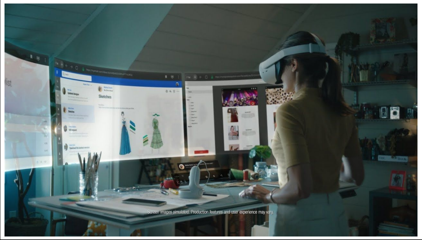
CMSC838C & ENEE759N: Advances in XR

TuThu 12:30-1:45 cs.umd.edu/class/spring2024/cmsc838C



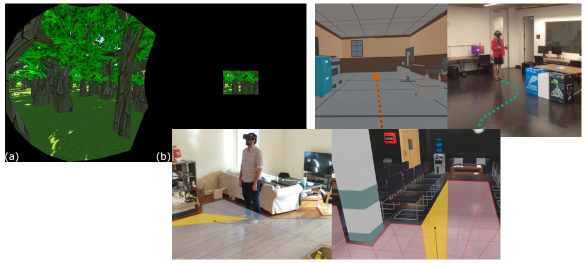
Ming C. Lin
http://www.cs.umd.edu/~lin/

- BS, MS, PhD in Electrical Engineering & Computer Science *University of California at Berkeley*
- B. Mersky & CapitalOne E-Nnovate Endowed Professor 2021-
- UMD Distinguished University Professor, 2019-
- Former Elizabeth Iribe Chair of Computer Science @ UMD, 2018-2020
- J.R. & L.S. Parker Distinguished Prof. Emeritus @ UNC Chapel Hill
- ACM, IEEE & Eurographics Fellow; ACM SIGGRAPH Academy
- Areas of Research: **Virtual Reality, Robotics, AI/ML/Vision & Autonomy** with focuses on *physically-based modeling/simulation*, *multimodal interaction* (haptics & audio technology), *animation*, and *human-computer interaction*; applications in autonomous driving, virtual try-on, healthcare, digital design, rapid prototyping, and personalized fabrication/cybermanufacturing

Niall Williams

niallw.github.io

- 5th PhD student in the GAMMA lab.
- Undergrad from Davidson College.
- Research interests in XR, human perception, HCI, and robotics.



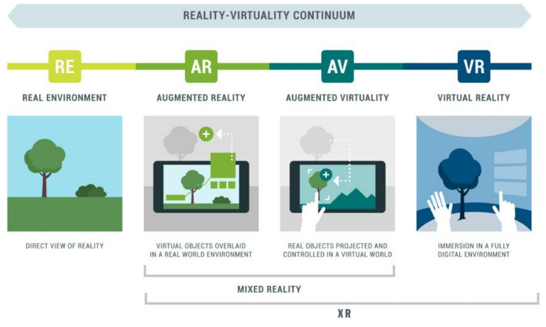
Geonsun Lee

http://www.cs.umd.edu/~gsunlee/

- 4th year PhD student in the GAMMA lab
- BA/BE, ME from Korea University
- Research interests in novel interaction interfaces for multi-user, social AR/VR systems

What is Extended Reality (XR)?

- Software & hardware that replaces or mixes real world stimuli with synthetic
- Different types of XR mostly differentiated by **display** and **tracking** methods


REALITY-VIRTUALITY CONTINUUM
RE
REAL ENVIRONMENT
AR
AUGMENTED REALITY
AV
AUGMENTED VIRTUALITY
VR
VIRTUAL REALITY
DIRECT VIEW OF REALITY
VIRTUAL OBJECTS OVERLAID IN A REAL WORLD ENVIRONMENT
REAL OBJECTS PROJECTED AND CONTROLLED IN A VIRTUAL WORLD
IMMERSION IN A FULLY DIGITAL ENVIRONMENT
MIXED REALITY
XR

Sutherland 1965: The Ultimate Display

It (the Ultimate Display, referring to VR and AR) is a looking glass into a mathematical wonderland……. If the task of the display is to serve as a looking-glass into the mathematical wonderland constructed in computer memory, it should serve as many senses as possible. - Ivan Sutherland, 1965

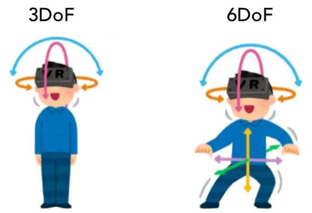
- A virtual world, through a HMD, appeared realistic through augmented 3D sound & tactile feedback
- Computer hardware to create the virtual word and maintain it in real time
- The ability users to interact with objects in the virtual world in a realistic way

Sutherland 1968: First XR headset

- “A head-mounted three dimensional display” Ivan Sutherland 1968
- Concept first introduced by a concept paper to DARPA in 1965

Degrees of Freedom (DoF)

- How many types of motion are tracked
  - Translational: x, y, z
  - Rotational: pitch, yaw, roll
- 3DoF in XR usually means rotational tracking only; 6DoF is rotation+translation
- Applies to display device (head-mounted display [HMD]/phone) and any other controllers

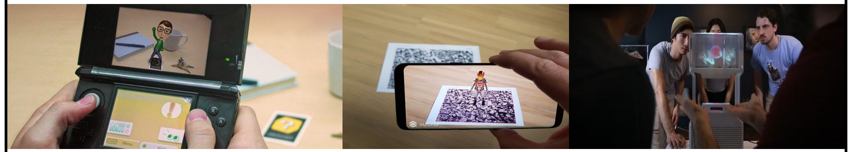


Virtual Reality (VR)

- Only handles virtual world and assumes real stimuli completely replaced with synthetic stimuli (e.g. Oculus Rift, HTC Vive, Google Cardboard)
    - If the synthetic stimuli not handled correctly, people can get simulator sickness, lack of "presence" or immersion, or worse (e.g. trauma)
- Main "illusions" needed for immersive experience (Mel Slater 2009):
    - **Place illusion**: feeling like you're in the virtual world and not the real world
    - **Plausibility illusion**: feeling like what happens in the virtual world is really happening

Augmented Reality (AR)

- Overlays digital 'elements' onto real world, including graphics, images, video, sound, GIS data, text, animation, etc.
- Through head-mounted display, hologram, video-passthrough, etc.
- Focus on simple experiences or information easily available
- e.g. Google Glass, old Snap Spectacles, marker-based tracking–Nintendo 3DS, volumetric AR display



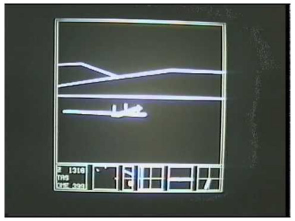
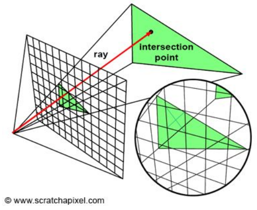
Mixed Reality (MR)

- Merge of real world and virtual world, co-existing & interacting in real time
- Focus on practicality, productivity, integration with day-to-day life
- e.g. tracking real world features & hands makes the **Oculus Quest** an MR headset

XR Trends Over Time

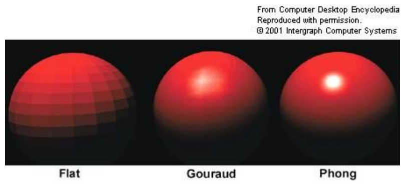
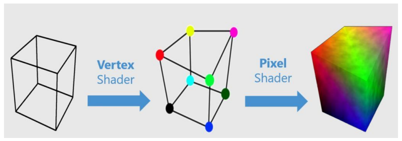
XR Trends Over Time: Pre-1970s

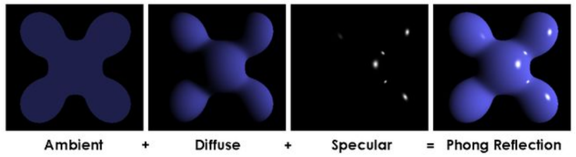
- Fundamental tracking, rendering, & display technology (esp. For simulators)
- First VR headsets (stereo headsets in the 1800s!)

