Game Technology

Lecture 1 – 24.10.2015
Input and Output

Dr.-Ing. Florian Mehm

Prof. Dr.-Ing. Ralf Steinmetz
KOM - Multimedia Communications Lab

Welcome!

Florian Mehm

- Favourite Game: The Longest Journey
- Studied Computer Science in Darmstadt
- PhD at Multimedia Communications Lab, Serious Games
  - Focus on authoring tools for games
- Since 2015: Game programmer @Subiculum Interactive GmbH (Limbic Entertainment)
- Working on an unannounced project with Unreal Engine 4

(Robert Konrad)

- Favourite Game: Super Hexagon
- Studied Computer Science in Darmstadt
- No PhD ☹
  - Open source game tech developer

Organization

Lecture (V2)

- Lecturer: Florian Mehm
- Attendance is not required
- The lectures will be recorded and provided for you

Exercise (Ü2)

- Theory and implementation (game programming)

Language

- Answers are accepted in German and English (exercises and exam)

Block format in 2015

Date	Lecture	Topic
24.10.2015	**1**	**Input and Output**
	2	**The Game Loop**
	3	**Software Rendering**
	4	**Advanced Software Rendering**
28.11.2015	5	Basic Hardware Rendering
	6	Bumps and Animations
	7	Physically Based Rendering
	8	Physics 1
19.12.2015	9	Physics 2
	10	Procedural Content Generation
	11	Compression and Streaming
	12	Multiplayer
23.1.2016	13	Audio
	14	Artificial Intelligence
	15	Scripting

Organization

Exam

- Saturday, February 20th, 2016
- 90 Minutes
- 11:30 – 13:00
- S101/A1

Organization

Sign up with TuCan

Consultation hour

- Planned to be online
- Details will be announced on the forum

Current news

- Website@KOM: http://www.kom.tu-darmstadt.de/teaching/current-courses/sg-lecture0/overview1/
- Wiki, including the lecture slides and script: wiki.ktxsoftware.com
- Fachschafts-Forum: https://www.fachschaft.informatik.tu-darmstadt.de/forum/viewforum.php?f=557

Exercises

Released after each block

- First exercise will be a special case, intended to bring everyone up to speed with git repositories, engine, …

Exercises will have due dates

- These dates are non-negotiable

Bonus Points

- >50%: 0.3; >70%: 0.7; >90%: 1.0
- The exam has to be passed without the bonus points – bonus is added only after the exam has been passed regularly
- Your bonus points will be uploaded to your git repository

Exercises

Group Exercises

- Allowed to complete exercises in groups up to **3 members**
- Turn in exercises via git until the noted time

Group Formation (1-3 people – please form teams!)

- Choose your own name
- Send group information to game-technology@kom.tu-darmstadt.de, including:
  - Group name
  - Names of all members
  - Mail adresses of all members
- Until Wednesday, October 28, 23:59

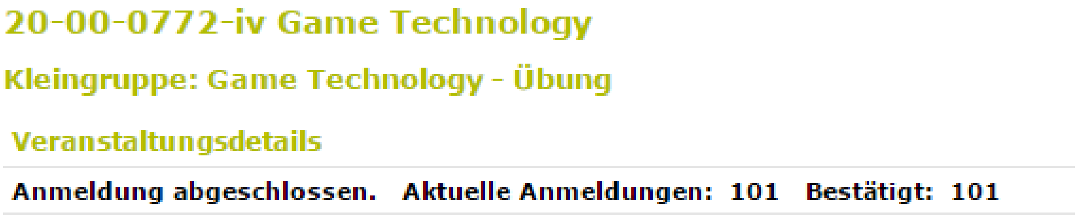
Turning in Solutions

- Theory: Digital, scan written answers or work digitally (PDF, txt, …)
- Source Code: See C++ lecture part

Please form teams!

Last winter term

- First time the lecture was offered
- Expecting a comfortable 30-40 students



20-00-0772-iv Game Technology

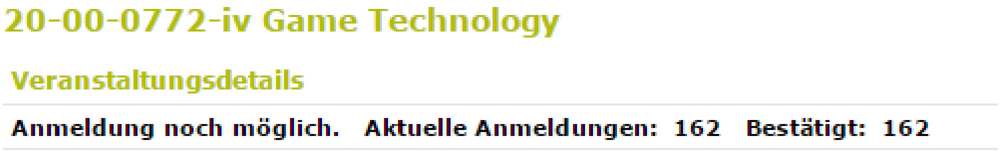
Kleingruppe: Game Technology - Übung

Veranstaltungsdetails

Anmeldung abgeschlossen. Aktuelle Anmeldungen: 101 Bestätigt: 101

This time

- Saturday, block lecture, …
- Expecting a bit less than 101…



20-00-0772-iv Game Technology

Veranstaltungsdetails

Anmeldung noch möglich. Aktuelle Anmeldungen: 162 Bestätigt: 162

Warning

This class will require programming

- C++
- GLSL

This class will be hands-on

- Exercises will be required to understand the topics
- Work sheets will include questions about practical problems and implementation issues

This class will cover a lot of information

- Large parts of the game engine stack
- With many detailed looks into the implementations

But, it will also be fun ☺

Relation to other lectures

Serious Games

- Lecture
- Seminar
- (Projekt)Praktikum

Urban Health Games

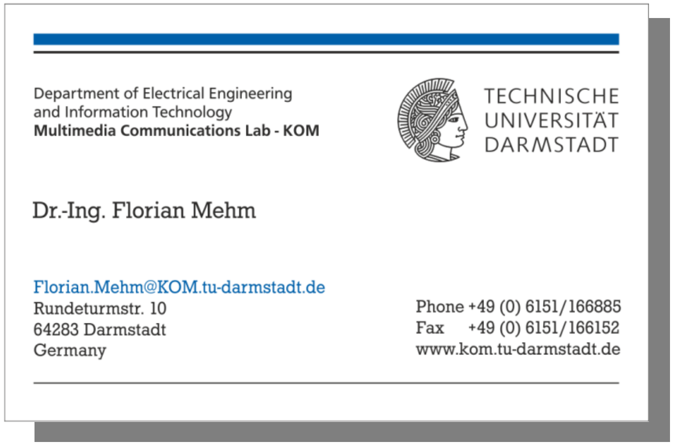
FIF Schwerpunkt Serious Games

fif FORUM
INTERDISZIPLINÄRE
FORSCHUNG

- http://www.fif.tu-darmstadt.de/fif_topics_structure/fif_serious_games_structure_ref/index.de.jsp

Computer Graphics

Questions & Contact



Department of Electrical Engineering
and Information Technology
Multimedia Communications Lab - KOM

