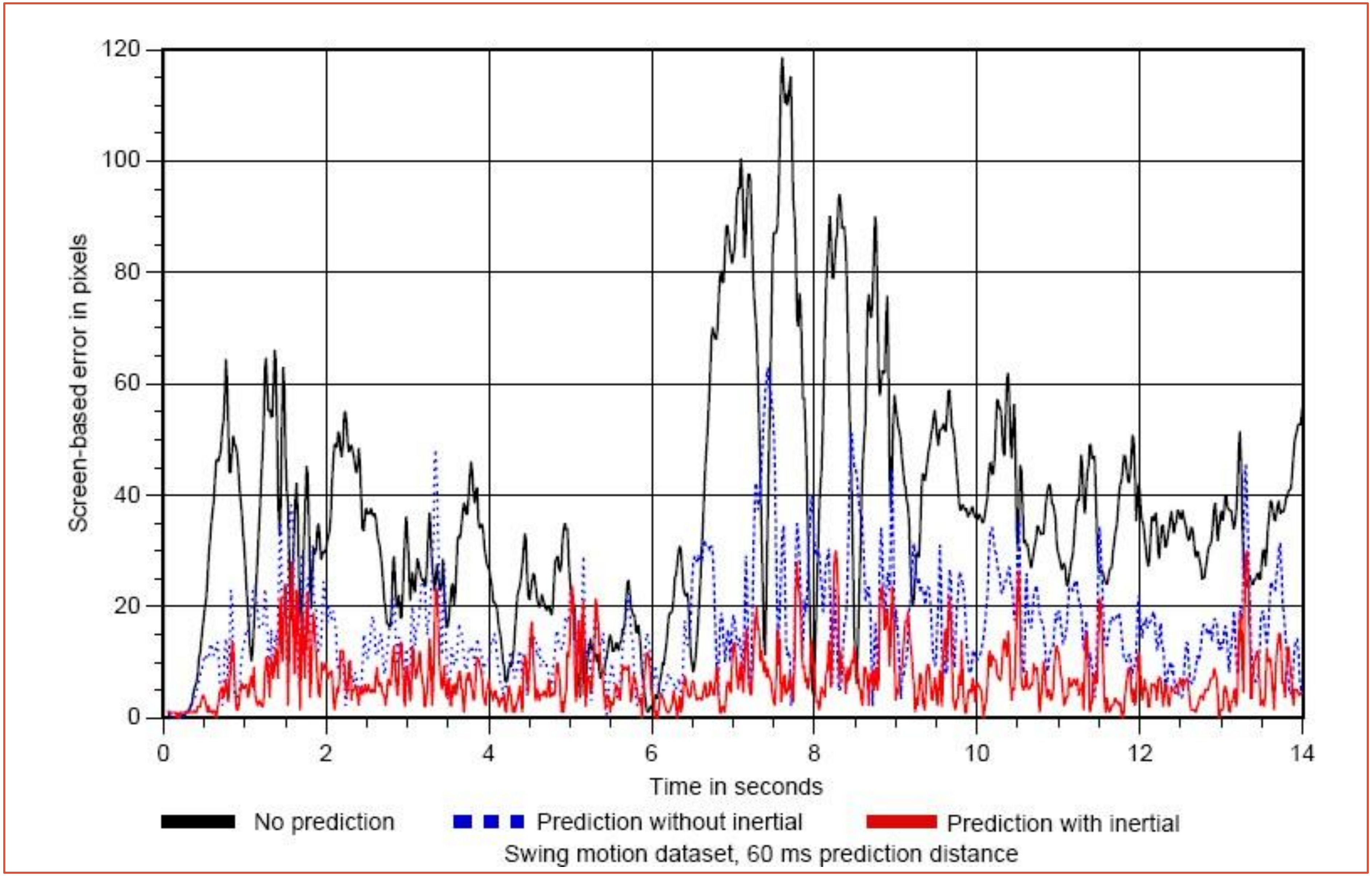
Predictive Tracking for Reducing Latency



Use additional sensors (e.g. inertial) to predict future position

- Can reliably predict up to 80 ms in future (Holloway)
- Use Kalman filters or similar to smooth prediction

Predictive Tracking Reduces Error (Azuma 94)

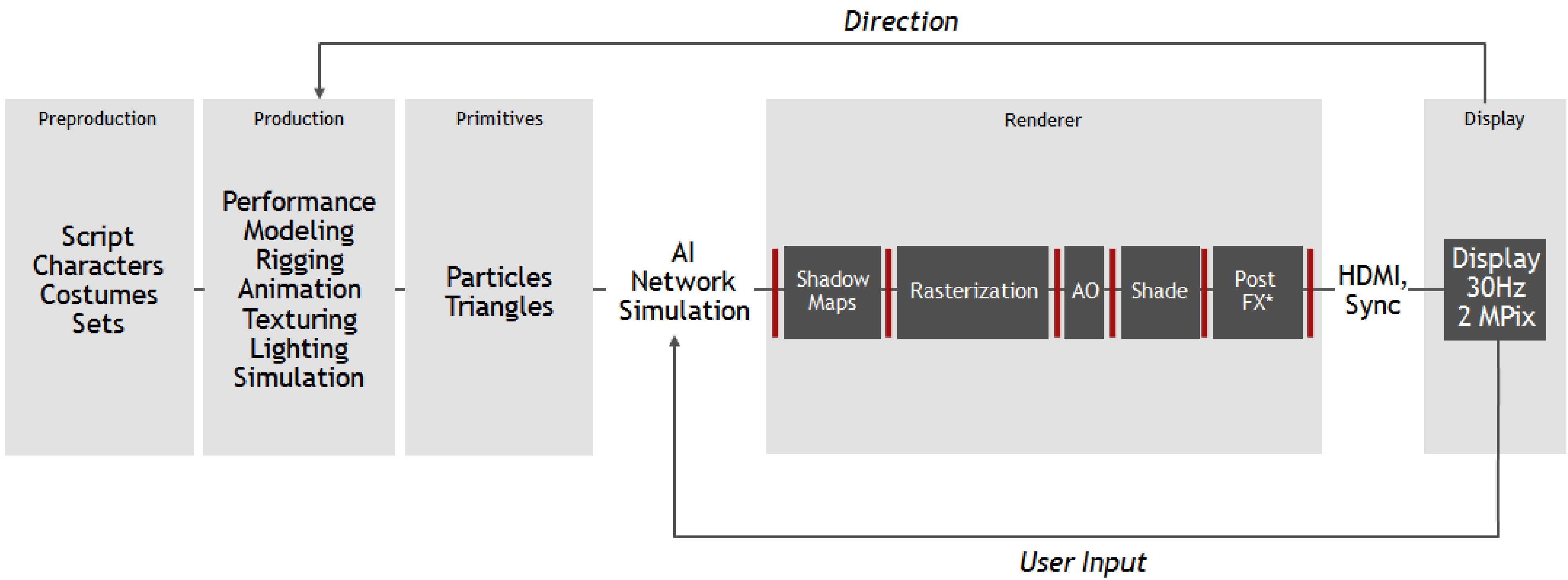


GRAPHICS

VR Graphics Architecture/Tools

- Rendering Layer (GPU acceleration) [OpenGL]
  - Low level graphics code
  - Rendering pixels/polygons
  - Interface with graphics card/frame buffer
- Graphics Layer (CPU acceleration) [X3D, OSG]
  - Scene graph specification
  - Object physics engine
  - Specifying graphics objects
- Application Layer [Unity, Unreal]
  - User interface libraries
  - Simulation/behaviour code
  - User interaction specification

