TIEA311
Tietokonegrafiikan perusteet
kevät 2018

(“Principles of Computer Graphics” – Spring 2018)

TIEA311 Tietokonegrafiikan perusteet – kevät 2018 (“Principles of Computer Graphics” – Spring 2018)

Adapted from: *Wojciech Matusik*, and *Frédo Durand*: 6.837 Computer Graphics. Fall 2012. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu/.

Frontpage of the local course version, held during Spring 2018 at the Faculty of Information technology, University of Jyväskylä:
`http://users.jyu.fi/~nieminen/tgp18/`

Some words about:

- programming (philosophical)
- learning programming (philosophical)

With some references to get us thinking

Postponed for a later time (likely next lecture):

- and “what on earth actually happened (or, was supposed to happen) in Assignment 0?” (philosophical and technical)
- C++ (technical)
- OpenGL (technical)
- GLUT (technical)

Abstruse Goose: “How to teach yourself C++ in 21 days” (`http://abstrusegoose.com/249`)

The above webcomic contains science fiction elements, “nerd humor”, and sarcasm – but also bits of truth to think about:

- Regardless of language, programming starts with learning basic structures and program flow.
- Learning takes time
- Learning needs **interaction** and **motivation**
- **Ten years** of **actively** doing something (ca. 3648 days) is likely to make you a professional in any skill – like programming (or theoretical physics, or genetics, or playing an instrument, for that matter).
- 21 days . . . well . . . is unlikely.

What is learning?

As of Spring 2017 (and also 2018) I'm thinking about this quite a lot, since we're discussing this formally in the "YPE" studies (university pedagogical studies)

My thoughts, and implications:

- Learning takes time – **use the 135 hours** allotted for any 5 ECTS credit points
- Learning needs **interaction** – use IRC, email, computer classes, face-to-face fellow students, friends
- Learning needs **motivation** – I can only (try to) spark and maintain yours, but the rest is all personal; if you don't **want** to learn something, a bigger question should arise: do you want a degree in IT; do you want to be an IT pro?
- Learning **is** painful – remember: "no pain, no gain", if you **don't** yet feel any pain, **raise the bar**! The Assignments of this course have all the potential to facilitate pain.

What is learning?

More thoughts:

- Learning comes from getting answers to questions – first thing is to **formulate the next question** – what do I know, what do I need to know **next**. Not so simple, though!!
- Sometimes just **thinking about the question** gives you the answer – discovery of an application!
- Usually it leads to **reading more information** – so simple these days; we have the WWW!
- At times, you just need to ask others.
- However, I doubt any deeper learning is possible if you don't **ask yourself first**, which leads to formulating the question.

What is it to learn programming?

More thoughts:

- Not sure about today, but back in the 1990's the school system did not really support active thinking and asking questions. "Book knowledge" is just not enough for programming (or scientific work, or other fundamentally creative activities). Fortunately, we are in **university** now!
- "Book knowledge" is definitely **required**, too – for example, we need tutorials to learn how things work, we need specifications and reference manuals to check things out.
- But none of this knowledge will **create** a program
- The real skill is to **locate knowledge** and to **apply** it to the task at hand.
- Impossible to remember how all the 10000 nuts and bolts work – the **target skill** is to **re-discover**, again and again. It gets easier every time. Things look similar (in C, C++, C#, Java, Javascript, Python, . . .)

Requirements in real-world programming jobs

- Standard pre-assignment in a job recruitement: You are given a new platform (something you might have never seen) and a task to create some “simple” application (something you might have never done before) in one week.
- You need to be able to do this – they want you to **be able to learn new things** (and of course, they’re likely to measure also the “quality” of your solution)
- Then, you might get to the actual interview – in which they’d probably like to know how you communicate with peers – for example, if you are able to **ask for help** and **help others** in a team.

Role of TIEA311 . . . **a step towards** the **real skills**

- New tools (language, libraries)
- New application
- Less “holding your hand” – a push towards new ways of thinking and acting (compared to what earlier levels of school may have been like)
- Towards being a pro!

Programming 1 and 2 can only get you so far. Soon after, there must be a shift towards the skills to survive in real world.

The Assignments of this course facilitate learning real-world skills. They give you some basic knowledge, but also “more or less vague clues” for finding out more from outside sources :).

Ok, we discussed a bit about:

- programming (philosophical)
- learning programming (philosophical)

Then onwards:

- C++ (technical)
- OpenGL (technical)
- GLUT (technical)
- “what on earth actually happens (or, is supposed to happen) in Assignment 0?” (philosophical and technical)

Abstruse Goose: “How to teach yourself C++ in 21 days” (`http://abstrusegoose.com/249`)

C++

- Old
- Alive and well
- Evolving
- Backwards-compatible “down to C”
- Cross-platform (through native compilation!)
- Multi-paradigm (structural, object oriented, “template metaprogramming”, some coarse “functional” approaches possible)
- Complex as hell. . .

OpenGL – “glNameOfSomeCall()”

- One interface for graphics (alternatives: Windows-only DirectX, upcoming/very new cross-platform Vulkan)
- Implemented as a library with specified function calls
- Cross-platform
- Widely used
- State machine!

OpenGL Utility Library – “gluNameOfSomeCall()”

- Some convenience functions on top of plain OpenǴL
- This is usually available along with OpenGL

GLUT – “glutNameOfSomeCall()”

- user interface library (Window + minimal keyboard and mouse controls)
- Designed to make it easy to learn OpenGL (UI library is as simple as possible)
- Not to be used for “production” (production-grade cross-platform alternatives: SDL2, Qt, fltk, etc. . .)
- Perfect for its intended job
- Event-driven, based on registering callbacks

Let us try to “dissect” Assignment 0 somewhat together, on-screen . . .

(in 2018, we started this on lecture 2, and we’ll continue the dive on lecture 3)