XR Trends Over Time: 1970s

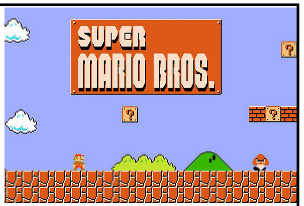
- Fundamental 3D graphics technology (e.g. Phong shading)


From Computer Desktop Encyclopedia
Reproduced with permission.
© 2001 Intergraph Computer Systems
Flat
Gouraud
Phong




XR Trends Over Time: 1980s

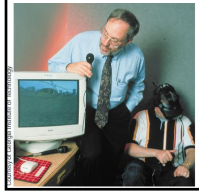
- Physically-Based Rendering (PBR)
- VR Simulators (esp. flight)
- Interactive Games
- Display adapters (old GPUs)
- Multimodal XR apps



microsurface
chart shown in linear space
0.0 - rough
glossy - 1.0
plaster linear: 0.14
wood linear: 0.32
plastic (glossy) linear: 0.46
steel linear: 0.75
mirror linear: 1.00

XR Trends Over Time: 90s & Early 2000s

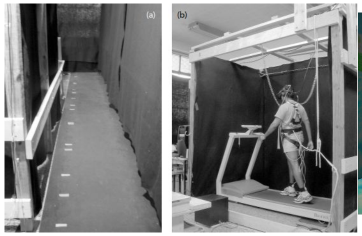
- Clumsy but effective XR headsets
- Boom of XR psychological studies
- 3D Game Engines
- Interactive 3D graphics
- Collision detection & interactive physics
- Boom in areas like haptics & locomotion

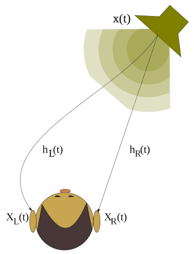


3 NASA-Houston's "Charlotte" Virtual Weightless Mass lets astronauts practice handling weightless massy objects.

XR Trends Over Time: Late 2000s

- Research on different modalities, audio, locomotion, rendering→all evolved close to what they are today
- Unity, Unreal 2/3, idEngine, Source Engine, Autodesk Maya, etc. established many interactive 3D graphics conventions



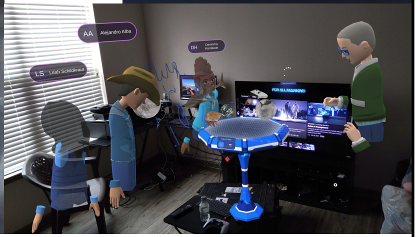


XR Trends Over Time: 2010s

- Good mobile technology
- Strong GPUs
- Decent commercial headsets
- Industry getting involved

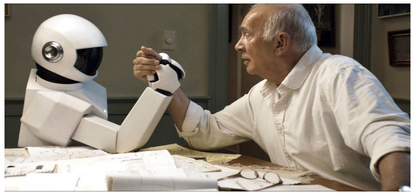
XR Trends Over Time: Early 2020s

- Everyone is making XR headsets
- Metaverse hype
- Social XR



XR Trends Over Time: Predicting Late 2020s

- Focus on application development
- Procedurally-generated content
- Better, large-scale HRI
- Virtual assistants
- Attempts at neural interfaces



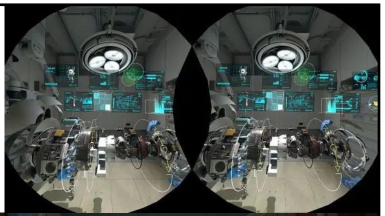
Challenges in XR

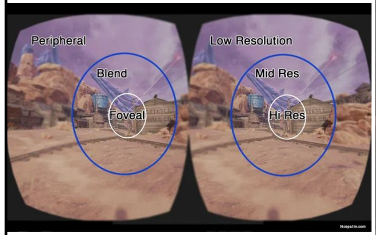
Challenges in XR: HMD Design

- Weight
- Physical dimensions & portability
- Field of view (FOV)
- Battery vs. computing power
- Pixel opacity
- Optics

Challenges in XR: Graphics

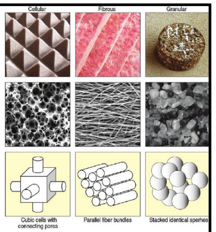
- Realistic materials
- Foveated rendering & optimization
- Physically-based interactions

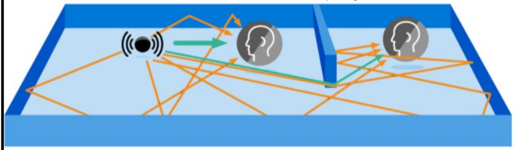


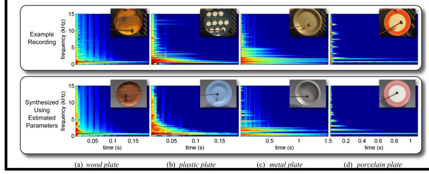


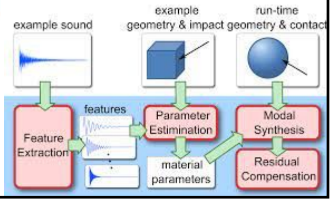
Challenges in XR: Audio

- Procedural generation vs. sound synthesis (esp. on mobile)
- Sound propagation & 3D acoustics (esp. on mobile)
- Personalized 3D audio display



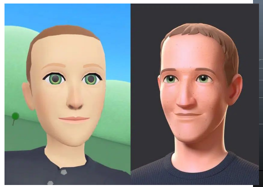
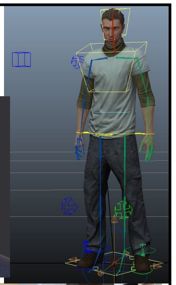






Challenges in XR: Virtual Humans

- Uncanny valley
- AI behavioral modelling
- Social cues
- Procedural animation & rigging
- Generative AI models
- UV mapping/texturing

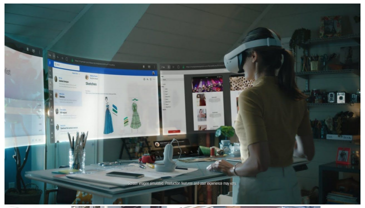



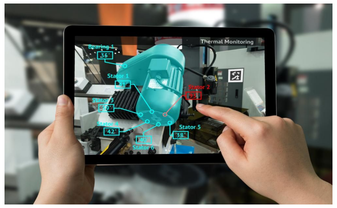
Challenges in XR: Content

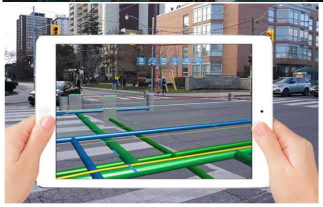
- Procedural generation
  - Doesn't only apply to environments! Used to speed up getting good textures, models, animation, audio, etc.
- Generative AI models
- User-generated vs. company-generated

Challenges in XR: Application Design & Utility

- How to keep people using it?
- How to reduce barrier of entry?
- How to integrate with people's lives?

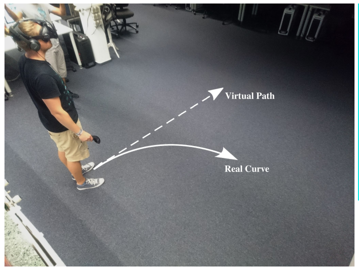






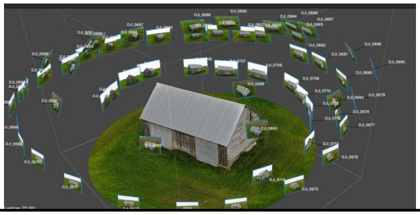
Challenges in XR: Natural Locomotion in VE

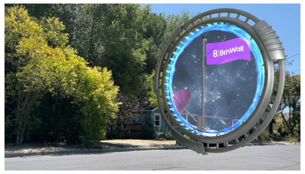
- Getting people to walk around naturally with limited space
- Handling sickness & perception (knowledge of the "illusion" changes its effect!)



