Course Information

- **Mailing List**
  aine.ati.arvutigraafika@lists.ut.ee
- **Learning Environment (CGLearn)**
  [https://cglearn.codelight.eu/](https://cglearn.codelight.eu/)
Instructors

- Raimond Tunnel – jee7@ut.ee
  Lectures, Basic I, Basic II, JavaScript, C++

- Jaanus Jaggo – jjaggo@ut.ee
  Blending, Procedural Generation, Noises, Game Engines, Blender

- Ats Kurvet – akurvet@ut.ee
  Game Engines, Unreal Engine 4
Organization

- 6 credit course
  - **Lectures**: $15 \cdot 1.5h = 22.5h = \sim 0.9$ credits
  - **Practice Sessions**: $16 \cdot 1.5h = 24h = \sim 0.9$ credits
  - **Homework**: $2.3$ credits $= \sim 60h = 15 \cdot 4h$
  - **Project**: $40h = 1.5$ credits
  - **Exam**: $2h + 8h = \sim 0.4$ credits
Organization

• Grade
  
  • **Homework:** 40%
    
    Solve the tasks in CGLearn
  
  • **Project:** 30%
    
    Do something fun in a 2-3 person team
  
  • **Exam:** 30%
    
    General knowledge from the lectures and CGLearn
Organization

- Grade
  - **Homework:** 40%
    Solve the tasks in CGLearn
  - **Project:** 30%
    Do something fun in a 2-3 person team
  - **Exam:** 30%
    General knowledge from the lectures and CGLearn
- **Bonus Points** – *eg* APT GG organized game jams
Organization

- **Lectures** – Tuesday, 16:15, r402

- **Practice Sessions**
  - Basic I (JS), Basic II – Thursday, 12:15, r003
  - Basic I (C++), Game Engines – Thursday, 14:15, r003

- **Project Presentation** – In the exam session. TBA.

- **Exam** – In the exam session. TBA
Project

• Deadlines and instructions are here:
  https://courses.cs.ut.ee/2018/cg/fall/Main/Projects

• Can be any graphical application, prototype or a research project (e.g. replicating results from some paper).

• Extra ideas & time in the Computer Graphics Seminar (MTAT.03.305) and Computer Graphics Project (MTAT.03.328) courses.

Modules

- **Basic I**
  Geometry, transformations, projection, lighting, texturing and blending.

- **Basic II**
  Environment mapping, curves, procedural generation, ray tracing, global illumination, shadows.

- **Game Engines**
  Unreal Engine 4 and Blender
You should finish two of the modules to the end
You can, of course, attend the other group and do their tasks for extra points (earning ~60% total).
Homework

- Each week we will cover topics in the practice sessions.

- **Deadlines**
  - Basic I – 28.10.2018
  - Basic II, Game Engines – Week before the exam

- It is recommended, however, to do the homework **weekly**, because:
  - You won't fall behind
  - You will get the points sooner
Homework

• We start doing the tasks together in the practice.

• You can probably get more up to speed with the tasks when attending the practice session.

• If you miss the practice, then there are instructional videos for the Basic I and Basic II tasks.
Material

- **CGLearn** – [https://cglearn.codelight.eu](https://cglearn.codelight.eu)
  - Material with interactive examples
  - Tasks (homework)

- In the lectures we will cover similar topics that are in CGLearn (but not 100% overlapping)

- In the practice sessions we will discuss the concepts in tasks and start implementing them.

CGLearn is my Master thesis work. There might be bugs, let me know immediately if you find some. :)
Material

- **Fundamentals of Computer Graphics**
  P. Shirley, M. Ashikhmin, S. Marschner, 2009

- Provides a good and systematic approach to many topics we cover.

- Explaines the math behind the topics.

- Who is interested in borrowing it?
Lectures

Hey you
Don't be so sad, I'm here for you
Lectures

1) Me talking about the topics, 
   Me answering questions (I do not ask you)

2) Me talking about the topics 
   Class answering the questions (I ask the class)

3) Me talking about the topics 
   You answering the questions (I ask people)

4) You talking about the topics (In small groups) 
   Each time groups prepare presentations
Computer Graphics

- So, what is computer graphics? Applications?
Computer Graphics

- Games and entertainment

Deus Ex, 2000

Deus Ex: Human Revolution, 2011
Computer Graphics

- Games and entertainment

Deus Ex: Mankind Divided, 2016

Cyberpunk 2077, 2019?
Computer Graphics

- Graphical user interfaces (GUI)

Microsoft Word

WinDirStat

Photoshop
Computer Graphics

- Computer Aided Design (CAD)

