Helge Rhodin

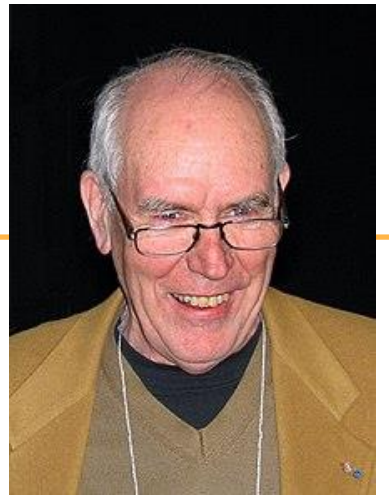
CPSC 427
Video Game Programming

History and Future of Game Technology

Today

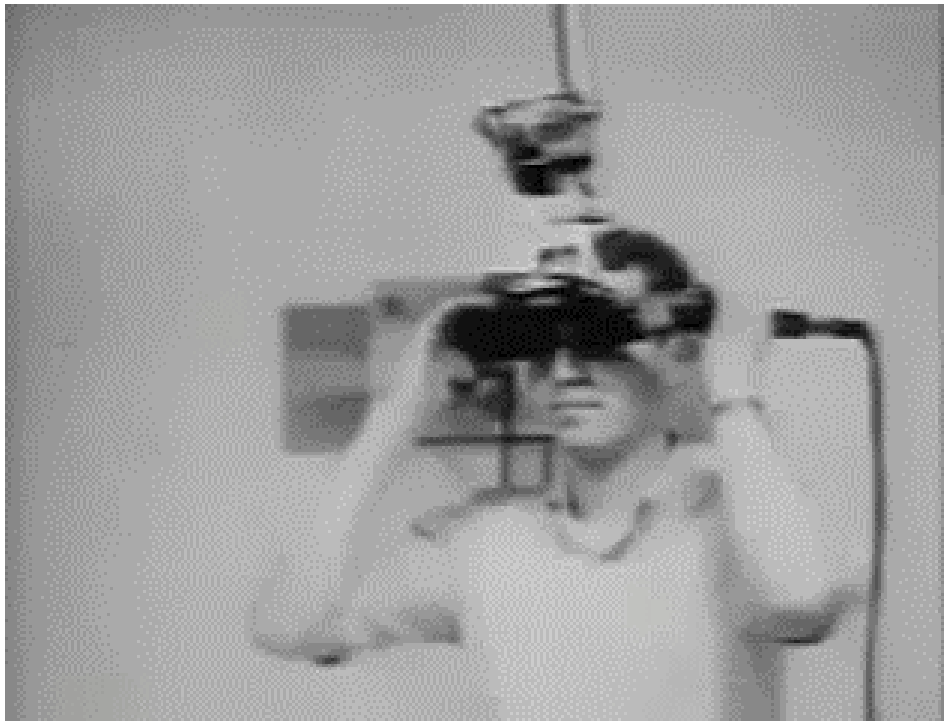
- ***Technical highlights in game history***
- ***Relations to computer science advances***
  - *computer graphics*
  - *computer vision*
  - *optics …*
- ***Course Summary***
- ***The future of gaming?***

The Sword of Damocles (1968)



- ***By Ivan Sutherland***

- ***First augmented reality head-mounted display (HMD)***
  - stereoscopic display
    - *see-through technology!*
  - viewpoint-dependent rendering
    - *required 6 DOF head tracking*
    - *some versions used ultrasound!*



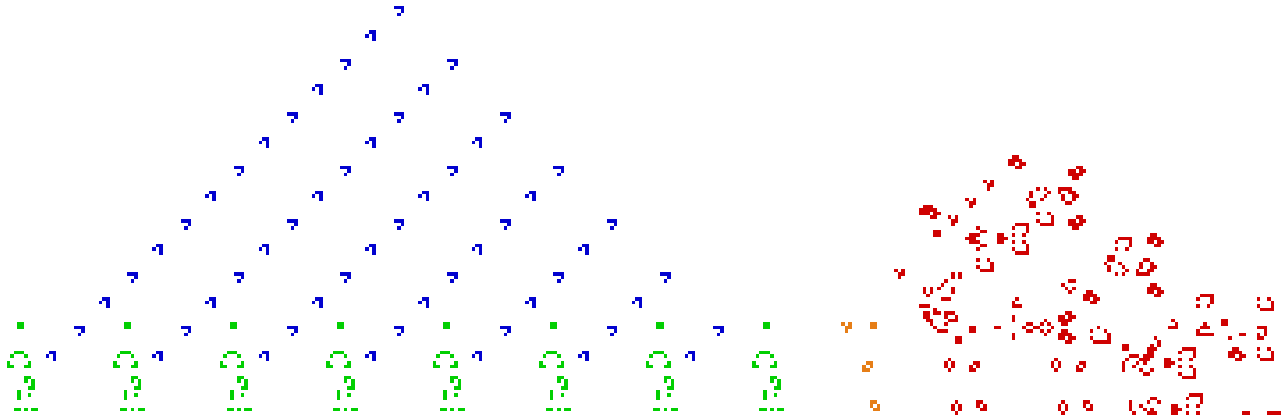
LIFE (1970)

By John Horton Conway

Rules:

- A pixel grid of active/live and inactive/dead cells
- Any live cell with two or three live neighbours survives
- Any dead cell with three live neighbours becomes a live cell
- All other live cells die in the next generation

The seed (initial condition) determines the evolution



Perlin noise (1983)

Ken Perlin
https://mrl.cs.nyu.edu/~perlin/
NYU

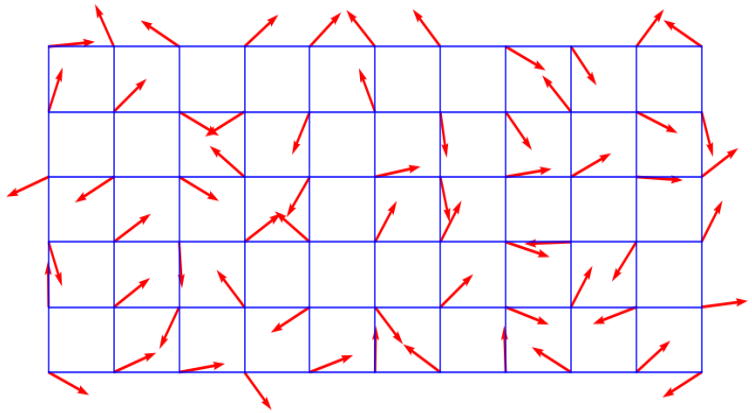
Check out his website!

