UMKC

COESC AI+AR/VR

Spring 2025

View these Slides:
https://docs.google.com/presentation/d/1utVYhSMCL8EviY2QNkuyc3JoUDwlz9j4vyebENd_kko/edit?usp=sharing

Luke Miller

Course Summary

- Session 1: Setting Up and Unity Basics -   Jan 25, 2025
- Session 2: ML Basics and Oculus Integration - Feb 1, 2025

UMKC

Staff

Instructor

Yugyung Lee, PhD

Professor of Computer

University of Missouri - Kansas City

Email: leeyu@umkc.edu

Phone: 816-235-5932

Assistant Instructor

Luke Miller

Computer Science PhD Student

University of Missouri - Kansas City

Email: ljmbm5@umkc.edu

Logistics and Materials

- **Prerequisites:**
  - Basic programming knowledge (C# preferred).
- **Tools and Setup:**
  - Unity (with Android SDK) for AR/VR sessions.
  - Oculus Link
  - Meta Oculuses provided for hands-on activities.

- **Team Size:**
  - Small groups (2–3 participants) to encourage collaboration.
- **Resources Provided:**
  - Presentation Materials
  - Pre-configured Software
  - Lessons
  - Tutorials

Objective:

- **By the end of the course, participants will:**
  - Understand the basics of ML and its applications in AR/VR.
  - Be familiar with using Meta Oculuses for AR development.
  - Create a simple, functional AR/VR application enhanced with an ML feature.

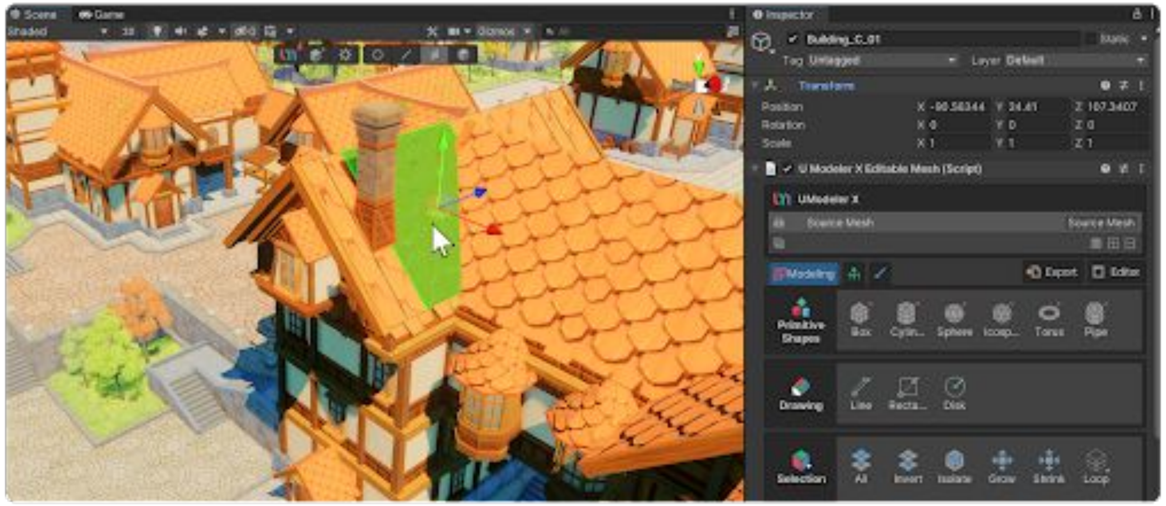
UMKC

Session 1: Setting Up and Unity Basics

- Getting Started with Unity
  - What is Unity
  - Overview of the Unity interface
- Building the First Scene
  - Adding objects (3D models, UI elements) and adjusting transforms.
  - Adding physics components: Rigidbodies and colliders.
- Basic Interactions with Unity
  - Adding scripts (in C#) to objects for interactivity.

Introduction to Unity

- What is Unity?
- Why is it used in AR/VR development?
- Installing and Configuring Unity



Building_C_01

Tag Untagged Layer Default

Transform

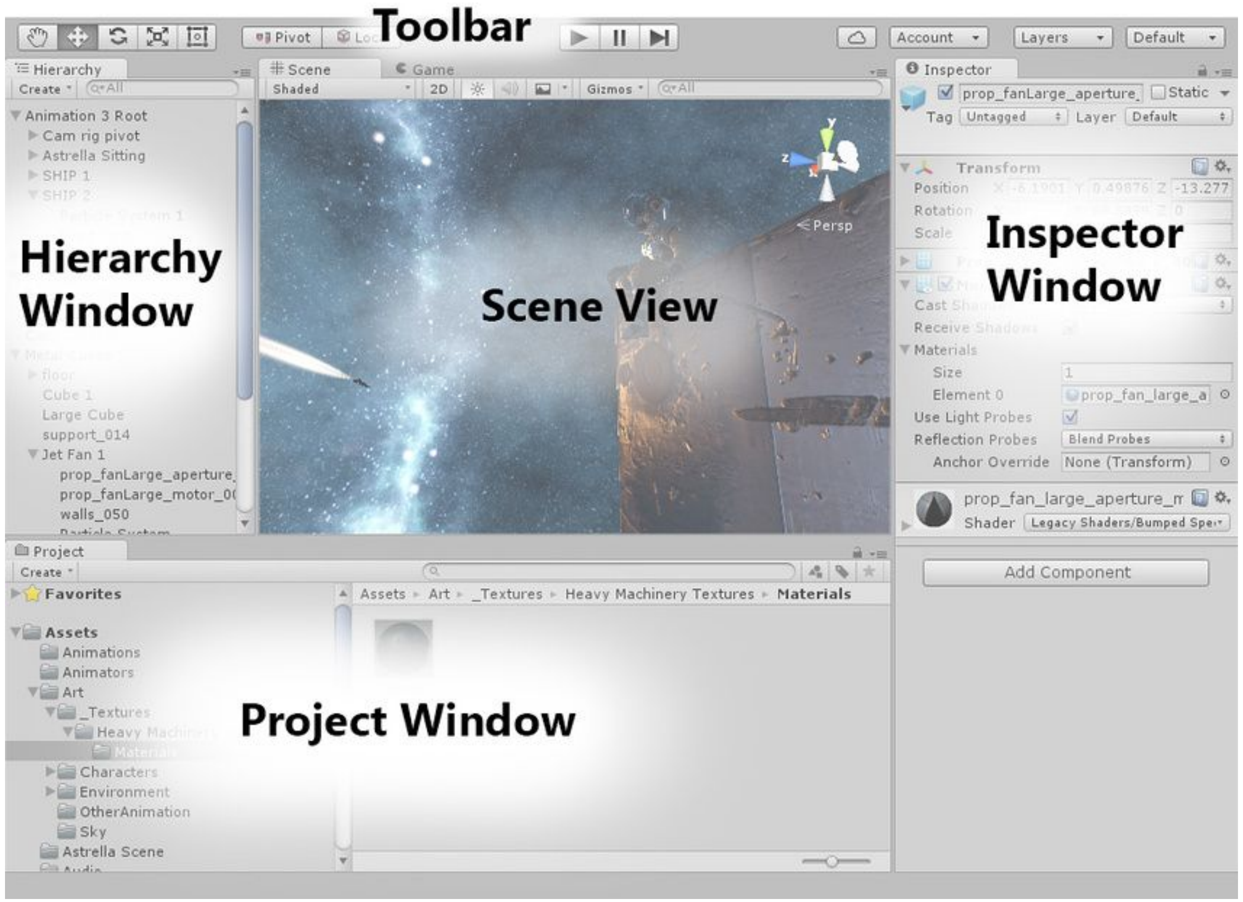
Position	X -90.58344	Y 34.41	Z 107.3407
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

U Modeler X Editable Mesh (Script)

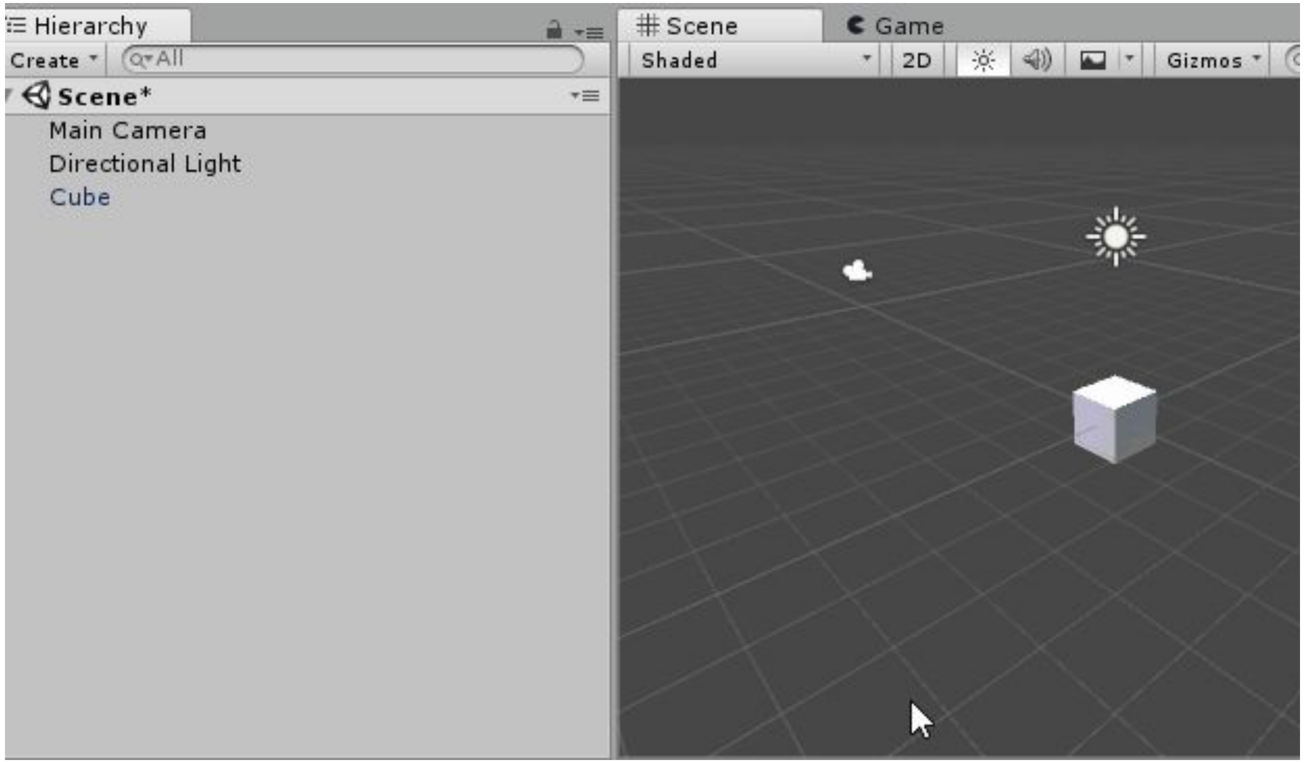
100 SECONDS OF

Unity Interface Overview

- Key components:
  - Hierarchy: List of all objects in the scene.
  - Scene View: Visual representation of the scene.
  - Inspector: Object properties and adjustments.
  - Toolbar: Provides options to the user
  - Project Window: Asset management.