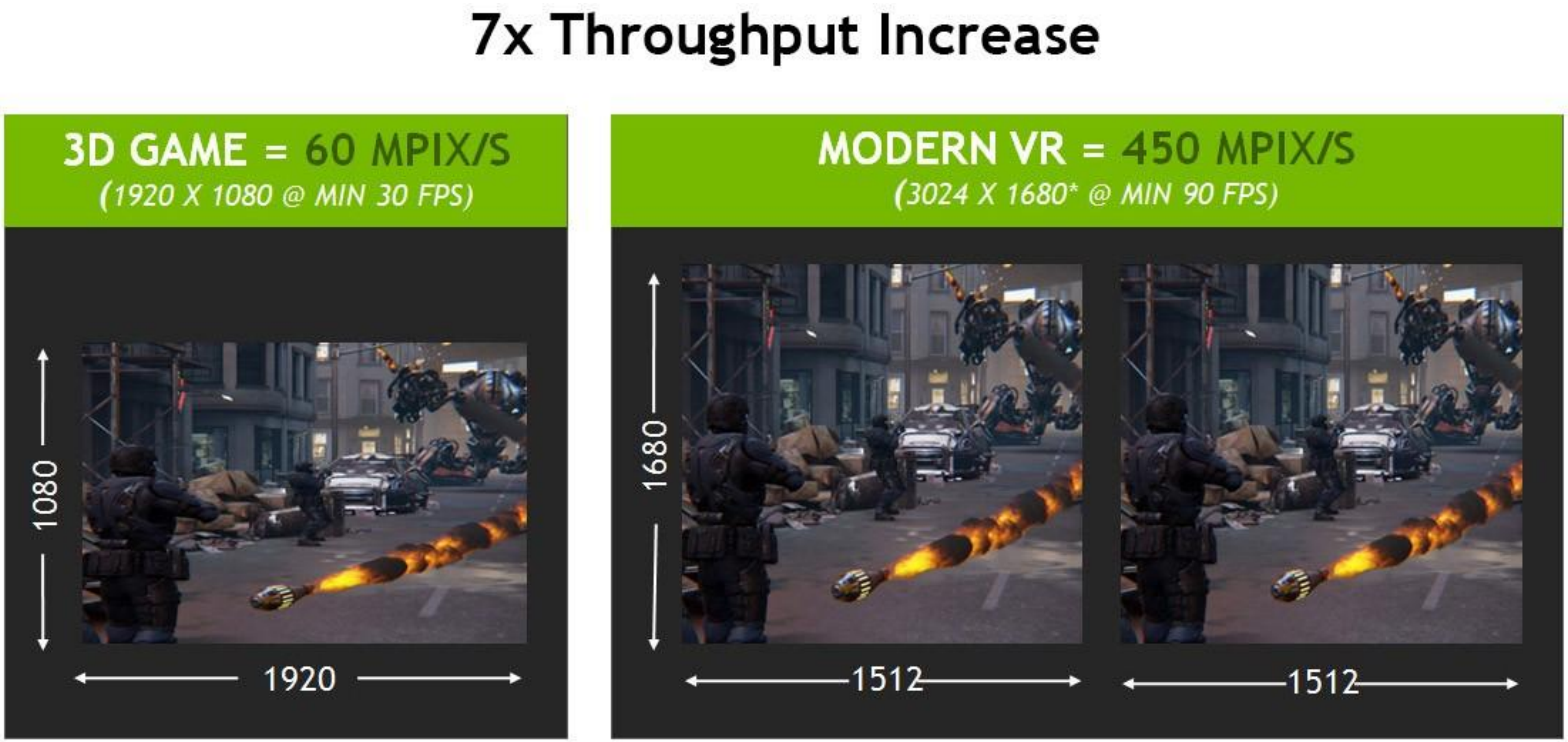
Traditional 3D Graphics Pipeline



* *Includes depth of field, reflections, fog, color grading, motion blur, antialiasing*

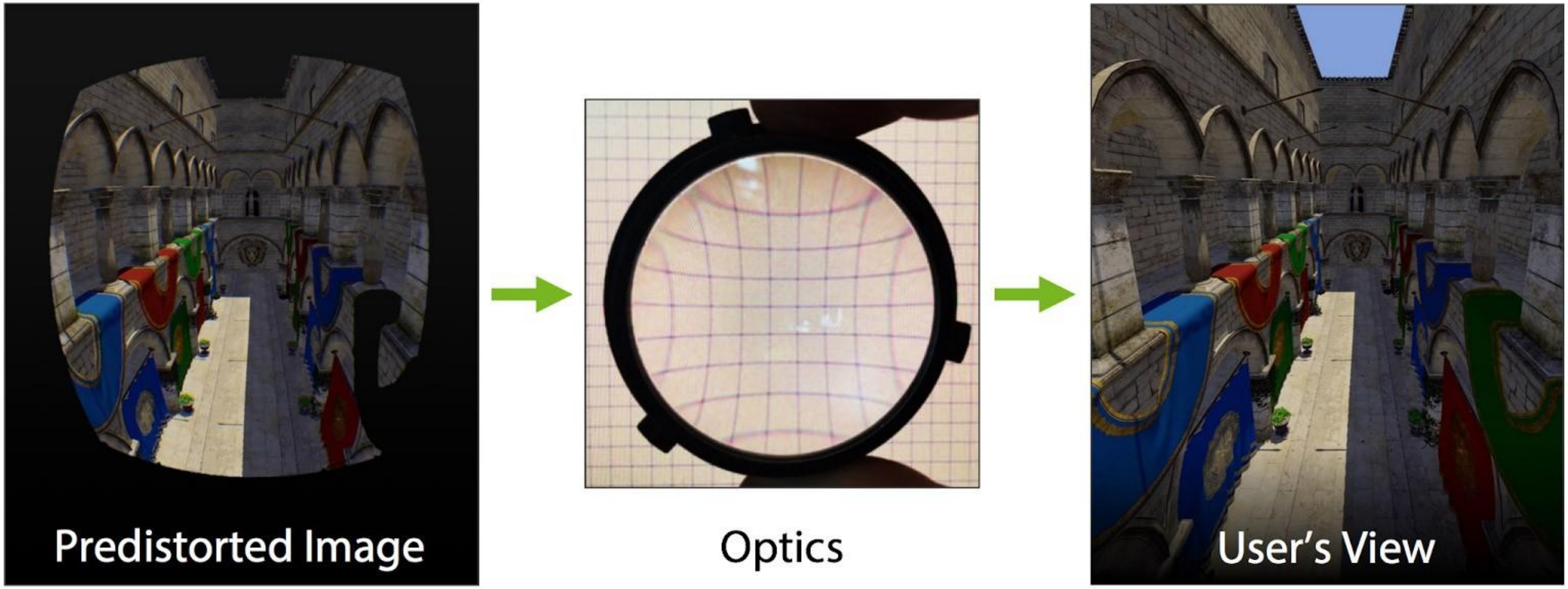
- Low level code for loading models and showing on screen
  - Using shaders and low level GPU programming to improve graphics

Graphics Challenges with VR



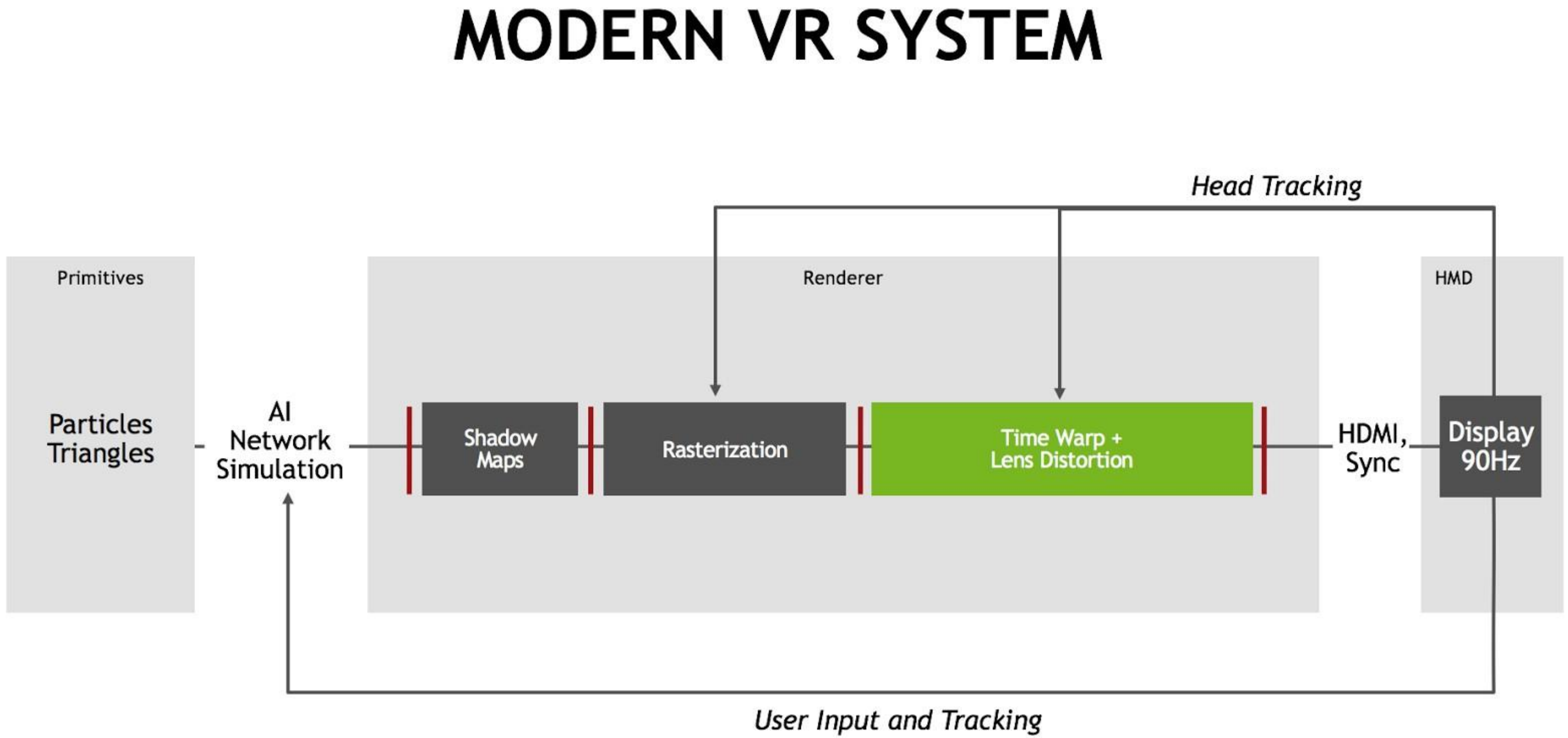
- Higher data throughput (> 7x desktop requirement)
- Lower latency requirements (from 150ms/frame to 20ms)
- HMD Lens distortion

