Game Technology

Lecture 1 – 17.10.2014

Dipl-Inf. Robert Konrad
Dr.-Ing. Florian Mehm

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

Welcome!

Introduction

Robert Konrad

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

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

Organization

Lecture (V2, weekly)

- Friday, 9:50 to 11:30, S103/9
- Lecturers: Robert Konrad, Florian Mehm

Exercise (Ü2, weekly)

- Friday, after lecture, 11:40 – 13:20, S103/100
- Theory and implemention (game programming)
- Each week 1 exercise, 1 week to work on the task

Exam

- 90 Minutes
- Date and location TBD

Organization

Sign up with TuCan

Consultation hour during the exercise

- In case no one shows up for the exercise, we will not wait the whole time slot

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

Each week a new exercise sheet

- Practical (programming) tasks
- Theoretical tasks

Exercise slot on Fridays

- Discuss the previous exercise
- Show interesting solutions

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

Exercises

Group Exercises

- Allowed to complete exercises in groups up to **3 members**
- Turn in exercises via FTP until Friday of the next week before the exercise starts (11:40)

Group Formation

- 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 Tuesday, 21.10.2014, 23:59

Turning in Solutions

- Theory: As PDF, scan written answers or work digitally
- Source Code: See today‘s C++ lecture…

Preliminary timetable

Lecture No.	Date	Topic
1	17.10.2014	Basic Input & Output
2	24.10.2014	Timing & Basic Game Mechanics
3	31.10.2014	Software Rendering 1
4	07.11.2014	Software Rendering 2
5	14.11.2014	Basic Hardware Rendering
6	21.11.2014	Animations
7	28.11.2014	Physically-based Rendering
8	05.12.2014	Physics 1
9	12.12.2014	Physics 2
10	19.12.2014	Scripting
11	16.01.2015	Compression & Streaming
12	23.01.2015	Multiplayer
13	30.01.2015	Audio
14	06.02.2015	Procedural Content Generation
15	13.02.2015	AI

Warning

This class will require programming

- C++
- GLSL
- (Lua)

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

- The whole game engine stack
- With many detailed looks into the implementations