Unity Interface: Hierarchy

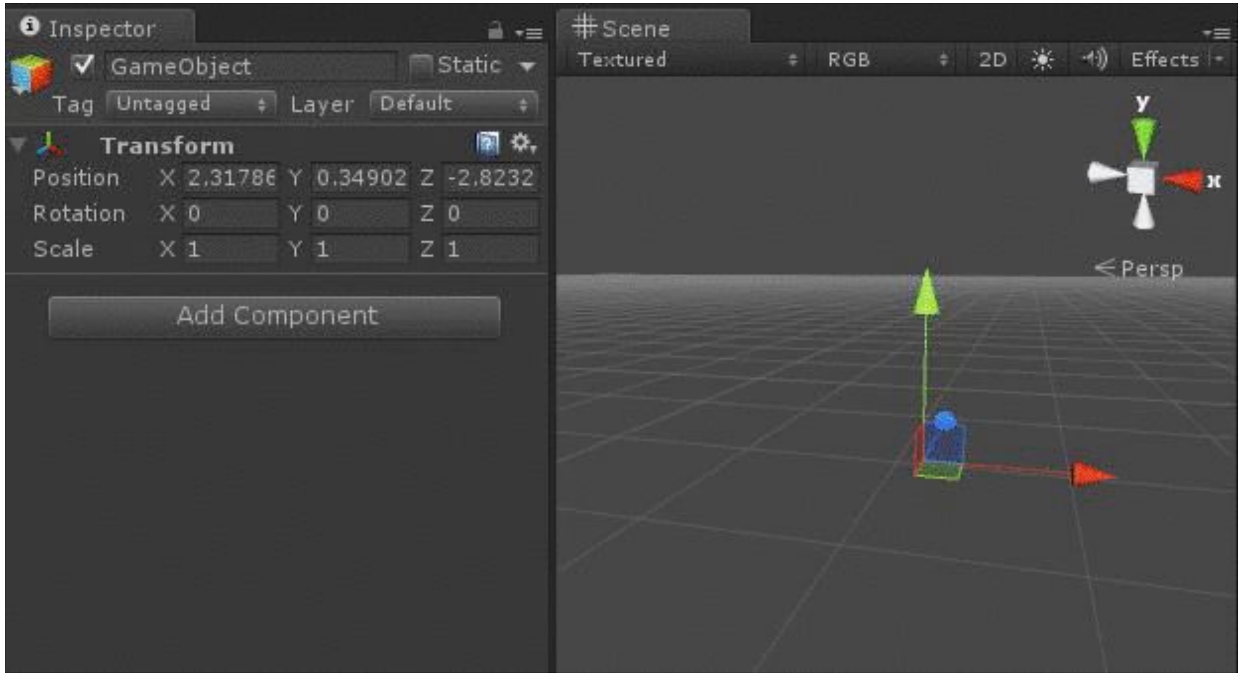


- The Hierarchy window displays every GameObject in a Scene
- You can use the Hierarchy window to sort and group the GameObjects you use in a Scene.
- When you add or remove GameObjects in the Scene view, you also add or remove them from the Hierarchy window.

Learn more: https://docs.unity3d.com/Manual/Hierarchy.html

Unity Interface: Scene View

- The Scene view is where you visualize and interact with the world you create in the Editor.
- In the Scene view, you can select, manipulate, and modify GameObjects: scenery, characters, cameras, lights, and more.



GameObject ☐ Static

Tag Untagged Layer Default

Transform

Position X 2.31786 Y 0.34902 Z -2.8232

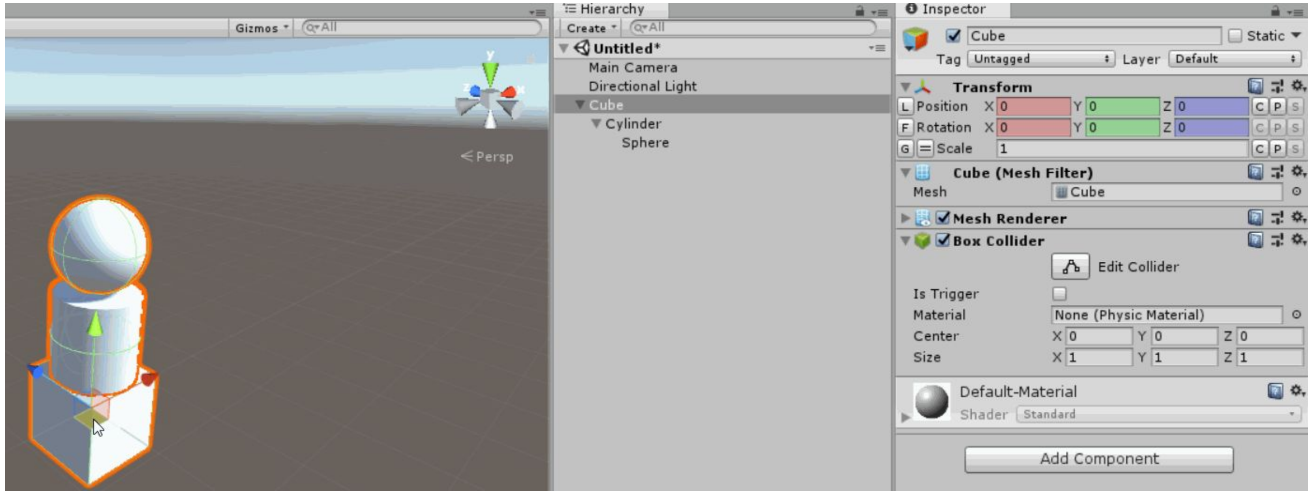
Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Textured RGB 2D

https://docs.unity3d.com/Manual/UsingTheSceneView.html

Unity Interface: Inspector



Inspector

☑ Cube □ Static

Tag Untagged Layer Default

Transform

Position X 0 Y 0 Z 0

Rotation X 0 Y 0 Z 0

Scale 1

Cube (Mesh Filter)

Mesh Cube

☑ Mesh Renderer

☑ Box Collider

Edit Collider

Is Trigger □

Material None (Physic Material)

Center X 0 Y 0 Z 0

Size X 1 Y 1 Z 1

Default-Material

Shader Standard

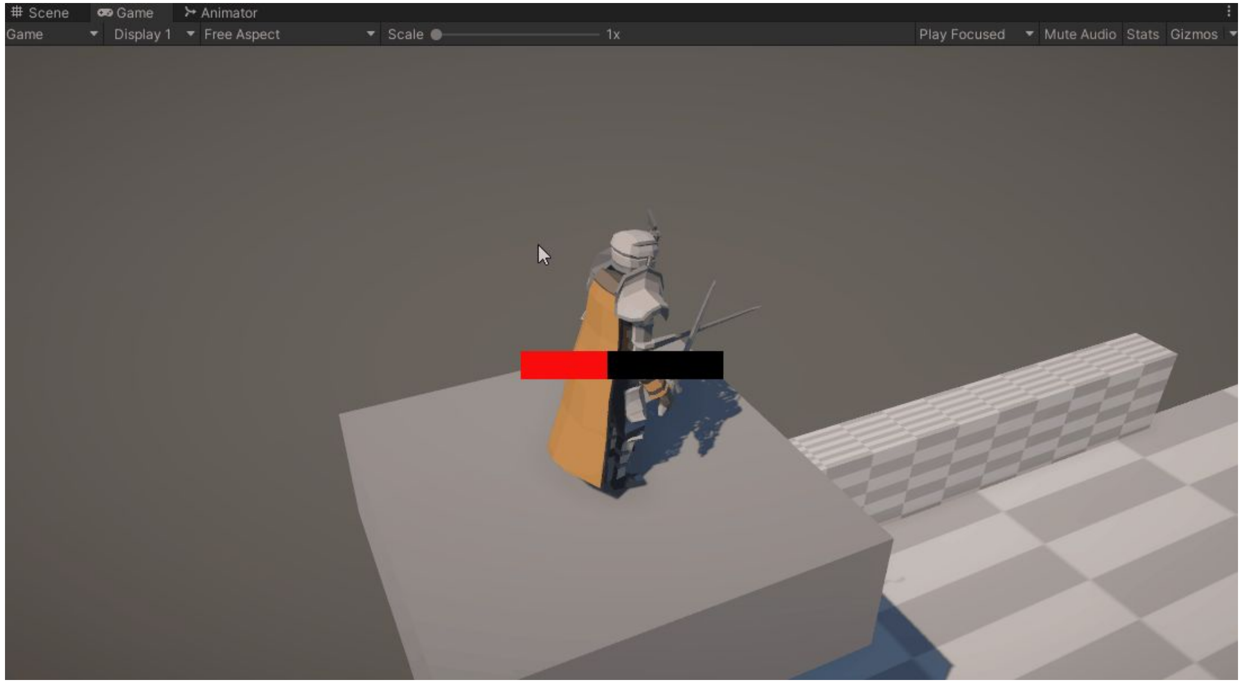
Add Component

- Use the Inspector window to view and edit properties and settings for almost everything in the Unity Editor:
  - GameObjects
  - Unity components
  - Assets
  - Materials
  - In-Editor settings and preferences