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Dr.-Ing. Florian Mehm

Florian.Mehm@KOM.tu-darmstadt.de
Rundeturmstr. 10
64283 Darmstadt
Germany

Phone +49 (0) 6151/166885
Fax +49 (0) 6151/166152
www.kom.tu-darmstadt.de

game-technology@kom.tu-darmstadt.de

Relation to other lectures

Serious Games

- Lecture
- Seminar
- (Projekt)Praktikum

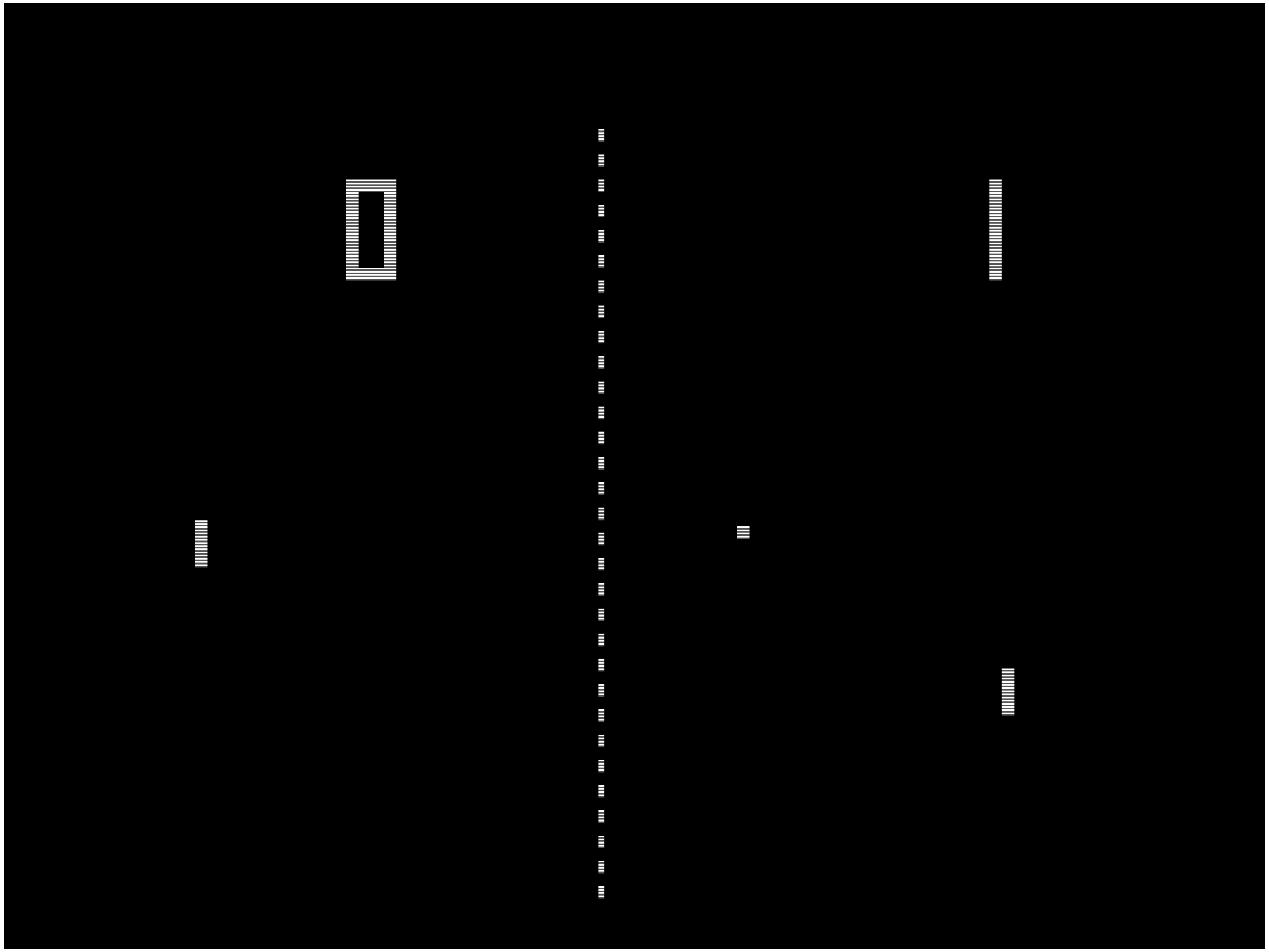
Urban Health Games

FIF Schwerpunkt Serious Games

- http://www.fif.tu-darmstadt.de/fif_topics_structure/fif_serious_games_structure_ref/index.de.jsp

Computer Graphics

Video Games



Pong, 1972

Focus on Performance

Manual memory management

- Pre-loading
- Cache optimization

Shader Programming

Custom data types

…

Separate lecture part for the first lectures

~1 hour theory

~30 minutes programming

Motivation

Shaded Pixels per Second

- 720p @ 30 Hz: 27 million pixels/sec
- 1080p @ 60 Hz: 124 million pixels/sec
- 30” Monitor 2560x1600 @ 60 Hz: 245 million pixels/sec
- 4k Monitor 4096x2160 @ 30 Hz: 265 million pixels/sec
- **VR 1512x1680x2 @ 90 Hz: 457 million pixels/sec**

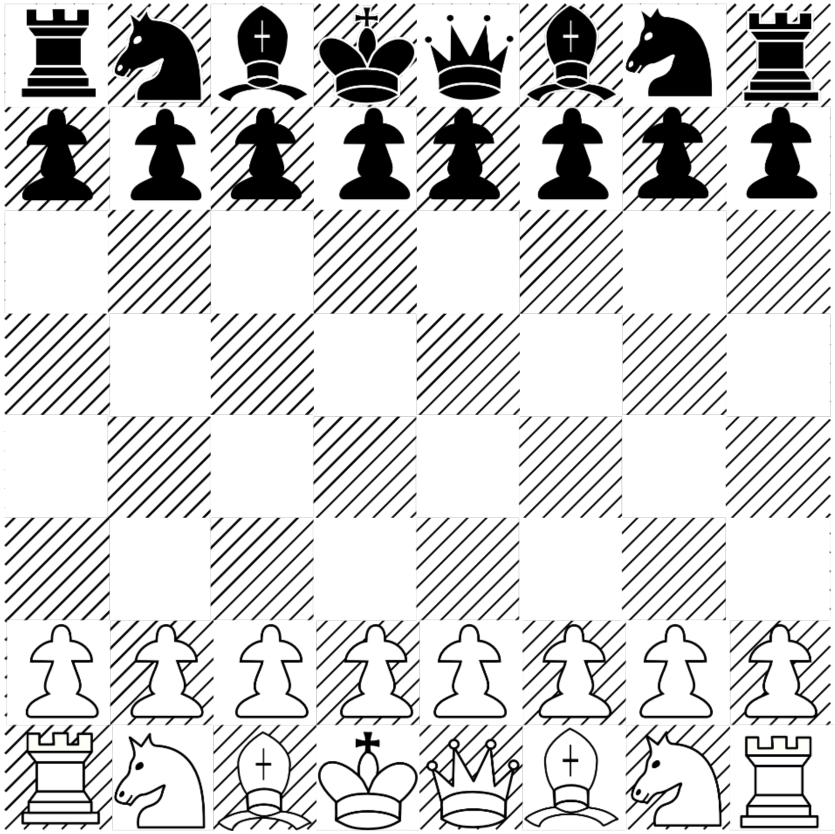
Even with Off-the-Shelf Engines, we have to make sure we feed them the right input

Pseudo-realistic realtime simulations

Pseudo-realistic realtime simulations

No chess

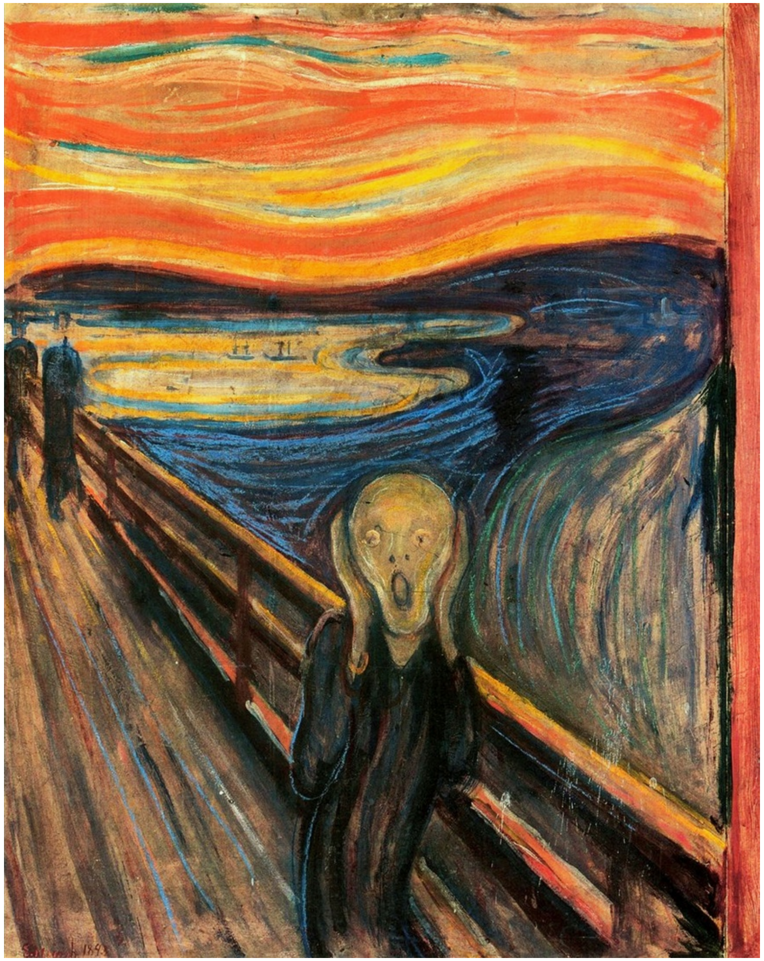
- Focus on fast/realtime apps
- Running in a game loop



Pseudo-realistic realtime simulations

No „artsy“ games

- But understanding how to make realistic games also helps with non-realistic games



Pseudo-realistic realtime simulations

No flight simulators for Lufthansa

- Actual realism not necessary
  - …and probably too slow
- Requires knowledge of human perception

Human-Machine data transfer

Human

- Output
  - Pushing
  - Talking
  - Moving
- Input
  - Staggering amounts of data