Relation to other lectures

Serious Games

- Lecture
- Seminar
- (Projekt)Praktikum

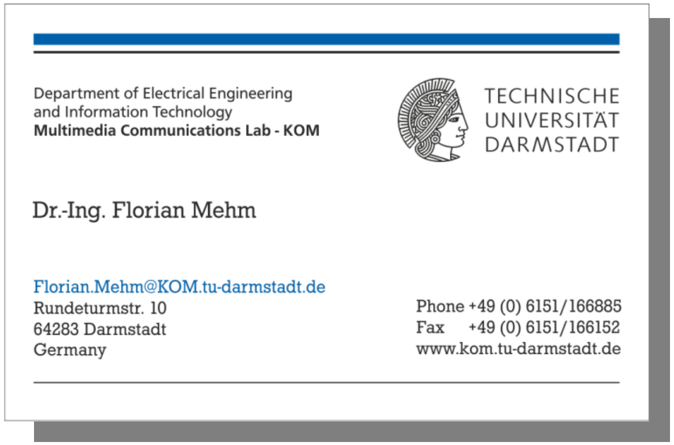
Urban Health Games

FIF Schwerpunkt Serious Games

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

Computer Graphics

Questions & Contact



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

TECHNISCHE
UNIVERSITÄT
DARMSTADT

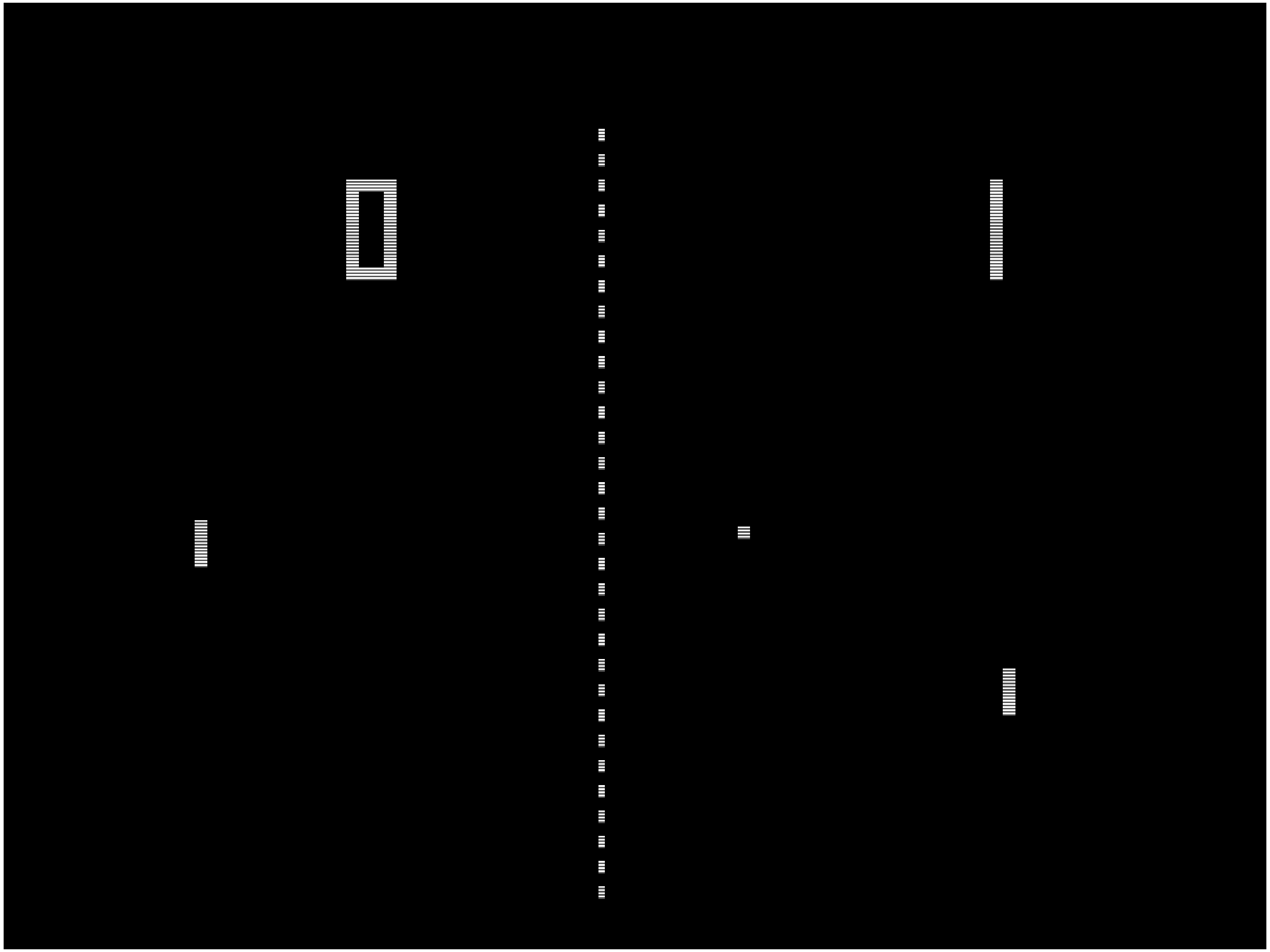
Dr.-Ing. Florian Mehm

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

Video Games



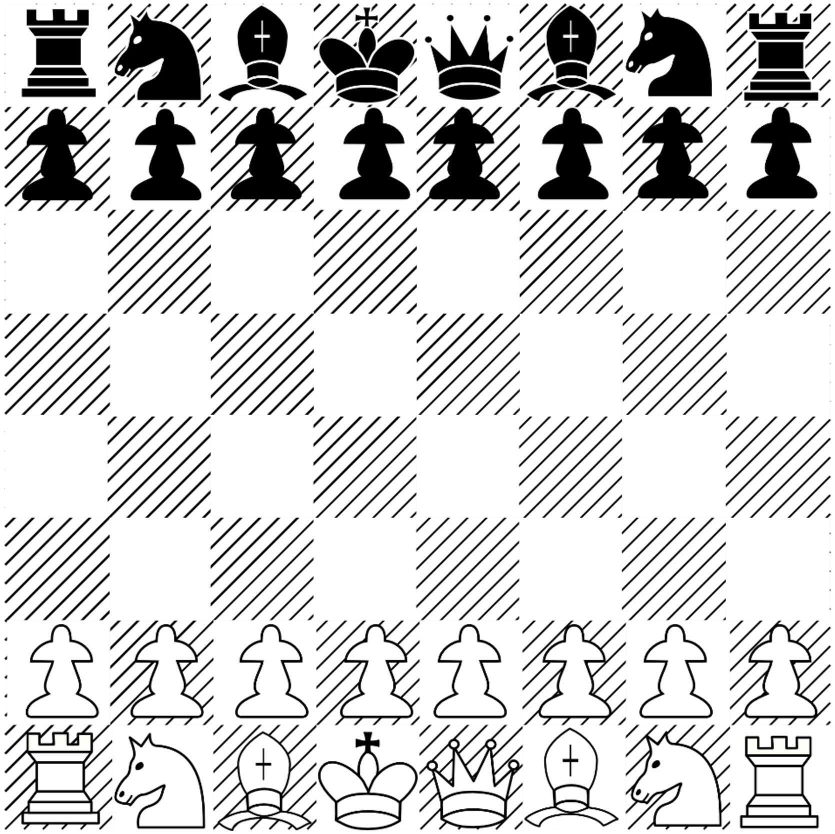
Focus on Performance

- **Manual memory management**
- **Shader programming**
- **…**
- **Separate lecture part**
  - ~1 hour abstract theory
  - ~30 minutes programming