Two-dimensional slice through 3D Perlin noise at z=0

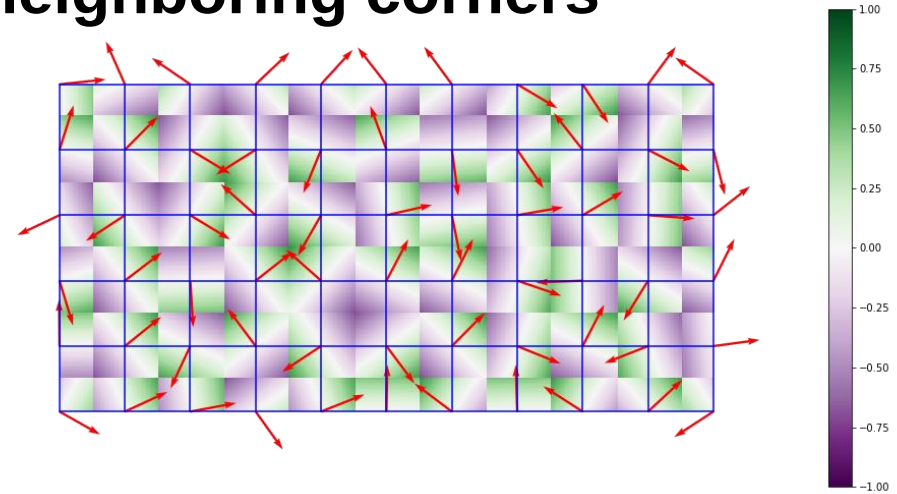
Landscape by Perlin noise

Perlin noise

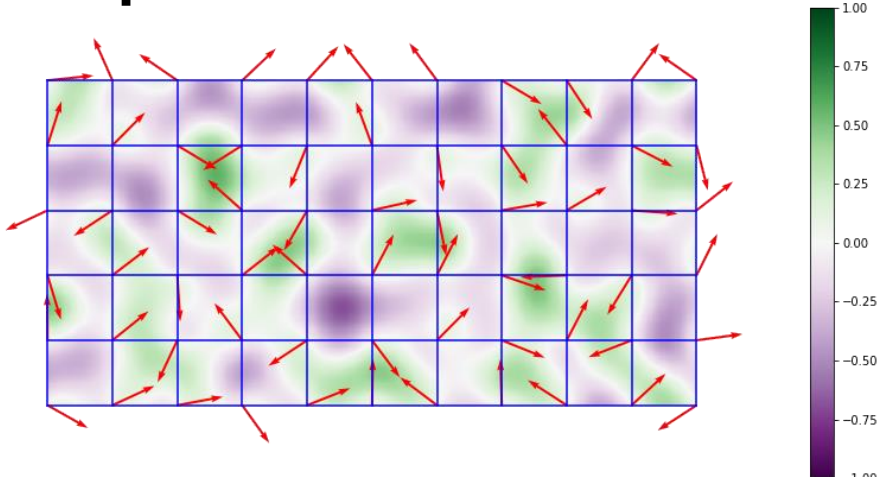
1. Generate random vectors



2. Dot product of rnd. vec. and offset to neighboring corners



3. Interpolate based on distance



4. Repeat at different resolutions and add displacements

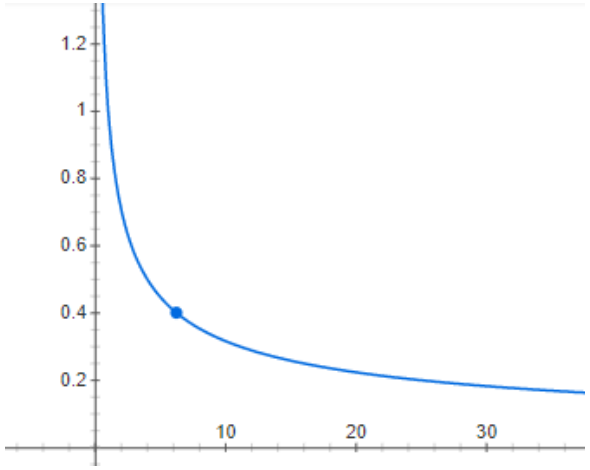
Code in Quake 3 (1999) – what??

```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y  = number;
    i  = * ( long * ) &y;
    i  = 0x5f3759df - ( i >> 1 );
    y  = * ( float * ) &i;
    y  = y * ( threehalfs - ( x2 * y * y ) );
//  y  = y * ( threehalfs - ( x2 * y * y ) );

    return y;
}
```

OpenArena (open source version of the original)

Fast inverse sqrt

- *1/sqrt(x)*



- *For normalizing a vector*

$$\hat{\boldsymbol{v}} = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}$$

- *For lighting and reflectance*
- ***How to speed it up?***

Light effects

Fast inverse sqrt

- ***How to speed it up?***
- *Only use addition and multiplication (at the time, division was very expensive)*

-> 4x speedup compared to division

```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y  = number;
    i  = * ( long * ) &y;                       // evil floating point bit level hacking
    i  = 0x5f3759df - ( i >> 1 );               // what the f***
    y  = * ( float * ) &i;
    y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
//  y  = y * ( threehalfs - ( x2 * y * y ) );   // 2nd iteration, this can be removed

    return y;
}
```

*Used elsewhere before but best known for its use in **Quake III Arena!***

Magic exploiting floating point representation and $\log(\frac{1}{\sqrt{x}}) = -\frac{\log(x)}{2}$

One step of Newton's method (root finding)

World of Warcraft - Corrupted Blood Incident

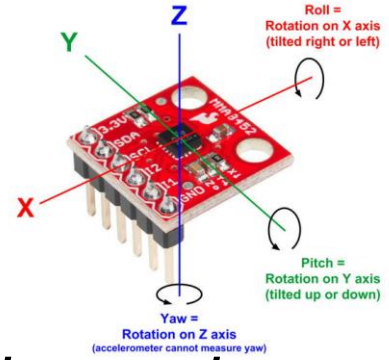
- *virtual pandemic*
- *spread by end boss Hakkar (intended to be local to a single dungeon)*
- *spread by pets and minions*
- *lasted one week*
- programmer-imposed quarantines
- players' abandoning of densely populated cities
- *Model for epidemic research*

WoW, September 13, 2005

[Balicer, Ran (2005). "Modeling Infectious Diseases Dissemination Through Online Role-Playing Games". Epidemiology. 18 (2): 260–261.]

Pokemon Go (2016)

- ***Augmented reality***
  - *requires tracking of the real world*
    - 6 DOF (3D position and 3D rotation)



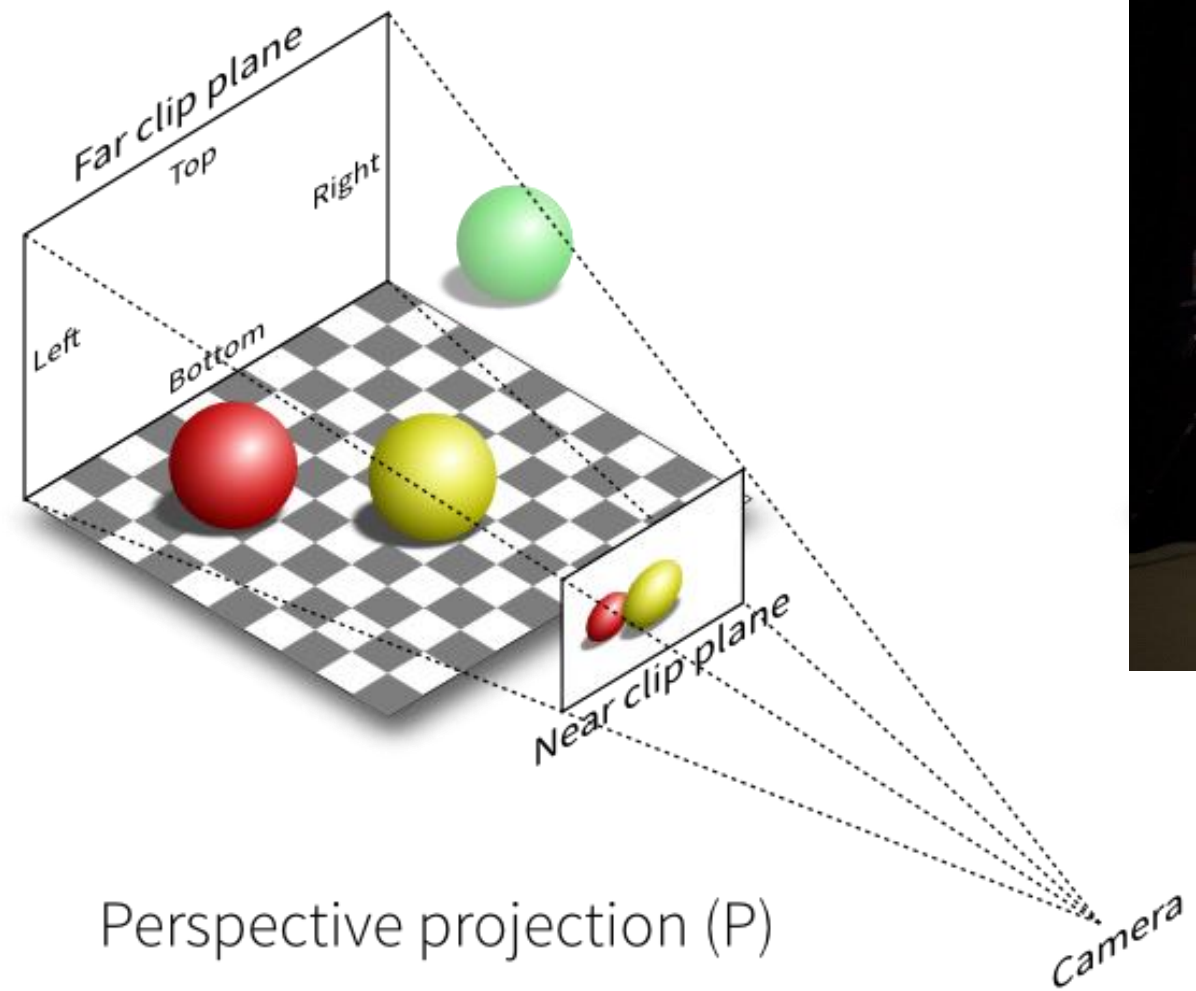
Options:

- Use device accelerometers
  - *Advantage: simple*
  - *Disadvantage: drift & no relation to the real world*
- Estimate camera angle relative to real objects

Which one is done in Pokemon Go?

HoloLens - Augmented Reality done right

Virtual Camera



Perspective projection (P)

Virtual camera registered in the real world
(using marker-based motion capture)

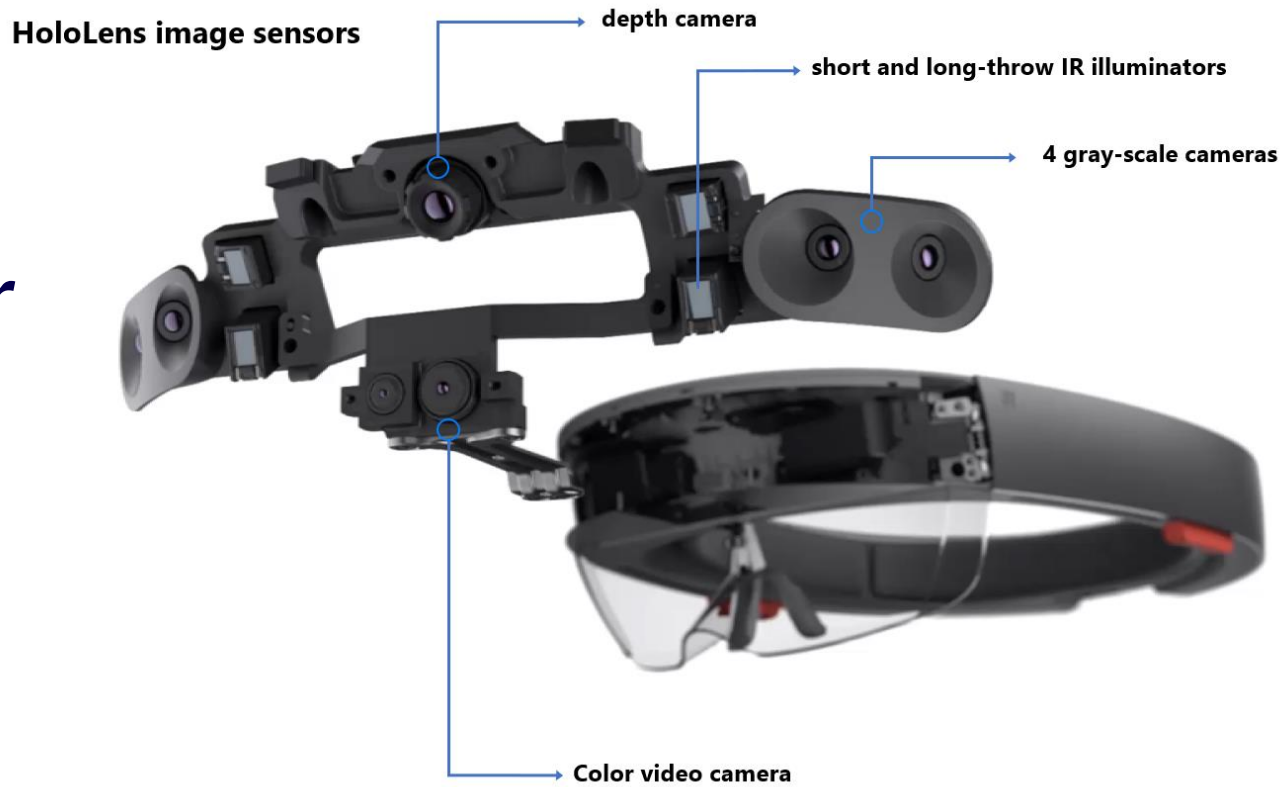
Spatial mapping and tracking

HoloLens --- Augmented Reality done right

Input: gray-scale fisheye cameras (paired with accelerometers)

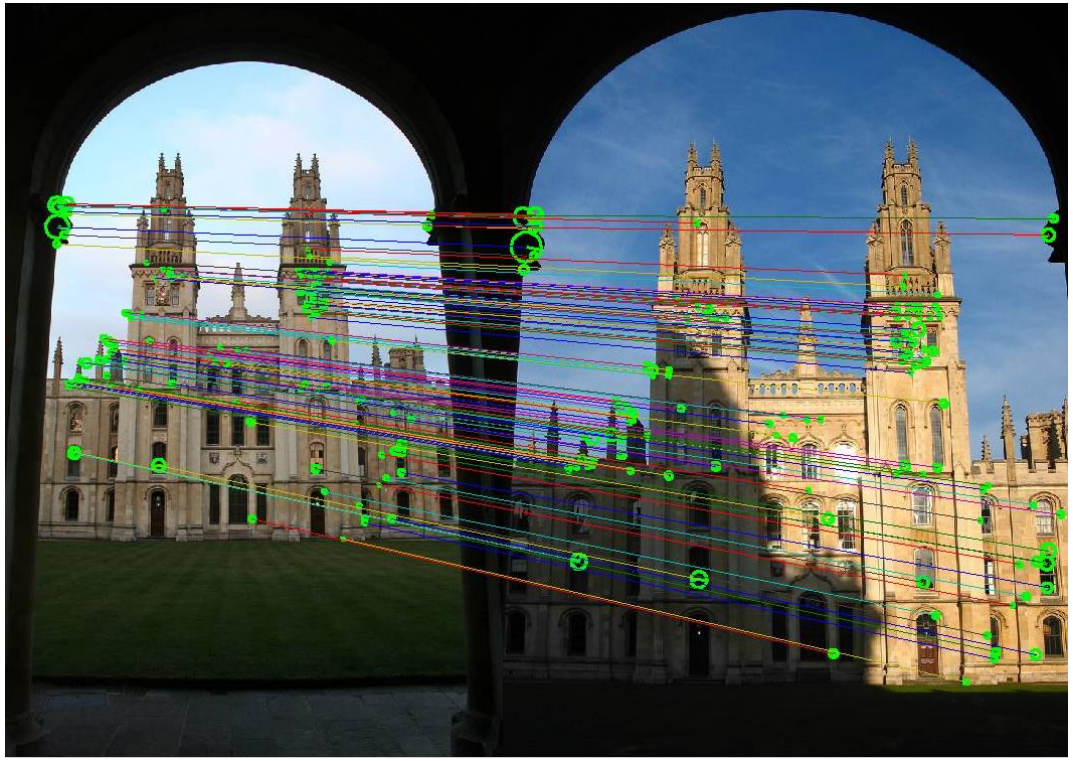
Method: track image features with on-board processor