Challenges in XR: Tracking & Reconstruction

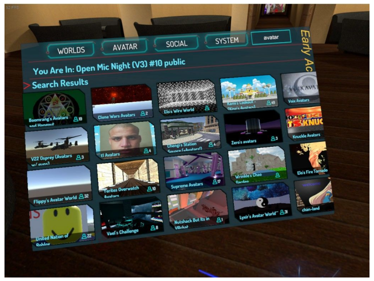
- **Local** tracking of headset more or less solved
- **Global** tracking still challenging
  - How to register virtual environments to real ones?
  - How to link the environments together?
  - How to overlay real and virtual worlds?
- Reconstructing people, real environments, etc. to make virtual world as convincing as real world
  - Photogrammetry is the state-of-the-art for high-fidelity asset creation but it's incredibly slow & tedious





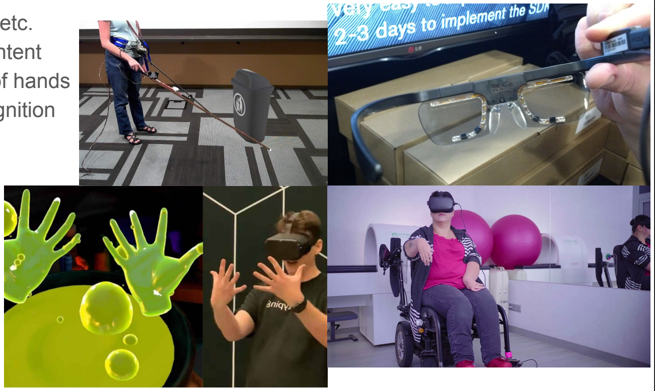
Challenges in XR: SW/HW Compatibility

- Huge issue with Metaverse
- How can assets trivially move between XR applications as if they are part of a unified metaverse?



Challenges in XR: Natural Interactions with Full-Body Tracking

- Eyes, hands, etc.
- Predict user intent
- HMD's FOV of hands
- Gesture recognition
- Accessibility



Current Research Directions

Research Directions: Natural Virtual Locomotion

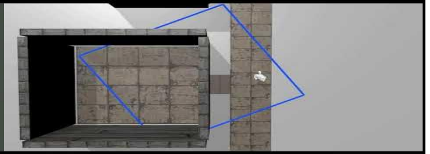
- Use distortions to trick people into walking differently than they realize.
- Maximize real walkable space.
- Must support a range of users and environment shapes.

Geometric Algorithms for Modeling, Motion, and Animation

Alignment-based Redirection Controller

Niall L. Williams, Aniket Bera, Dinesh Manocha
University of Maryland, College Park
https://gamma.umd.edu/arc/
niallw@cs.umd.edu

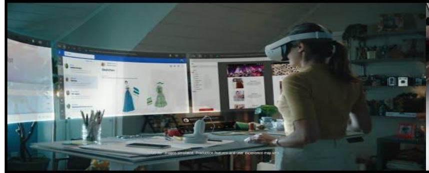
IEEE VR 2021 VIRTUAL



Research Directions: Intelligent Virtual Environments

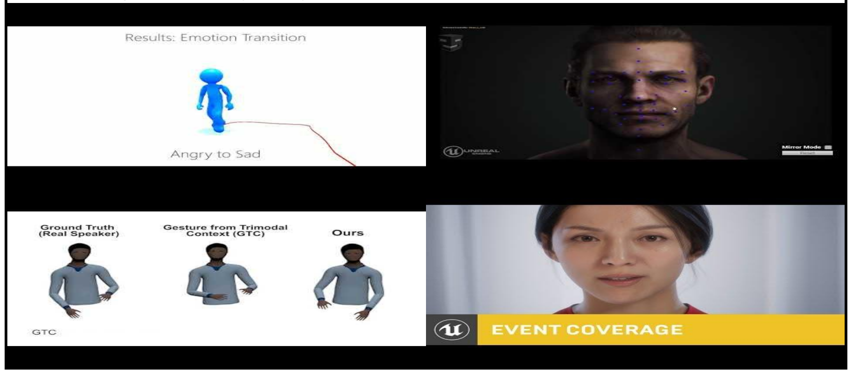
- Future of work
- Context-aware interaction
- Integration with daily life

Here is the kitchen table. There's a glass top and eight chairs.
There's also a display cabinet behind it. And there are three windows behind it. That go into the backyard. And here we also see the patio door.
You can see it over here, too, the water is just kind of piling up in the middle of the driveway.



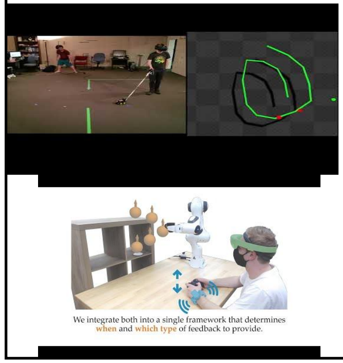
Research Directions: Emotive Virtual Humans

- Realistic virtual assistants w/ procedural animation
- Passing the uncanny valley


Results: Emotion Transition
Angry to Sad
Ground Truth (Real Speaker)
Gesture from Trimodal Context (GTC)
Ours
GTC
EVENT COVERAGE

Research Directions: Human-Robot Interaction in XR

- Robot dogs to train the visually-impaired spatial mapping skills
- Robots in warehouses for collaborative XR


We integrate both into a single framework that determines when and which type of feedback to provide.

Research Directions: Metaverse

- (Seamlessly-Integrated Real & Virtual Environments)
- **Human-centric mapping between spaces**: pocket dimensions/ portals/ symbolic links, human-centric motion planning, context, behavior prediction
- **"Internet of XR"/ "IoX"**: sensor/device fusion, connected HMDs, maximizing knowledge of real world

Research Directions: Intelligent, Responsive XR

- Immersive, XR Interfaces for autonomous systems
  - Driverless vehicles
  - Autonomous drones
  - Intelligent sensor networks
- XR Systems for rapid design and personalized fabrication/manufacturing
- XR Systems for remote collaboration, tourisms, social events
- XR Systems for personalized healthcare and wellness
- XR Interface for virtual try-on
- XR Systems for social good
  - Immersive scenario replay for Police Training, Bias Training, AI for Fairness, etc.
- Audio-Visual Reconstruction

Course Information

Course Goals

- Understand **multimodal XR design**
  - Basic principles of audio, haptic, and visual rendering
  - How multiple modalities work together and interact with each other
  - Challenges of Metaverse & future of XR
- Understand “realism”, “illusion”, and “presence”
  - Roles of locomotion
  - Sense of ‘being there’
  - Avatar & object animation/simulation
  - Social interaction via virtual humans
- Application design & development
- Use the state-of-the-art APIs & XR tools

Course Components

- Lectures & Participation
- Homework Assignments [30%]
- Final Project [35%]
- Midterm & Final exams [35%]

Lectures

- Will be recorded, but with emerging tech, the discussion is very beneficial.
- Rough sequence:
  - Basic 3D graphics
  - Basic game development
  - Immersion & presence
  - Virtual Locomotion
  - 3D Audio
  - 3D Animation
  - Virtual agents & AI
  - Tracking & Reconstruction
  - Displays, Optics, Lenses, etc.
  - Advanced Topics (light fields, haptics, olfactory, GANs, holography, etc.)
- Invited speakers from industry & academia

Game Engines

- **Game engines**: powerful, realtime, interactive, multimodal 3D applications
- De facto standard for XR development and most interactive 3D consumer programs
- We will provide support for Unity development, but not for Unreal Engine.
  - Use Unreal at your own risk

Headset Logistics

- The plan:
  - Oculus Quest rental system
  - We need to share!
  - Use your own device, if you own one
- Details being worked out, more info to come
- Anyone using personal devices?

