MTAT.03.015 Computer Graphics (Fall 2013) Exercise session I: Programming a 2D graphics system.

Konstantin Tretyakov, Ilya Kuzovkin

September 09, 2013

The aim of this exercise session is to get a first feel for how graphical applications are made. The first part of the session is devoted to a brief overview of the C language, which might be helpful to those who have not used (or have forgotten how to use) it before. If you feel comfortable with C you may safely skip it. In the second part we shall do some of the actual Allegro programming (and that is the part that gives you credit points).

The code base for the exercise session is provided in the file `practice01.zip`, available on the course website[1]. You should download and unpack this archive to some writable directory (e.g. the desktop) before proceeding. All the solutions should be implemented in code within the file `helloallegro.cpp`, located in the `src` directory. The homework is submitted by uploading the zip archive of the project directory via the homework submission interface accessible at `https://courses.cs.ut.ee/2013/cg/fall/Main/Practices`.

Make sure that your code compiles and runs fine. Submissions that do not compile may get zero points.

1 C Programming Primer

We shall be programming in C during this course[2]. Note that there is nothing special about C as far as graphics programming goes. Indeed, most mainstream graphics libraries have bindings for many languages, so you could, for example, use the Allegro library in a Python, Lisp or C# project. The correspondence is straightforward: whenever you use the function `al_init` in C you would have to use `allegro.al_init` in Python or `(al_init)` in LISP, etc. None the less, C, being a low-level compiled language, often offers performance benefits which are important for visually rich applications, hence it is still among the main tools of computer graphics practitioners. You simply would not be able to claim that you know computer graphics if you haven't had experience with it in C.

[1]Alternatively, all lecture slides and practice session materials are also available at Github: `http://github.org/konstantint/ComputerGraphics2013`

[2]... with a tiny sprinkle of C++ in a couple of places where it makes things easier.

Compilation and linking. C is a *compiled* language. You write the C program in one or more text files with the extension *.c* or *.cpp*. This text can then be translated into binary code (a file with extension *.o*) using a compiler. Typically compilation is performed by invoking something like

```
        $ gcc -c program.cpp -o program.o
```

One or more *.o* files are then *linked* together into the final executable (adding all the necessary external libraries along the way) using the *linker*. The linker is typically invoked as follows (note the missing `-c` option):

```
        $ gcc program.o -l somelibrary -o final_executable
```

IDE. The task of having to invoke the compilation routine is simplified by using a smart editor (an *Integrated Development Environment, IDE*). We shall be using the CodeBlocks IDE. To get a feeling of it, please, download and unpack the practice codebase, then open the file `0_HelloWorld.cbp` in CodeBlocks. Note by browsing the *Project tree* on the left pane, that the project contains a single file, `helloworld.cpp`. Build the project by pressing the *Build* button on the toolbar. Observe the output in the *Build log* on the lower pane – you should see how the compiler and linker are invoked. Then press the *Run* button to launch the resulting executable.

Build options. You must have noted that the compiler and linker invocations have some additional options. Indeed, there are many possibilities to fine-tune the way compilation and linking is performed using *Build options*. You should not care much about them so far, but in case you are interested, try looking at the *Project/Build options* and *Project/Properties* menus. Those are the two dialogs where everything about the project is specified.

The C language. Study the code in `helloworld.cpp`. This should give you an idea about the things you will need to know about C. Try playing with the code, changing things and looking at the result. If you feel completely new to the language, check out the “How C programming works” tutorial, referenced from the practice session page on the course site. Also, the “C reference card” can provide a two-page overview of more C than you’ll ever need in this course.

2 Allegro. Programming a 2D Graphics System

Allegro is a popular cross-platform library for (mainly) 2D graphics and game programming[3]. This is in no way the only such library out there, nor is it the best for all purposes[4], but it is good enough to understand the general principles.

Start by opening the file `1_HelloAllegro.cbp` in CodeBlocks, building and running the project.

Exercise 1 (1pt). Study the code and introduce a couple some simple changes:

1. On the first scene add a square of your favourite color with an inscribed filled circle.
2. On the second scene draw a rotated version of the cat using

   ```
   al_draw_rotated_bitmap()
   ```

3. On the third scene, change the code so that the kitten could not leave the screen.

Exercise 2 (1pt). Add another scene to the program, that draws an image using the following algorithm (in pseudocode):

```
    points  = { (0, 0), (320, 640), (400, 0) }
    curpoint = points[0]
    for i = 1..4000:
           put_pixel(curpoint, RED)
           r = random integer from {0, 1, 2}
           curpoint = (points[r] + curpoint)/2
    end
```

Hint. You can generate random numbers using

```
    int r = rand() % 3;
```

Exercise 3 (1-2pt). Modify the third scene into a basic "platformer" game, namely:

1. Add a background picture. Make one of your own or find from the internet.
2. The main character (can be a person, a kitten, etc, whatever comes to your mind) should be able to move left, right *and jump* (using the arrow keys).

[3]The latest version, Allegro 5, actually provides proper access to hardware-accelerated OpenGL for 3D graphics.

[4]SDL, Cocos2D-x and SFML are among the closest alternatives, and there are literally hundreds of more distant relatives in C and other languages.

3. The character cannot leave the bounds of the screen or fall below the ground.

The most simplistic implementation of this specification gives one point. Any displays of creativity (e.g. animated character, scrolling background, obstacles or platforms, sound effects, etc) are awarded by another point.