Immersive Systems I
Hardware & Software

Developing Immersive Applications

Created by: Chek Tien TAN

Learning Objectives:

- describe common hardware and software components across XR devices
- differentiate device capabilities and form factors
- explain general system architecture for immersive applications
- describe the software component types in WebXR applications

Inside an HMD: Meta Quest 2

Key Hardware Components:

- Display screen (1800 × 1920 per eye)
- Magnifier lenses (+ optional corrective lenses)
- Specialized controllers (IR light arrays, motion sensors)
- Motion tracking sensors (IMU - accelerometer, gyroscope)
- Cameras (inside-out tracking, passthrough)
- Battery, speakers, microphones
- CPU/GPU, motherboard

Source: iFixit Quest 2 Teardown - ifixit.com

Smartphone vs HMD Hardware

Similarities:

- Display screen
- CPU/GPU
- Motion sensors (IMU)
- Cameras
- Battery

Key Difference: Magnifying Lenses

- HMDs use lenses to magnify the close display
- Creates wide field of view
- Enables virtual image at optical infinity
- Smartphone cardboard viewers add lenses to phones

AR Hardware Landscape

Augmented Reality devices come in various form factors:

1. **AR Glasses** — lightweight, everyday wearable
2. **AR Headsets** — higher capability, professional use
3. **Mobile AR** — smartphone/tablet cameras

Let’s explore recent AR hardware across these categories

AR Glasses

Meta Ray-Ban Smart Glasses (2023)

- Consumer-friendly glasses form factor
- Built-in cameras, speakers, microphone
- AI assistant integration
- *Not true AR — no display overlay*

Xreal Air / Air 2 Pro

- Virtual display glasses (130” equivalent screen)
- 1080p OLED per eye, 46° FOV
- Tethered to phone/computer
- Consumer entertainment focus

Rokid Max

- Similar virtual display approach

AR Headsets (Professional)

Microsoft HoloLens 2

- Optical see-through AR headset
- 52° diagonal FOV, hand tracking
- Enterprise/industrial use (training, remote assist)
- Standalone device, no tether required

Magic Leap 2

- Waveguide-based AR display
- 70° diagonal FOV, eye tracking
- Enterprise/medical applications
- Dimming technology for varying light conditions

Apple Vision Pro (2024)

- High-resolution displays (23M pixels total)

AR Hardware: Key Differences

Form Factor Trade-offs:

- **Glasses** — lightweight, socially acceptable, limited capability
- **Headsets** — more powerful, wider FOV, bulkier
- **Mobile** — ubiquitous, no extra hardware, limited tracking

Display Technology:

- **Optical see-through** — direct view + overlay (HoloLens, Magic Leap)
- **Video passthrough** — cameras + digital merge (Vision Pro, Quest 3)
- **Virtual display** — screen projection only (Xreal, Rokid)

Use Cases:

- **Enterprise** — Training, remote assistance, visualization
- **Consumer** — Entertainment, productivity, communication

Common Software Components

At its core, an immersive application is a real-time interactive simulation.

1. Rendering System
2. Physics System
3. Input Handler
4. Audio Processor
5. AI System

1. Rendering System

Responsibilities:

- Drawing graphics to the display
- Managing 3D models, textures, materials
- Configuring lighting and cameras
- Post-processing effects (bloom, DOF, etc.)
- **Stereo rendering** for VR

Under the Hood:
Scene graph traversal, culling, clipping, shader execution, rasterization

In WebXR: Babylon.js provides high-level rendering API and handles stereo rendering automatically with WebXR session.

2. Physics System

Responsibilities:

- Collision detection and response
- Rigid body dynamics
- Gravity and forces
- Constraints and joints
- Raycasting for interaction

In WebXR: Babylon.js integrates physics engines like Cannon.js, Ammo.js, or Havok for realistic physical simulations.

3. Input Handler

Responsibilities:

- Reading controller button presses
- Tracking hand/controller positions
- Processing gestures and interactions
- Managing input events
- Supporting multiple input devices

In WebXR: WebXR Device API provides standardized access to VR/AR controllers and hand tracking.

4. Audio Processor

Responsibilities:

- Spatial audio (3D sound positioning)
- Sound effects playback
- Music and ambient audio
- Audio attenuation with distance
- Head-related transfer function (HRTF)

In WebXR: Web Audio API provides spatial audio capabilities for immersive sound experiences.

5. AI System

Responsibilities:

- NPC behavior and decision-making
- Pathfinding (A* algorithm, navigation meshes)
- Behavior trees for complex AI
- Procedural content generation
- Adaptive difficulty

Note: Not all immersive applications need AI systems — it depends on the experience design.

Which Components Are Essential?

Consider the **EDIS** (Experience Dementia in Singapore) application:

Essential components:

- Rendering System — display 360° environments
- Input Handler — navigate through scenes
- Audio Processor — ambient sounds, narration

Less critical / not needed:

- Physics System — no physical interactions required
- AI System — no NPCs or adaptive behavior

The importance of each component depends on your application's design goals.

Entity Component System (ECS)

Modern immersive applications often use **ECS architecture**

Traditional OOP approach:

- Objects inherit from base classes
- Tight coupling between behaviors
- Hard to add new behaviors dynamically

ECS approach:

- **Entities** — unique IDs (e.g., player, enemy, light)
- **Components** — data containers (position, mesh, physics)
- **Systems** — logic processors (rendering, physics, AI)

Key advantage: Composition over inheritance — easily mix and match components

Why ECS Over OOP?

1. **Flexibility**
   - Add/remove behaviors at runtime
   - No rigid inheritance hierarchies
2. **Performance**
   - Data-oriented design enables cache-friendly processing
   - Systems process components in batches
3. **Maintainability**
   - Clear separation of data and logic
   - Easier to test and debug
   - Team members can work on different systems independently

Top reason: Composition enables flexible, reusable, and performant

WebXR Application Architecture

Typical WebXR app structure:

1. **Scene Graph**
   - Hierarchical tree of objects
   - Parent-child transformations
2. **Rendering Loop**
   - `requestAnimationFrame` or `XRSession.requestAnimationFrame`
   - Update logic → render frame (60-90+ FPS)
3. **Component System**
   - Meshes, materials, lights, cameras
   - Physics bodies, colliders
   - Custom behaviors and scripts

Summary

Today we covered:

- HMD hardware components (Quest 2 example)
- Smartphone vs HMD — key difference is magnifying lenses
- AR hardware landscape (glasses, headsets, mobile)
- Common software components in immersive apps
- ECS architecture advantages over traditional OOP
- WebXR application structure

Next: Week06 — Deep dive into HMD optics theory