Assignments

- Address the major modalities & concepts of XR design
  - Goal is **breadth** of experience; final project is opportunity to dive deep
- Build on each other to result in complete multimodal XR application
- Planned sequence
  - **A1:** Introduction to XR and game engines
  - **A2:** Interaction with virtual environments
  - **A3:** 3D audio for virtual environments
  - **A4:** Natural virtual locomotion
  - **A5:** Virtual avatars and agents
  - **A6:** Introduction to augmented reality
- May include non-technical readings
- Grad section will have slightly extra requirements & higher expectations
- Submission includes code/Git repo & video showing it works

Introducing Assignment 1

- Introduction to the Unity game engine
- 2 fundamental readings:
  - **"What's Real About VR?"** Fred Brooks 1999
  - **"Recent advances in augmented reality"** Ron Azuma 2001
  - Will answer some short-answer questions about them

Final Project

- Will draw upon the concepts you learned in the course
- Planning a competition with prize winner(s)
- Details to be announced later

Exams

- Midterm
- Final
- Relatively high-level design questions
  - E.g. How would you design an XR app for a particular application? What are the important considerations?
  - E.g. Someone wants to make an XR app a certain way; what problems are they not addressing correctly?
  - E.g. Why do we need to do [a particular thing] for XR?
  - E.g. How did [this paper] likely accomplish [this feature]?
- No programming, but will ask high-level questions about technical problems
  - E.g. why are VR lenses designed the way they are
  - E.g. what do waveguides and lightboxes do in near-eye AR displays?

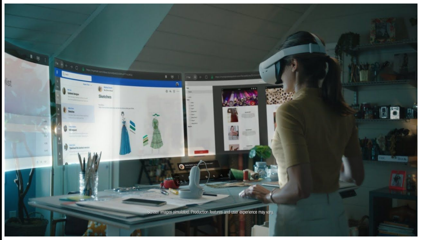
Collaboration & General Policies

- Pairs for regular assignments
- 1-5 people in a group for final project
  - Final report must describe what each team member does
  - Each will be graded based on individual contributions specified
  - Group efforts should show higher levels of complexity and components
- Recommendation: try to stay with the same OS & platform combination
- ***No collaboration on in-class exams***
- Lateness
  - Total of 5 “late credits” (1 credit = extend deadline by 24 hours)
  - Linear late penalty, up to 3 days
  - Late assignments due to illness or unexpected events can be excused with doctor’s notes or other forms of written indication

Office Hours

Instructor:	Immediately after class or by appointment
TA’s:	To be announced

Questions?



Course Website:

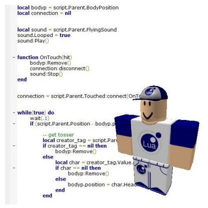
https://www.cs.umd.edu/class/spring2024/cmsc838C

Outline

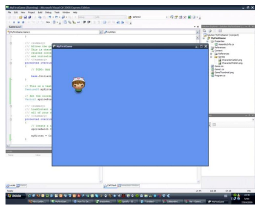
- Introducing instructional team
- Components of class
- Interactive 3d graphics/Games/ game engines
- metaverse/digital ecosystem
- What is xr? Ar vs vr vs mr. slater. Plausibility vs place illusion makes sense for vr/mr, practicality makes more sense for ar
- Problems in xr: rendering, audio, animation, virtual agents & AI uncanny valley, procedural generation/GANs, multimodal design, tracking, reconstruction, headset weight/design/fov/power, locomotion, virtual world mechanics/suspension of disbelief, compatibility (vrchat addresses well)/walled garden, eye/hand-tracking
- Class: goals
- Timeline
- Logistics, headsets, OS (build os and phone)
- Grade breakdown
- Assignments
- Final project
- Exams
- Collaboration?
- Office hours tr
- A1. come to office hours if any problems
- Mask policy

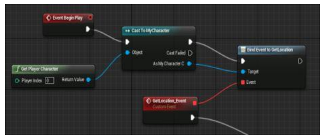
Nick

- BS/MS student, graduated UNC 2019 with BS in CS and BA in Chinese
- Programming games for ~11 years
  - Lua/OG Roblox -> VB/text-based -> Java/Android -> C#/XNA/Unity -> C++/Blueprint/Unreal 4 -> Python/Visualization



```
local bodyp = script.Parent.BodyPosition
local connection = nil

local sound = script.Parent.FlyingSound
sound.Looped = true
sound:Play()

function OnTouch(hit)
	bodyp:Remove()
	connection:disconnect()
	sound:Stop()
end

connection = script.Parent.Touched:connect(OnTo

while(true) do
	wait(.1)
	if (script.Parent.Position - bodyp.p

		-- get tosser
		local creator_tag = script.Par
		if creator_tag == nil then
			bodyp:Remove()
		else
			local char = creator_tag.Value.
			if char == nil then
				bodyp:Remove()
			else
				bodyp.position = char.Head
			end
		end
```

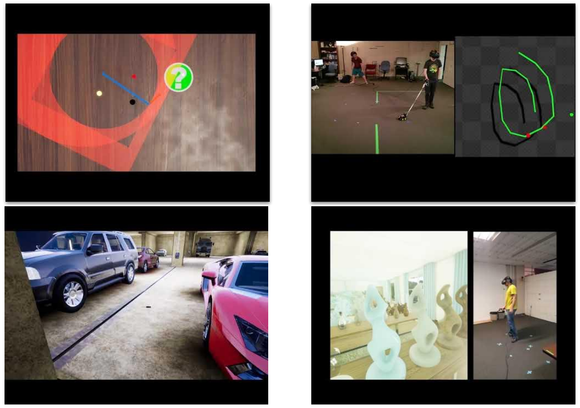




(reverse image search for image sources)

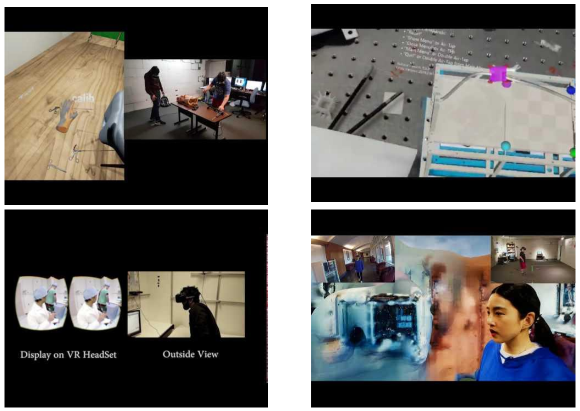
GAMMA/Oculus

3D audio, synthesized audio, AI, perception, VR



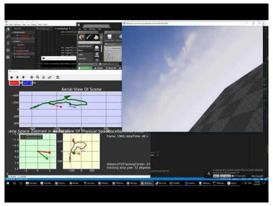
Telepresence

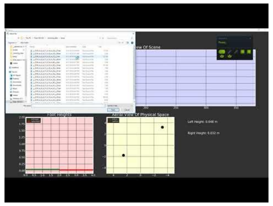
3D reconstruction, multi-user systems, AR, surgical apps

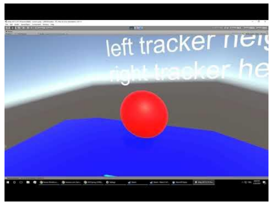


EVE

Customized locomotion, CAVEs, walking-in-place, distortion







Nick

- From Brooklyn, NYC
- Travel a lot!
    - Studied and interned: Beijing, HK, Seoul

CAPITAL NORMAL UNIVERSITY CNU 1954

CUHK 香港中文大學

YONSEI UNIVERSITY 1885 연세대학교

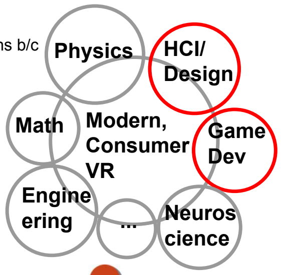
YONSEI UNIVERSITY

Components of the class

- Game development
- Virtual reality (VR)
- Human-computer interaction (HCI)

Related somehow?