Lens Distortion



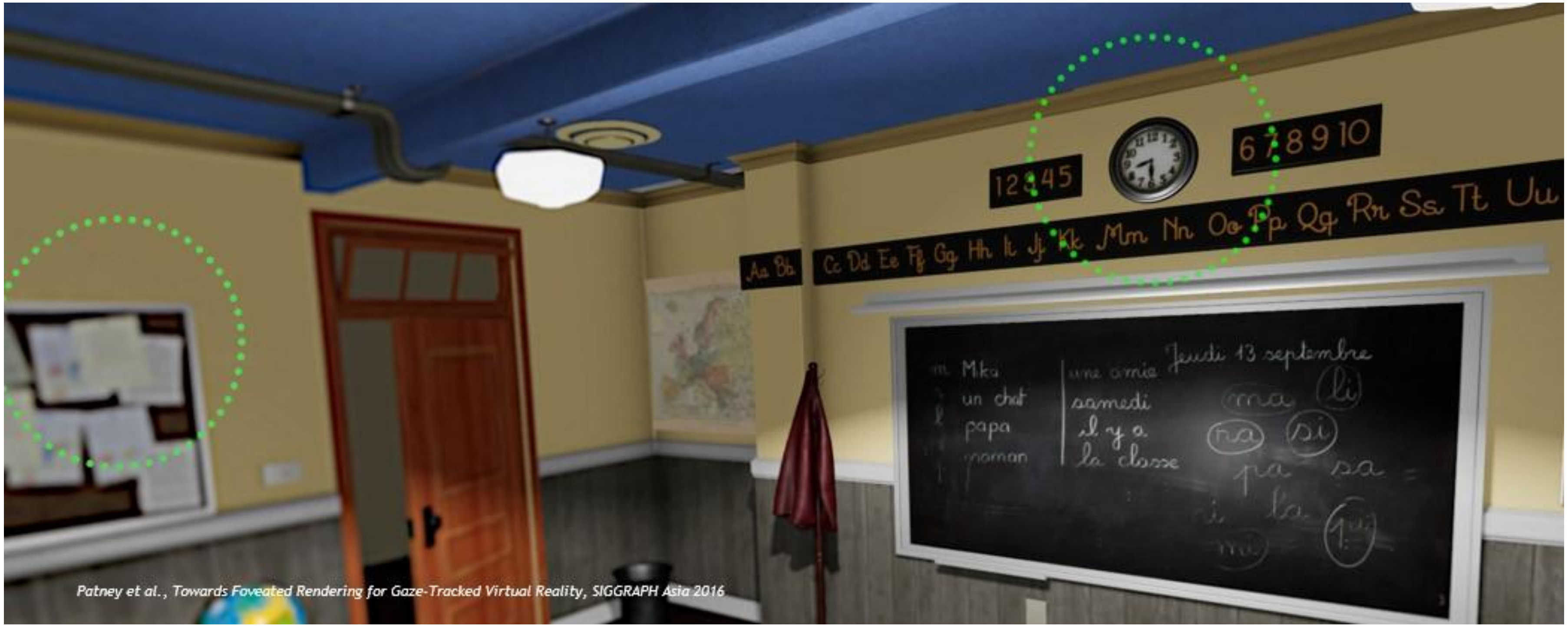
- HMD may have cheap lens
  - Creates chromatic aberration and distorted image
- Warp graphics images to create undistorted view
  - Use low level shader programming

VR System Pipeline



- Using time warping and lens distortion

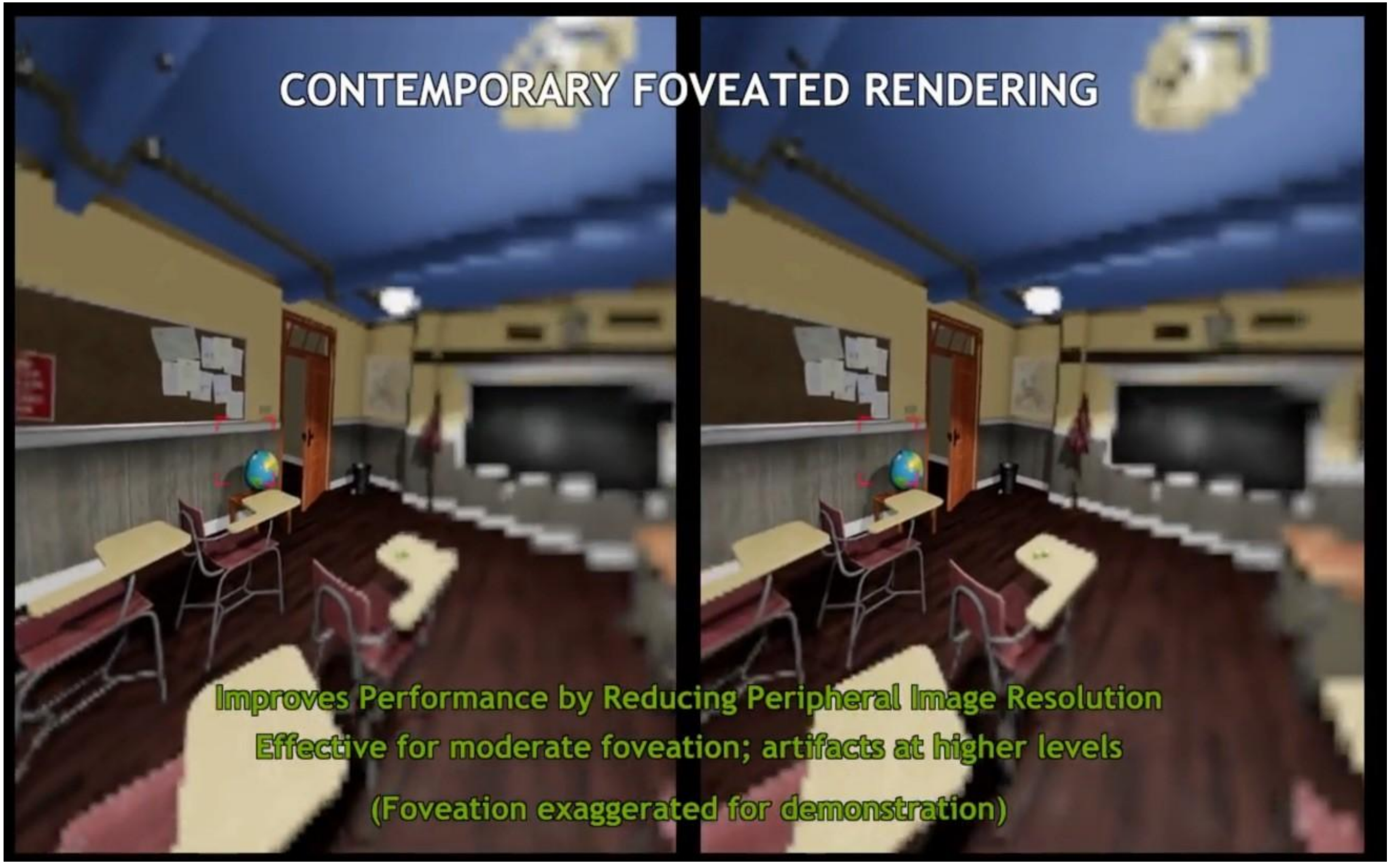
Perception Based Graphics



Patney et al., *Towards Foveated Rendering for Gaze-Tracked Virtual Reality*, SIGGRAPH Asia 2016