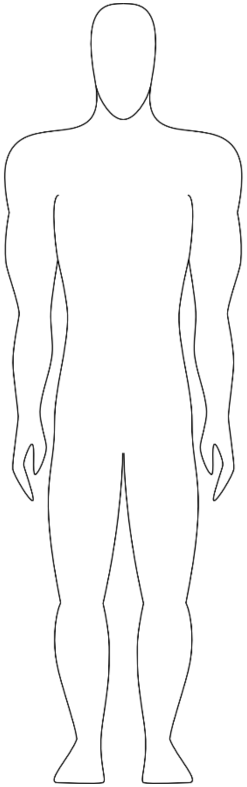
Machine

- Output
  - Monitor
  - Speakers
- Input
  - Buttons

Humans

Five senses

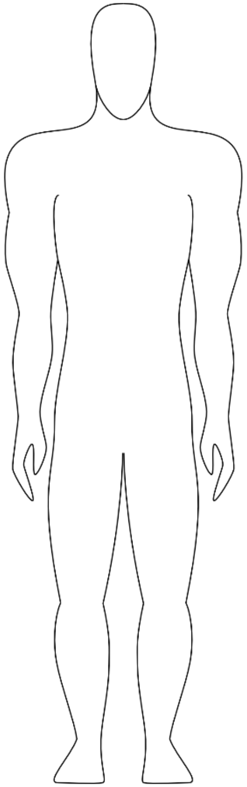
- Sight
- Hearing
- Touch
- Smell
- Taste



Humans

Many senses

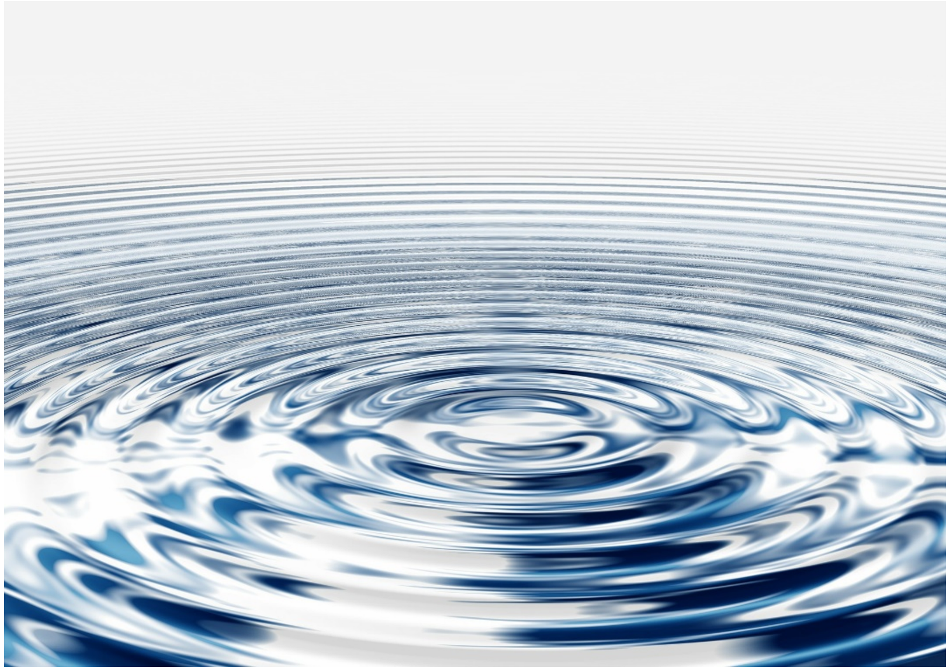
- External
  - Sight
  - Hearing
  - Touch
  - Smell
  - Taste
  - Acceleration
  - Temperature
- Internal
  - Kinesthetic
  - Pain
  - …



Eyes and Ears

Most dominant sensors

Measure different kinds of waves



Waves

Wave Direction

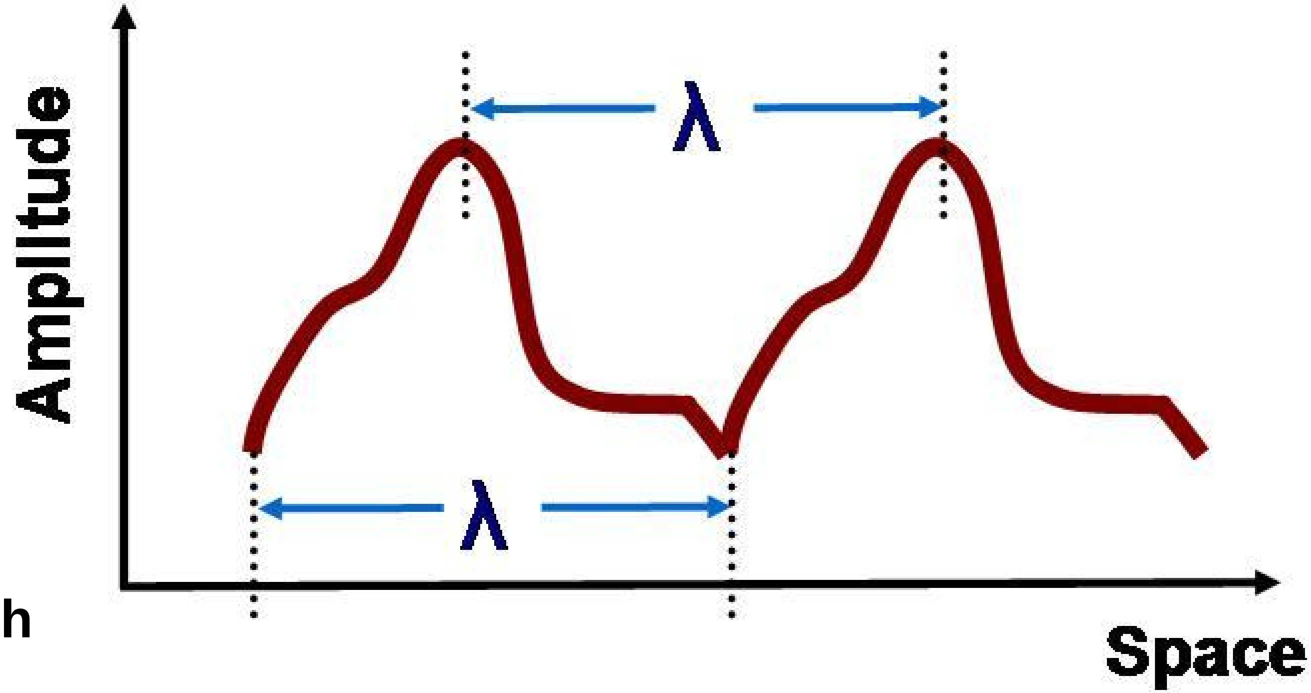
Oscillation Direction (for transverse waves)

Amplitude

Speed (often constant)