Output: reconstruct the 3D scene and camera pose



Related computer vision concepts

- ***Feature detection***
- ***Feature tracking and re-identification***
- ***Perspective projection***



- ***Hand Gesture recognition (as input)***

Virtual and Augmented Reality Issues

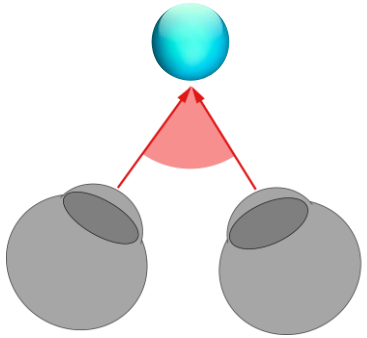
Open questions:

- ***Why are headsets so bulky?***
- ***Why do I get motion sick or perceive discomfort?***
- ***Why is the field of view (FOV) and resolution so low?***

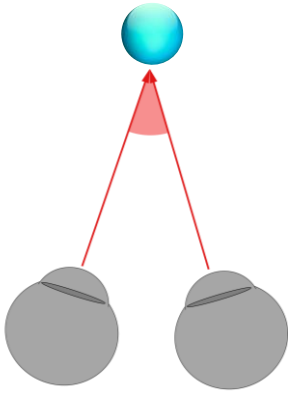
3D perception — Binocular

Convergence and accommodation

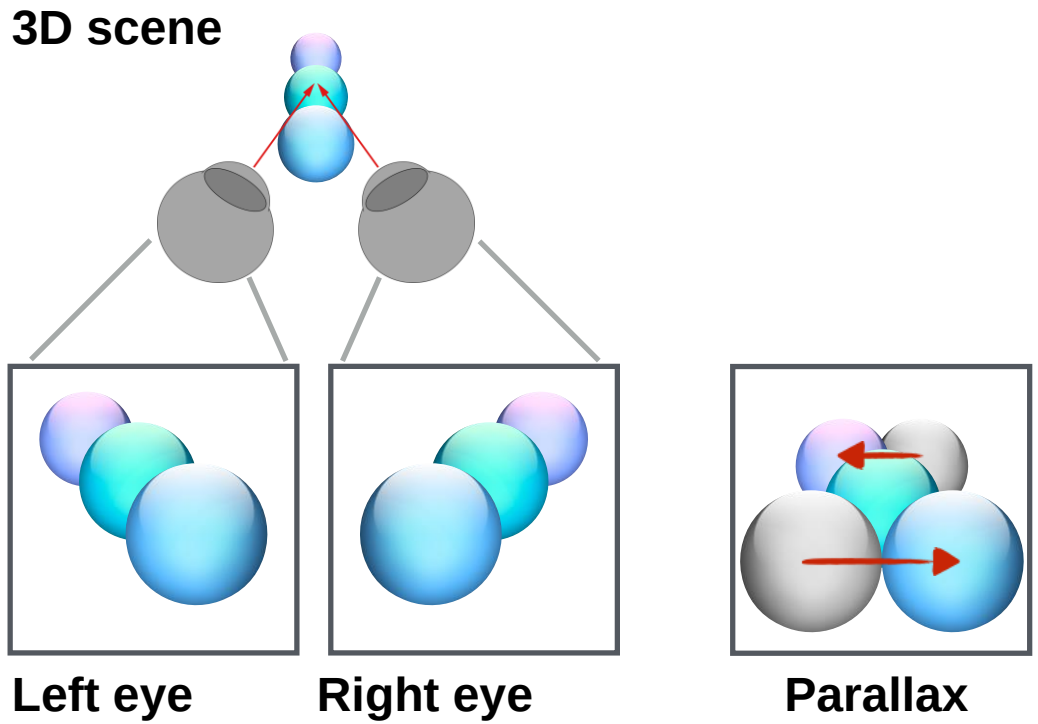
Near object,
Large angle



Far object,
Small angle

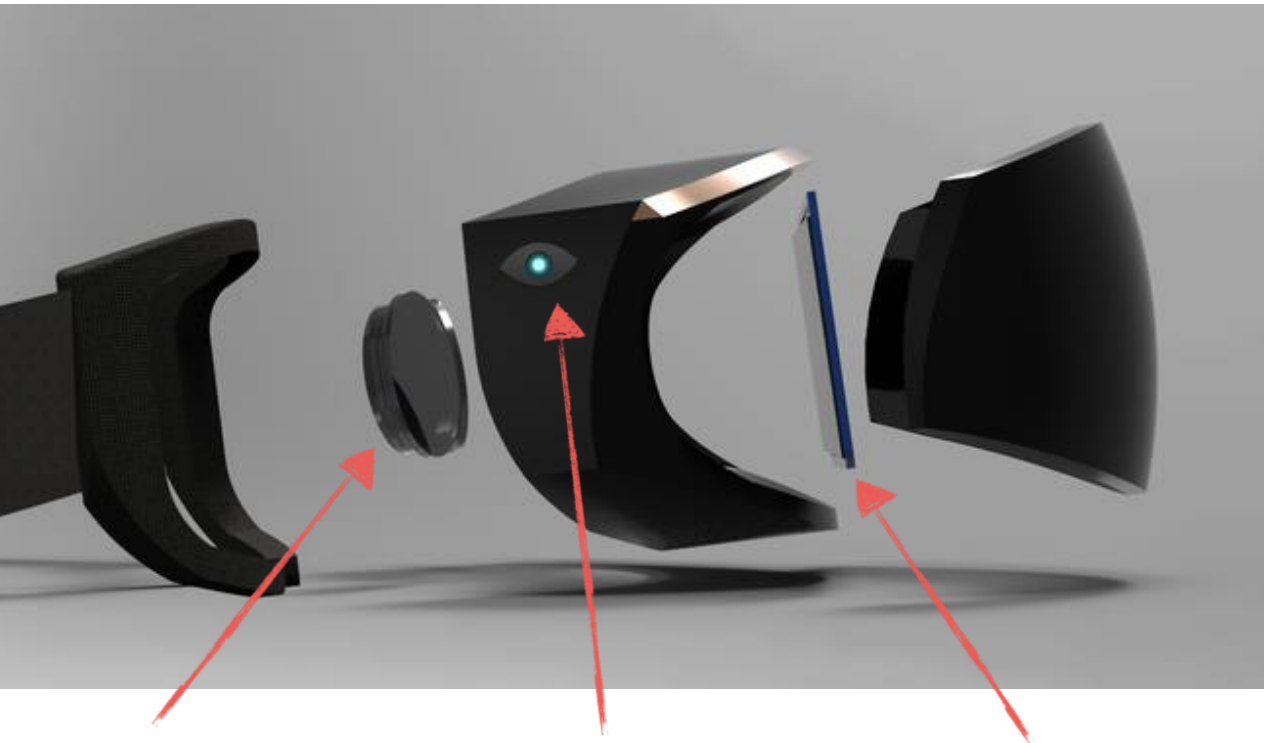


Binocular parallax

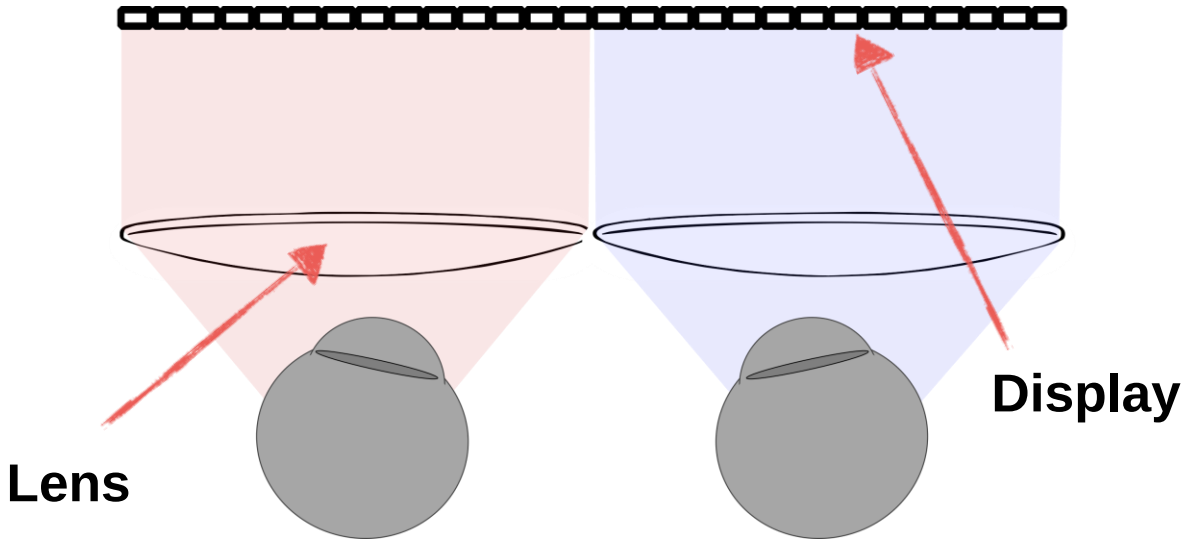


Head-Mounted Display

Head-Mounted Display (HMD)

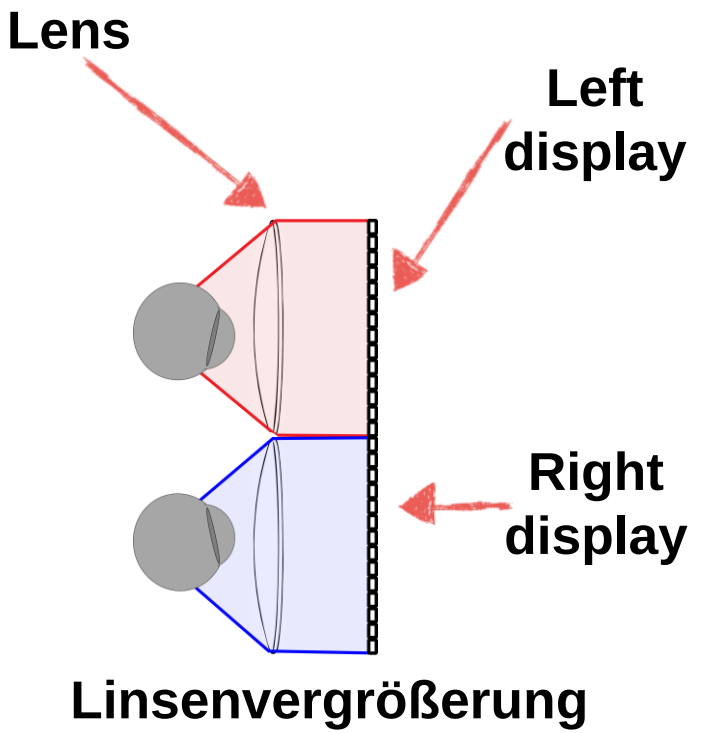


Optical setup

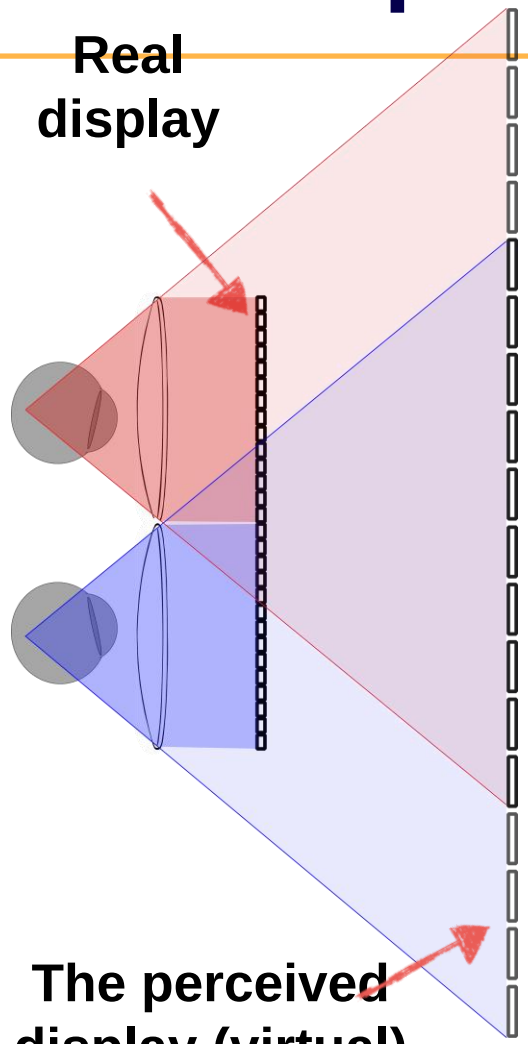


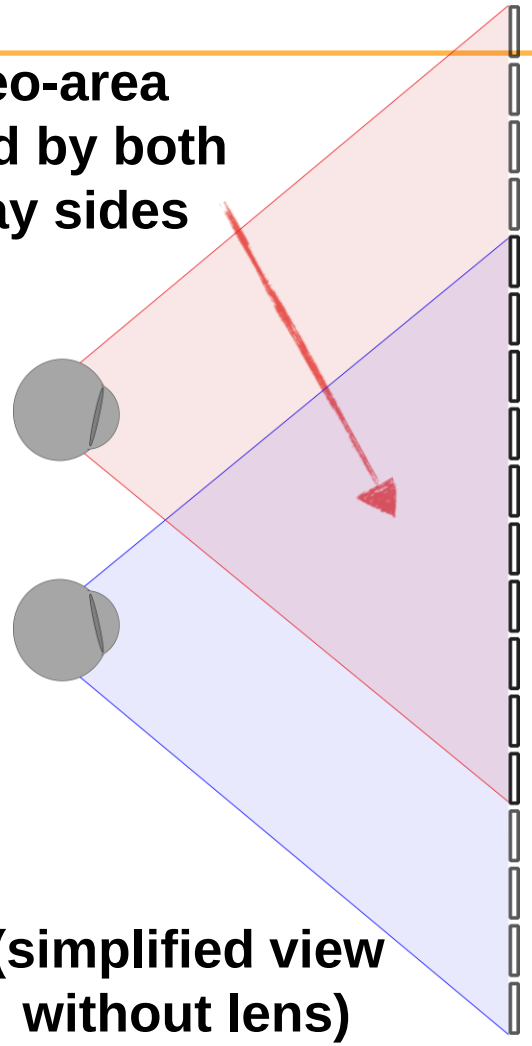
[HMD Aufbau: minimalicity.com]