- Eye Physiology
  - Rods in eye centre = colour vision, cones in periphery = motion, B+W
- Foveated Rendering
  - Use eye tracking to draw highest resolution where user looking
  - Reduces graphics throughput

Foveated Rendering

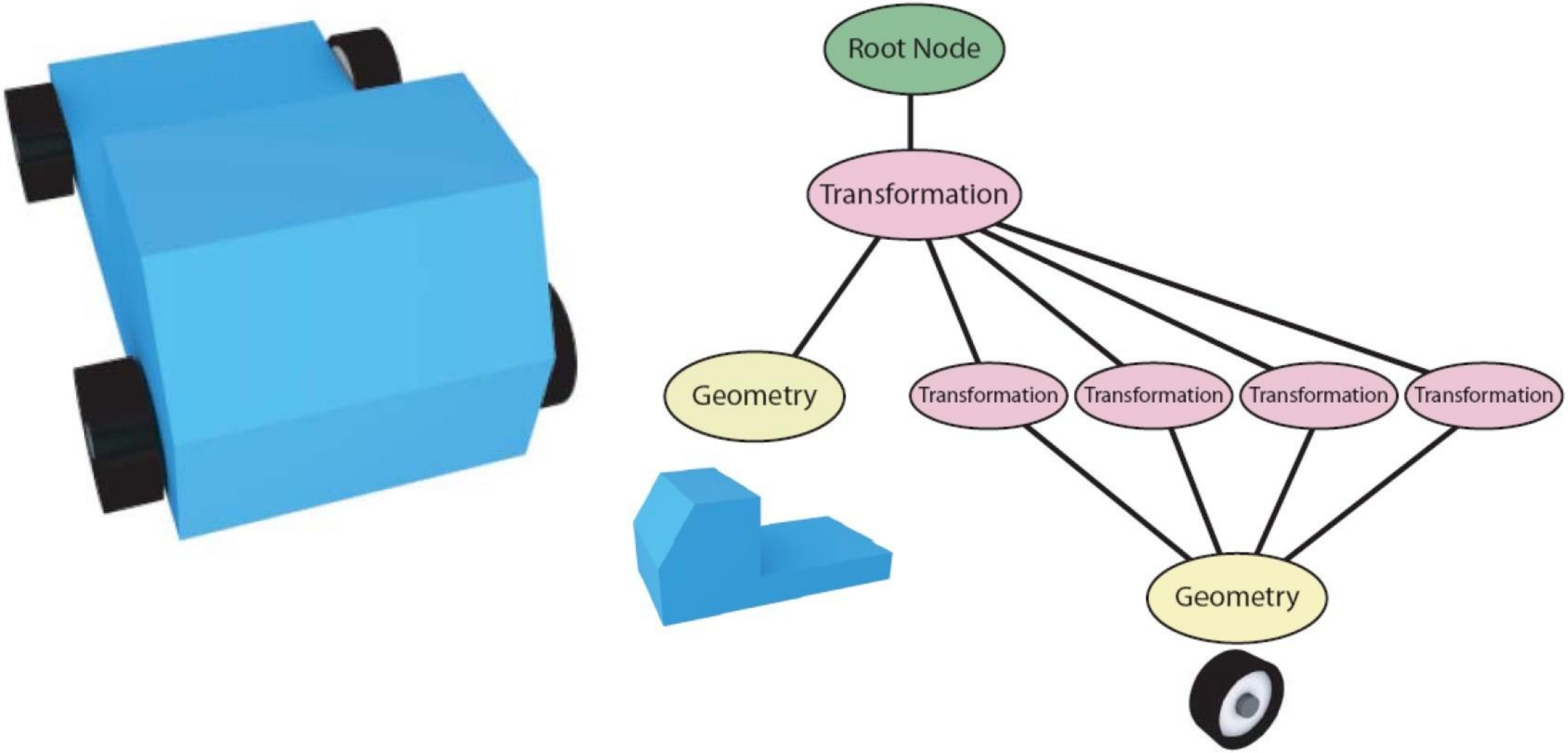


- https://www.youtube.com/watch?v=lNX0wCdD2LA

Scene Graphs

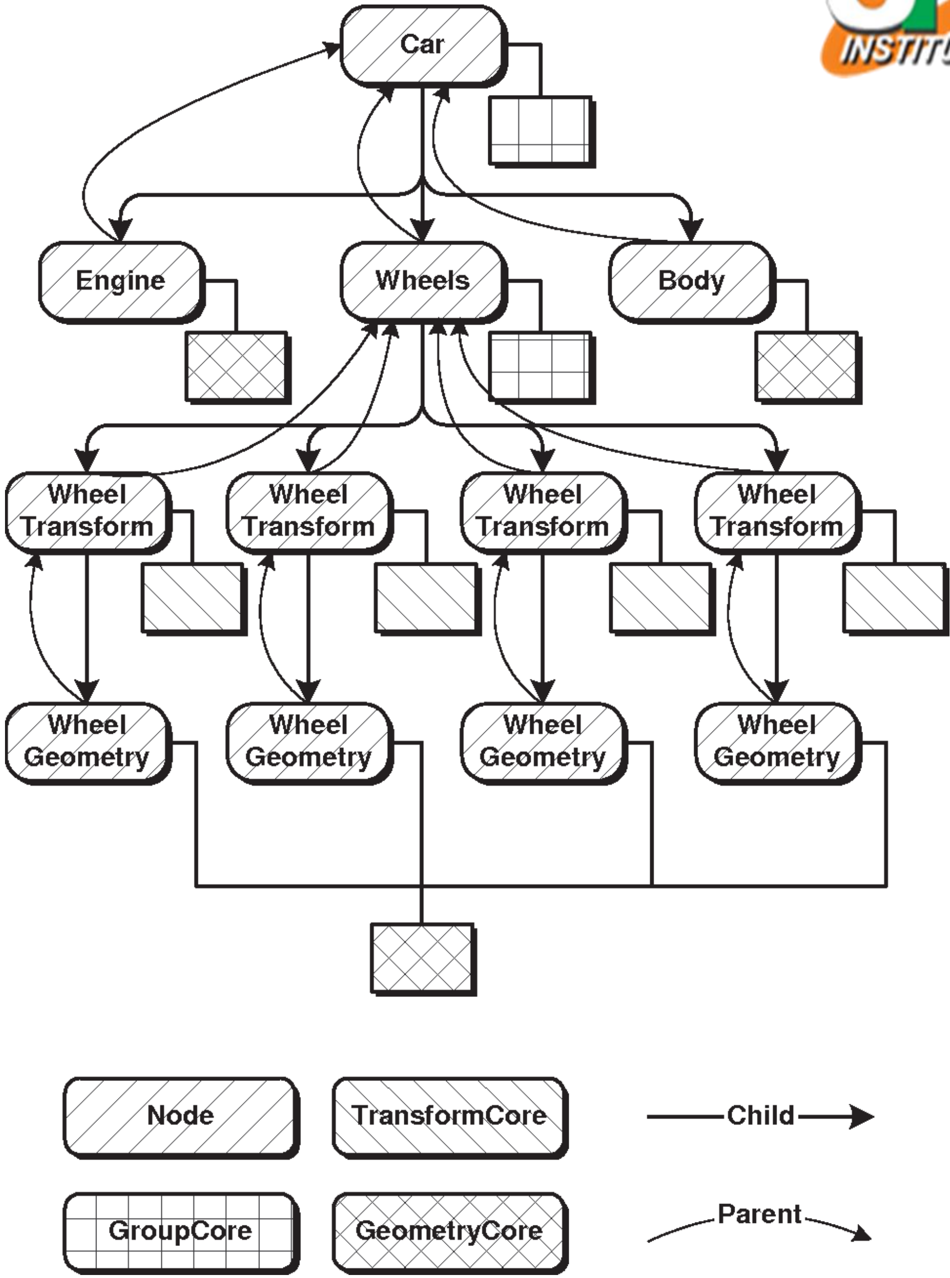
- Tree-like structure for organising VR graphics
  - e.g. VRML, OSG, X3D
- Hierarchy of nodes that define:
  - Groups (and Switches, Sequences etc…)
  - Transformations
  - Projections
  - Geometry
  - …
- And states and attributes that define:
  - Materials and textures
  - Lighting and blending
  - …