Wavelength

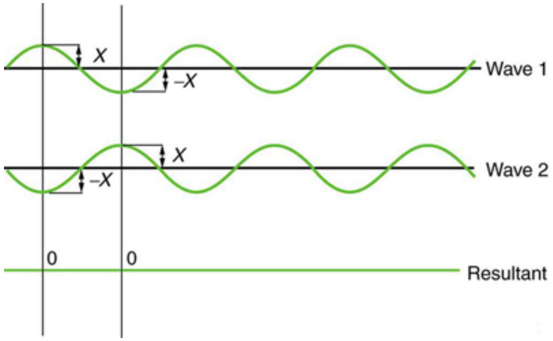
Waveform

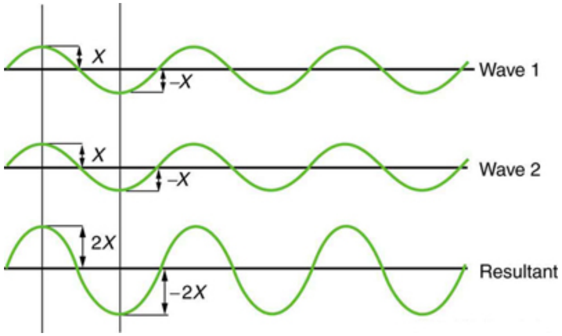


Frequency =
Speed / Wavelength

Wave Interaction

Superposition





Light Waves

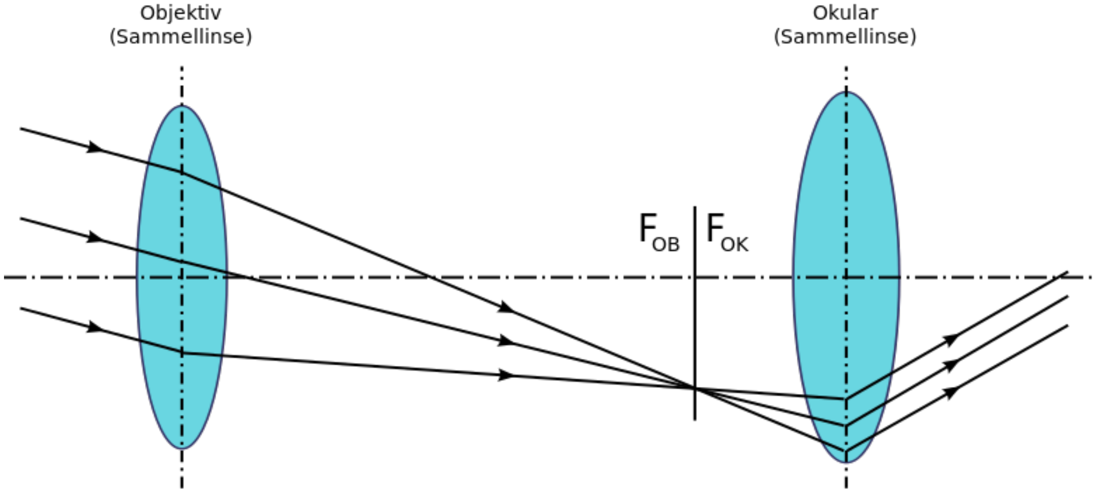
Electromagnetic waves

Transverse waves

- Direction of oscillation orthogonal to wave direction

Very fast

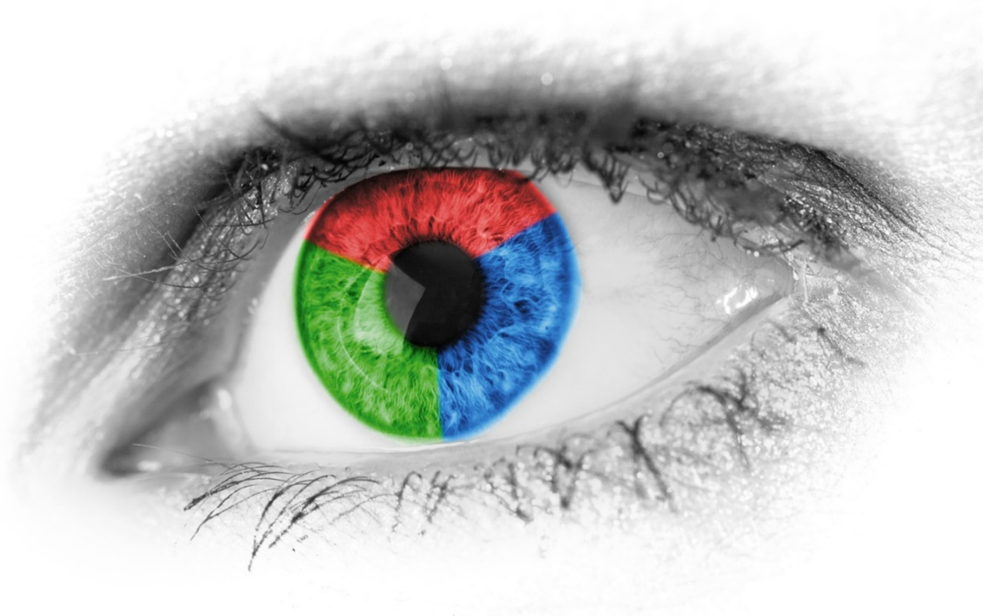
Usually discussed using simplified models



Optical Sensors

Two units

- Surround view or 3D view depending on arrangement



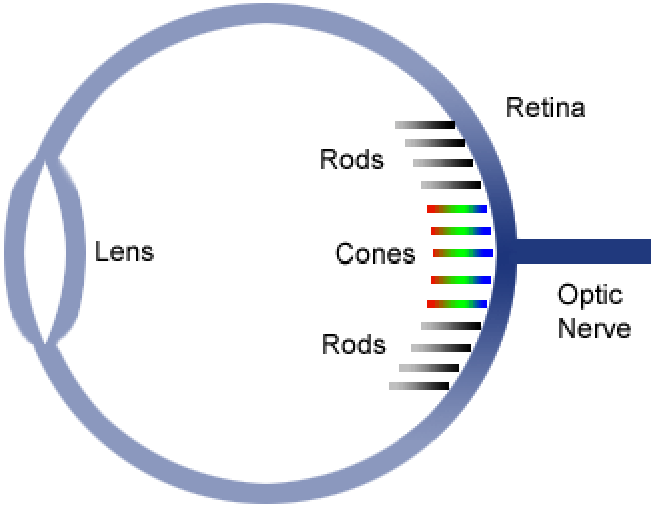
The eye

The lens focuses light on the retina

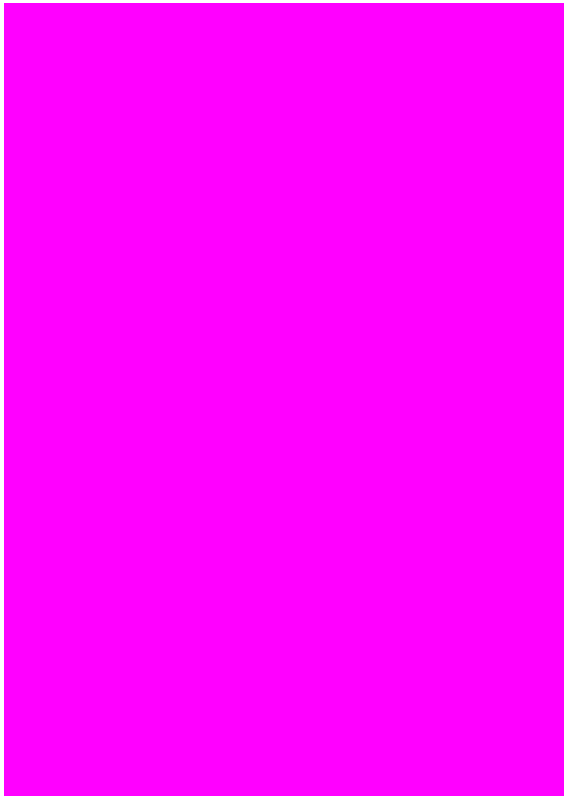
Rods measure light intensity/energy (wave amplitude and frequency)

Cones only react to specific wavelengths

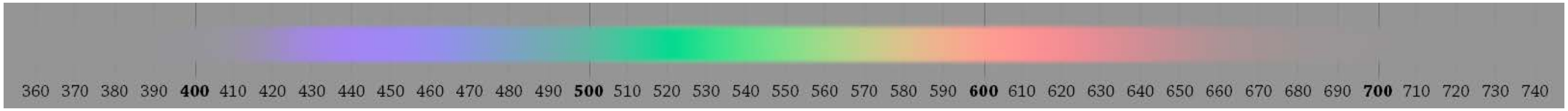
- Three different kinds
  - Red,
  - green, and
  - blue



What do you see?



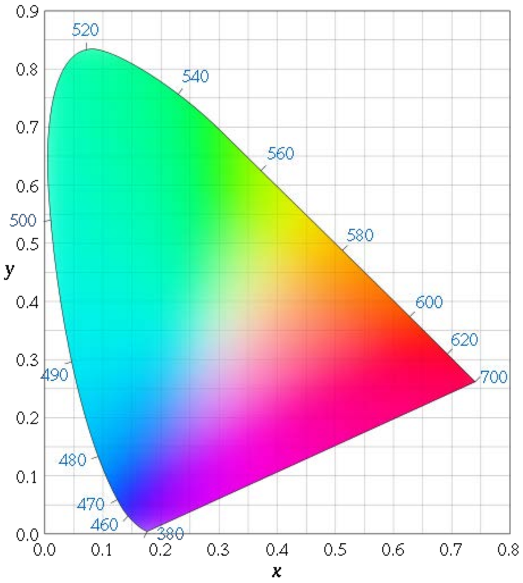
Red, green and blue



Brain interpolates colors

Brain sees magenta when interpolation fails

- Same amounts of blue and red but no green
- See http://richannel.org/colour-mixing-and-the-mystery-of-magenta