https://docs.unity3d.com/Manual/UsingTheInspector.html

Unity Interface: Game View

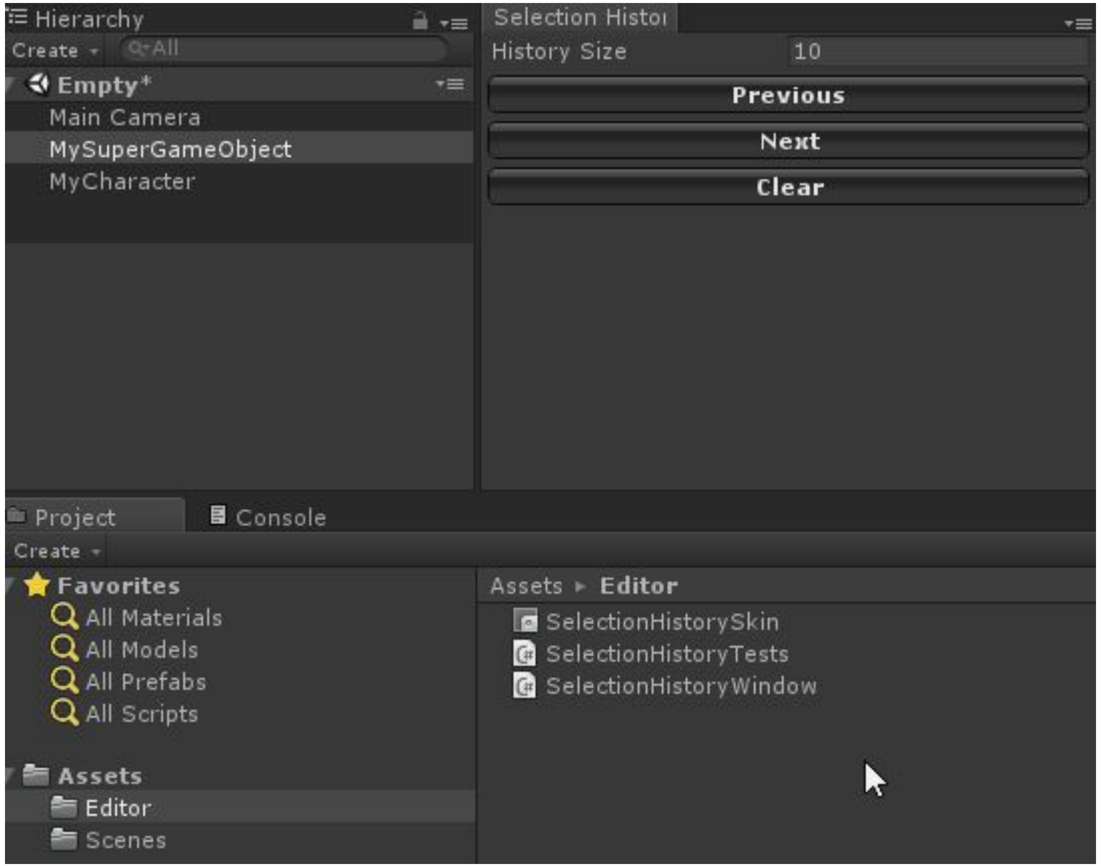
- The Game view is rendered from the Cameras in your application.
- The Game view displays how your final, built application looks.
- You need to use one or more Cameras to control what the player sees when they’re using your application.



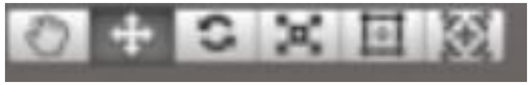
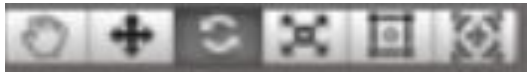
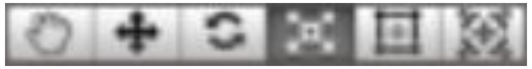
https://docs.unity3d.com/Manual/GameView.html

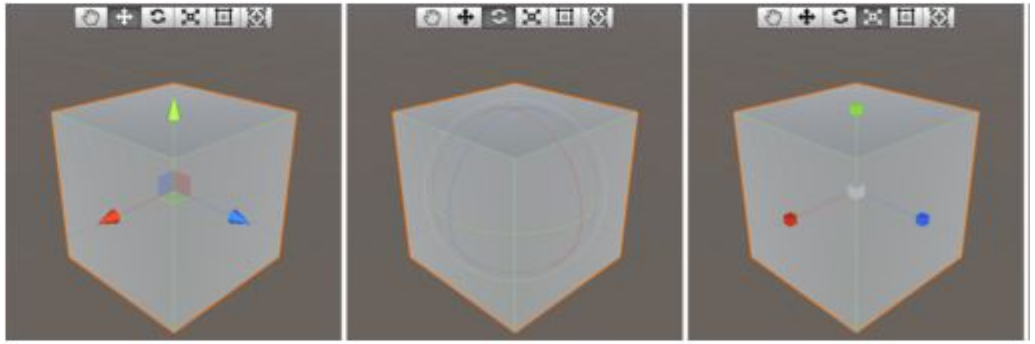
Unity Interface: Project Tab

- The Project window displays all of the files related to your Project
- The main way you can navigate and find Assets and other Project files in your application.
- When you start a new Project by default this window is open.

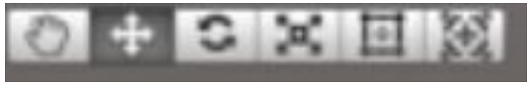
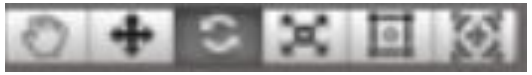
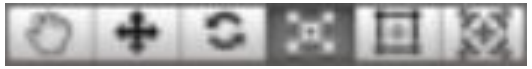


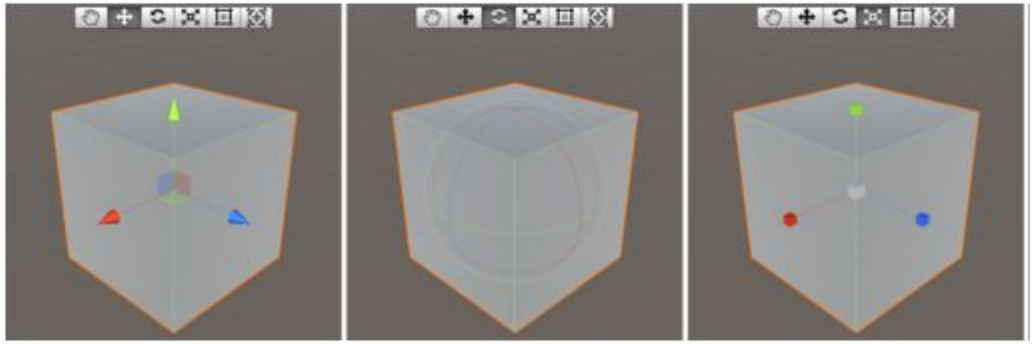
Building the First Scene

- Add 3D objects (cube, sphere, plane).
- Move, Rotate, and Scale tools.
  - Hand Tool. Pans around.
  - Move tool 
  - Rotate Tool 
  - Scale Tool 
-

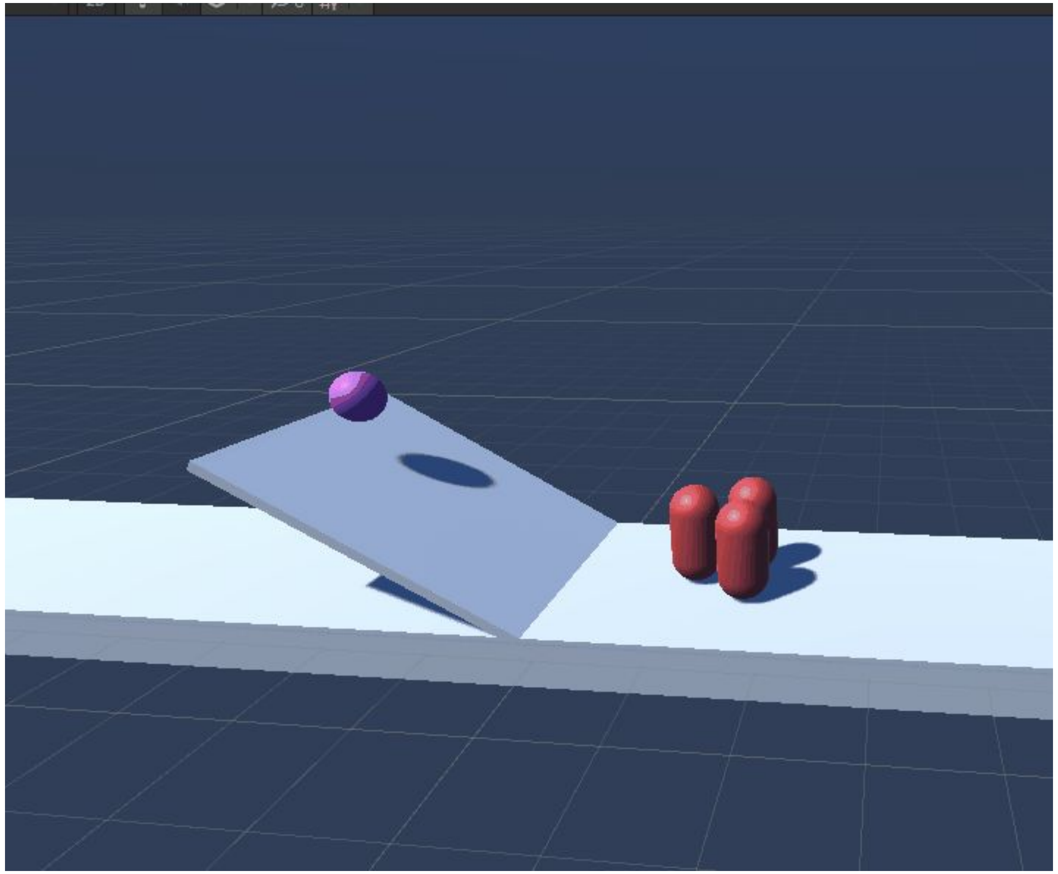


Building the First Scene: Objects

- Add 3D objects (cube, sphere, plane).
- Move, Rotate, and Scale tools.
    - Hand Tool. Pans around.
    - Move tool 
    - Rotate Tool 
    - Scale Tool 