SolidEdge

AutoCAD
Computer Graphics

- Scientific simulation visualization

HIV particle

Parker Solar Probe
Computer Graphics

• Data visualization

World ocean currents, NASA
http://deepbluehome.blogspot.com/2012/03/cool-cat-currents.html

World's Biggest Airlines
http://spatial.ly/2012/06/mapping-worlds-biggest-airlines/
Computer Graphics

- Simulations

Road Planer Simulator (Maanteehöövli simulaator)
Estonian Road Museum (Eesti Maanteemuuseum)
http://muuseum.mnt.ee/

http://en.wikipedia.org/wiki/Flight_simulator
Computer Graphics

• Art

*Blithe Certainty* by Kerry Mitchell

See also: http://flam3.com/flame.pdf
And: https://screen.aptgg.ee/

Buddhabrot
https://en.wikipedia.org/wiki/Buddhabrot
Technologies

• What to use to make computer graphics?
• 2D graphics? 3D graphics?
• Any previous experience?
Technologies

- We will start with lower-level technologies and libraries like:
  - C++ – Allegro, OpenGL, GLFW, GLM
  - JavaScript – Canvas, WebGL, Three.js

- Then you have a choice to continue with those or proceed with high-level modeling software and game engines:
  - Blender, Unreal Engine 4
2D Technologies

- `<canvas>`
- Language
- HTML Element
- Browser

Practice Session Group 1

- Language
- Library

Practice Session Group 2

- C/C++
- Allegro
3D Technologies

- Three.js
  - Language
  - Library
  - API

- WebGL
  - Language
  - Library
  - API

- C/C++
  - GLFW, glm
  - Language
  - Library
  - API

Practice Session Group 1

Practice Session Group 2
OpenGL, WebGL

- Those are API-s for communicating with the GPU drivers.
- WebGL is based on OpenGL ES 2 (Embedded Systems).
- They come with GLSL (Shader Language) which is used to program code executed on the GPU. It is similar to the C language.
OpenGL, WebGL

OpenGL 3D API Family Tree

Fixed function 3D Pipeline
Programmable vertex and fragment shaders
ES3 is backward compatible so new features can be added incrementally

OpenGL ES 1.0 → OpenGL ES 1.1 → OpenGL ES 2.0 → OpenGL ES 3.0 → OpenGL ES 3.1

OpenGL 1.3 → OpenGL 1.5 → OpenGL 2.0 → OpenGL 2.1 → OpenGL 3.0 → OpenGL 3.1 → OpenGL 3.2 → OpenGL 4.1 → OpenGL 4.2 → OpenGL 4.3 → OpenGL 4.4

WebGL 1.0 → WebGL 2.0

Open GL 4.4 is a superset of DX11

Standard Graphics Pipeline

Construct geometry
Define transformations
Assign material properties

Vertex Transformations

Culling & Clipping
Determine front-facing triangles
Determine which vertices are visible

Vertex Shader
Object's local space → viewport space

Rasterization
Fill the triangle with fragments

Fragment Shading
Calculate correct color values

Visibility Tests
Is the fragment visible?

Blending
Blend together multiple fragments
Standard Graphics Pipeline

1. Construct geometry
   - Define transformations
   - Assign material properties

2. Vertex Transformations
   - Object's local space → viewport space

3. Culling & Clipping
   - Determine front-facing triangles
   - Determine which vertices are visible

4. Rasterization
   - Fill the triangle with fragments

5. Fragment Shading
   - Calculate correct color values

6. Visibility Tests Blending
   - Is the fragment visible?
   - Blend together multiple fragments
Geometry

- Everything starts with the geometry
- Geometry defines the 3D objects in the scene
Geometry

- Although a cube has only 6 faces, we define it as a set of 12 triangles (12 faces).
- Why triangles?
Geometry

- Triangle very useful scientific facts:
  - Triangle vertices always form a plane
  - Every polygon can be converted to triangles
  - Triangles are easy to rasterize
  - Triangles are convex and simple
  - Three vertices always form a convex and simple triangle

This is a scientific term.
Geometry

- **Convex polygon** (*kumer hulknurk*) —

  1) All convex combinations of the vertices are inside the polygon
  2) Straight paths to all vertices from any point inside the polygon, are also inside the polygon
  3) The polygon is equal to its convex hull
  4) All interior angles are $\leq 180^\circ$
Geometry

- **Simple polygon** (*lihtne hulknurk*) — no intersecting edges

- What is the difference between those triangles?
Geometry

- Coordinate system can be left- or right-handed.
- We usually use the right-handed system.
Geometry

- In a right-handed coordinate system, the positive direction of an angle is counter-clockwise.

This means that the order of polygon vertices will define a front face and a back face.
What did you learn today?

What more would you like to know?

Next time: Geometry
(points, vectors, coordinate systems, ...)