DM842 Computer Game Programming

Rolf Fagerberg and Marco Chiarandini

Fall 2017

Why Computer Game Programming?

- Fun, attraction, curiosity
- Career goal
- Great display of use of many Computer Science subjects and courses:
  - Programming
  - Algorithms and data structures
  - Linear algebra and other math
  - Numerical analysis
  - Finite Automatons
  - Computer architecture

Computer Game Development

- Large game company (100 persons):
  - Game programmers: 30–40
  - Game artists, model designers: 30–40
  - Game level designers, testers: 10–30
  - Game designers: 2
  - Game producers: 4
  - Business and management persons: 5–10

Computer Game Development

- Large game company (100 persons):
  - Game programmers: 30–40
  - Game artists, model designers: 30–40
  - Game level designers, testers: 10–30
  - Game designers: 2
  - Game producers: 4
  - Business and management persons: 5–10
- Casual game company (1–3 persons):
  - Each person has many roles.

Computer Game Development

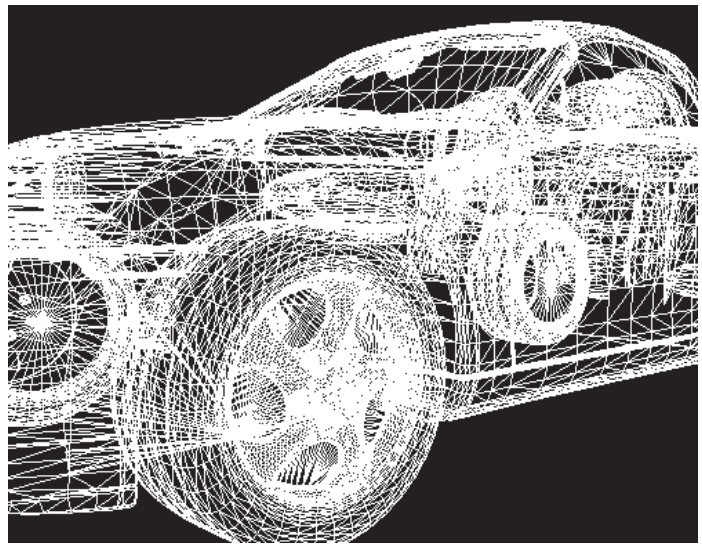
- Large game company (100 persons):
  - Game programmers: 30–40
  - Game artists, model designers: 30–40
  - Game level designers, testers: 10–30
  - Game designers: 2
  - Game producers: 4
  - Business and management persons: 5–10
- Casual game company (1–3 persons):
  - Each person has many roles.

Computer games in Computer Science: the study of

> Methods and principles of game programming

Subjects covered in DM842

1. The graphics pipeline:
   - 3D geometry (transformation, projection)
   - Shading (color, textures, lights, lighting models)
   - Image based techniques (skyboxes, billboards,. . .)
   - Polygonal techniques (culling, level of detail)
2. Game AI (path finding, chasing and evading, fighting, flocking, decision making, game trees,. . .)
3. Collision detection
4. Rigid body physics simulation



Subjects NOT covered

- Graphics APIs (self-study, is included in textbook)
- Software engineering, testing
- Game engines
- Level editors, scripting
- Modeling
- Artwork
- Animation
- Sound, music
- Gameplay, narrative, study of genres

Subjects NOT covered

- Graphics APIs (self-study, is included in textbook)
- Software engineering, testing
- Game engines $\leftarrow$ NOT used in course
- Level editors, scripting
- Modeling
- Artwork
- Animation
- Sound, music
- Gameplay, narrative, study of genres

Formal Course Description

Prerequisites:	Algorithms and data structures (DM507), programming proficiency, knowledge of vectors and matrices
Literature:	Textbooks
Evaluation:	Implementation projects (pass/fail), oral exam (7-scale)
Credits:	10 ECTS
Course language:	Danish and/or English

Project

Small project (in groups of 2–3) must be passed to attend the oral exam:

> Implement a 3D visualization of a (very) simple game, including some AI and physics simulation.

Programming language and graphics API of own choice. Must run on either Imada machines (Linux), or a recent Windows.

Some suggestions for API and language:

- C++ and OpenGL
- Java and OpenGL-binding (e.g. JOGL)
- C++/C# and DirectX

Project

Small project (in groups of 2–3) must be passed to attend the oral exam:

> Implement a 3D visualization of a (very) simple game, including some AI and physics simulation.

Programming language and graphics API of own choice. Must run on either Imada machines (Linux), or a recent Windows.

Some suggestions for API and language:

- C++ and OpenGL ← choice aligned with textbook
- Java and OpenGL-binding (e.g. JOGL)
- C++/C# and DirectX

Disclaimer

- Includes reading quite a number of pages
- Includes actual (but fairly simple) math
- Includes programming
- Includes work on issues not taught explicitly in course (graphics APIs)

Rather heavy - but fun - workload.

Textbook for Graphics Part

Computer Graphics Through OpenGL, 2nd edition. Sumanta Guha, Chapman and Hall/CRC. 2014.

- University level.
- Right coverage of subjects.
- Integrates theory and OpenGL.
- Lots of figures, code, examples, exercises.
- Instructions for installing OpenGL on Ubuntu, Mac, and Win on its website.
- Uses the "legacy" features of OpenGL, for pedagogical reasons. Coverage of the OpenGL 4.3 (shaders) in the last chapters (20 and 21).

Suggestion: run OpenGL in "compatibility mode" (see book and website for instructions) for first part of th course, change to shader based in later parts of the course.

Other Resources

- The OpenGL site at `www.khronos.org` (in particular their Wiki).
- The OpenGL Programming Guide (the "Red Book") for a clean explanation of modern shader-based OpenGl.
- Lots on the web.
- Lots of other books.

For the last two points: Beware which of the legacy/deprecated vs. modern (shaders) OpenGL main styles is addressed.