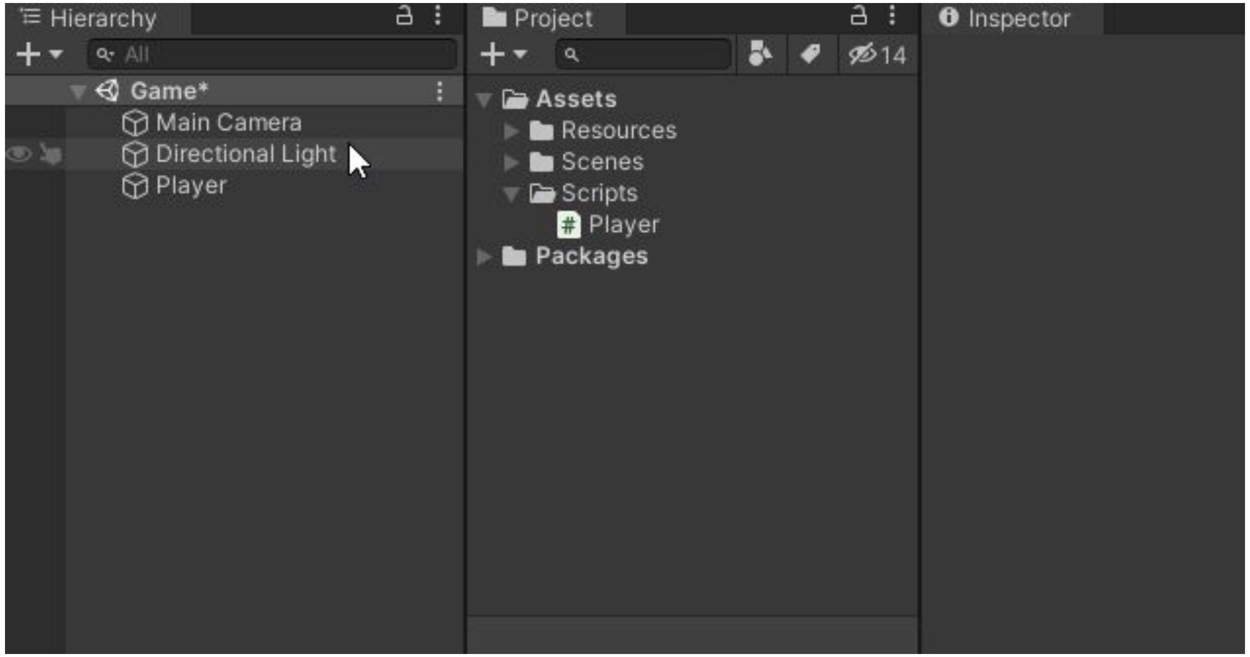


Building the First Scene: Physics



- Rigidbodies
  - Adding a Rigidbody component to an object will put its motion under the control of Unity's physics engine.
  - Even without adding any code, a Rigidbody object will be pulled downward by gravity and will react to collisions.
- Colliders
  - Collider components define the shape of an object for the purposes of physical collisions.
  - A collider is invisible
  - A rough approximation of the visual shape is often more efficient and indistinguishable in gameplay.

Basic Interactions with Unity



- What are Scripts
  - Scripts allow you to customize and extend the capabilities of your applicaton with C# code.
  - Unlike most other assets, scripts are usually created within Unity directly.
- Adding Scripts to Entities
  - From the main menu: go to **Assets > Create > Scripting** and select the type of script you want to create. *OR*
  - From the Create menu (plus sign) in the Project window toolbar: go to Scripting and select the type of script you want to create.

VR Best Practices - Optimizing Performance in VR

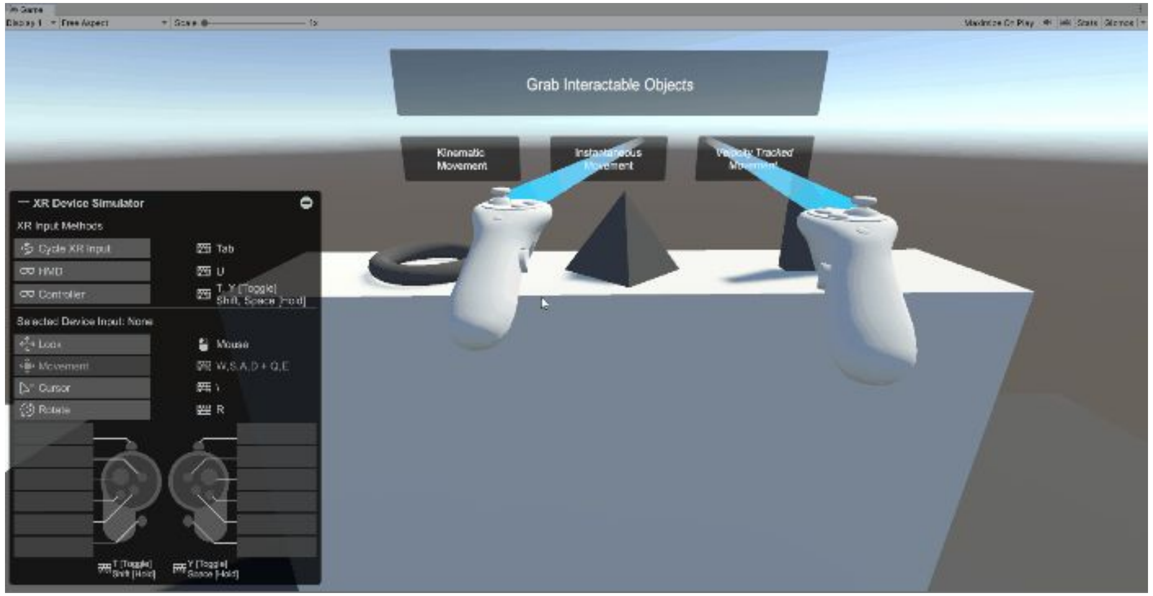
- **Understanding Frame Rates and Latency**
  - VR requires a stable frame rate of 90 FPS or higher to avoid motion sickness.
  - Latency below 20 ms is critical for maintaining immersion and comfort.
- **Techniques for Reducing Draw Calls and Poly Counts**
  - Combine meshes and use texture atlases to reduce draw calls.
  - Simplify 3D models to lower poly counts while maintaining visual quality.
- **Efficient Use of Lighting and Shadows**
  - Use baked lighting instead of real-time lighting for static environments.
  - Optimize shadow settings: lower resolution or use shadow cascades selectively.

VR Best Practices - User Experience (UX) in VR

- **Designing Intuitive User Interfaces**
  - Use gaze, hand tracking, or controllers for navigation.
  - Avoid clutter and focus on clear, readable text and visuals.
- **Addressing Motion Sickness**
  - Minimize rapid camera movements and sudden acceleration.
  - Use teleportation over continuous movement for navigating VR environments.
- **Importance of Audio in Enhancing Immersion**
  - Implement spatial audio for directional sound cues.
  - Use ambient sounds to make virtual environments more lifelike.

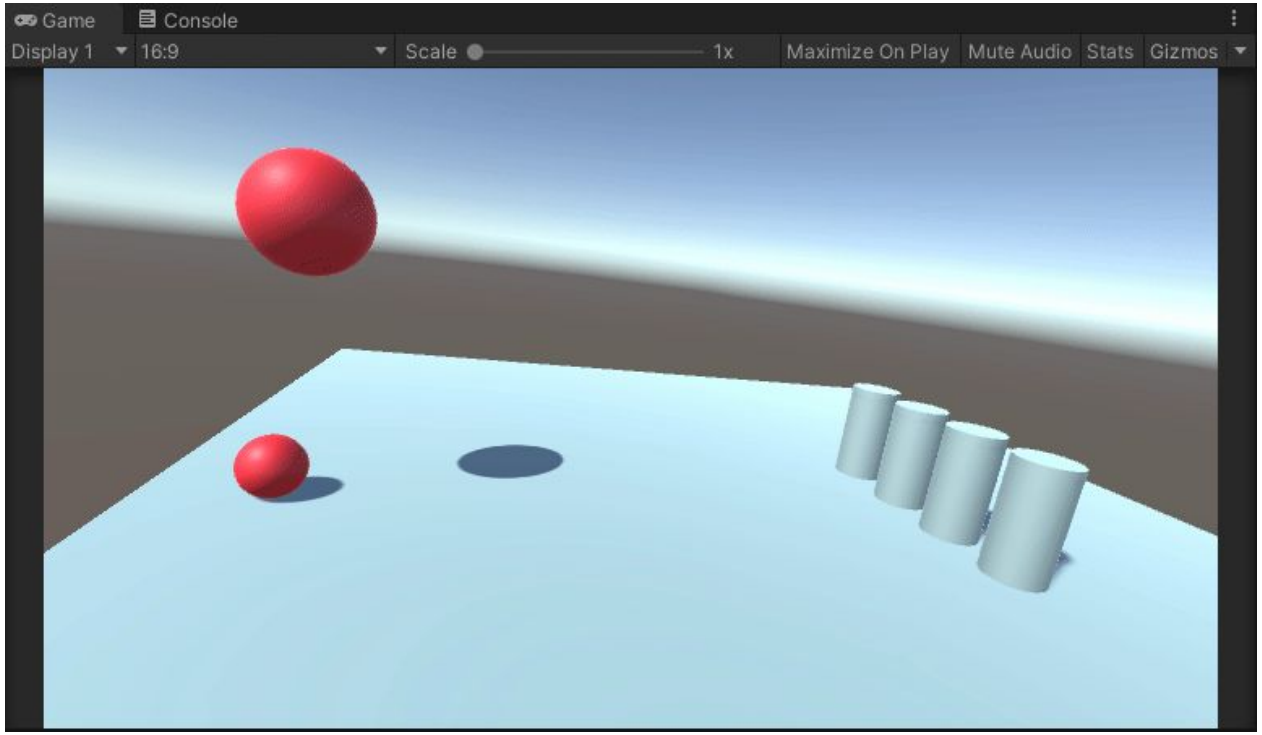
Unity Capabilities -
Unity's XR Interaction Toolkit

- **Overview and Setup**
  - Provides prebuilt interactions for VR such as teleportation and object manipulation.
  - Easy integration with Unity's XR framework and supported devices.
- **Implementing Teleportation and Direct Interactions**
  - Set up XR Interaction Manager and Ray Interactor for teleportation.
  - Configure interactable objects to enable grabbing, throwing, and touching.