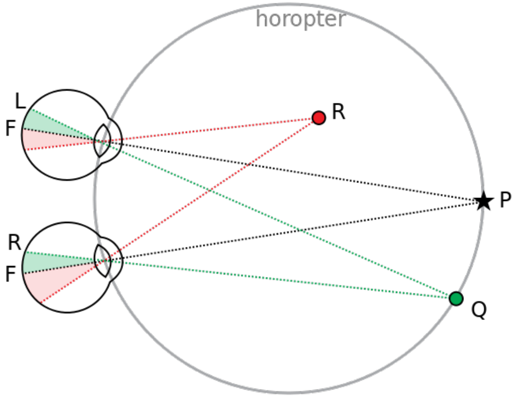
Stereo Vision, Depth Perception

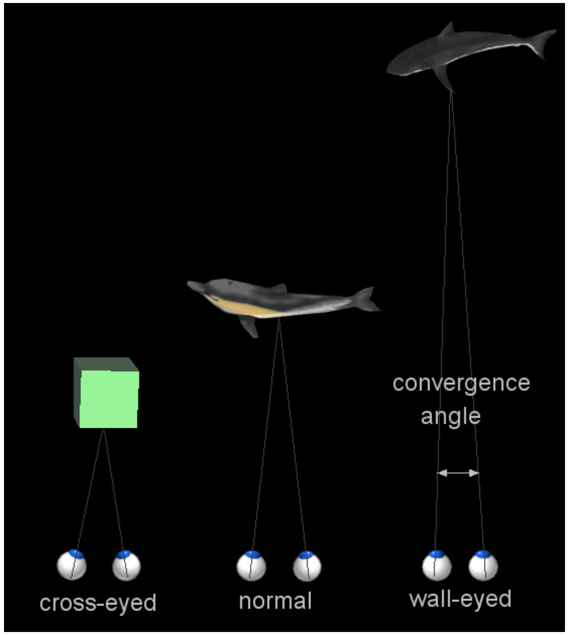
Binocular cues

- Stereopsis: Triangulation using difference in both eyes
- Convergence: Using muscles in the eyes
- Shadow Stereopsis

Distance between eyes

- Interpupillary Distance
- ~6.5 cm in humans





Stereo Vision, Depth Perception

Monocular Cues

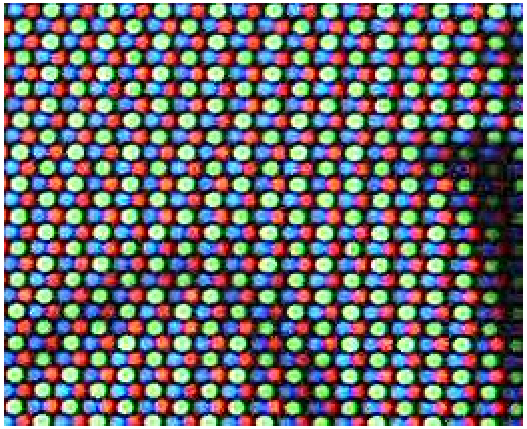
- Motion parallax
- Depth from motion
- Kinetic depth effect
- Perspective
- Relative size
- Familiar size
- Absolute size
- Accommodation
- Occlusion
- Curvilinear perspective
- Texture gradient
- Lighting and shading
- Defocus blur
- Elevation

Monitors

Exact counterpart to human eye

Red, green and blue emitters

No physically accurate picture reproduction



Computer → Monitor

Designated memory area which is transferred to the monitor

- The framebuffer

Structurally equivalent to the pixel structure

- 1 red byte
- 1 green byte
- 1 blue byte, …

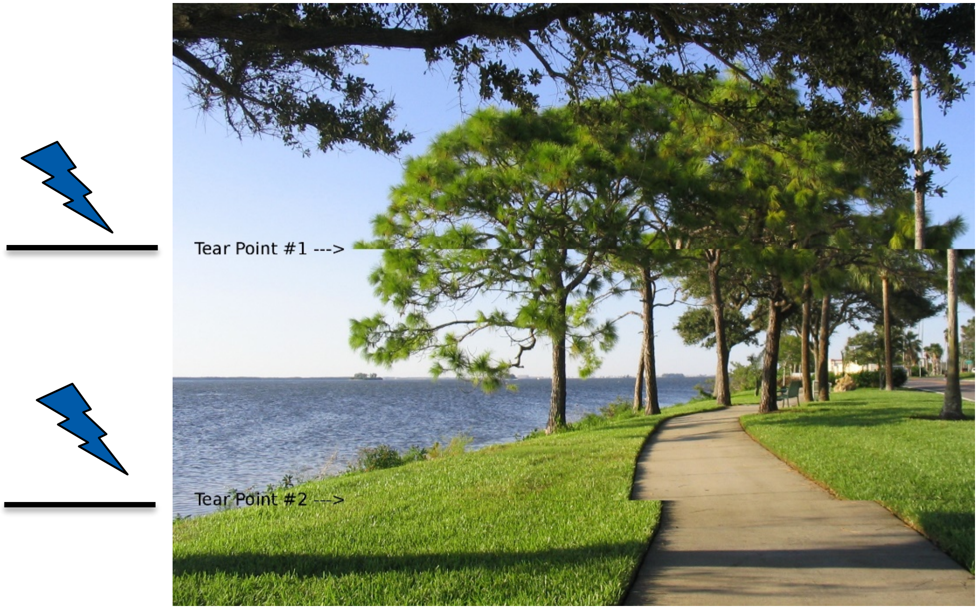
Vertical Sync

Monitors typically operate at framerates of 60 Hz

Picture is transfered during a designated timeslot (vblank)

Game has to wait for that timeslot after image calculations are done, or else…

- Tearing
- Display of different images intermixed



Double Buffering

Render image to off-screen buffer

Wait for vblank signal

Set buffer as monitor input array

Switch to previous buffer

Repeat

Triple buffering

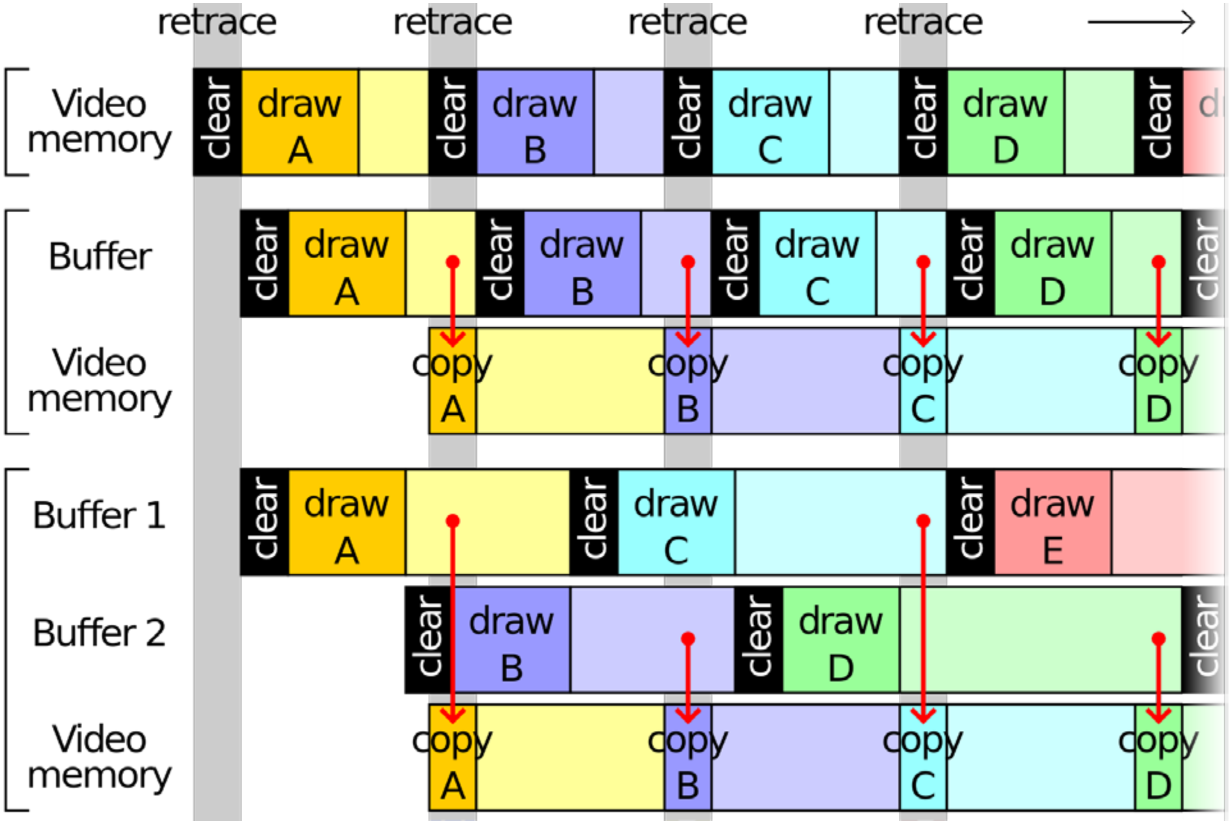
- Additional buffer to avoid waiting time

The new thing - G-Sync (nVidia), Freesync (AMD)

