CPE 101
Fall 2009
Laboratory 3 (Functions)

Due Date

- **By the end of your last scheduled lab of week.**
- **You must turn in your source electronically on vogon using the handin command – instructions are provided in below.**

Objectives

- To practice writing functions and their prototypes.
- To practice problem solving.
- To develop a complete C program, compile it, and turn it in electronically.

Resources

- This is an individual lab and should be completed individually
- You may use your instructor, peers, texts, and your own innate capabilities and resourcefulness!

Ground Rules

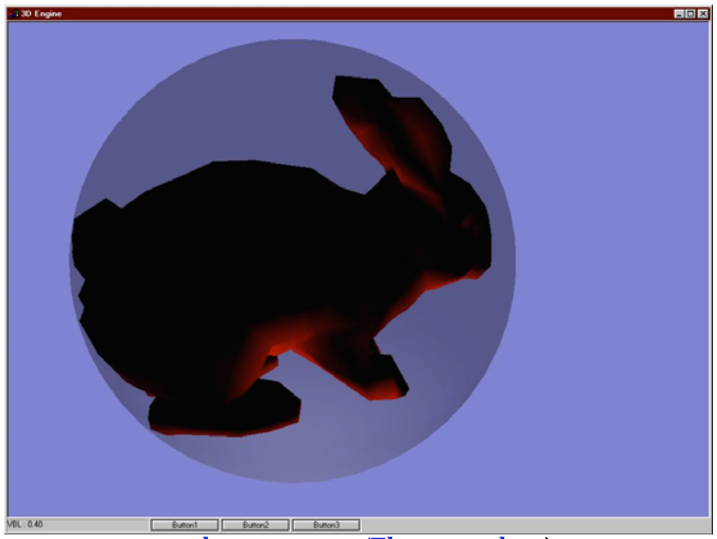
- None

Orientation…

This lab will involve developing a small C program to assist in collision detection. In 3D computer games (and other applications that involve a computational representation of 3 dimensional space), the game (or software) needs to be able to detect when two different 3D objects are close to one another._ Specifically, the software needs to detect when two objects may run into one another (for example, the software needs to know when the main character has run into another character, instead of letting the character walk through the other character)._ This process is called "collision detection"._ To accurately compute collision detection of complex 3D objects is a very complex calculation that is computationally expensive._ Therefore, many applications use an approximation for some aspects of the collision detection._ Your assignment today is to write a program which detects whether two 3D objects are colliding._ You will accomplish this by testing whether the "bounding spheres" around each object are colliding with one another._ A "bounding sphere" is a sphere which approximates the real object and which encapsulates that object (see the below image for an example of a bounding sphere around a bunny character)._ The bounding sphere has a center, an x, y, z coordinate, and a radius._ Two spheres are colliding if the distance between their two centers is less then or equal to the sum of their radii.

Your assignment is to develop a program which takes input from the user about two bounding spheres and tests if they are colliding or not._ Please develop the below specified functions._ Note that unlike your previous labs, you will be expected to develop the names, arguments and return values for all functions in this lab. There is example output (including prompts) below.

Example bounding sphere:



(image credit: **www.codercorner.com/Flexporter.htm**)

Part 1: Develop a 3D Euclidean Distance Function

Similar to what you will develop for your program 2._ Develop a 3D Euclidean Distance function._ If you are unsure what this equation is, feel free to reference Math World, an excellent online resource for math information._ Specifically, see: http://mathworld.wolfram.com/Distance.html_ Please test this function with coordinates and a distance value you can reason about (you may use the floating point comparison test from program 2)._

Part 2: Write a main function which prompts the user for bounding sphere information

Develop a main function in which you prompt the user to input the center and radius for a bounding sphere._ The sequence of prompts should appear twice such that your program will prompt the user for the location of two different objects._ We have been learning to use functions for code re-use._ Why, given what we currently know, can you not write a prompt function to gather the bounding sphere information, and just call that function twice in main?

Part 3:_ Develop a function to test for collision

Develop a function that uses the user’s input regarding the locations of the bounding spheres and the distance function you developed in order to determine if the two objects are colliding._ The function should return a boolean (1=true, 0=false) which indicates whether the two objects are colliding (specifically, return 1 if the objects are colliding and return 0 if the objects are not colliding)._ Please test this function with values you can reason about._ Similar to in Program 2, you will want to test this function to make sure it correctly evaluates both true and false collisions.

Part 4: Complete the program

Your main function should already prompt the user for the specifics of two different game objects . Now, finish the program by adding a printed report of whether the objects are colliding._ In other words, after you have prompted the user for the location of two bounding spheres, use that information to call your collision test to see if they are colliding and report their status._ Be sure to report a different message based on whether they are colliding or not._ ***Show your completed program*** *(including your code to test the distance and collision function)* ***to your instructor.***_ Two different example runs appears below (user input shown underlined):

```
This program will test if two objects are colliding
please enter the center of one object
Object 1 x: 1.2
Object 1 y: 5.7
Object 1 z: 6.3
Object 1 radius: 1.0
Please enter the center of the second object
Object 2 x: 4.2
Object 2 y: 5.7
Object 2 z: 2.3
Object 2 radius: 4.5
Objects are colliding!

This program will test if two objects are colliding
please enter the center of one object
Object 1 x: 0
Object 1 y: 0
Object 1 z: 0
Object 1 radius: 3.1
Please enter the center of the second object
Object 2 x: 5
Object 2 y: 0
Object 2 z: 0
Object 2 radius: 1.8
Objects are not colliding.
```

Part 5: Handing in Your Source Electronically…

1. Transfer your file (lab3.c) to vogon as you've done for previous labs
2. Log on to vogon using the Secure Shell Client program (or your favorite equivalent).
3. Change directory (cd-command) to the directory containing the source file or files to hand in.
4. Execute the following command:

```
handin zwood cac101lab03 lab3.c
```

5. You should see messages that indicate handin occurred without error. You can (and should) always verify what has been handed in by executing the following command:

```
handin zwood csc101lab03
```