Example Scene Graph



- Car model with four wheels
  - Only need one wheel geometry object in scene graph

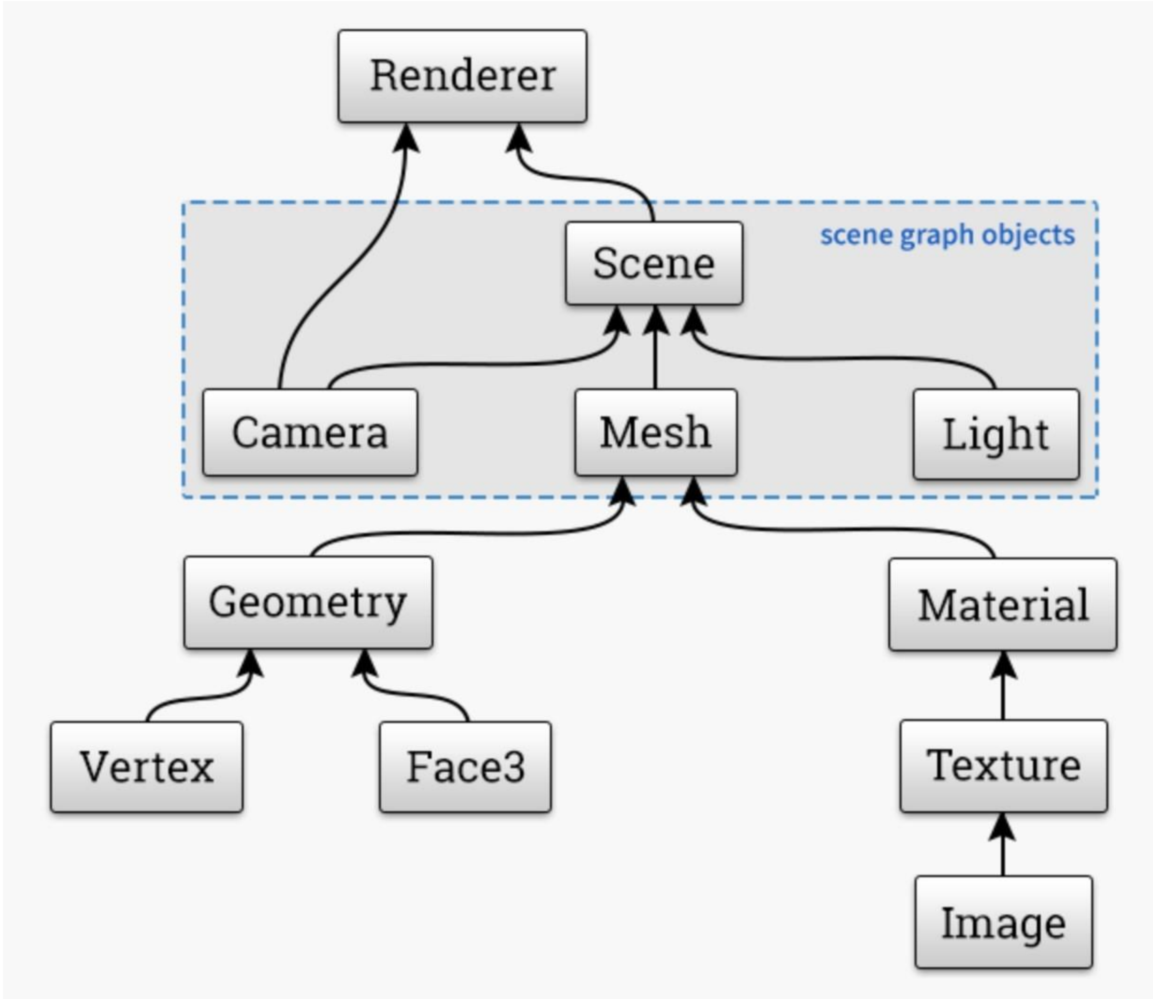
More Complex



- Everything off root node
- Parent/child node relationships
- Can move car by transforming group node

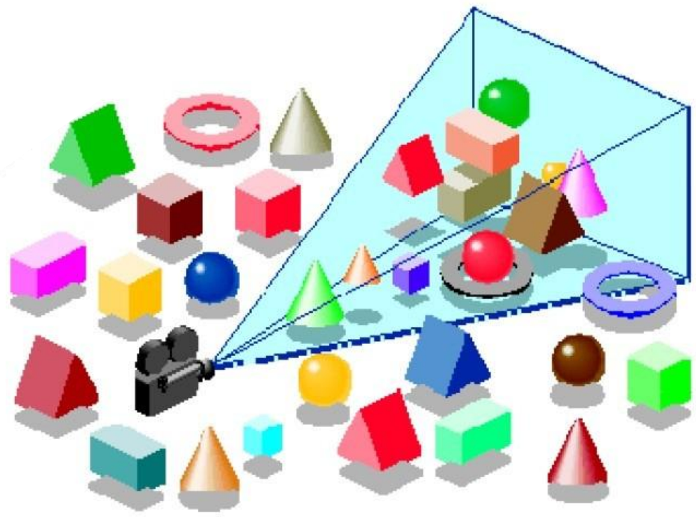
Adding Cameras and Lights

- Scene graph includes:
  - Cameras
  - Lighting
  - Material properties
  - Etc..
- All passed to renderer



Benefits of Using a Scene Graph

- Performance
  - Structuring data facilitates optimization
    - Culling, state management, etc…
- Hardware Abstraction
  - Underlying graphics pipeline is hidden
- No Low-level programming
  - Think about objects, not polygons
- Supports Behaviours
  - Collision detection, animation, etc..



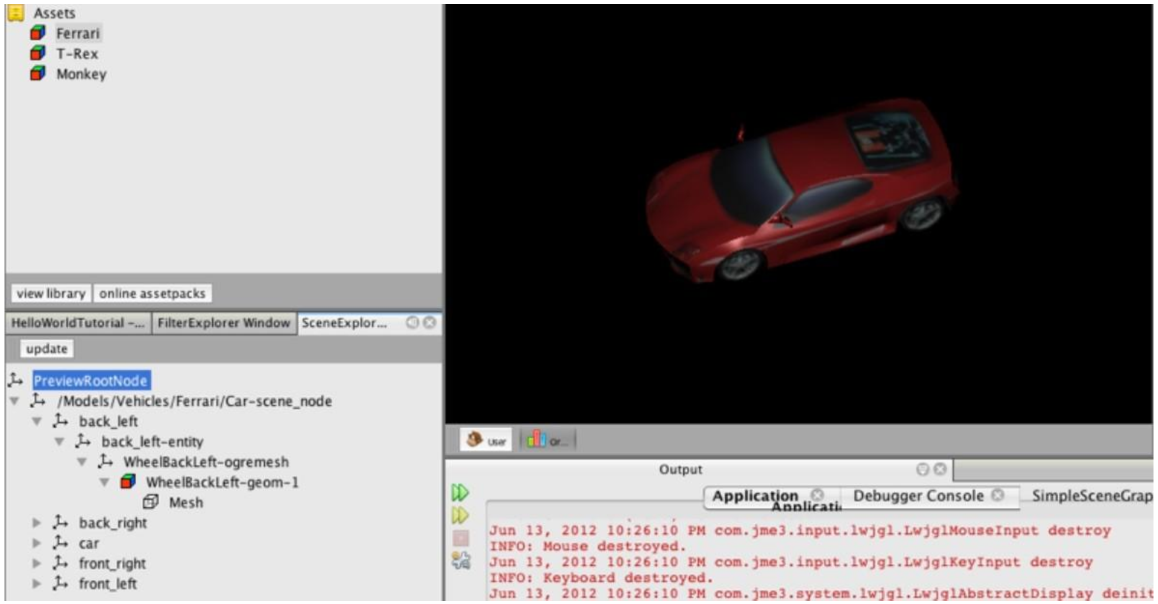
Scene Graph Libraries

- VRML/X3D
  - descriptive text format, ISO standard
- OpenInventor
  - based on C++ and OpenGL
  - originally Silicon Graphics, 1988
  - now supported by VSG3d.com
- Java3D
  - provides 3D data structures in Java
  - not supported anymore
- Open Scene Graph (OSG)
- Various Game Engines
  - e.g. JMonkey 3 (scene graph based game engine for Java)