Optik und Virtuelles Display

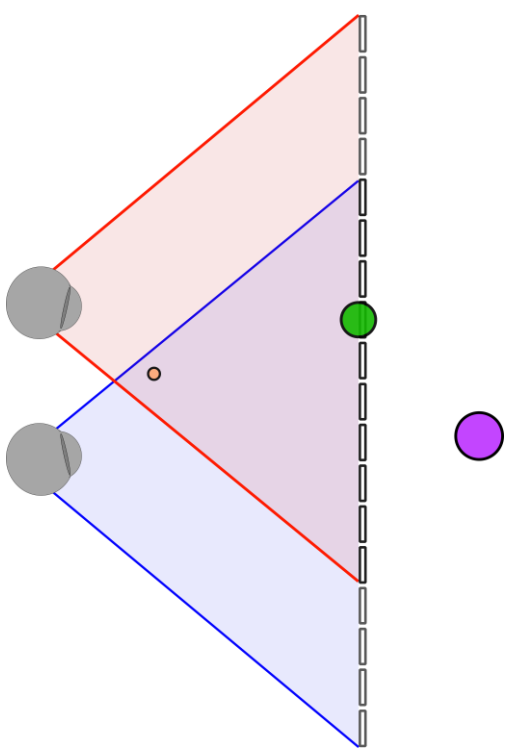


Linsenvergrößerung

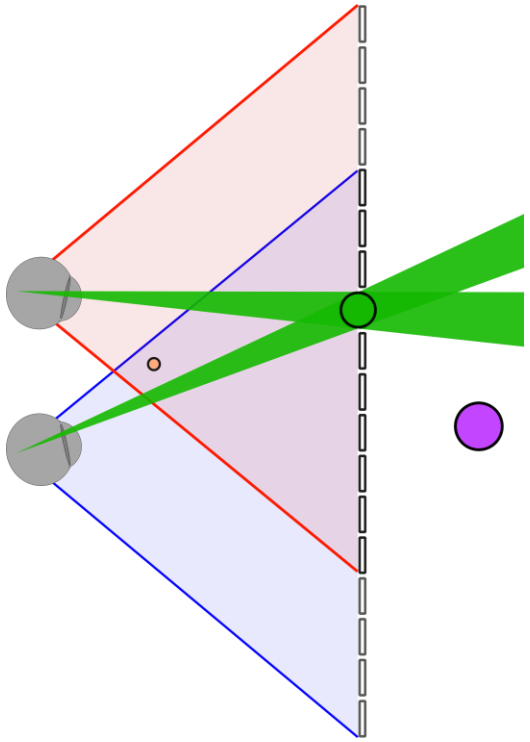




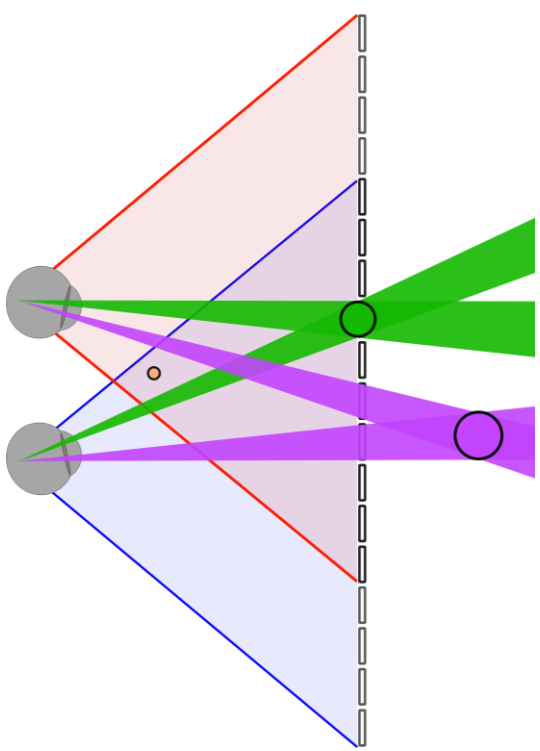
Stereo-Display



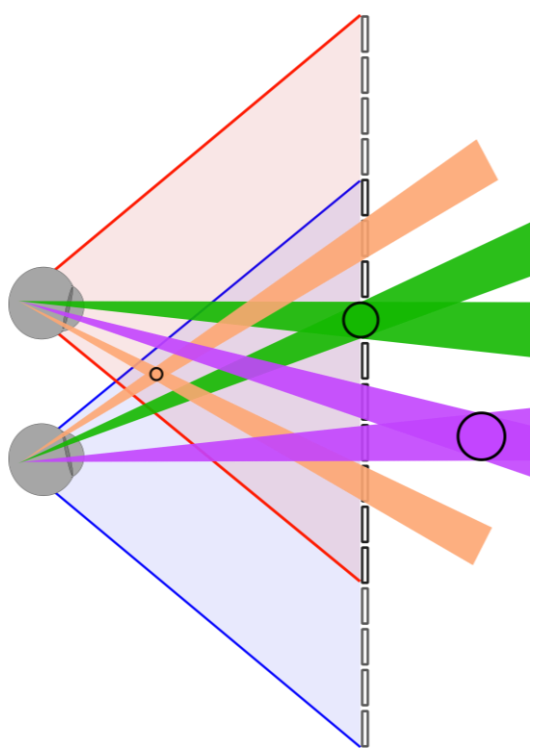
Desired/real object positions



Case I: Object projected on display



Case II: Object is behind the screen



Case III: Object is in front of screen

Issues of VR?

- ***What does this imply for us (video game programming)?***

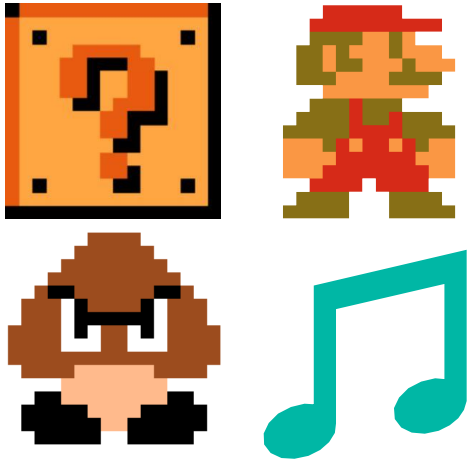
Light field displays (3D without glasses)

Sony

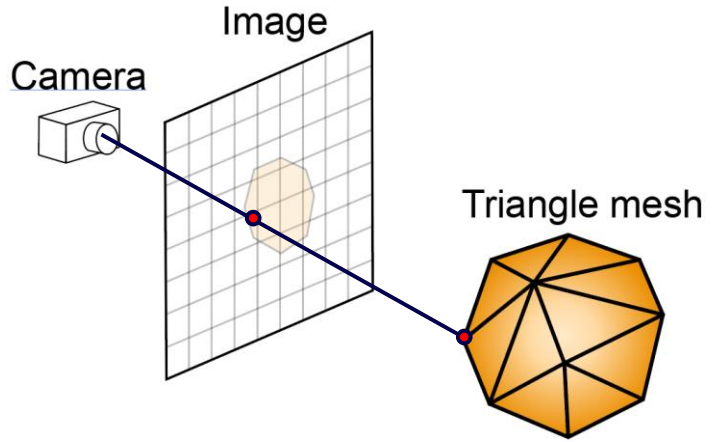
Principle: a diplay that emits a different color dependent on the view direction
Difficulty: i) Render an independent image for each view direction (and position).
ii) hardware realization.