UMKC

Unity Capabilities -
Advanced Physics and Interactions

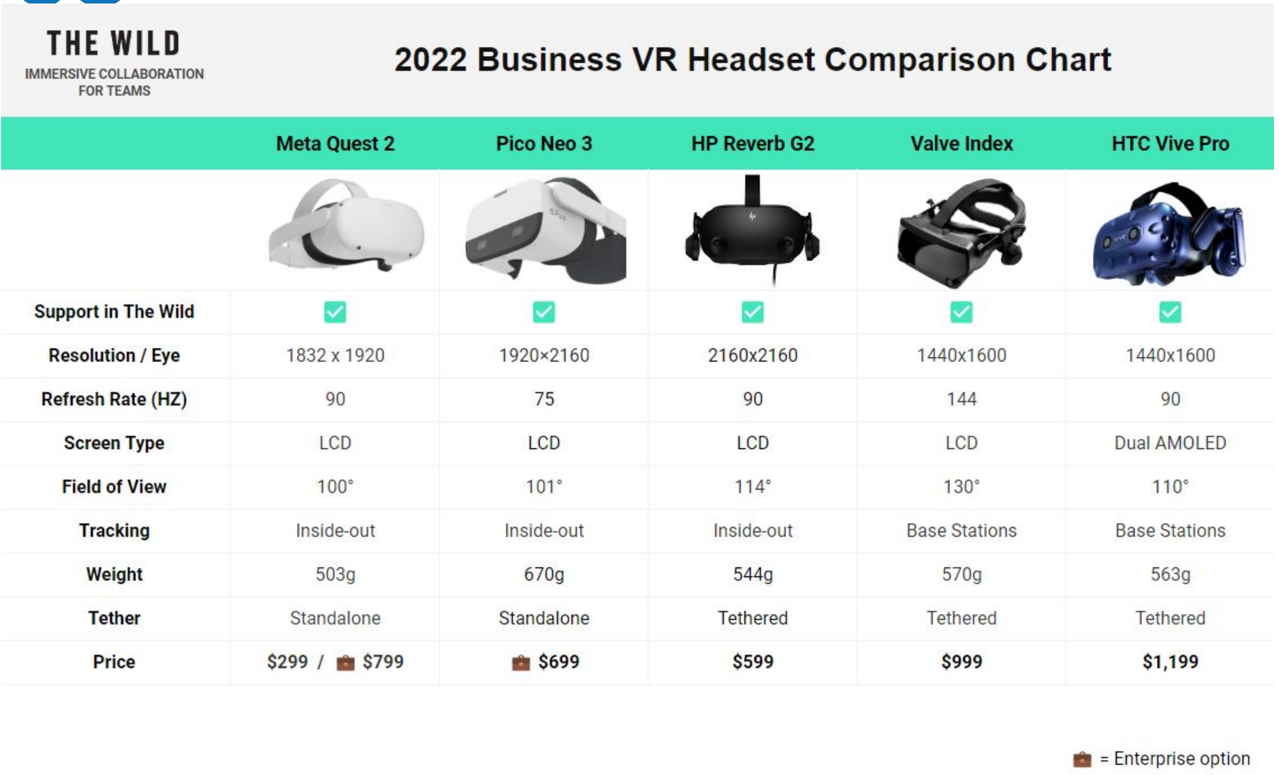


- **Using Unity's Physics Engine for Realistic Interactions**
  - Apply Rigidbody and Collider components to objects.
  - Use physics materials to control friction and bounciness.
- **Implementing Grab Mechanics and Object Manipulation**
  - Create custom grab logic using Unity Events.
  - Enhance interaction realism with dynamic hand poses and feedback.

UMKC

VR Hardware and Platforms - Overview of VR Devices

- **Comparison of Popular VR Devices**
  - **Oculus Quest**: Standalone device with inside-out tracking.
  - **HTC Vive**: PC-tethered with room-scale tracking.
  - **Valve Index**: High-fidelity visuals and advanced hand tracking.
- **Hardware Specifications and Capabilities**
  - Consider factors like resolution, refresh rate, and tracking system.



THE WILD
IMMERSIVE COLLABORATION FOR TEAMS

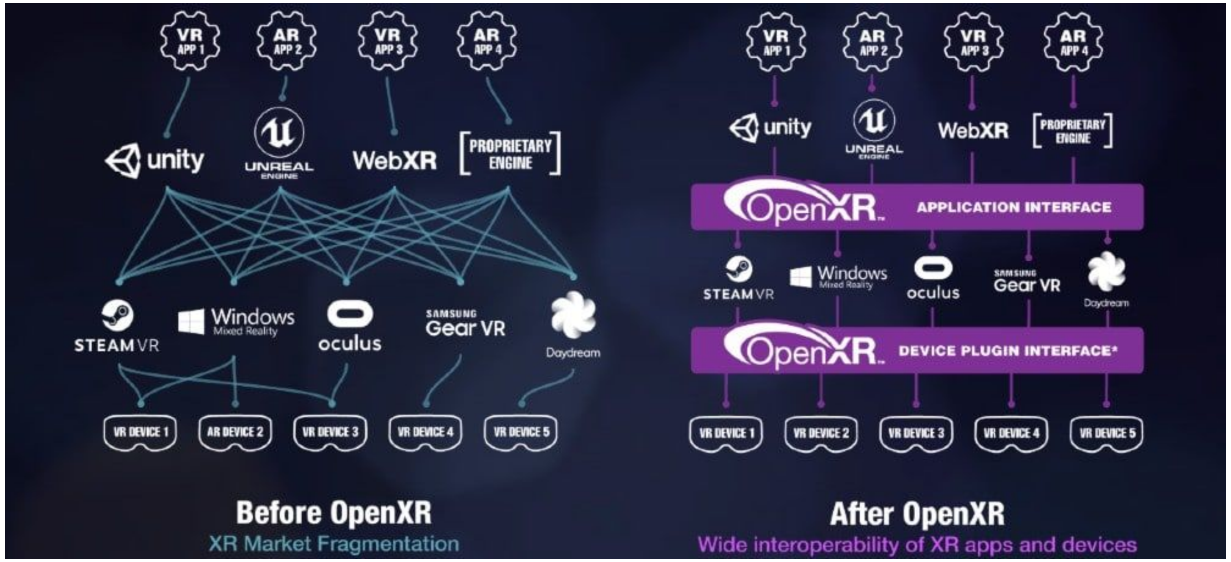
2022 Business VR Headset Comparison Chart

	Meta Quest 2	Pico Neo 3	HP Reverb G2	Valve Index	HTC Vive Pro
Support in The Wild	✓	✓	✓	✓	✓
Resolution / Eye	1832 x 1920	1920×2160	2160x2160	1440x1600	1440x1600
Refresh Rate (HZ)	90	75	90	144	90
Screen Type	LCD	LCD	LCD	LCD	Dual AMOLED
Field of View	100°	101°	114°	130°	110°
Tracking	Inside-out	Inside-out	Inside-out	Base Stations	Base Stations
Weight	503g	670g	544g	570g	563g
Tether	Standalone	Standalone	Tethered	Tethered	Tethered
Price	$299 / 💼 $799	💼 $699	$599	$999	$1,199

💼 = Enterprise option

UMKC

VR Hardware and Platforms - VR Platforms and SDKs



- **Differences Between OpenXR, Oculus SDK, and SteamVR**
  - OpenXR: Cross-platform standard for VR development.
  - Oculus SDK: Tailored for Meta's devices with unique features.
  - SteamVR: Broad device support for PC-based VR systems.
- **Choosing the Right Platform for Your Project**
  - Evaluate target audience, device compatibility, and feature requirements.

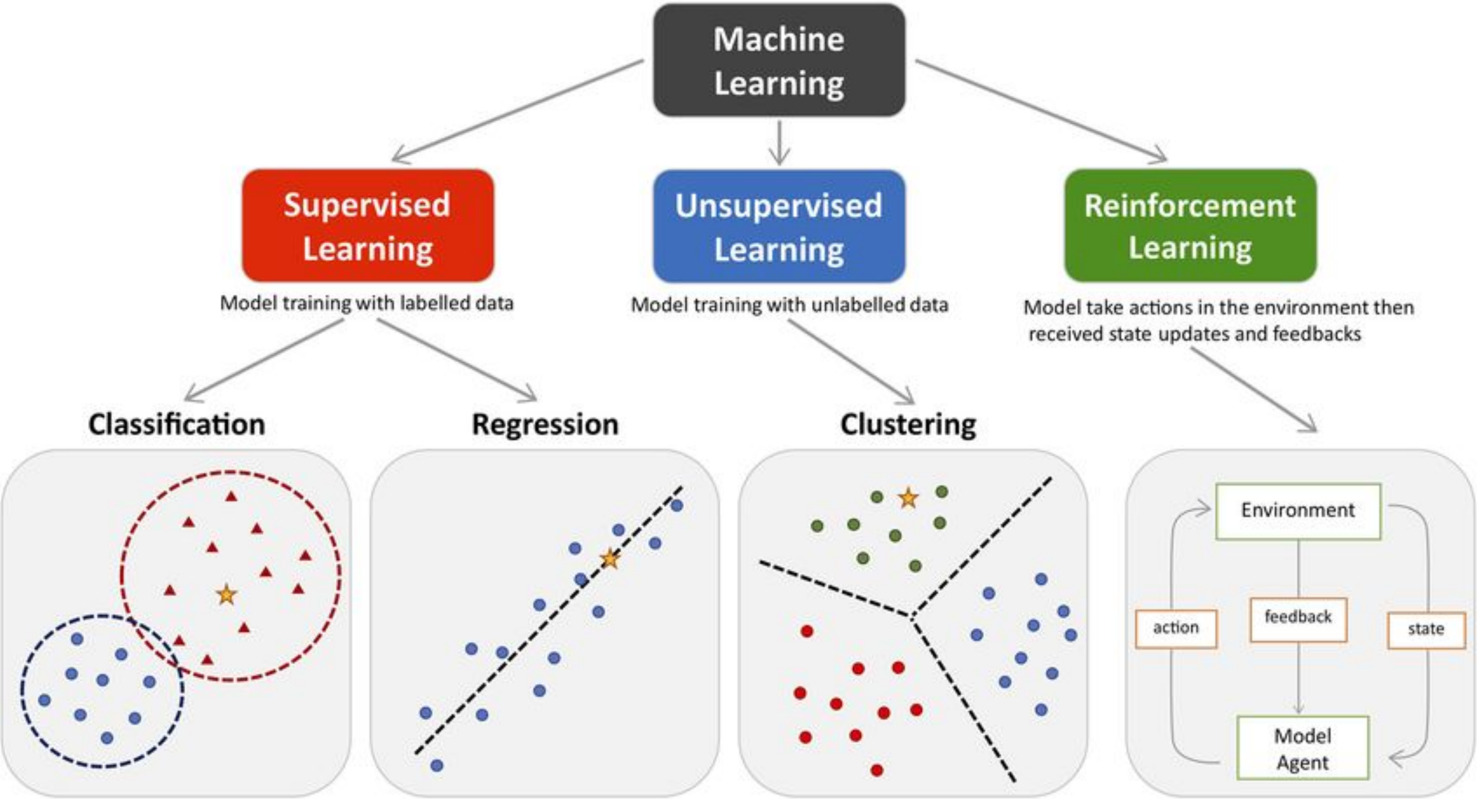
UMKC

Session 2: ML Basics/Oculus Integration

- Introduction to ML
  - What is Machine Learning
  - ML applications in AR/VR
- Pre-Trained ML Models
  - Why use pre-trained models?
  - Examples of pre-trained models
  - How models process images
- Integrating ML into Unity
  - Tools for integration
  - Real-time object detection in Unity