- Dynamic monitor framerate
- Transfer image when finished

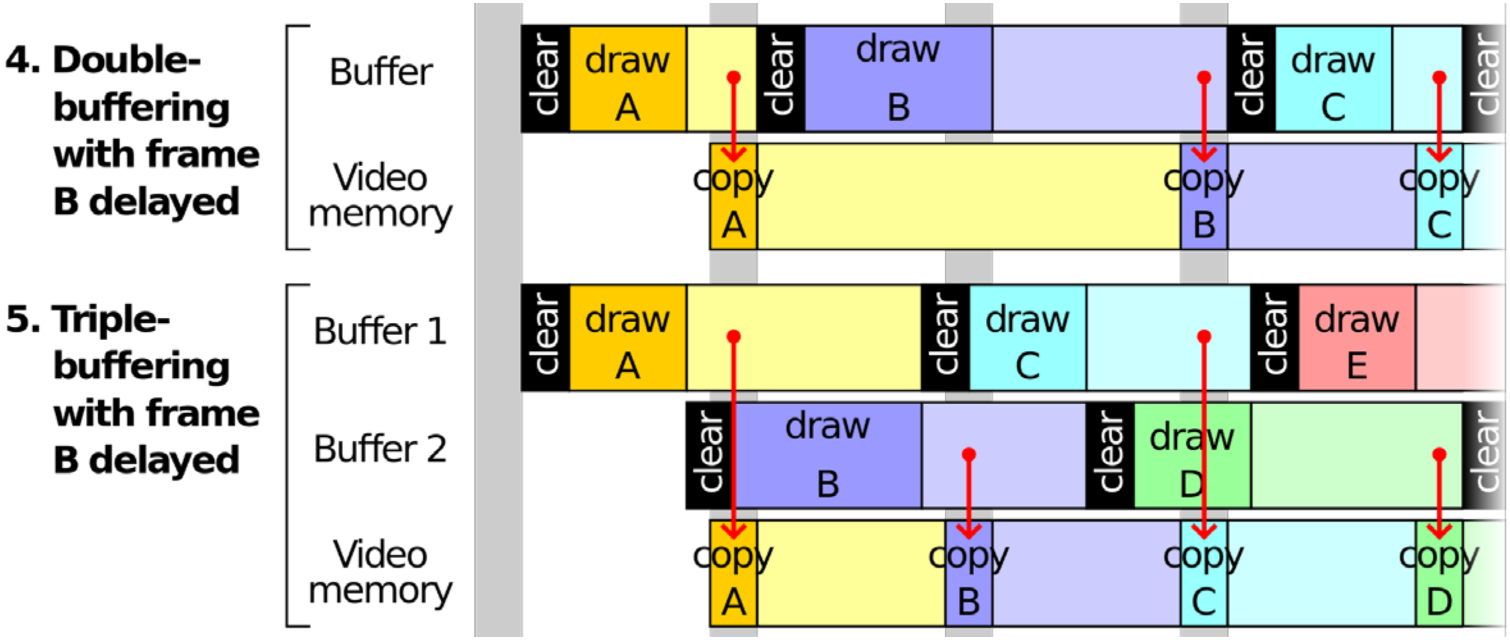
Buffering 1/2

- **Single-buffering**
- **Double-buffering**
- **Triple-buffering**



Source:
https://commons.wikimedia.org/wiki/File:Comparison_double_triple_buffering.svg

Buffering 2/2



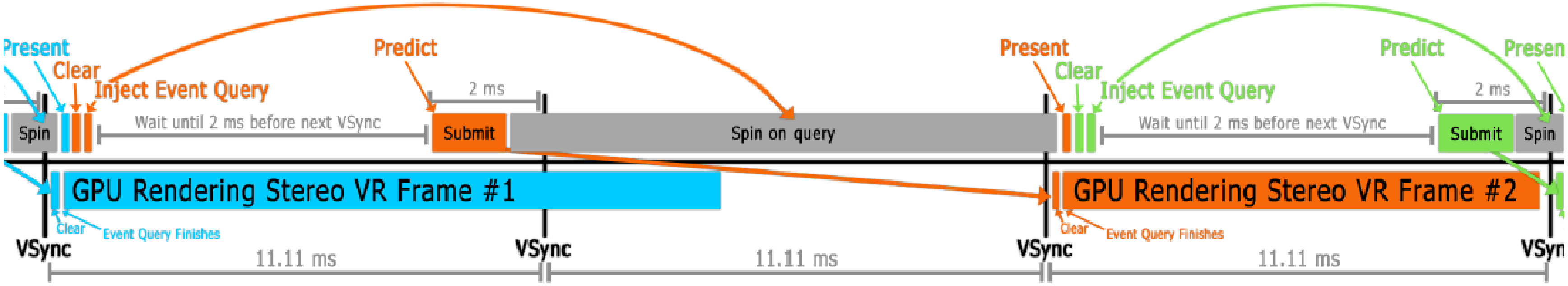
Virtual Reality

„Motion to Photons“ important metric

- The less time it takes for user actions to result in changed images, the better

Double (Triple!) Buffering introduce delays

- E.g. we take a bit longer to render our frame
- Wait for the rest of the frame



http://www.gdcvault.com/play/1017797/Why-Virtual-Reality-Is-Hard

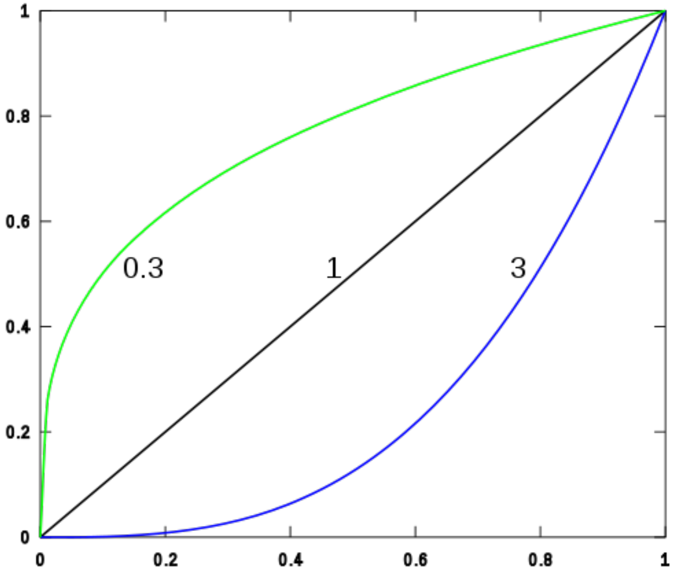
http://www.gdcvault.com/play/1021771/Advanced-VR

Gamma

Monitors do not emit 50% light intensity for a 50% light value

Work according to a gamma function

$$I_{out} = {I_{in}}^{\gamma}$$



Monitor color space is not ideal for lighting calculations

Usually we choose $\gamma = 2.2$

More info: http://http.developer.nvidia.com/GPUGems3/gpugems3_ch24.html

Gamma correction

Input from uncorrected images

- Raise values to the power of $\boldsymbol{\gamma}$

Handle calculations in linear space

Output to the monitor

- Raise output values to the power of $\frac{1}{\boldsymbol{\gamma}}$

Sound Waves

Air compression

Longitudinal Waves

~343 m/s

Sound sensors

Also two units

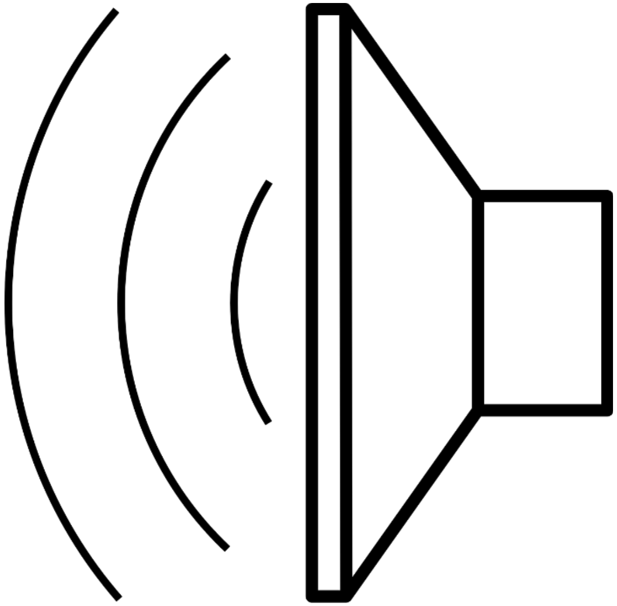
Infer direction by measuring time differences