Course Summary

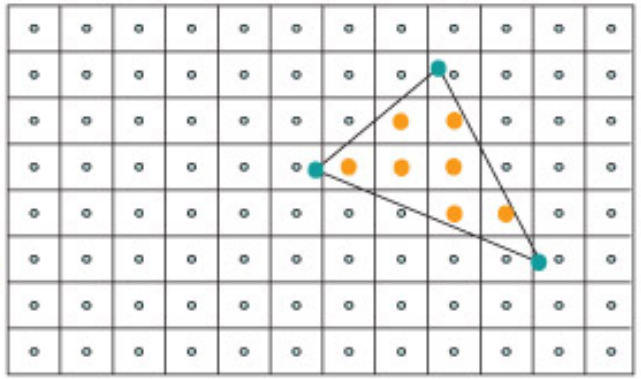
1. Intro



2. ECS

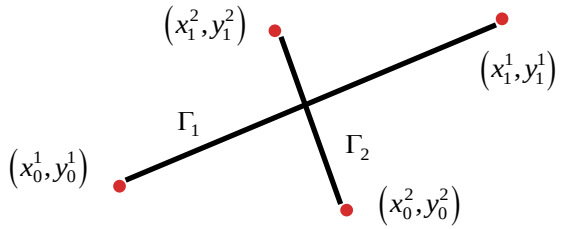


3. Rendering



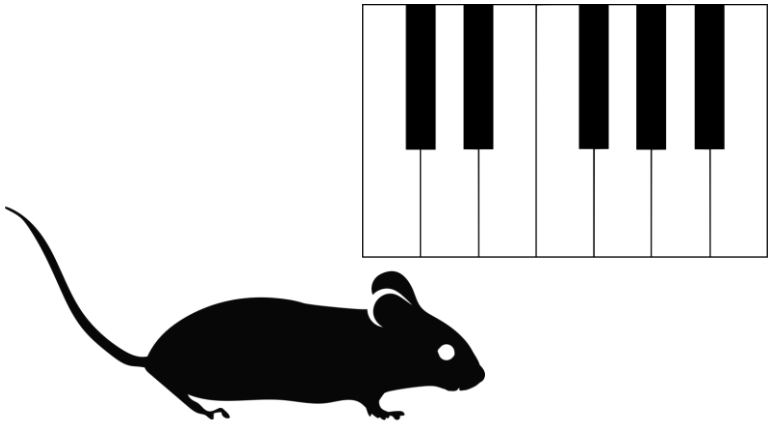
4. Rendering Pipeline

5. Advanced OpenGL



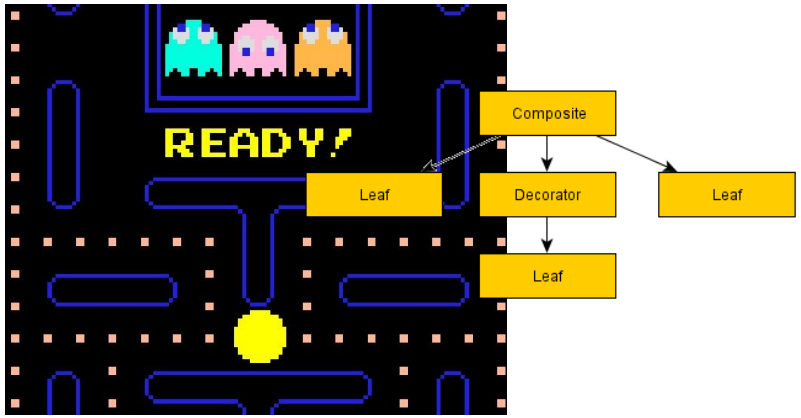
6. Collisions

Course Summary

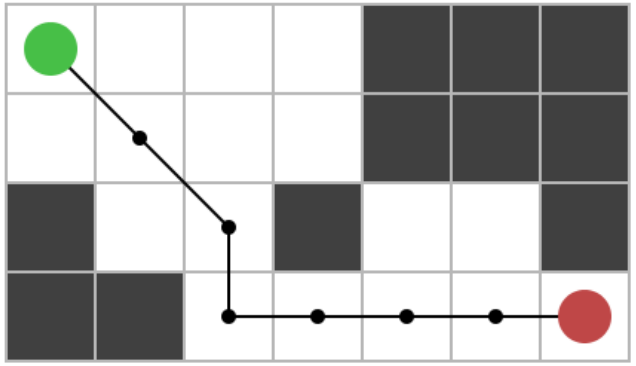


7. IO & Observer Pattern

8. User Interfaces

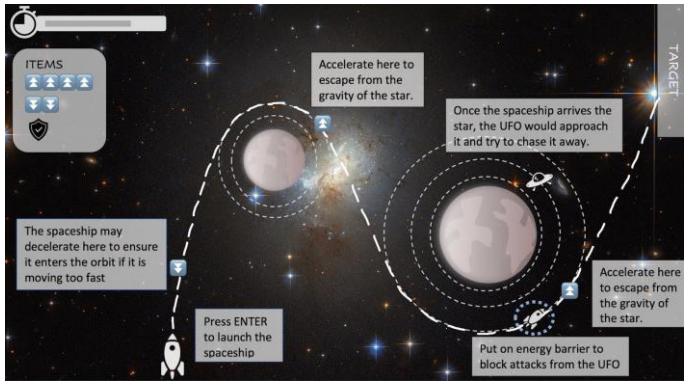


9. AI and Trees



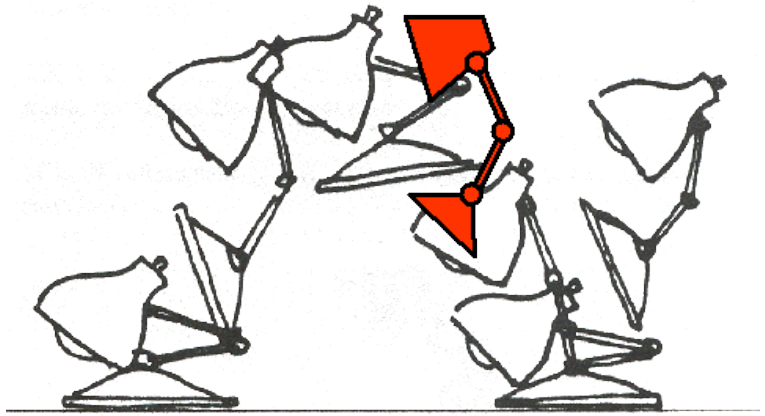
10. Path finding

11. Debugging & Simulation

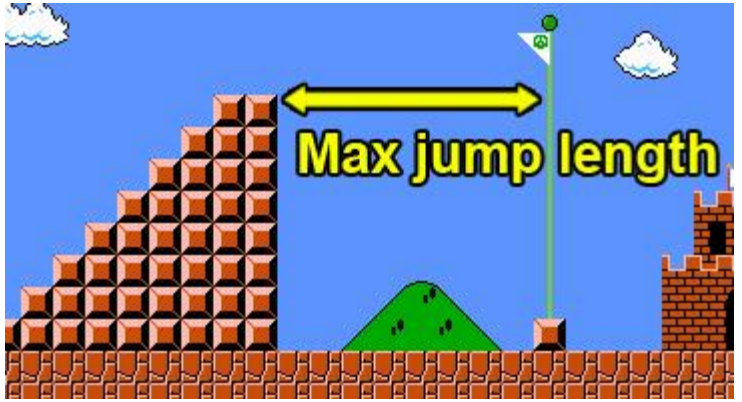


12. Simulation

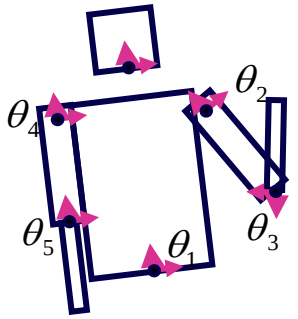
Course Summary



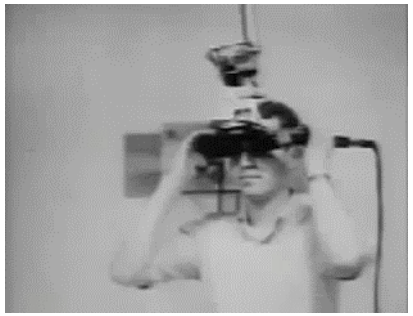
13. Curves & Animation

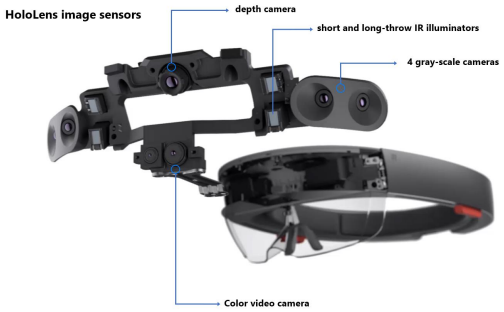


14. Game Balancing



15. Skeleton Animation

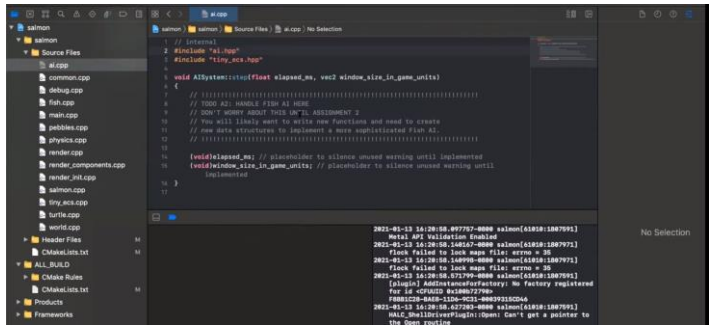




16. History & Future

Tutorials

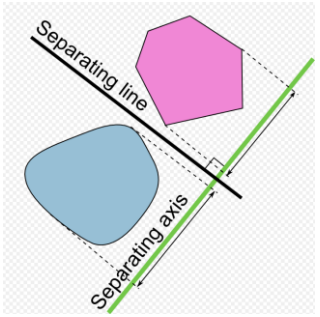
A&B. Modern C++ by Tim



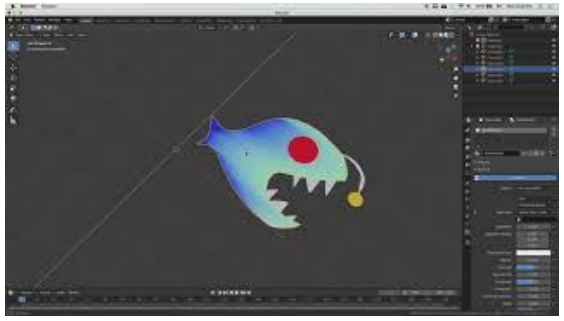
C. Tools and Game Framework by Grace and Andrew

++

D. Behaviour Trees in c++



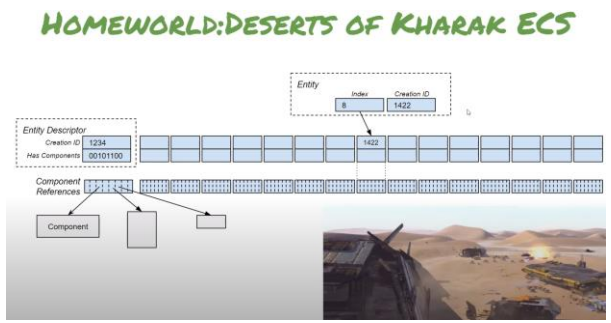
E. Advanced Collision by Tim



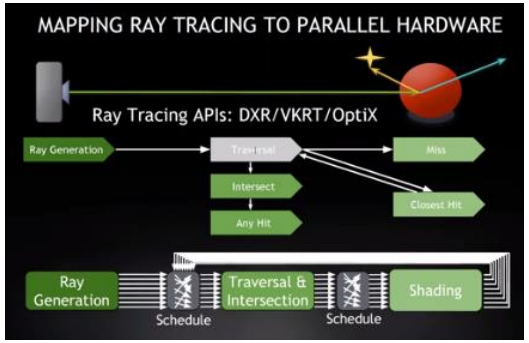
F. Mesh editing & OpenGL integration by Dave

G. Team feedback by all TAs!

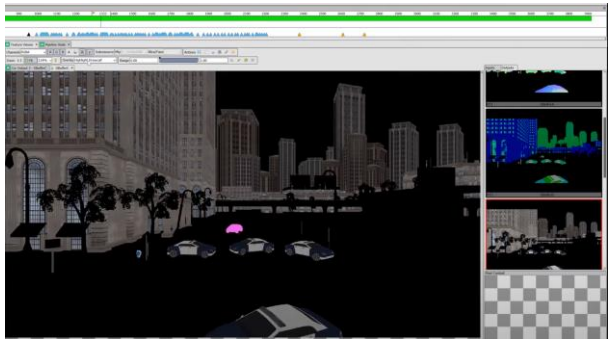
Guest Speakers