Introduction to Machine Learning

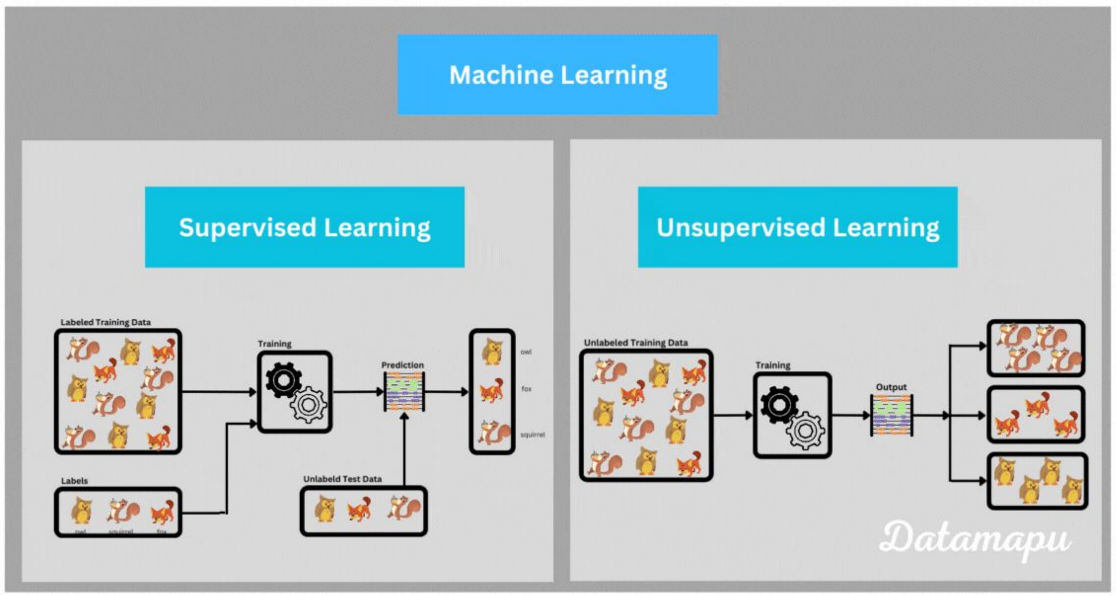
Introduction to Machine Learning



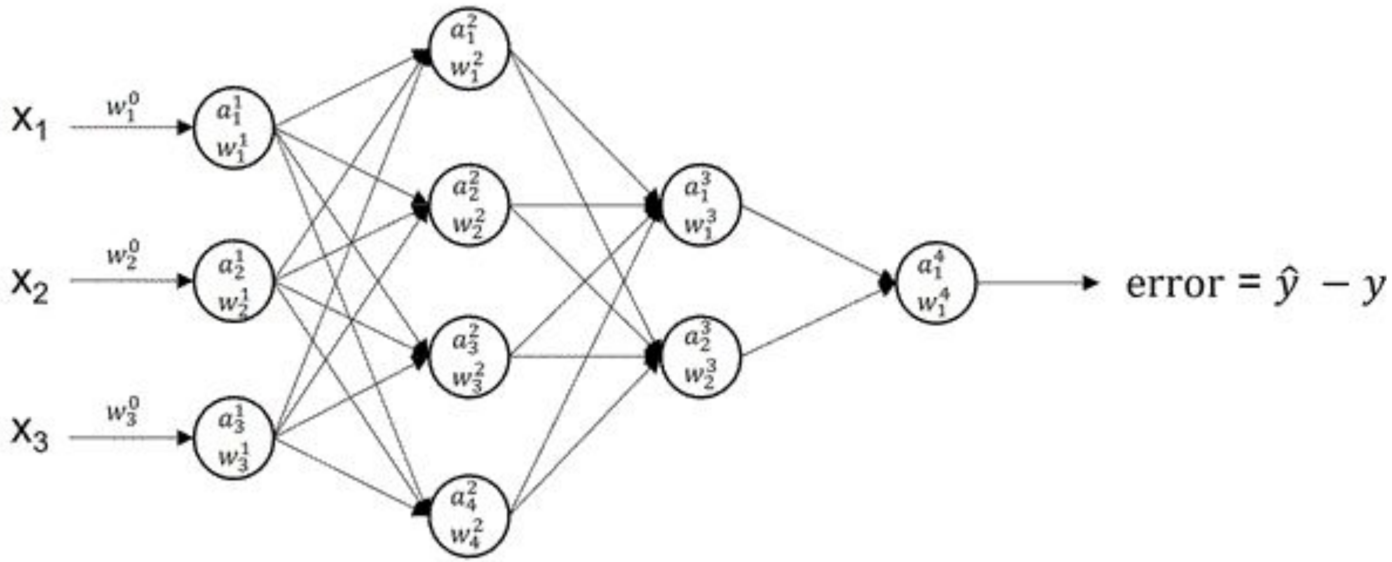
- Key concepts:
  - Definition
  - training data
  - models
  - predictions.
- Types of ML:
  - supervised
  - unsupervised
  - reinforcement learning.

Machine Learning Algorithms - Supervised vs. Unsupervised Learning

- **Definitions and Key Differences**
  - Supervised Learning: Training models with labeled data.
  - Unsupervised Learning: Identifying patterns in unlabeled data.
- **Examples of Algorithms**
  - Linear Regression (Supervised)
  - K-Means Clustering (Unsupervised)



Machine Learning Algorithms - Neural Networks

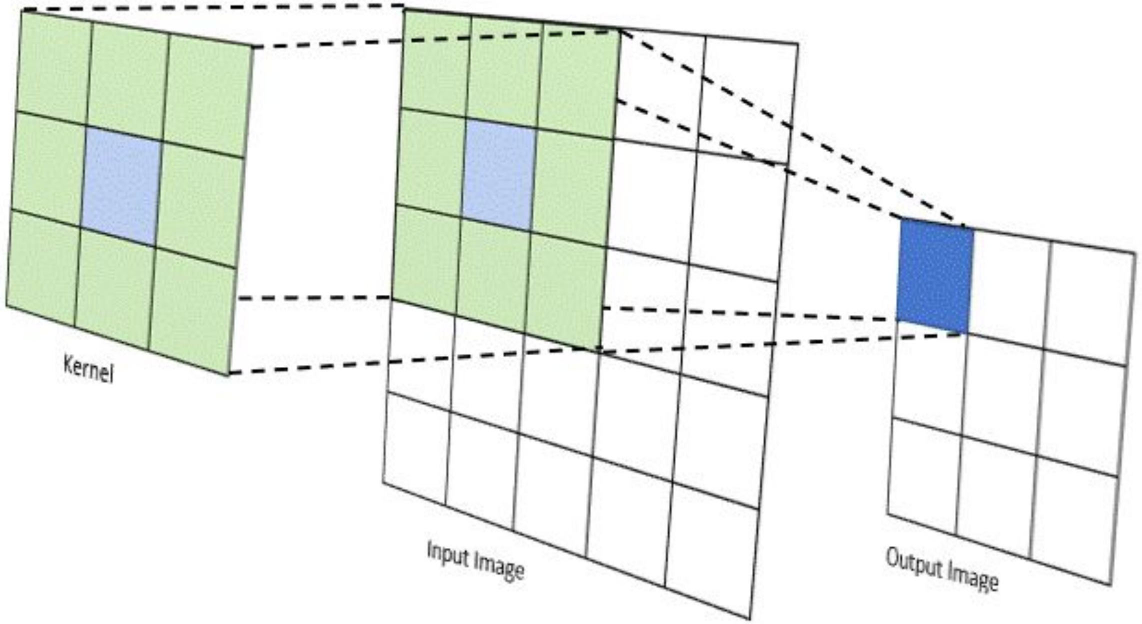


- **Understanding Perceptrons and Activation Functions**
  - Perceptron: Basic unit of a neural network.
  - Activation functions introduce non-linearity (e.g., ReLU, Sigmoid).
- **Building Blocks of Deep Learning Models**
  - Layers: Input, hidden, and output.
  - Weights and biases: Parameters learned during training.

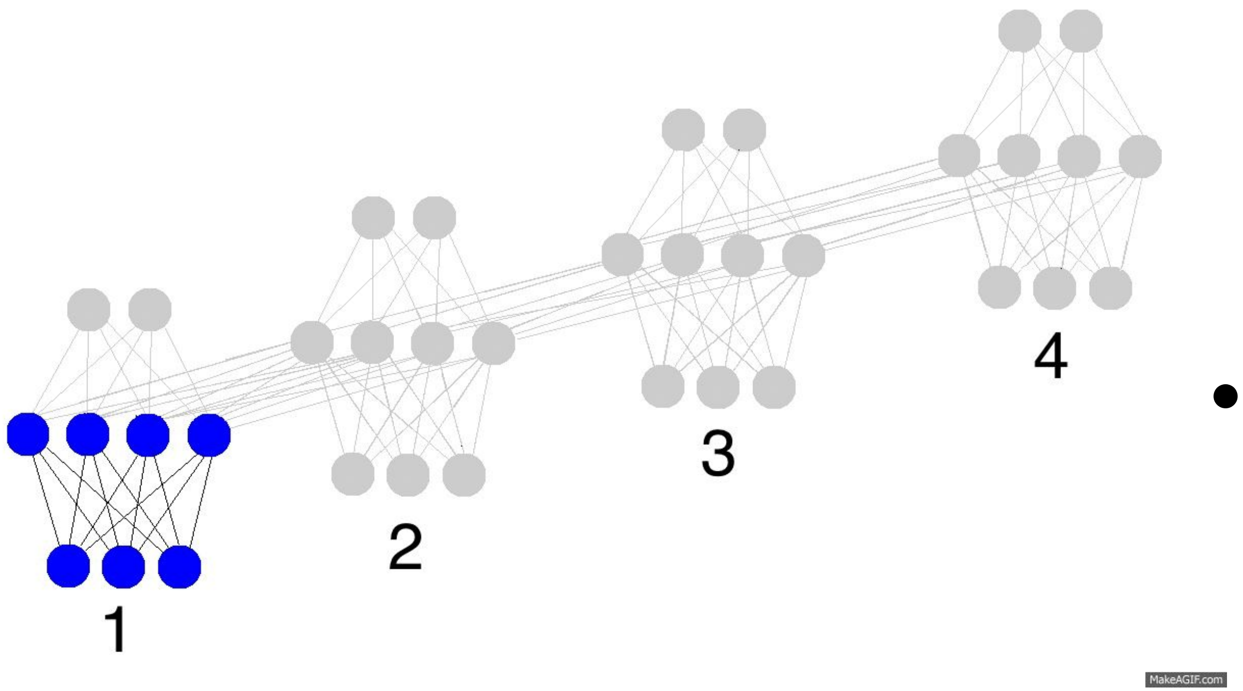
UMKC

ML Models Relevant to VR - Convolutional Neural Networks (CNNs)

- **How CNNs Process Visual Data**
  - Extract features through convolutional layers.
  - Pooling layers reduce spatial dimensions.
- **Applications in Image Recognition Within VR**
  - Object detection, texture recognition, and environmental analysis.