Pseudo-realistic realtime simulations

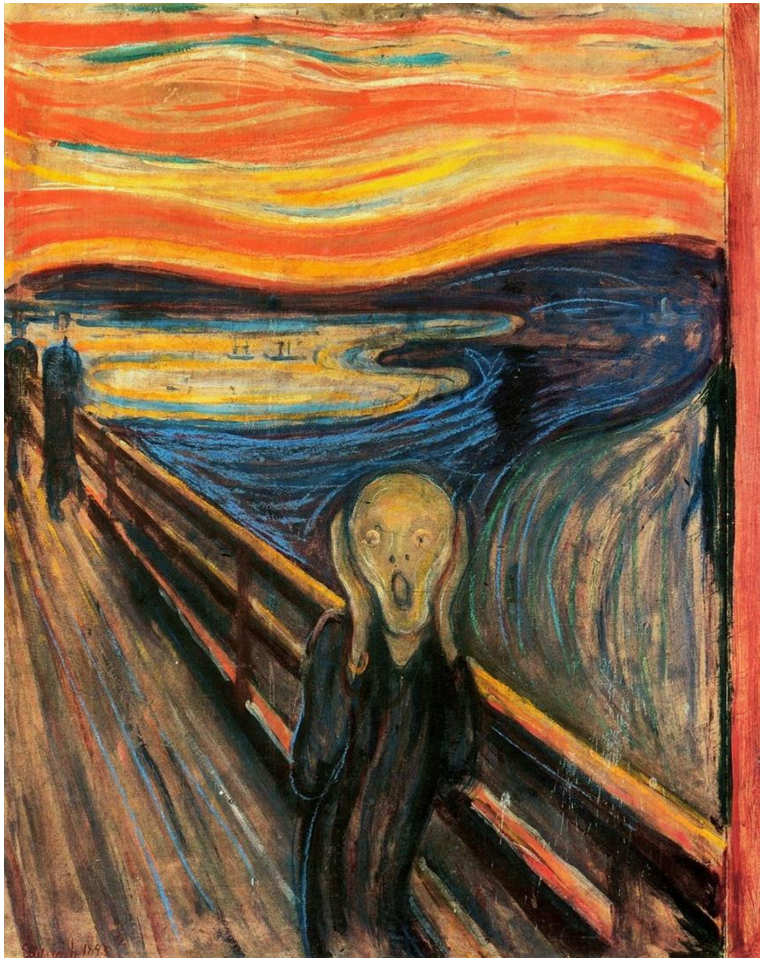
Pseudo-realistic realtime simulations

- **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

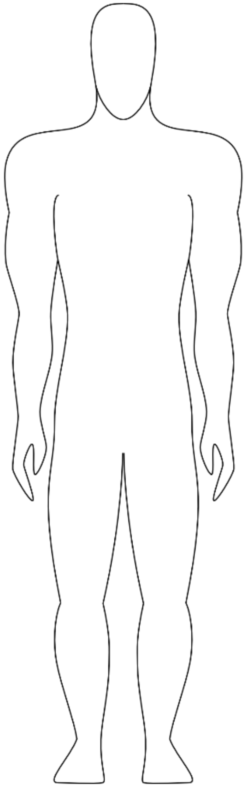
Human-Machine data transfer

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

  - Machine
    - Output
      - Monitor
      - Speakers
    - Input
      - Buttons

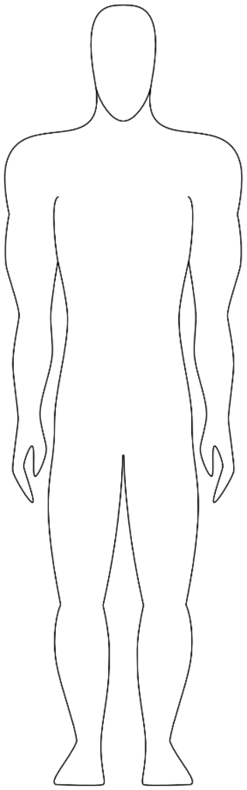
Humans

- **Five senses**
  - Sight
  - Hearing
  - Touch
  - Smell
  - Taste



Humans

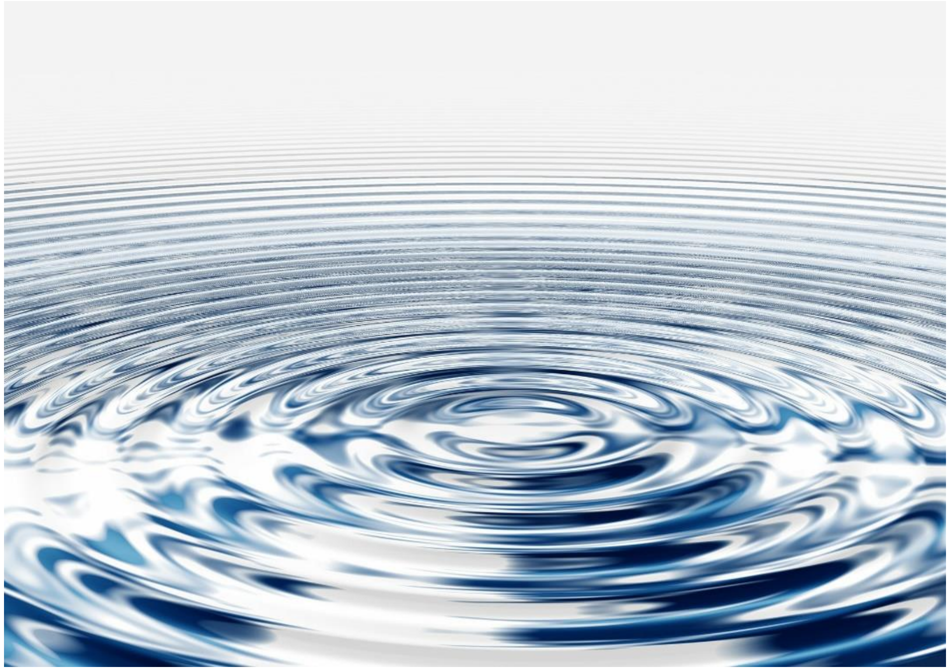
- **Many senses**
  - External
    - Sight
    - Hearing
    - Touch
    - Smell
    - Taste
    - Acceleration
    - Temperature
  - Internal
    - Kinesthetic
    - Pain
    - …