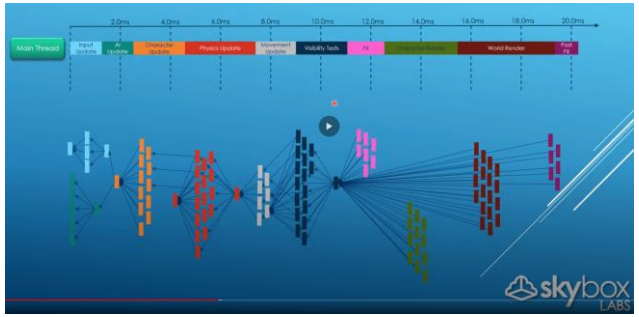
ECS by Yggy King
(Blackbird Entertainment)



Raytracing by Ralf Karrenberg
(NVIDIA)



Debugging by Craig Peters
(Electronic Arts)



Multithreading by David Hiscock
and Pete Quickert (Skybox)

UI by Ben Humphreys
(Brace Yourself Games)

VR by Dinos Tsiknis
(Charm Games)

The Future?

Streamed games?

- ***Cloud gaming (Stadia, GeForce Now, PlayStation Now, xCloud, Luna)***
- ***Streaming content at 60+ fps, up to 4k resolution***
- ***Can that work?***
  - *Multi-player games worked for decades now*
  - *Internet throughput has increased dramatically*
  - *Compression has improved too*
  - *Yes!*
- ***Minimal delay remains***
  - *Predictive input?*

AI

- AI characters

  “I do see a future where, within 10 years, whether it’s through mixed-reality headsets or looking at AR through our phones, we’ll have this concept of, ‘Oh, I hang out occasionally with this NPC who remembers me and who I have this conversation with.’” --Mitu Khandaker

- Infinite content creation
  - Worlds, quests, art, …
  - Interactive with user preferences / guidance

Natural Communication

- ***Body language***
  - Explicit gestures
  - Subtle emotions
- ***Voice control***
- ***Haptic feedback***
- ***Brain interfaces?***
  - *natural???*

How do you see the future?