Measure actual wave forms

Loudspeakers

Construct actual sound waves

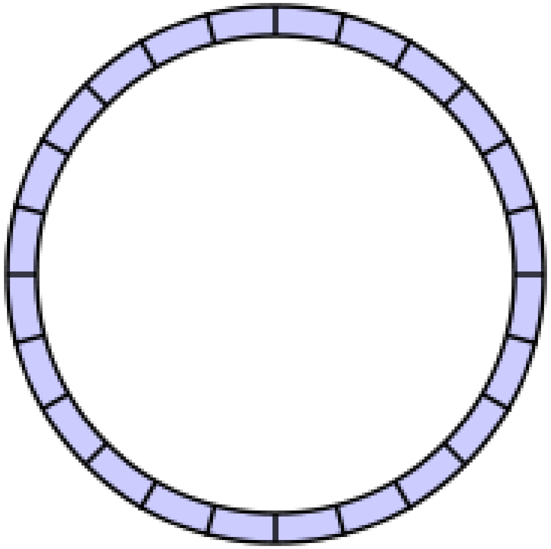
Physically accurate reproduction of original waves



Computer → Speaker

Small ring buffer

- Write samples into the buffer
- Read back during playback



Discretely sampled waveform

Pointer to last sample written

Pointer to next sample to read

Sound Mixing

Superpositioning

- Adding waves

Again physically accurate

Actual danger of superposition effects

- Avoid mixing identical sounds
- In reality, events rarely/never happen at the exact same time

Superposition effects

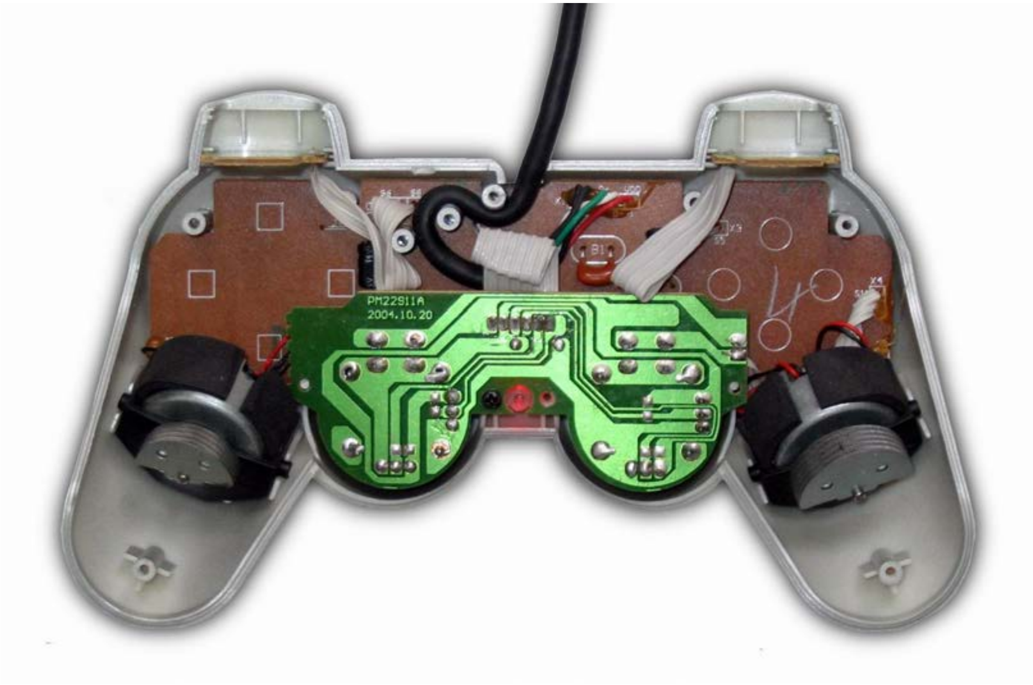
Not just in sound

Easy to spot by human observers

```
int numSpawn;
for (int i = 0; i < numSpawn; i++)
{
	NPC* npc = new NPC();
	World.PlaceActorRandomly(npc);
	npc->StartAnimation("Dance");
}
```