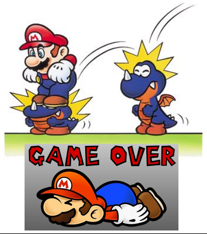
Complex interactions b/c all circles



What is a game? How are games different from other apps?

Games

- Adapted from Diane Pozefsky's COMP585:
  - **Play**: something not required to do...leisure
  - **Pretend**: synthetic stimuli
  - **Goal**: some reason to play, some way to win
  - **Rules**: how to reach the goal, gameplay, etc.



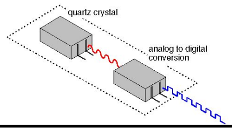
What is game dev? What makes it different from other types of dev?

My Parentheses Notation

- E.g. (most) vr devices
- Implies there are technicalities
- Refers to the subset we care about in this class

(Most) (Modern) Game Engines

- Essentially realtime systems
  - **clock** (in this case tied to framerate) handles continuous logic
  - **Synchronization** between subsystems (physics, listeners, collision detectors, ray-tracing, etc.)
  - Tries to be as synchronous as possible (e.g. each frame lasts the same amount of time)
  - As opposed to solely reacting to input through **callbacks**, like in most mobile apps, text-based games, etc. which have no continuous logic beyond an OG clock
- Game engines are mostly **fully-featured APIs** containing many sub-APIs
  - Physics, collision logic, rendering, materials, optimized data structures, AI, audio etc.
- **Playground** to experiment with abstract CS concepts like clocks, state machines, data structures, black boxes, etc.

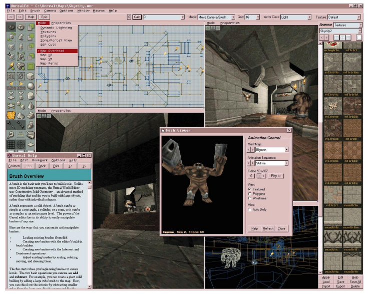


Titans of the Modern Game Dev Industry

- Unity
- Unreal

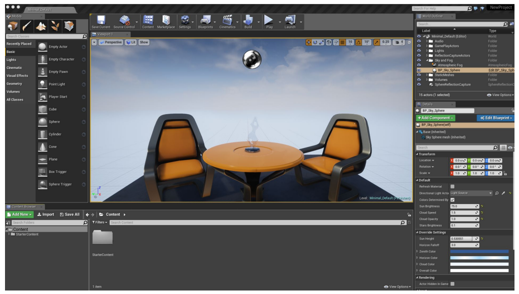
Unreal Engine by Epic Games

- Originally, a highly-optimized but uglier competitor to idEngine and Source Engine (UE1 and 2)



Unreal Engine by Epic Games

- Now, the highly-optimized and way higher-tech (closer to state-of-the-art) competitor to Unity and CryEngine (UE3 and 4)



Unreal Engine by Epic Games

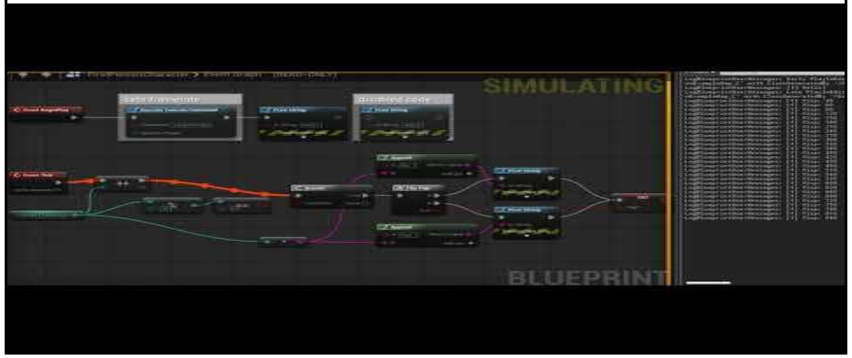
- Most recent version, UE4, released in 2014. Developed since 2003
- Source code available!
- Relies heavily on NVIDIA libraries and Windows SDK

Unreal Engine by Epic Games

- Most profitable game engine and popular in AAA companies (for reasons we'll talk about…)
- Many internal game engines are some variant of Unreal or similar engine

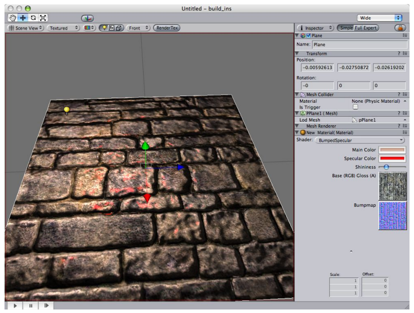
Unreal Engine by Epic Games

- Uses C++ and Blueprint (node-based C++). Plugins to use most interpreted languages like Python (i/c) and JS



Unity 3D by Unity Technologies

- Originally, a Mac-exclusive engine made to combat widespread incompatibility of 3D engines with Macs and web platforms.



Unity 3D by Unity Technologies

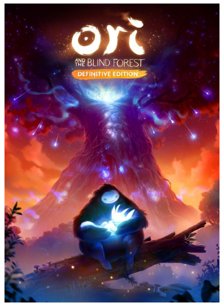
- Now, a low-resource, modular engine that is (considered) easier to learn

Unity 3D by Unity Technologies

- Released in 2005. Each year has tons of changes (this is **good & bad**)
- Doesn't rely on many third-party libraries and pretty barebones so it works on many more devices
- They opted instead to implement **buggy but low-resource** versions of them (e.g. broken physics with bad collision detection, but high framerate)
- Uses **C#** (essentially a better-optimized Java) and once used JS. Not compatible with anything else (largely b/c of lack of source code)

Unity 3D by Unity Technologies

By far the most popular with indie studios



Why is Unity so popular?

Many superficial reasons for this IMO

- Indie devs often students with **less dev experience and senior guidance**
- Universities for some reason stick to **Java**, abandoned C++ >10 years ago
- UE4 very **resource intensive**, meaning **expensive** to develop on in terms of equipment
- Some claim Unreal contracts are **less profitable** (which is usually incorrect)
- UE4 original pricing system **turned devs away** (like Xbox One debacle)
- Unity still useful, but Unity devs often dishonest about popularity (and why AAA companies won't touch it!)
- **Be careful about Unity vs other engine articles. Build your own opinion!**

Game quality and performance

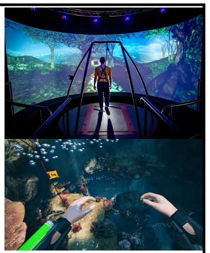
- Quality of experience
  - Mostly subjective user reviews
    - Is it fun? Did you enjoy it? Would you play again? Etc.
    - Good graphical style?
    - Gameplay is good?
- Performance
  - Framerate
  - Frame latency
  - Quality of graphics (highest resolution, aliasing, baked vs realtime lighting, etc.)
  - Required specs relative to graphics/framerate

What is VR? How is it different from AR?

(class question)

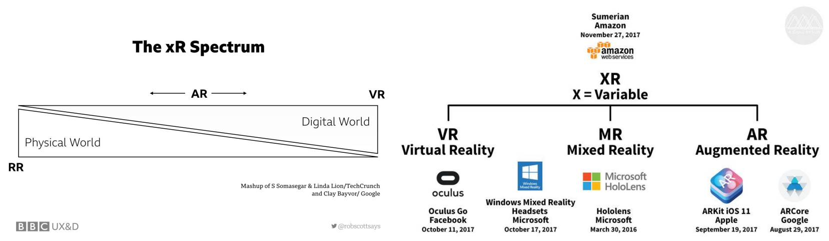
Virtual Reality

<-misleading!



