T-637-GEDE Game Engine Architecture
Problem Set 2- Due Tuesday February 4th, 2014

Problem 1 – Design and Human Interface Devices (25%)

Human interface devices (HIDs) bring game players into the heat of action; they are therefore an incredibly important piece in the whole game experience. But how do you know whether a certain HID is good for a certain game? In an attempt to understand the connection between game engine design and the wide variety of HIDs available, discuss the following two questions:

a) If you are designing a game engine for a certain genre (you can pick one you like, e.g. RPG, Sports, RTS, etc.), what range of HIDs should you support? Do certain HIDs fit that type of game better than others? Why or why not?
b) If the game world and game play is complex, how can the game engine help developers make use of a very simple HID (think of one of the simplest possible HIDs: one button)? Would this work for every game genre?

Problem 2 – Reference Counting (25%)

Imagine that your resource manager is about to display a new level in a game and it needs to determine what resources to unload and what resources to load. To do this it uses reference-counting. The current level is using resources ID3, ID6, ID7, ID9, ID11 and the new level will use resources ID5, ID6, ID7, ID10, ID12. Fill out a resource usage table like Table 6.2 in the textbook to demonstrate how the reference-counting occurs and what gets unloaded and loaded in the process of changing the levels.

Problem 3 – Hashing Game Object Names (25%)

Consider a game engine where all game objects are identified at design time using a short string, maximally 10 characters in length. We want to avoid as much as possible to use string comparisons in our code to identify objects and therefore decide to hash each string into a unique 32 or 64 bit integer (your choice) for faster comparisons.

a) Create OR find two different hash functions and calculate the hash value for the IDs "BUILDING1", "BUILDING2","P1SPAWN" and "M2SPAWN".
b) Now imagine you have a fixed hash table of 1024 entries where you wish to store pointers to the actual objects along with their string IDs. What indices in the table do the above IDs map onto? (If probing is necessary, explain what probing method you are using.)
c) What are the advantages and disadvantages of using each of the two has functions you used?

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 int 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);
}
```