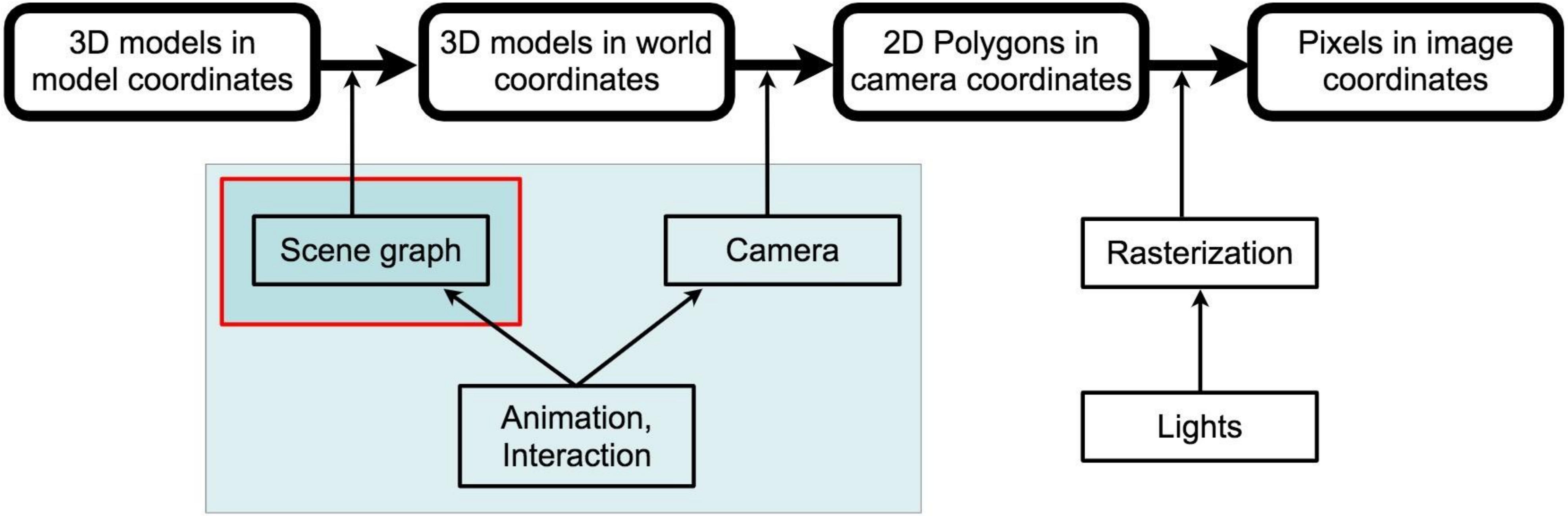
http://www.shlomifish.org/open-source/bits-and-bobs/open-inventor-bsd-daemon/

Creating a Scene Graph

- Creation of scene graph objects
  - Authoring software (e.g. Blender, 3DS Max)
- Assets exported to exchange formats
  - E.g. (X3D,) Wavefront OBJ (.obj), 3ds Max (.3ds), Ogre XML (.mesh)
- Objects typically are tesselated
  - Polygon meshes
- Create XML file
  - Specify scene graph
- Example:
  - JME Scene



Scene Graph in the Rendering Pipeline



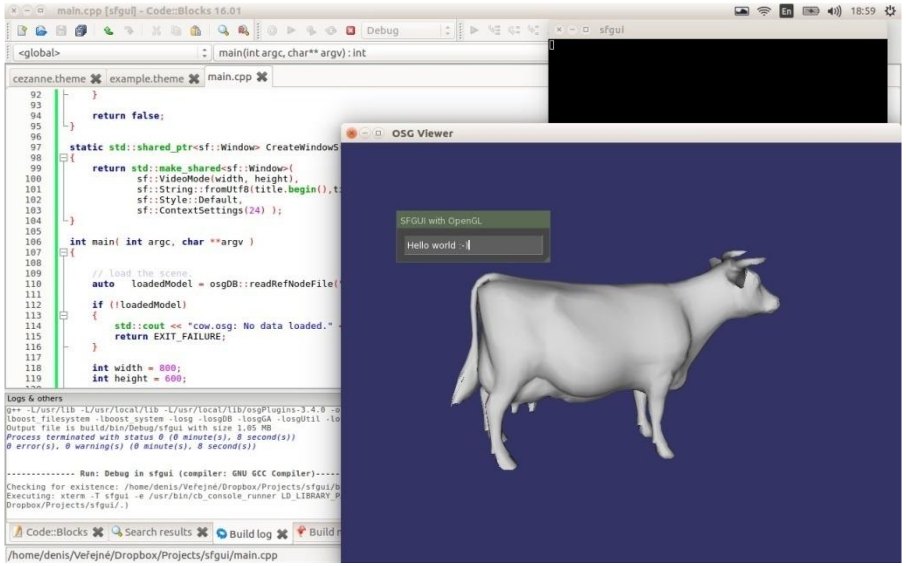
- Scene graph used to optimize scene creation in pipeline

OpenSceneGraph

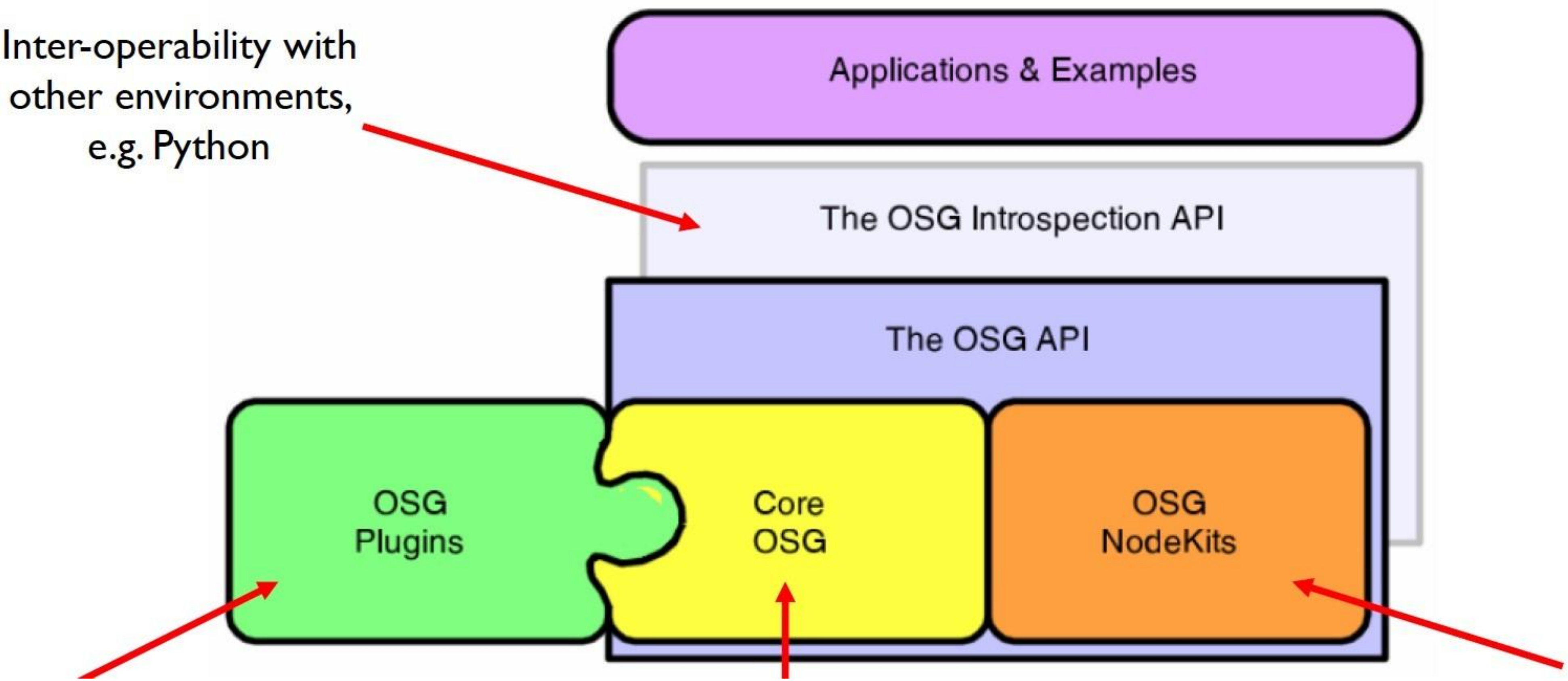
- http://www.openscenegraph.org/
- Open-source scene graph implementation
  - Based on OpenGL
- Object-oriented C++ following design pattern principles
  - Used for simulation, games, research, and industrial projects
- Active development community
  - mailing list, documentation (www.osgbooks.com)
- Uses the OSG Public License (similar to LGPL)