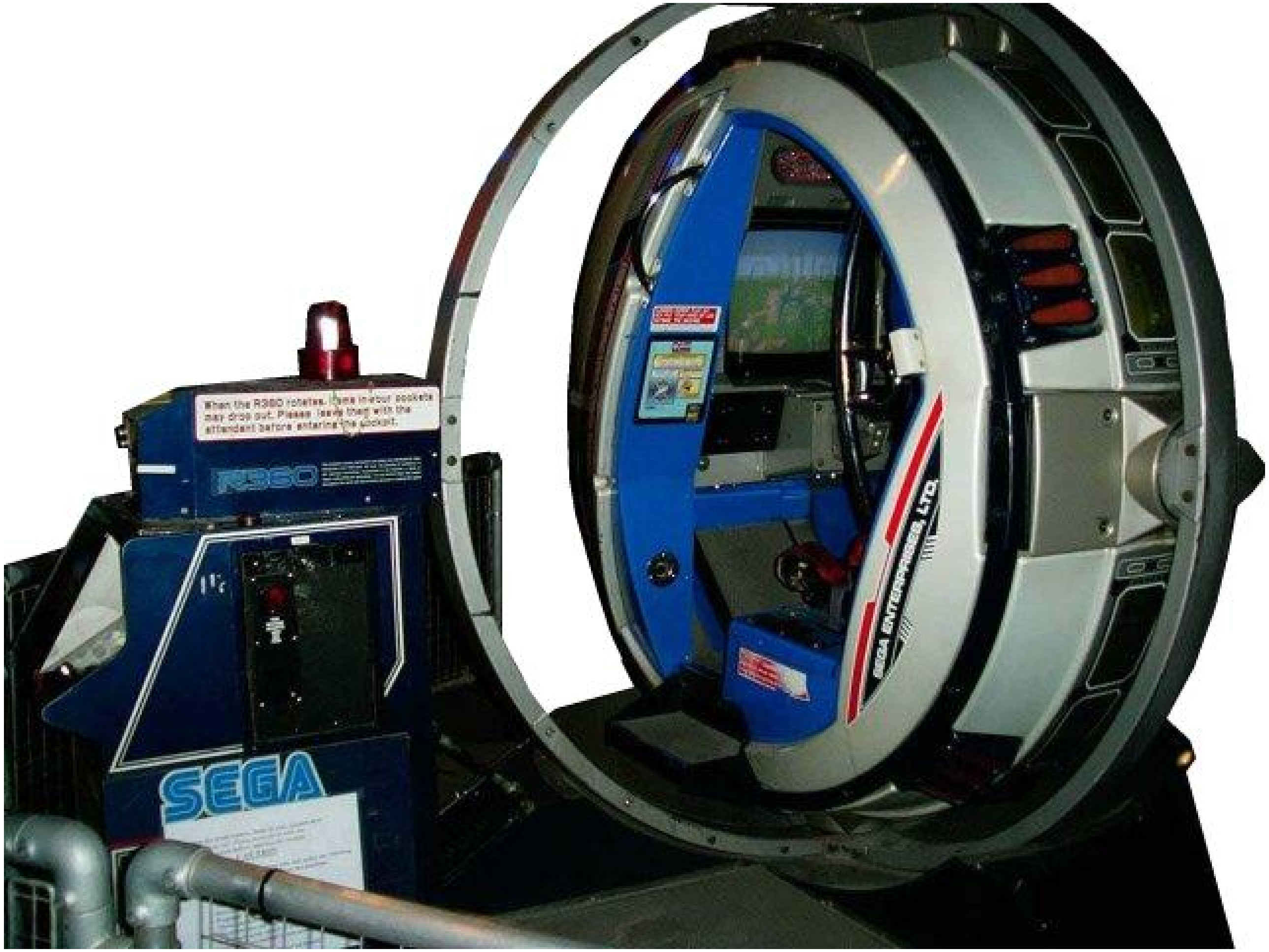
Rumble / Force Feedback

Very restricted „touch“ output



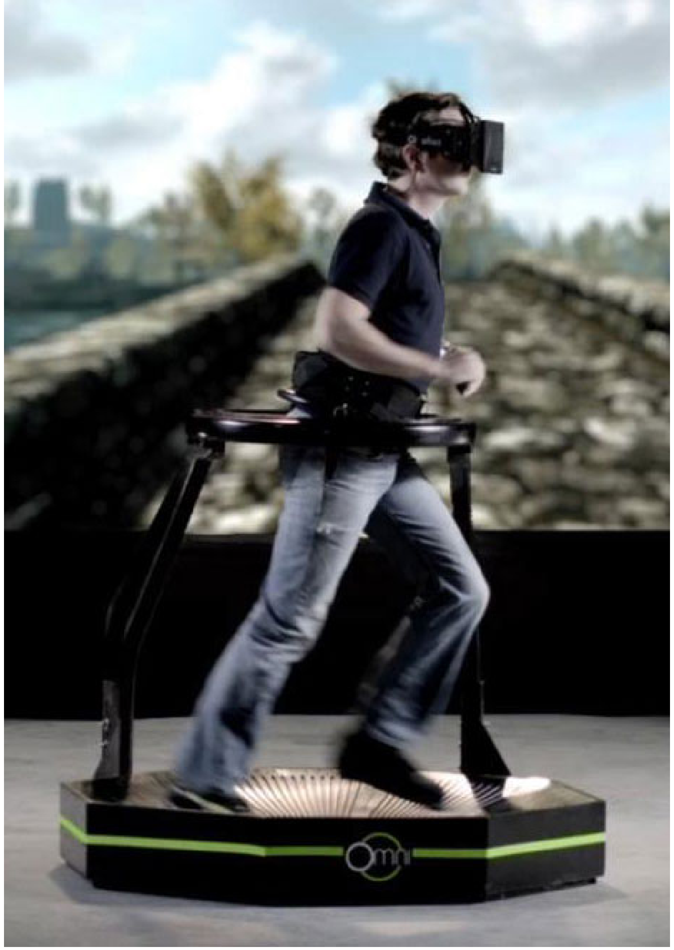
Acceleration output

Sega R-360


When the R360 rotates, items in your pockets may drop out. Please leave them with the attendant before entering the cockpit.
R360
SEGA
SEGA ENTERPRISES, LTD.

Kinesthetic

Virtuix Omni



Computer input

Mouse, Keyboard, Gamepad, …

Mostly trivial

Important to reduce input lag

- Minimize time from input to output
- Triple buffering harmful

Complex computer input

Input inaccuracies

- Compensate by being overly optimistic

https://www.youtube.com/watch?v=KWbLOFGSEDo

C

Portable assembler

Developed for/with UNIX

From 1969

Dennis MacAlistair Ritchie (September 9, 1941 – c. October 12, 2011)

C/C++

Open standards, not bound to a company

Available almost anywhere

- Even in the browser (Emscripten)

Bjarne Stroustrup

C++

Adds higher level concepts to C

No performance regressions

Originally „C with classes“

From 1979

Much work since then

- C++11
- Latest: C++14 – to be covered later
- C++17?

Classes

```
class Foo {
public:
  Foo() {
          x = 2;
  }
private:
  int x;
};
```

Free functions

```
int main(int argc, char** argv) {
  return 0;
}
```

Main entry point

- But not on every system

* is a pointer

- A memory address

char* is used for strings

char** - multiple strings

Header files

Using multiple source files is complicated

Compiler compiles single cpp file to object file

- Files can #include other files in a preprocess
- Use separate, minimal header files for #include

A separate linker application links multiple object files

No standard to tell the linker what to do

Primary reason that compiling C/C++ is slow

Foo.h

```
#pragma once

class Foo {
public:
  Foo();
private:
  int x;
};
```

#pragma once is not part of the standard, but widely adopted

- Easier to write and read than other way of include guards

Foo.cpp

```
#include “Foo.h”

Foo::Foo() {
    x = 2;
}
```

C++ in 20XX

Very big language

Complex features

- Templates (similar to Java‘s generics) are turing complete

Contains fancy library

- Automates memory management somewhat
- std::string, std::vector, …

boost Library

- Widely used
- Big, std style library

C/C++ in Games

Typically C with just a few C++ features

Avoid templates

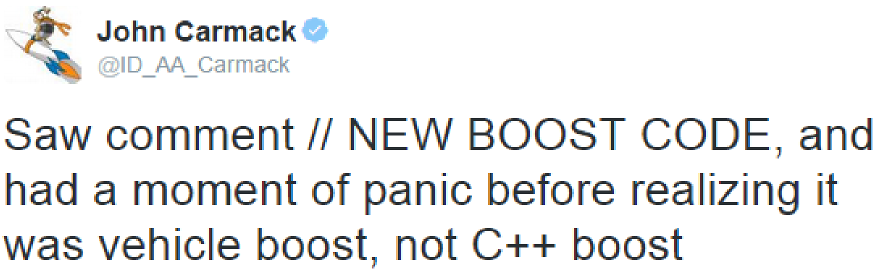
- Very hard to debug

Avoid exceptions

- Can have performance impact
- Can introduce resource leaks

Avoid C++ standard library

- Different implementations
- Unpredictable allocations



John Carmack
@ID_AA_Carmack

Saw comment // NEW BOOST CODE, and had a moment of panic before realizing it was vehicle boost, not C++ boost

Hardware Access

Files

- That‘s it

No support for

- Special directories
- Memory mapped files
- …

OpenGL

Standard API for Graphics Hardware

Many different versions

Not on consoles

Questionable support by Apple and Microsoft

GPU Programming Languages

GLSL

- Part of OpenGL

HLSL

- Microsoft (Direct3D and Xbox)
- Sony (all PlayStations)

Metal

Apple

Audio, Keyboard

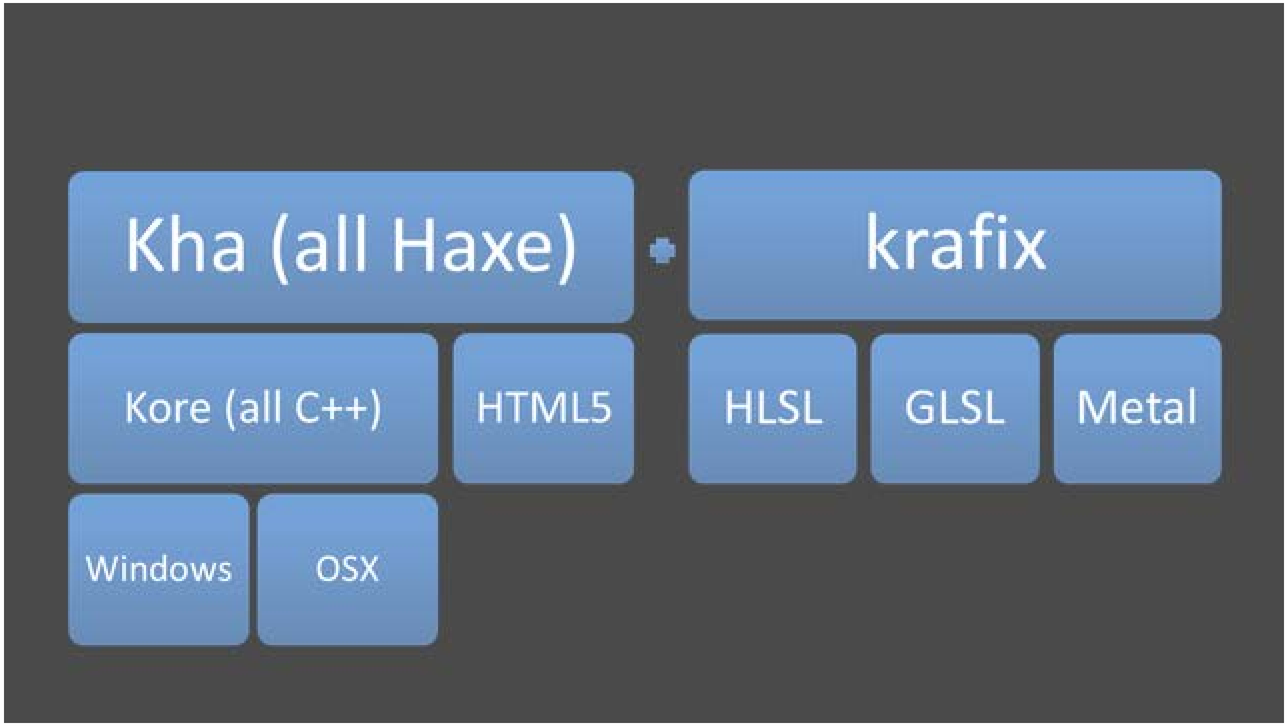
Practically no standards

SDL can do the job

Kore

- **APIs for**
  - Graphics
  - Audio
  - Input Devices
  - File Access
  - …
- **GLSL cross compiler**
- **https://github.com/KTXSoftware/Kore**
- **Introductions at http://wiki.ktxsoftware.com**

Kha



Kha (all Haxe)

krafix

Kore (all C++)

HTML5

HLSL

GLSL

Metal

Windows

OSX

https://www.youtube.com/watch?v=vGQjIfq7BwI

http://tech.ktxsoftware.com/wwx-new-part-3-the-slides/