- As an idea:
  - (Ideally, completely) substitutes real stimuli with synthetic
    - E.g. real visuals replaced by 3D CG visuals
    - E.g. real audio replaced by 3D synthesized audio
    - As opposed to AR & MR, which blend real and CG
  - Attempts to evoke a sense of presence
    - A feeling of "being there," in the 3D world **instead of** the real world (Minsky 1980)
- As a technology (most consumer devices..not all!)
  - **Stereoscopic Near-eye displays, engineering magic** to address vergence-accommodation conflict
  - Place constraints in the virtual world **based on** constraints in the real world, like walkable space or physical obstacles
  - **Track objects** like the HMD, controllers, etc. with (usually) 6 degrees of freedom (DOF) (**Location**: (x,y,z); **Rotation (aka Orientation)**: (pitch, yaw, roll))
  - Sense of presence is hard to perfect and some senses are difficult (if not impossible) to reproduce synthetically atm, so do the best we can

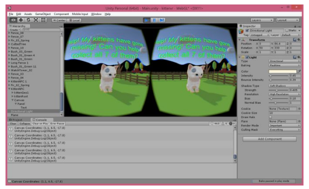
XR spectrum



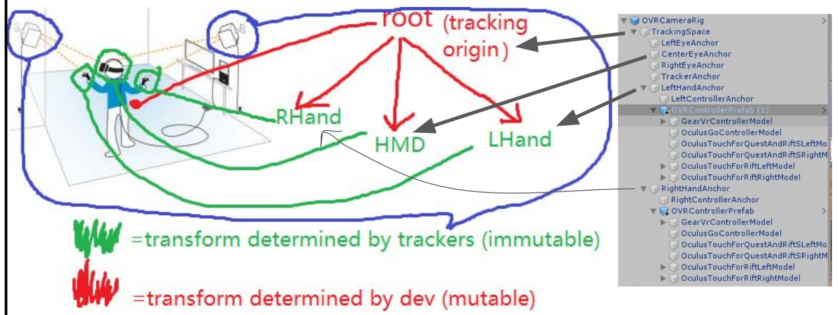
- Citations in images. Can also reverse image search
- Can argue that devices like Vives and some Oculuses are MR b/c they use real-world tracking constraints, muddying the waters
- Programming more or less the same for entire spectrum

XR dev is usually a special case of game dev

- (usually) Stereoscopic “camera” (viewpoint of player)
- Tracked camera and devices with (usually) **immutable transform** (location,rotation,scale)
  - Normally, all components of a single tracked space are attached to a **mutable root/rig**
- APIs to optimize areas more important to VR, like smoothness of movement, rendering (VR essentially renders twice), etc.
- **Same** game engines are used
- Developing VR requires knowledge of general-purpose game dev.



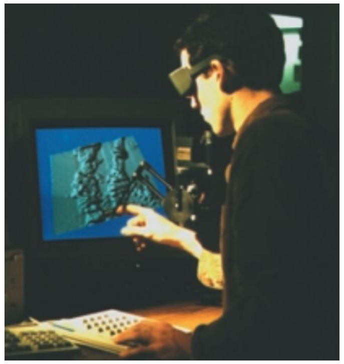
ART


root (tracking origin)
RHand
HMD
LHand
=transform determined by trackers (immutable)
=transform determined by dev (mutable)
OVRCameraRig
TrackingSpace
LeftEyeAnchor
CenterEyeAnchor
RightEyeAnchor
TrackerAnchor
LeftHandAnchor
LeftControllerAnchor
OVRControllerPrefab (1)
GearVrControllerModel
OculusGoControllerModel
OculusTouchForQuestAndRiftSLeftMo
OculusTouchForQuestAndRiftSRightM
OculusTouchForRiftLeftModel
OculusTouchForRiftRightModel
RightHandAnchor
RightControllerAnchor
OVRControllerPrefab
GearVrControllerModel
OculusGoControllerModel
OculusTouchForQuestAndRiftSLeftMo
OculusTouchForQuestAndRiftSRightM
OculusTouchForRiftLeftModel
OculusTouchForRiftRightModel

Why VR specifically?

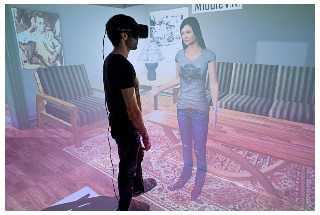
- VR tech is more widespread right now...in consumer stage, WAY less expensive
- Easier to work with (less API limitations)
- Same optimizations as non-VR games (not true for AR)
  - AR: more focus on tracking stability, reconstruction, etc
- AR can be harder to grasp b/c of real world component (need to have wider basic knowledge)
- VR is more natural transition for game devs, since games are mostly entirely virtual
- Most VR concepts and dev experience apply to AR

VR apps

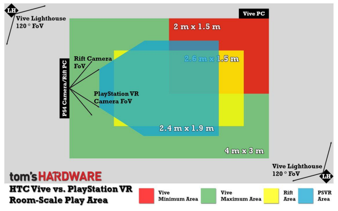


Mostly from Brooks '99: "What Real about Virtual Reality?"

VR app quality and performance



- Quality of experience
  - Sense of presence
    - Subjectively measured by SUS questionnaire...getting into HCI part!
    - Objectively measured by body temperature, conductance, heart rate, etc.
  - Naturalness of locomotion; amount of sickness (measured by SSQ)
- Performance
  - (Accidental) Collisions with tracking bounds or physical objects
  - Allowed walkable area
  - Walk speed allowed
  - Other objective measures of game quality; frame latency, framerate, clipping, etc.

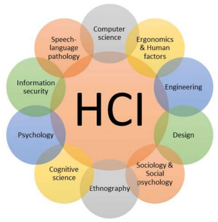


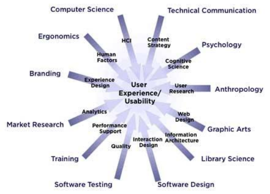
Challenges of the VR field

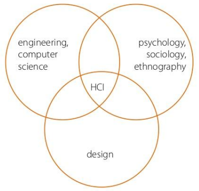
- Costly (getting cheaper...slowly)
- Lack of good killer gaming applications
  - Some longer, fully-featured apps like Skyrim and Fallout, but mostly copycat rail shooters and VR theaters
- Lack of fully-featured serious applications
  - Some training games that are a sneak peek of potential
- People are lazy. VR requires physical movement

What is human-computer interaction (HCI)?

Class question





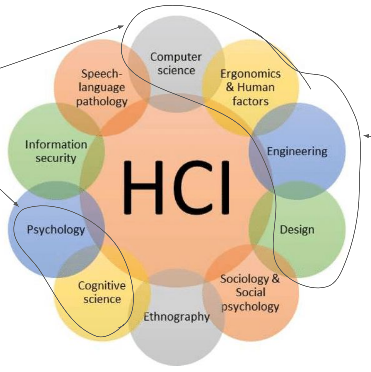

engineering, computer science
psychology, sociology, ethnography
HCI
design

Human-computer interaction (HCI)

- Study of interaction between computers and their human users
- Similar, if not a subset of design thinking
    - From Interaction Design Foundation: “non-linear, iterative process which seeks to understand users, challenge assumptions, redefine problems and create innovative solutions to prototype and test”
- Create novel applications with computers to solve **human problems**
- Understand how and why computer apps work, iteratively improve them
- Focus on the human factor instead of **solely** optimization or design.
- If your app doesn’t work for its target users, then it **doesn’t work**. Even if it’s O(1)
- HCI ≠ UX. UX is often more business-focused (e.g. how to keep your users coming back. The FEEL of the app rather than applicability. How does Candy Crush get people to play 100s of levels? Or FB to get users to keep scrolling? Who cares why?)

HCI vs UX

VR HCI focus (relative to other HCI fields)



UX’s focus

Some problems in VR addressed by HCI

- Sickness
  - We now know to **never use game controller input** to move!
  - “Scary” things in regular games could actually give users in VR heart attacks or trauma
- Applicability
  - Companies are learning to provide **multiple** locomotion **options**, graphical tweaks, etc.
- Novelty
  - Things that are usually boring in regular games can be more **addicting** in VR (e.g. rail shooters)
- Important questions to ask yourself in game and VR dev:
  - How do I know this works?
  - How do I go about testing practicality, sickness, user response, etc?
  - How do I improve this for the user?