ML Models Relevant to VR - Recurrent Neural Networks and LSTMs

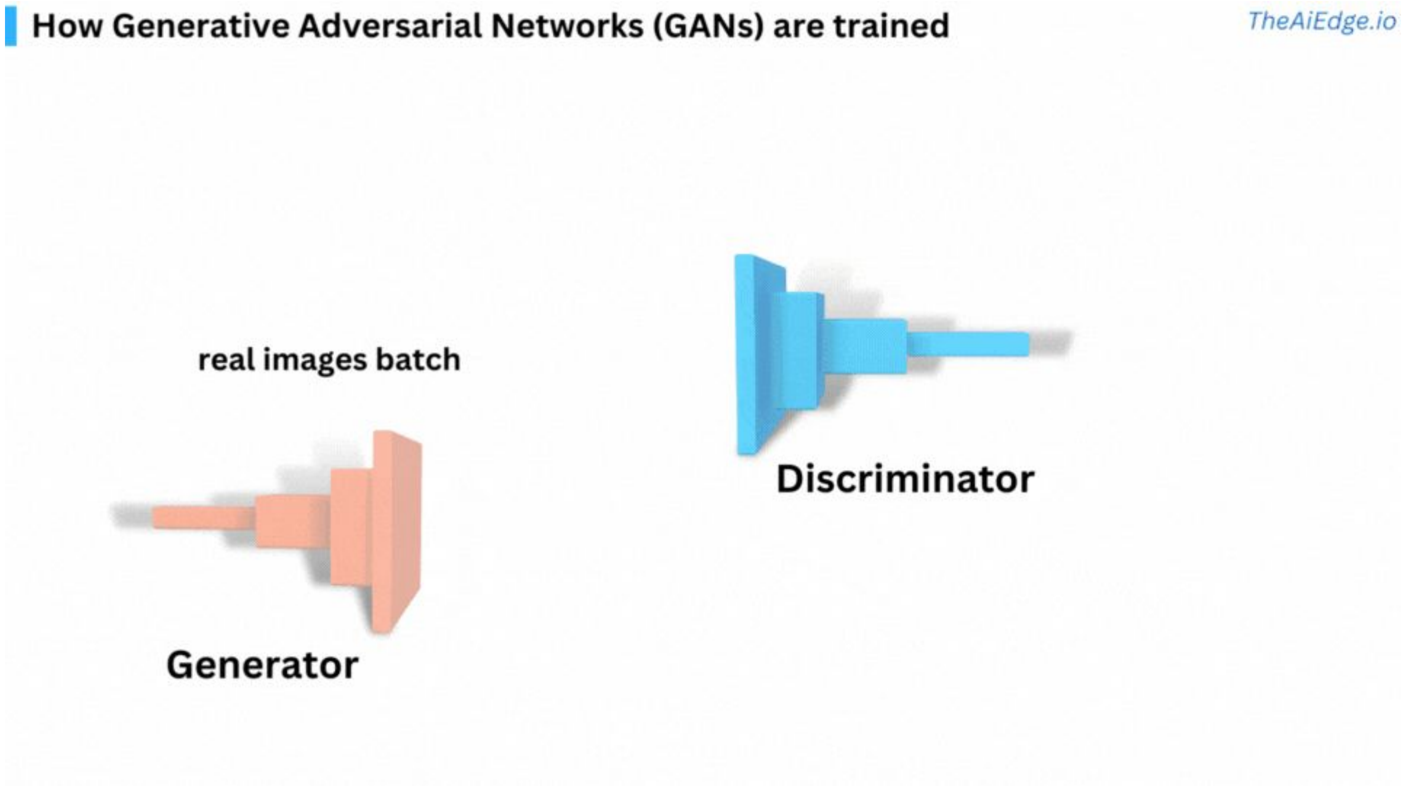


- **Handling Sequential Data**
  - RNNs process time-series or sequential data efficiently.
  - LSTMs address vanishing gradient issues for long sequences.
- **Potential Uses in VR**
  - Predicting user behavior or movement patterns.
  - Analyzing speech or text in real-time interactions.

UMKC

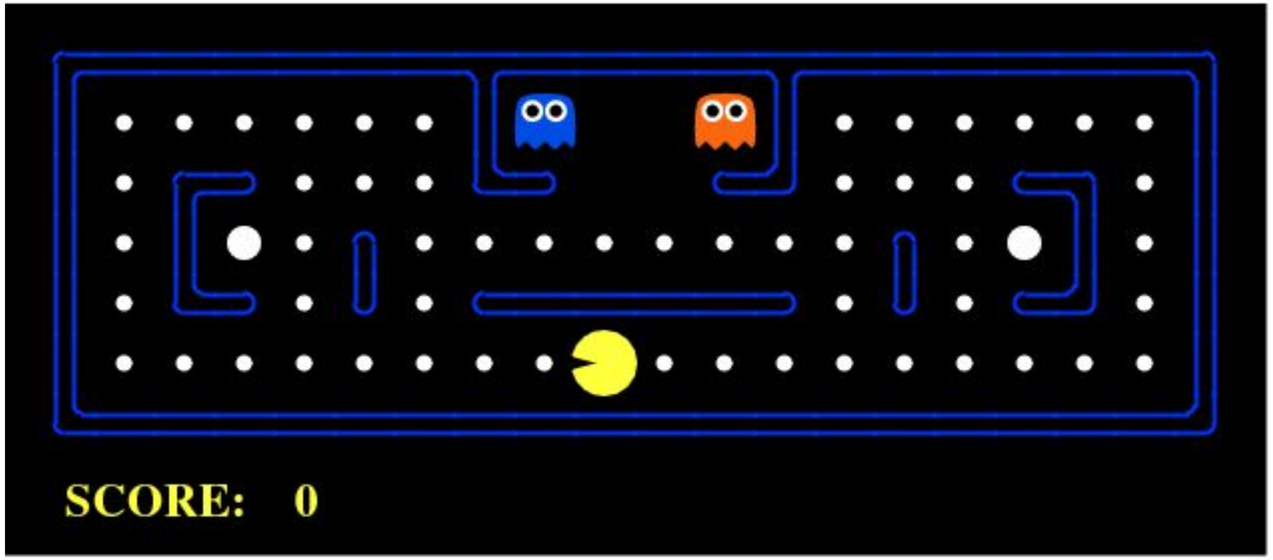
ML Models Relevant to VR - Generative Adversarial Networks (GANs)

- **Concepts of Generators and Discriminators**
  - ○ Generator: Creates synthetic data.
  - ○ Discriminator: Distinguishes real from synthetic data.
- **Generating Realistic Textures and Environments in VR**
  - ○ Use GANs to create immersive, procedurally generated landscapes.



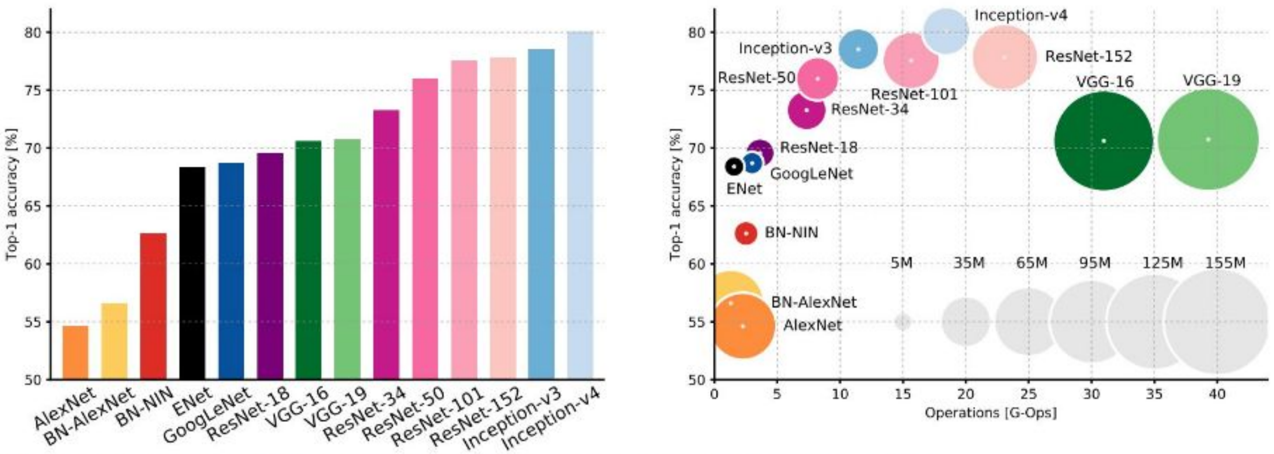
ML-Agents in Unity - Reinforcement Learning in VR

- **Concepts of Rewards and Policies**
  - Agents learn by maximizing cumulative rewards.
  - Policies define action strategies based on state observations.
- **Use Cases**
  - Autonomous NPC behavior and environmental interaction.
  - Adaptive difficulty scaling for personalized experiences.