Eyes and Ears

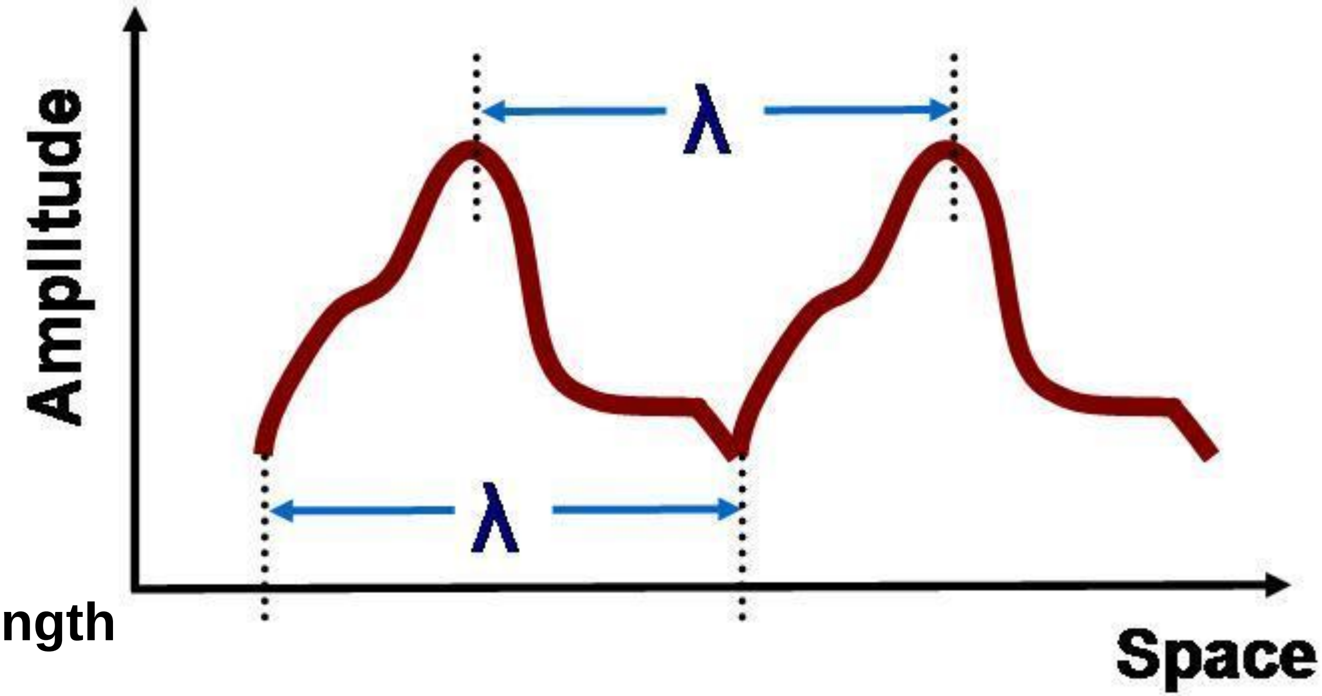
- **Most dominant sensors**

- **Measure different kinds of waves**



Waves

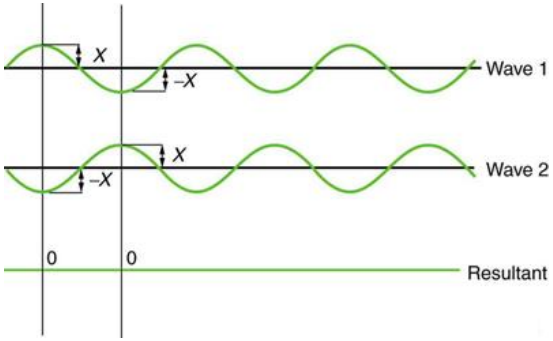
- **Wave Direction**
- **Oscillation Direction (for transverse waves)**
- **Amplitude**
- **Speed (often constant)**
- **Wavelength**
- **Waveform**

