T-637-GEDE Game Engine Architecture

Problem Set 1

Problem 1 – Engine Design (25%)

Imagine that you are developing a game engine that you want to use with your own game but also license to other developers. At some point you have to implement an **in-game inventory system**, where the player can bring up a decorative window that contains a visual representation of all held objects, which he can then equip/use, combine, drop or examine. How much of this inventory system would you implement as part of your game engine and how much would be game specific? How does this depend on what genre of games your engine should support? What other game engine components would you be relying on? What possible platform specific things might you have to consider?

Problem 2 – Ogre Project Setup in VS (25%)

Follow the instructions on the wiki page for creating your own Ogre application from the Ogre source code (http://cadia.ru.is/wiki/public:t-gede-12-1:buildingogrefromscratch). There are a few things you can improve about this project setup:

1. MAKE YOUR PATHS SYSTEM CONFIGURABLE: Notice that in the build configuration given above, you are using fully specified paths to the various directories. Change this so that you can define system wide environment variables that hold the locations of the Ogre source directory and the Ogre built directory, and then use those environment variables in your build configurations. E.g. one could define the variable $OGRE_SOURCE (at the system/windows level) and give it the value "C:\Development\ogre_src_v1-7-3" and therefore not have to mess with any of the build configurations when moving the project to a different machine, only change the value of $OGRE_SOURCE and $OGRE_BUILT.
2. PREPARE A RELEASE BUILD: The build configuration given above is only for a DEBUG build. Create a working RELEASE build configuration as well. Notice that some of the configuration is exactly the same for both targets – Make use of the "All Configurations" configuration for everything that remains the same for both configurations. The main difference between them for this application is that libraries that are compiled with debug information typically have "_d" appended to their names. Clearly those versions of the libraries should not be used in a release build. Remember to copy the release versions of the Ogre configuration files into your application directory (`ogre.cfg`, `resources.cfg` and `plugins.cfg`).
3. COMPILE WITH A DIFFERENT MODEL FOR RELEASE AND DEBUG: Tell the preprocessor to pick a different model to load in the "CreateScene" function, depending on whether you are building for DEBUG or RELEASE. Use the "ogrehead.mesh" for RELEASE and "cube.mesh" for DEBUG.

Submit a zipped copy of your application directory (that includes your solution, project, configuration and source files).

Problem 3 – Representing Numbers (25%)

Show how you find the decimal value of the 32-bits 0x8000000F

1. ...as an unsigned integer
2. ...as a signed two's complement integer
3. ...as a IEEE-754 floating point number (may need more than the text book)

Problem 4 – Memory Layout (25%)

Consider the following C program. Draw **two** diagrams: One that represents a possible arrangement of the **executable image on disk** and one that represents the **possible program memory layout** right after the line `value = value + answer;` has been executed. Use names of variables and functions to identify memory locations they refer to.

main.c

```
#include "calculate.h"
void go() {
  calculate_answer();
  printf("%d\n", answer);
}
int main() {
  char name[] = "answer";
  s = calloc(10, sizeof(char));
  strncpy(s,name,6);
  go();
  free(s);
  return 0;
}
```

calculate.h

```
#include <malloc.h>
#include <string.h>
#include <stdio.h>

int answer;
char* s;
void calculate_answer();
```

calculate.c

```
#include "calculate.h"
static int b = 10;
static int c;

static void print_answer(int answer) {
  static value = 0;
  value = value + answer;
  printf("%s: %d\n",s,value);
}

void calculate_answer() {
  int a = 3;
  c = a*b;
  answer = 20;
  print_answer(answer+c);
}
```