UMKC

Integration of ML Models into Unity - Importing and Using Pre-Trained Models

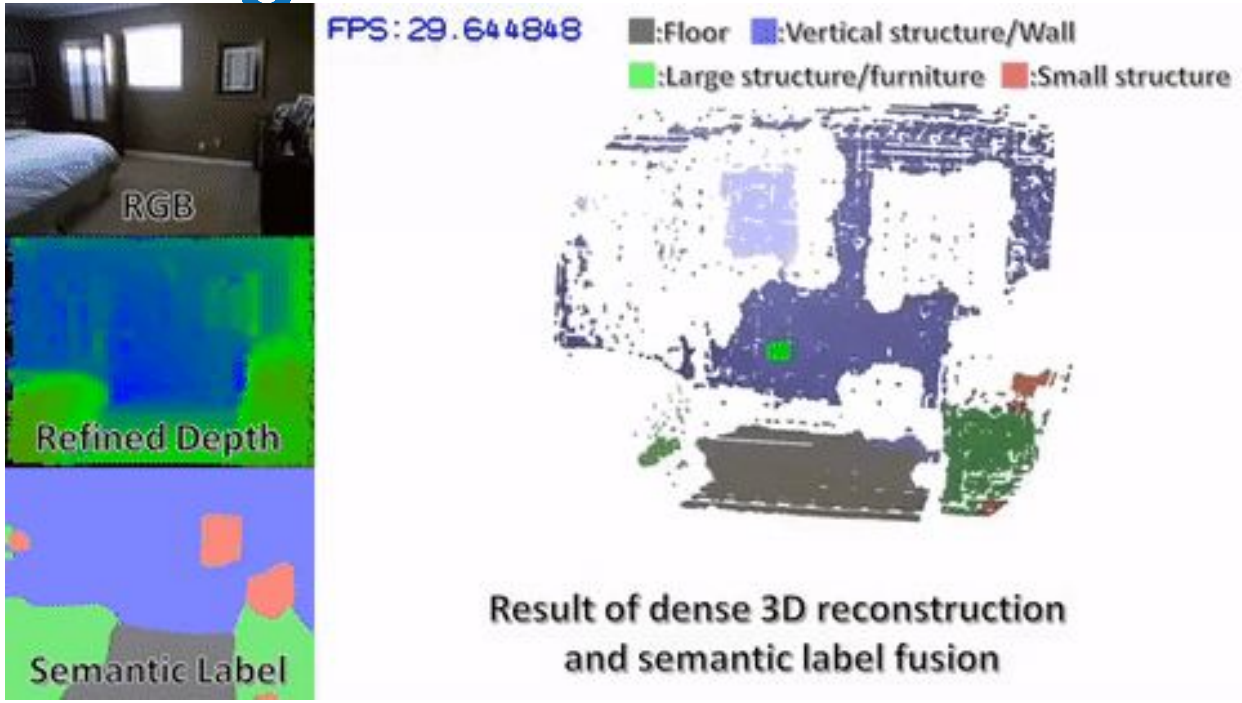


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

- **Steps to Integrate TensorFlow and PyTorch Models**
  - Convert models to ONNX format for compatibility.
  - Import using Unity Barracuda or TensorFlow for Unity plugins.
- **Converting Models for Unity Compatibility**
  - Simplify complex models for faster inference.

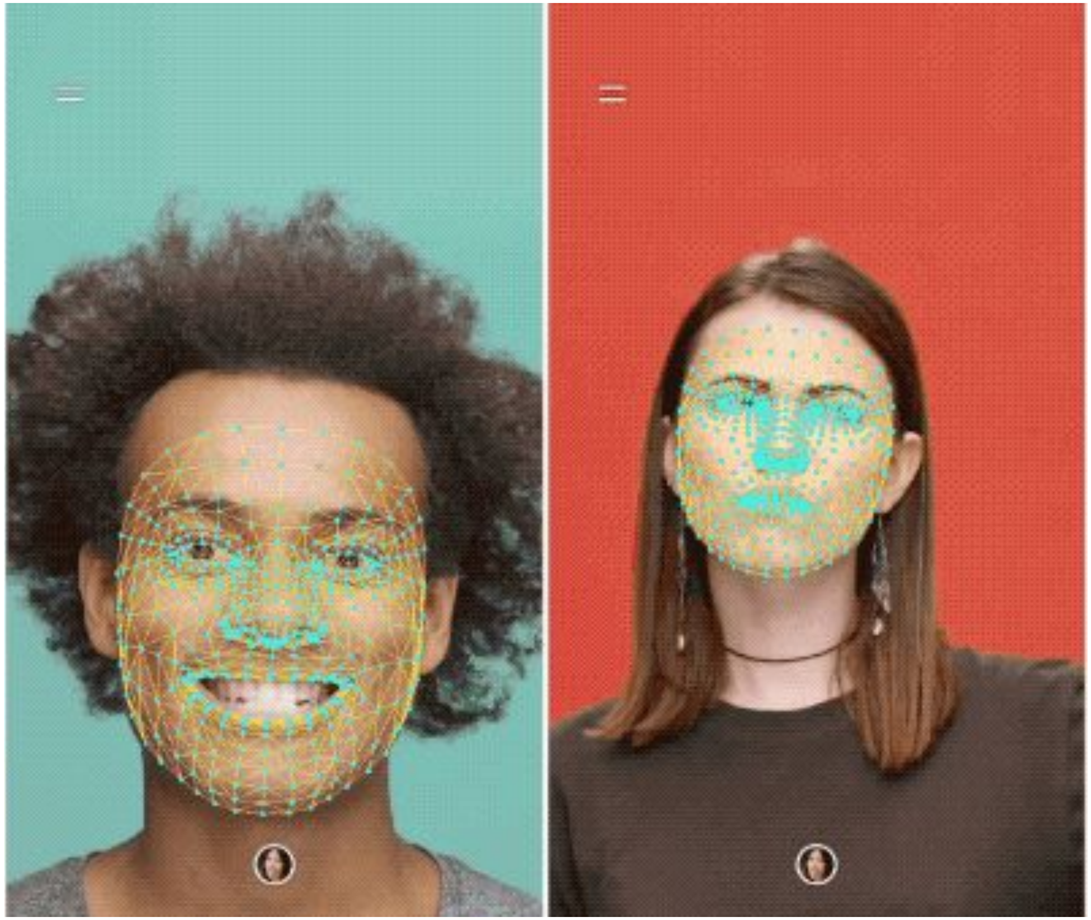
Integration of ML Models into Unity - Real-Time Data Processing

- **Techniques for Handling Live Sensor Data**
  - Use Unity’s asynchronous data pipelines.
  - Optimize for real-time performance.
- **Ensuring Low-Latency ML Inference in VR Applications**
  - Leverage GPU acceleration and efficient ML models.



Result of dense 3D reconstruction and semantic label fusion

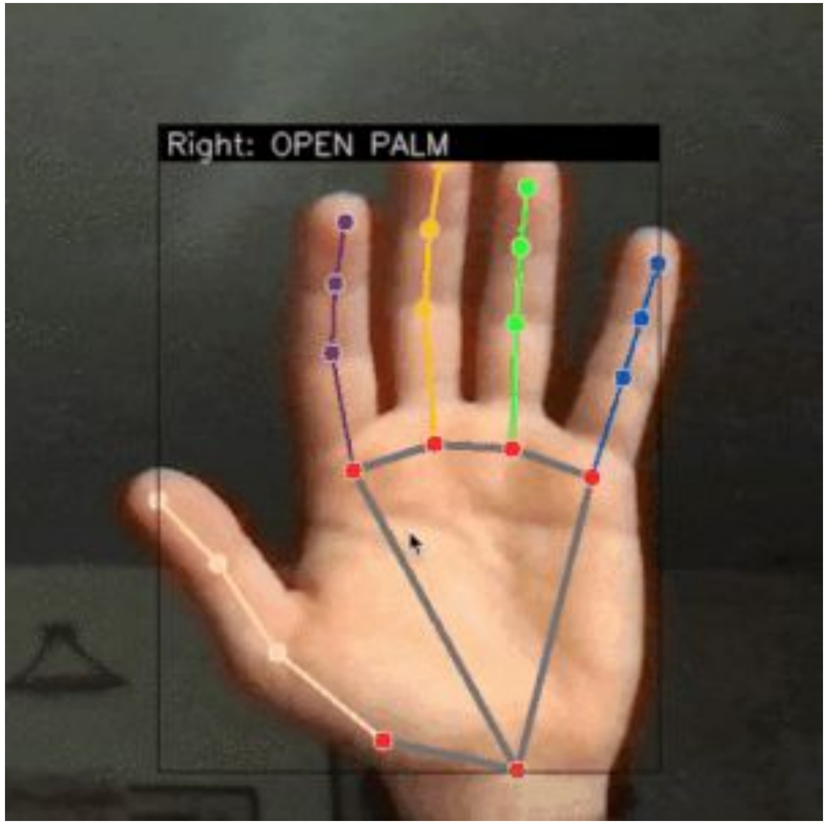
Gesture and Speech Recognition in VR - Implementing Gesture Recognition



- **Data Collection for Gestures**
  - Record user hand and body movements.
  - Annotate data for supervised model training.
- **Training Models to Recognize Hand and Body Movements**
  - Use CNNs or RNNs for feature extraction and classification.

Gesture and Speech Recognition in VR - Voice Commands and NLP

- **Integrating Voice Recognition APIs**
  - Incorporate third-party APIs like Google Speech-to-Text.
  - Process voice commands for real-time interaction.
- **Enhancing VR Interactions Through Speech**
  - Enable hands-free controls and natural communication.



Object Detection

Object Detection

- **Overview**
  - Identifies and localizes objects within an image or video.
  - Outputs bounding boxes and class labels for detected objects.
- **Applications in VR**
  - Enable dynamic interactions with real-world objects.
  - Enhance immersion by recognizing and reacting to user surroundings.

Object Segmentation

Object Segmentation

- **Overview**
  - Classifies each pixel in an image as belonging to a specific object.
  - Provides more detailed object understanding compared to detection.
- **Applications in VR**
  - Used for scene understanding and environmental mapping.
  - Supports realistic occlusion of virtual objects with real-world elements.
- **Popular Tools and Models**
  - YOLO and SSD for detection.
  - Mask R-CNN for segmentation.
  - Integration with Unity through Barracuda or TensorFlow plugins.
-