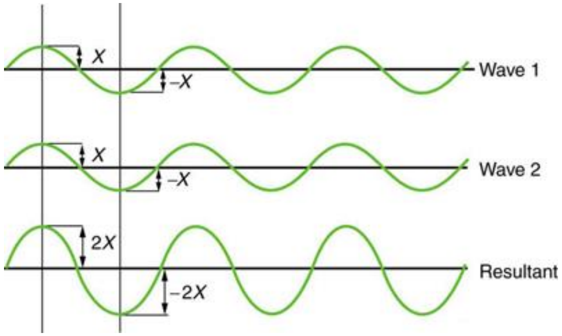


- **Frequency = Speed / Wavelength**

Wave Interaction

- **Superposition**





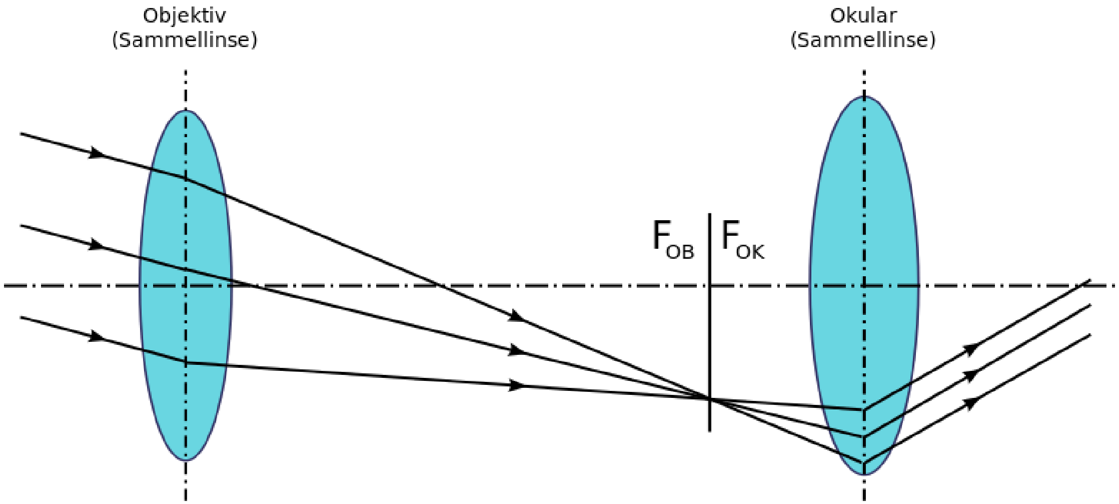
Light Waves

- **Electromagnetic waves**

- **Transverse waves**
  - Direction of oscillation orthogonal to wave direction

- **Very fast**

- **Usually discussed using simplified models**

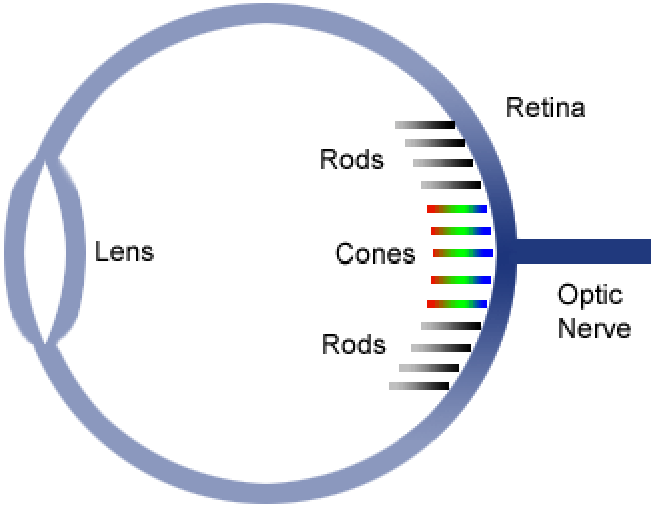


Optical Sensors

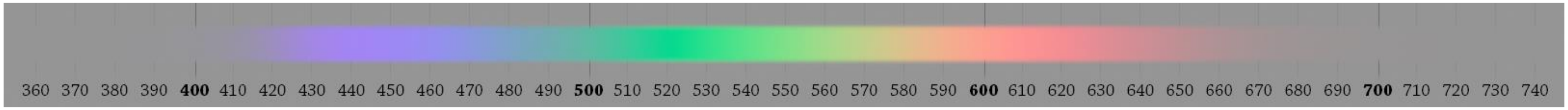
- **Two units**
  - Surround view or 3D view depending on arrangement

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

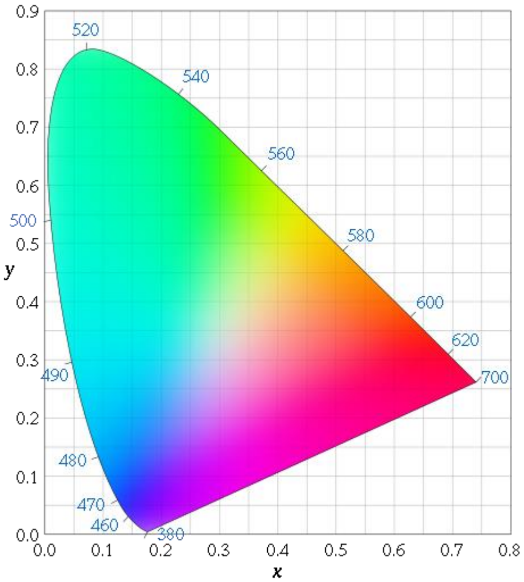


Red, green and blue



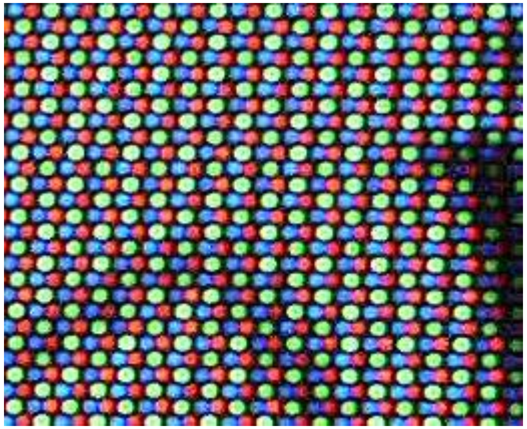
- **Brain interpolates colors**
- **Brain sees magenta when interpolation fails**
  - Same amounts of blue and red but no green