OpenSceneGraph Features

- Plugins for loading and saving
  - 3D: 3D Studio (.3ds), OpenFlight (.flt), Wavefront (.obj)…
  - 2D: .png, .jpg, .bmp, QuickTime movies
- NodeKits to extend functionality
  - osgTerrain - terrain rendering
  - osgAnimation - character animation
  - osgShadow - shadow framework
- Multi-language support
  - C++, Java, Lua and Python
- Cross-platform support:
  - Windows, Linux, MacOS, iOS, Android, etc.



OpenSceneGraph Architecture

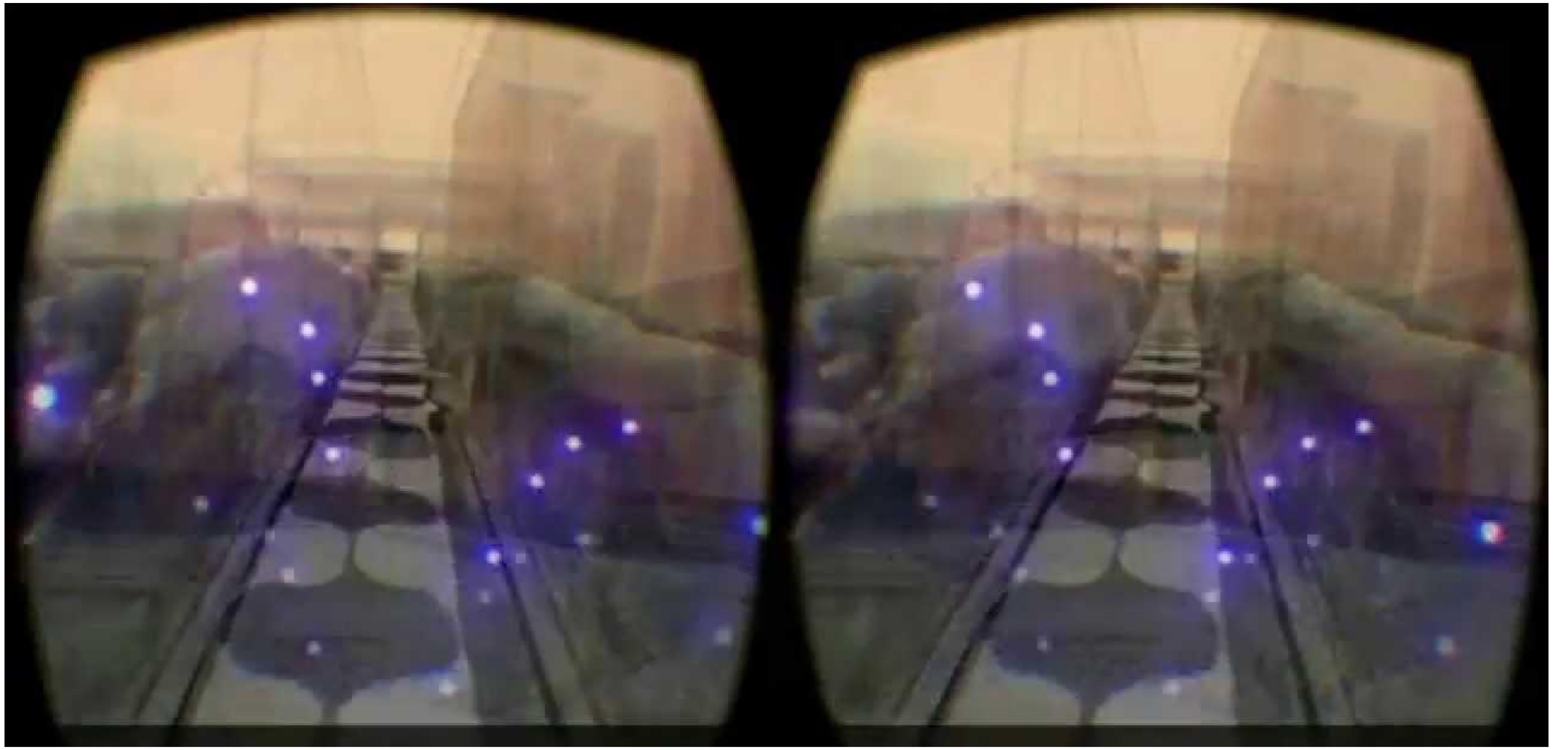


Plugins read and write 2D image and 3D model files

Scene graph and Rendering functionality

NodeKits extend core functionality, exposing higher-level node types

OpenSceneGraph and Virtual Reality

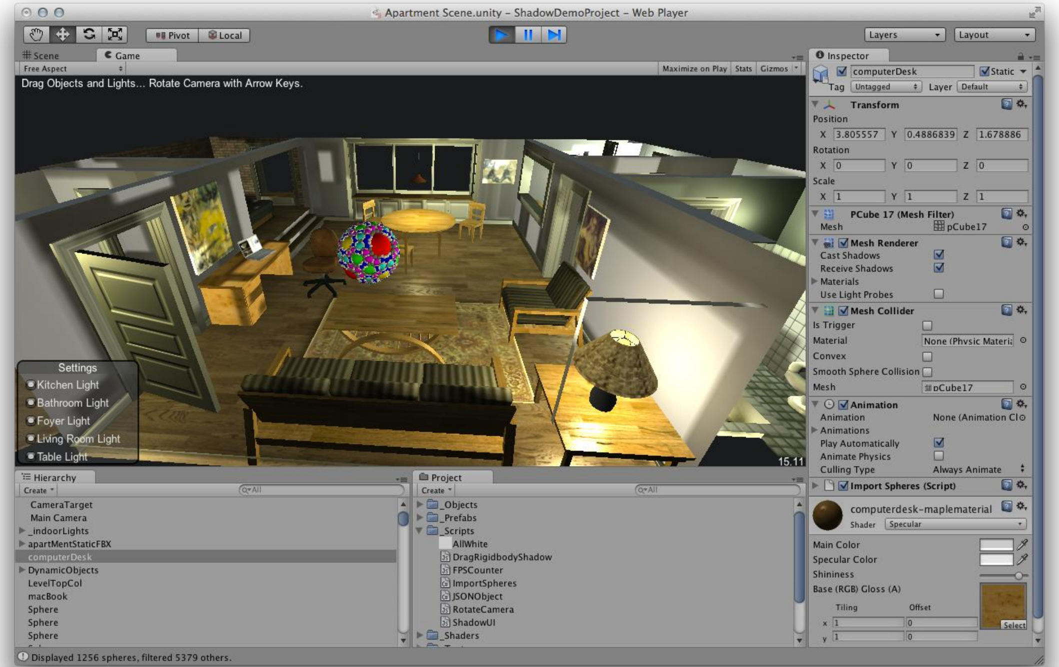


- Need to create VR wrapper on top of OSG
  - Add support for HMDs, device interaction, etc..
- Several viewer nodes available with VR support
  - OsgOpenVRViewer: viewing on VR devices compatible with openVR/steamVR
  - OsgOculusViewer: OsgViewer with support for the Oculus Rift

Examples

- Using OsgOculusViewer, Leap Motion and Oculus Rift HMD
- https://www.youtube.com/watch?v=xZgyOF-oT0g