Now, about this class....

- We’ll study all of these problems (to some extent)!
- We’ll learn how to build impressive VR apps that actually work!
- We’ll learn how to figure out if they work!

Slides will rarely be so wordy!

Rough class timeline

Broken down by senses

Topic/ Week #	Topic(s)
1	• **Introduction to course and the main topics: VR, game dev, HCI**
2	• **Basics of graphics** • **Complexity and optimization** • **Basics of Unity** • **Basics of 3D models and Blender**
3	• **Basics of VR development**
4	• **Data structures** • **Finite state machines and planning gameflow** • **Troubleshooting**
5	• **Locomotion: The Vestibular and Kinesthetic Senses** • **Navigation and sense of place** • **Distance compression/scaling** and why it's worth considering in ALL of the fields we talk about • **Smoothness/continuity** in VR and why it's also worth considering everywhere
6	• **Audio Rendering: The Sense of Hearing**
7	• **Immersion: The Sense of Presence** • **Training, learning effect, and skill transfer**
8	• The previous topics are likely to overflow here
9	• **Social VR and Presence of Others: The "Sixth" Sense**
10	• **Animation**
11	• **Physics and Collision Detection**
12	• **Haptics: The Sense of Touch** • **Olfactory systems: The Sense of Smell**
13	• **Systems related to VR and other XR systems**
14	• **Intro to Advanced topics** ○ Rendering ○ Trackers and synchronization. Inside -out tracking vs external. Intrinsic and extrinsic calibration ○ Iterative closest point, ball pivot, and other reconstruction methods, point clouds ○ Hand/skeletal tracking ○ Particle simulation (cloth sim and soft-bodies for surgery, fluids, etc) ○ Networking (RPC functions, client-server model, difficulty of networking skeletons and particle sims ○ Particle systems and function-based design ○ Marker tracking and computer vision/machine learning applications to improve the experience ○ Eye-tracking

Logistics

- Class time: MWF 4-4:50 in FedEx Global Center (unless some Sitterson room magically becomes available)
- Friday is optional lab/office hours (EXCEPT FOR THIS WEEK)
  - Some variant of the assignment solution might appear during this time :)
- Assignments are due on Sakai
  - I will almost always ask for videos and/or screenshots.
  - Pls post videos on Youtube (unlisted if you prefer no one else to see).
  - For best recording quality (I find), use OBS Studio with FLV file format (less CPU intensive), which uploads to Youtube directly (OG Youtube preferred FLV)
  - Latex is preferable for writeups and is good to know (Overleaf ♡), but whatever you use, EXPORT TO PDF
  - These instructions will be in the assignment instructions
- Class materials will primarily be stored on my site and maybe on Sakai (with delay): www.nickvr.me/comp590s20

Grade breakdown

- Class assignments: **60%** (~9-10 assignments)
    - Grade breakdown for each assignment might be different, but estimate 75% technical part, 25% HCI part
- Final project: **15%**
- Midterm proposal: **5%**
- Midterm assignment: **10%**
- Final assignment: **10%**
- Late policy: 7 late days. Afterwards, up to 1 week late=**25%** off, up to 2 weeks late=**50%** off, not accepted after (unless documented excuse)
- Not a fan of exams, esp. in an application class. Also, these topics doesn't have many set-in-stone answers so it's generally not fair to objectively test on

Assignments

- Usually, a technical, programming part and some kind of small HCI part (analysis, mini-user study, etc.)
- **Shouldn't be that hard** in terms of programming, but practice skills you probably haven't used before (e.g. design, ANOVA analysis)
- Assignments are **mostly disjoint** extensions of the project you start in the beginning (so Unity setup assignment is very important)
- **Advice**: if something breaks and doesn't seem programming related, post about it on Piazza. Unity is buggy and has a lot of errors with easy solutions (e.g. delete cache). Unreal similarly has annoying linker errors (common in C++)
- **My basic principle when designing assignments: try not to assign something that you're very unlikely to do something similar to in the field**

Rough Assignment Plan

1. Unity/Blender/Android setup
2. Build a simple 3D environment and make a moving character
3. Build some 3D props in Blender and complete a task in Blender's Python
4. Convert character to a VR character and use Oculus standard assets to implement a simple collection game (e.g. put objects in a bin to win)
5. Make a VR main menu, give collectables different metadata such as points (modularly), allow for winning
6. Implement physical locomotion system such as redirected walking or walking-in-place
7. Put props in a maze-like environment and add audio propagation and occlusion filters with an existing library such as SteamAudio
8. Add features in the environment that evoke physiological response
9. Add virtual AI that avoid the player with something like RVO
10. Implement a basic avatar that has simple inverse kinematics such as feasible elbow movement when the controllers are moved
11. Final project time

Final Project

- Some implementation of a class topic that is either
  - a strong extension of a class assignment
  - Rudimentary to intermediate implementation of a non-assigned topic (depending on topic, e.g. active haptics is much harder to implement well than passive)
- Consult with me and/or get feedback from midterm proposal
- Can turn into a bigger project if you wish (e.g. publication or real game)

Midterm and final assignment (not exams!)

- Midterm
  - **Part 1**: in-class proposal of final project
  - **Part 2**: take-home (essentially a longer non-programming assignment)
    - Subjective analysis of something (e.g. maybe I'll propose some really bad VR app or a poorly-design user study). Grade is based on whether or not your analysis makes sense.
- Final
  - Similar to the midterm but based on the 2nd-half-of-class material, which are almost entirely emerging technologies and require more critical thinking
  - Still take home :)
  - Finals period will be final project presentations
- Midterm and final assignment are use-whatever-you-want-except-each-other

Possible midterm/final scenarios

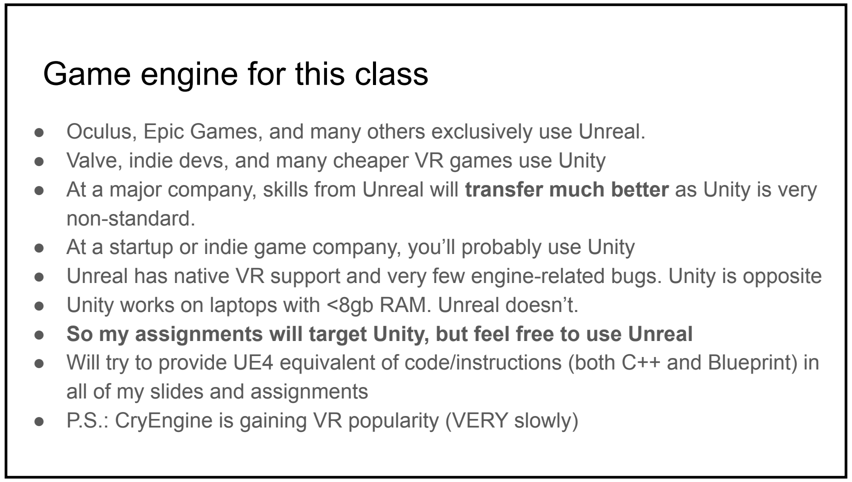
- You're a VR/game design consultant. I'm a indie dev who wants advice about making my game VR-compatible. You answer with considerations about each area you learned about, methodology, etc.
  - E.g. client wants Oculus Quest for dungeon crawler RPG. In terms of audio, I'd recommend using occlusion/audio filters instead of propagation b/c CPU resources.
- I propose to you a heavily-flawed user study to test some VR feature. You tell me the flaws and how you would do it (+ rationale). Think GRE essay