- **Colors are not just wavelengths**
- **2D value**



Monitors

- **Exact counterpart to human eye**
- **Red, green and blue emitters**
- **No physically accurate picture reproduction**



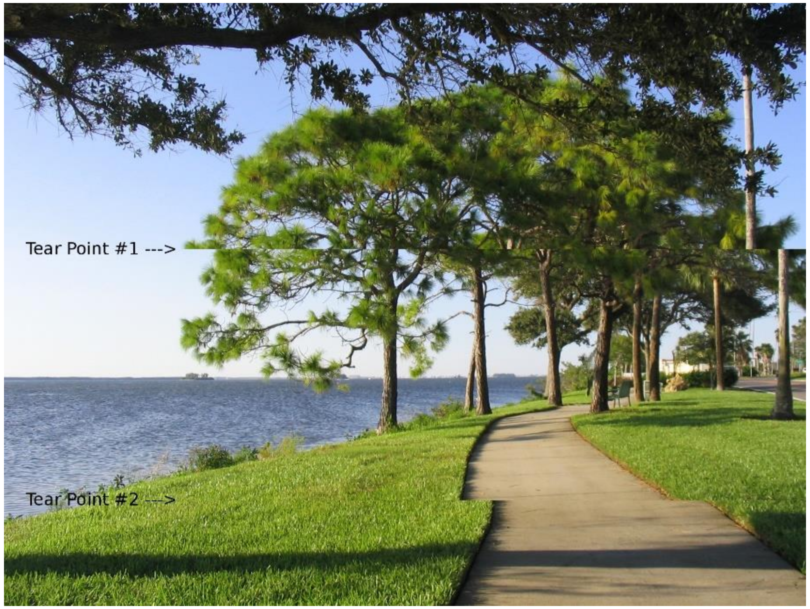
Computer -> Monitor

- **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…



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

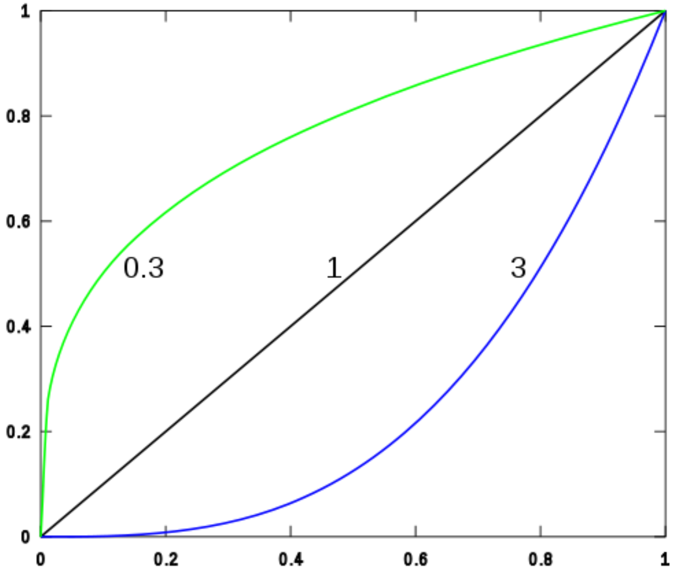
- **The new thing - G-Sync (nVidia), Freesync (AMD)**
  - Dynamic monitor framerate
  - Transfer image when finished

Gamma

- Monitors do not emit 50% light intensity for a 50% light value
- Work according to a gamma function

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

- Monitor color space is not ideal for lighting calculations



Sound Waves

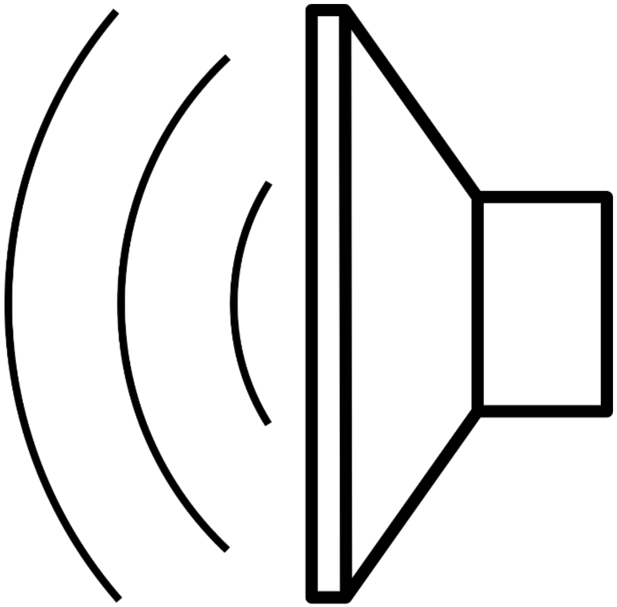
- **Air compression**
- **Longitudinal Waves**
- **~343 m/s**