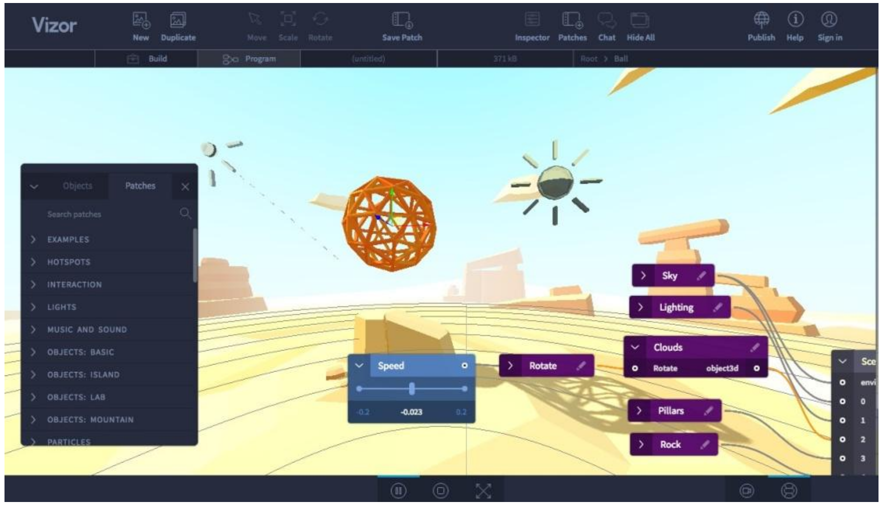
High Level Graphics Tools

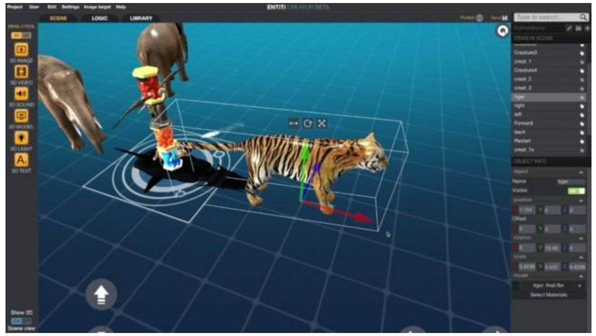


- Game Engines
  - Powerful, need scripting ability
    - Unity, Unreal, Cry Engine, etc..
- Combine with VR plugins
  - HMDs, input devices, interaction, assets, etc..

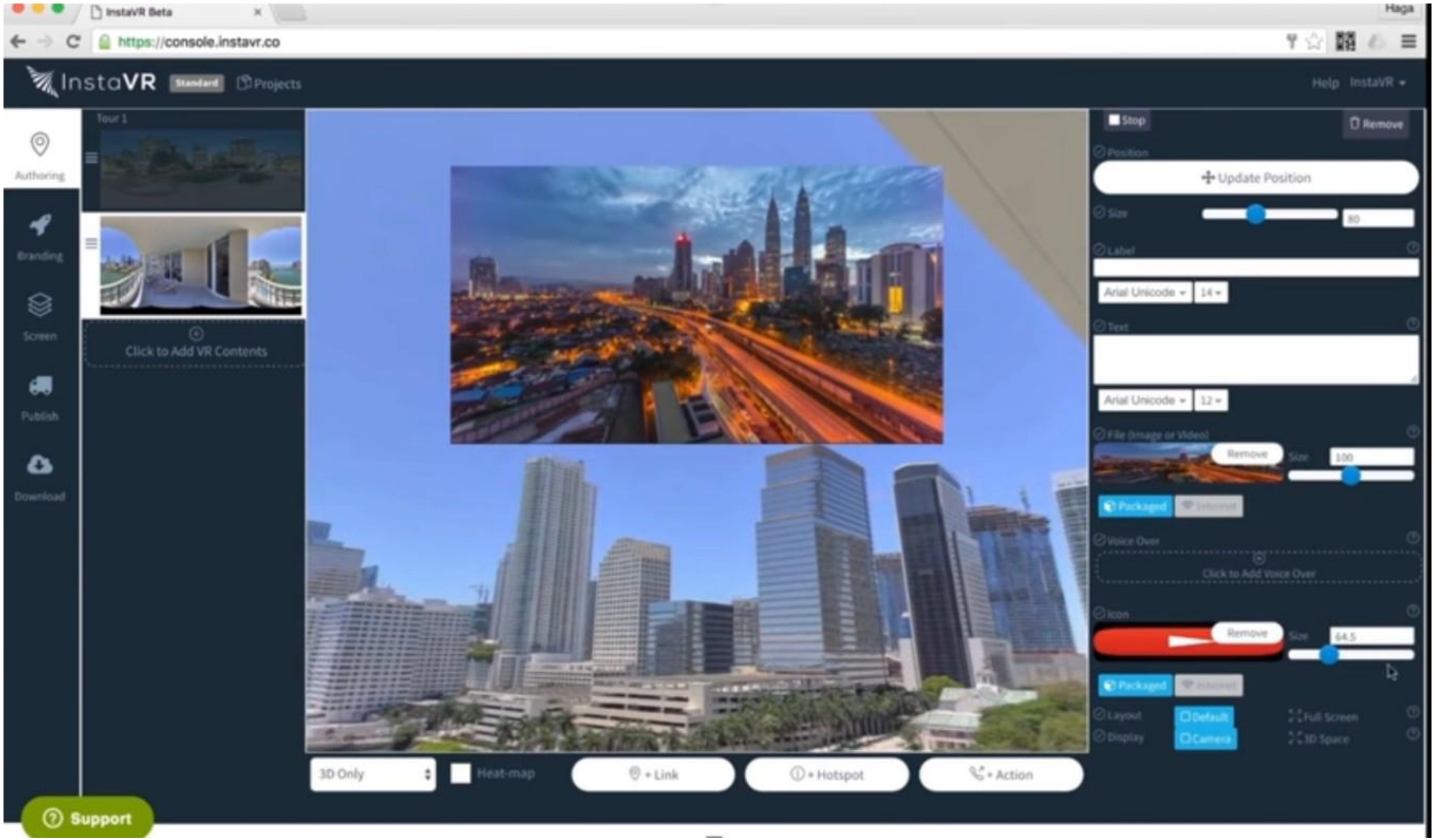
Tools for Non-Programmers

- Focus on Design, ease of use
  - Visual Programming, content arrangement
- Examples
  - Insta-VR – 360 panoramas
    - http://www.instavr.co/
  - Vizor – VR on the Web
    - http://vizor.io/
  - A-frame – HTML based
    - https://aframe.io/
  - Eon Creator – Drag and drop tool for AR/VR
    - http://www.eonreality.com/eon-creator/
  - Amazon Sumerian – WebGL, multiplatform
    - https://aws.amazon.com/sumerian/





Example: InstaVR (360 VR)



- https://www.youtube.com/watch?v=M2C8vDL0YeA

Example: Amazon Sumerian (3D VR)

- https://www.youtube.com/watch?v=_Q3QKFp3zlo

SYSTEM DESIGN GUIDELINES

System Design Guidelines - I

- Hardware
  - Choose HMDs with fast pixel response time, no flicker
  - Choose trackers with high update rates, accurate, no drift
  - Choose HMDs that are lightweight, comfortable to wear
  - Use hand controllers with no line of sight requirements
- System Calibration
  - Have virtual FOV match actual FOV of HMD
  - Measure and set users IPD
- Latency Reduction
  - Minimize overall end to end system delay
  - Use displays with fast response time and low persistence
  - Use latency compensation to reduce perceived latency

Jason Jerald, *The VR Book*, 2016

System Design Guidelines - II

- General Design
  - Design for short user experiences
  - Minimize visual stimuli closer to eye (vergence/accommodation)
  - For binocular displays, do not use 2D overlays/HUDs
  - Design for sitting, or provide physical barriers
  - Show virtual warning when user reaches end of tracking area
- Motion Design
  - Move virtual viewpoint with actual motion of the user
  - If latency high, no tasks requiring fast head motion
- Interface Design
  - Design input/interaction for user's hands at their sides
  - Design interactions to be non-repetitive to reduce strain injuries

Jason Jerald, *The VR Book*, 2016

www.empathiccomputing.org

mark.billinghurst@unisa.edu.au

@marknb00