Collaboration

- Assignments are not individually complex enough to warrant more than 1 partner. So groups of 2 max for assignments. Can discuss **high-level** concepts with others and on forum.
  - I'm pretty good at recognizing copied code :)
  - I'm also pretty well-versed on online example code. I use it myself :) :) :)
- Writeup can be done individually or with your partner/group.
- Github project link **must** be submitted with assignment, as well as Overleaf **EDIT** link if you did the writeup with partners
  - Overleaf has edit history that is only visible to editors (Don't worry, I won't edit)
  - It will be more effort to cheat by faking the Overleaf edit history than to just do the assignment
  - Why: you should both be doing both parts of assignments collaboratively, not splitting it
- Final project: Groups of 4 max. Bigger group=bigger expectations & better writeup.
- Any collaboration beyond this is an honor code violation

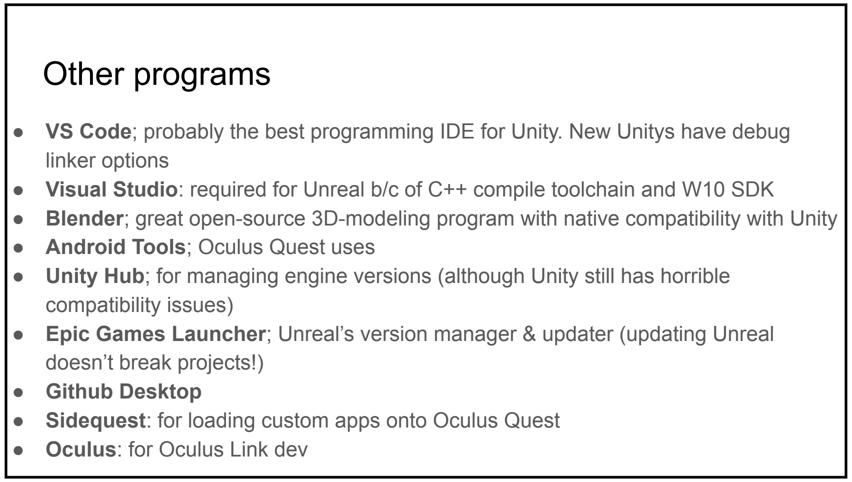
Copying Code

- I specifically designed the assignments to be **hard/impossible to find online implementations of**
- You can copy code that has a simple purpose (e.g. I always copy code that creates Debug Arrows in Unity), but you MUST cite it (write a comment at the top of the file that it's used in with the link)
- You can browse whichever online code you want, but you're unlikely to find the solution without the help of these slides or cited papers.



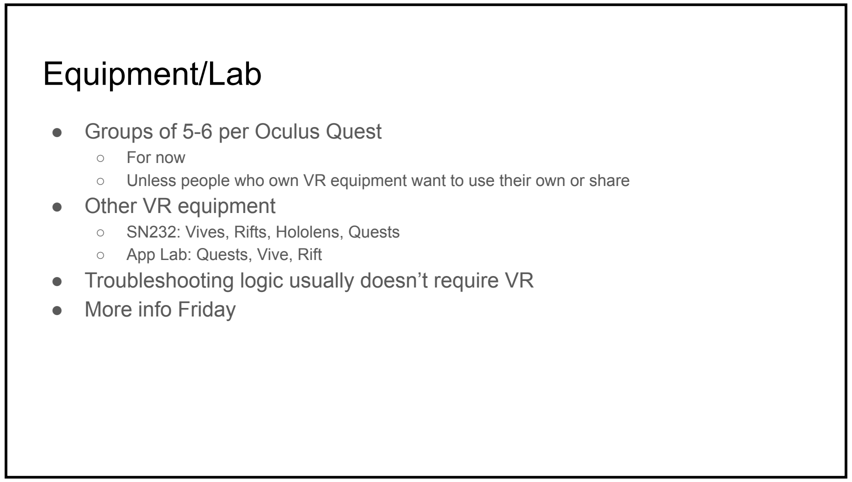
Game engine for this class

- Oculus, Epic Games, and many others exclusively use Unreal.
- Valve, indie devs, and many cheaper VR games use Unity
- At a major company, skills from Unreal will **transfer much better** as Unity is very non-standard.
- At a startup or indie game company, you'll probably use Unity
- Unreal has native VR support and very few engine-related bugs. Unity is opposite
- Unity works on laptops with <8gb RAM. Unreal doesn't.
- **So my assignments will target Unity, but feel free to use Unreal**
- Will try to provide UE4 equivalent of code/instructions (both C++ and Blueprint) in all of my slides and assignments
- P.S.: CryEngine is gaining VR popularity (VERY slowly)



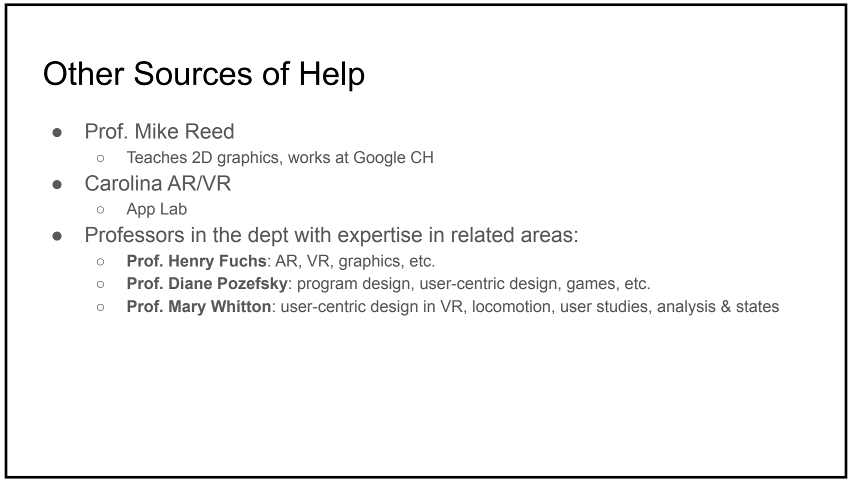
Other programs

- **VS Code**; probably the best programming IDE for Unity. New Unitys have debug linker options
- **Visual Studio**: required for Unreal b/c of C++ compile toolchain and W10 SDK
- **Blender**; great open-source 3D-modeling program with native compatibility with Unity
- **Android Tools**; Oculus Quest uses
- **Unity Hub**; for managing engine versions (although Unity still has horrible compatibility issues)
- **Epic Games Launcher**; Unreal's version manager & updater (updating Unreal doesn't break projects!)
- **Github Desktop**
- **Sidequest**: for loading custom apps onto Oculus Quest
- **Oculus**: for Oculus Link dev



Equipment/Lab

- Groups of 5-6 per Oculus Quest
  - For now
  - Unless people who own VR equipment want to use their own or share
- Other VR equipment
  - SN232: Vives, Rifts, Hololens, Quests
  - App Lab: Quests, Vive, Rift
- Troubleshooting logic usually doesn’t require VR
- More info Friday



Other Sources of Help

- Prof. Mike Reed
  - Teaches 2D graphics, works at Google CH
- Carolina AR/VR
  - App Lab
- Professors in the dept with expertise in related areas:
  - **Prof. Henry Fuchs**: AR, VR, graphics, etc.
  - **Prof. Diane Pozefsky**: program design, user-centric design, games, etc.
  - **Prof. Mary Whitton**: user-centric design in VR, locomotion, user studies, analysis & states

Final thoughts

- May be first class of its type (that can be found online at least)
  - Focus on VR-specific design instead of using VR as a by-feature (e.g. Brown CS137)
  - Less focus on stuff that's already done (e.g. UMich SI559/659, Stanford EE267, USC CS522)
- Only game dev class that I can find that will essentially teach Unreal and Unity programming **simultaneously**
  - Interesting to see if it helps!
- 3 hackathons at UNC: PearlHacks (for non-males), Global Game Jam, HackReality
- Present on Maze Day?
- **This Friday's lab (here)**: getting acquainted with our Oculus Quests
- **Next week**: basic graphics and Unity
- **Registration**: Can only make exceptions for seniors who need the class to graduate
- Want to record lectures to make public for successors. Just voice (Dewan style)
- I'll post debatable or interesting Unity/Unreal articles with my thoughts for debate
- Future lectures more interactive/realtime