Sound sensors

- Also two units
- Infer direction by measuring time differences
- Measures actual wave forms

Loudspeakers

- Construct actual sound waves
- Physically accurate reproduction of original waves



Computer -> Speaker

- **Small ring buffer**
- **Discretely sampled waveform**
- **Pointer to last sample written**
- **Pointer to next sample to read**

Sound Mixing

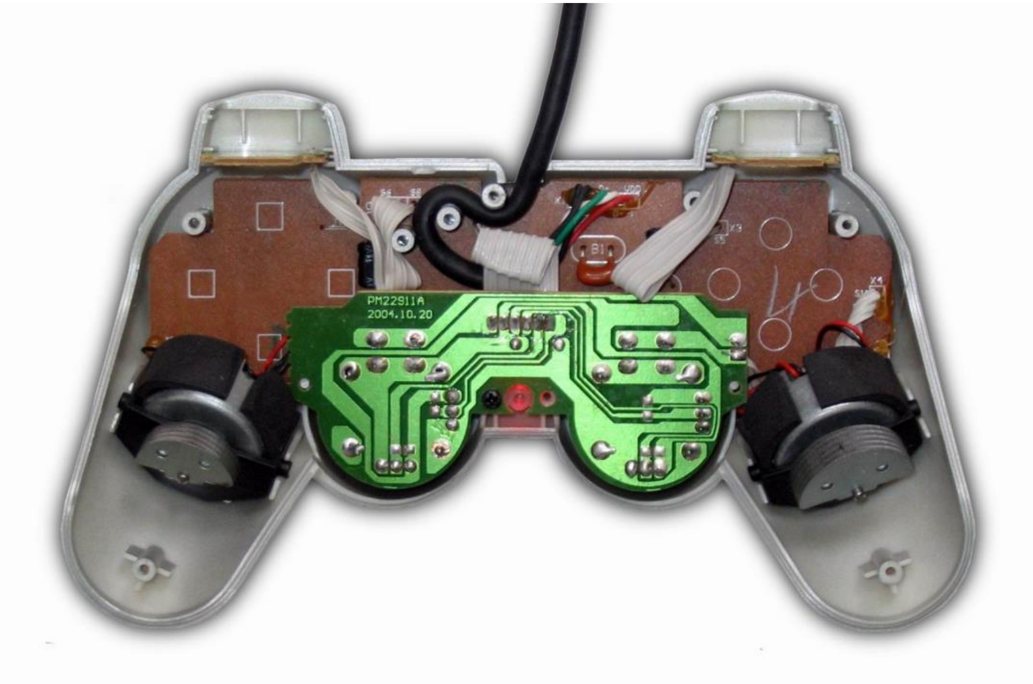
- **Superpositioning**
  - Adding waves

- **Again physically accurate**

- **Actual danger of superposition effects**
  - Avoid mixing identical sounds

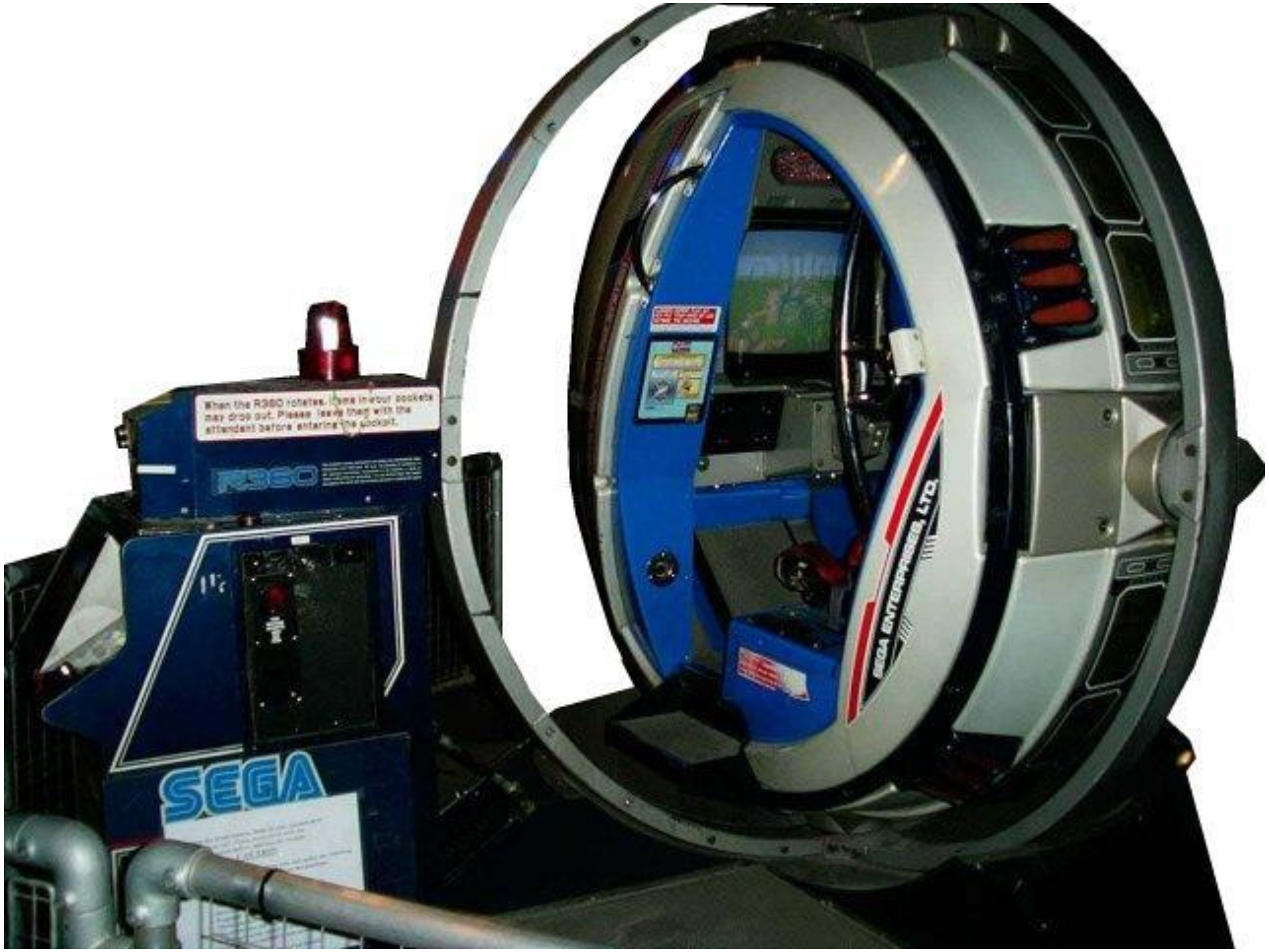
Rumble / Force Feedback

- **Very restricted „touch“ output**



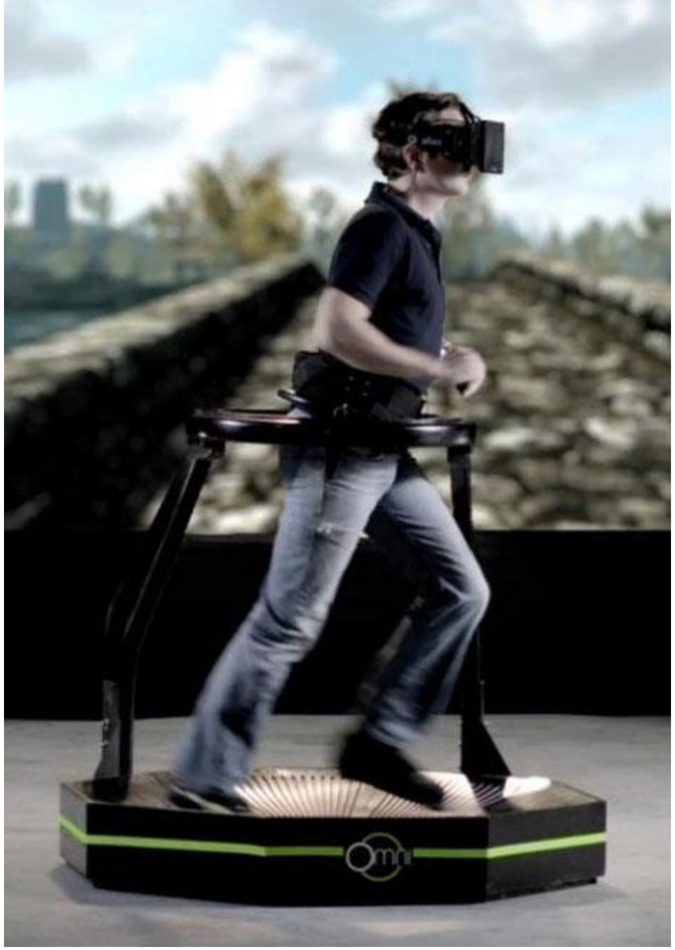
Acceleration output

- **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/C++

- **Open standards, not bound to a company**

- **Available almost anywhere**
  - Even in the browser (Emscripten)

C

- Portable assembler
- Developed for/with UNIX
- From 1969

C++

- **Adds higher level concepts to C**
- **No performance regressions**
- **Originally „C with classes“**
- **From 1979**

Classes in C++

```
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

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

Foo.cpp

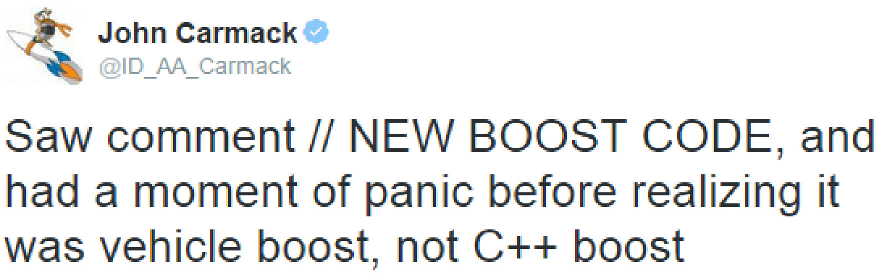
```
#include “Foo.h”

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**