Labirinti3D
Computer Graphics I (CSA2207) Assignment

Sandro Spina / Colin Vella

March 15th 2011

This is the assignment description of unit CSA2207, Computer Graphics I, for the year 2010/2011. Your work on this assignment may contribute up to 40% of the total mark for this unit and should be carried out individually. Under no circumstances should code be shared. Assignments should be submitted by hand on optical medium; each student will be asked to demonstrate the implementation submitted with the documentation. Please remember that plagiarism will not be tolerated; the final submission must be entirely your work.

Deliverables

You will submit your project source code, executables and a PDF file containing the project documentation on optical medium. Ideally binaries should be made to run straight from CD/DVD-ROM without the need for any installation unless specified in the accompanying document, in which case a detailed installation guide must also be provided. **DO NOT** submit with the assignment a printout of your source code.

Background / Description

For this assignment you are asked to create, render and navigate a 3D virtual world consisting of a set of labyrinths. Detail of how these labyrinths are described is provided further on. You are allowed to create (or use any that are freely available given this is not an arts course) 3D geometry to populate your world. All your geometry should be contained within a spatial data structure which is used to optimise the rendering of your world. You are allowed (or rather expected) to be creative and come up with different ideas in the design of this virtual world.

The application may be developed in C#, Java, C or C++.

Task Breakdown and Marking

The project is broken down into a number of tasks, each of which outlines a different aspect of the final deliverable. I am not attaching any marks to each component simply because each one of them can be dealt with at varying levels of difficulty. I will then determine the mark when reading the assignment report.

Task 1: Labyrinth Creation

Labyrinths may consist of multiple levels. Each level is described as a text file in the following format.

```
16, 8
3, 2
Level0
################
#.........#$..@#
#....####.####.#
#....#$.#......#
#..###..#.######
#...#...#.#@...#
#......@#...##$#
################
```

The text above would generate a 16 by 8 size labyrinth with a fixed maximum height. This height can be anything you decide. The camera (avatar) is initially placed at location 3, 2. More than 1 level may be defined using the Levelx delimiter. The following illustrates a small two level labyrinth.

```
16, 8
3, 2
Level0
################
#.........#...y#
#....####.####.#
#....#..#......#
#..###..#.######
#..X#...#.#....#
#.......#...##.#
################
Level1
################
#........Y#....#
#....####.####.#
#....#..#......#
#..###..#.######
#...#...#.#....#
#.......#..x##.#
################
```

In the example above the levels use the same floor layout. The pairs (X,x) and (Y,x) symbols in the different levels indicate one way portals i.e. moving the camera (user) on cell X (in Level 0) will automatically transport you to cell x in Level1. In order to move back to Level 0, the user has to move on tile Y in order to appear on tile x in Level0.

Labyrinths can be constructed from multiple levels and use as many portals as required.

Task2: The Avatar Camera

A basic camera class needs to be implemented here so that the user is able to travel inside the labyrinth.

Task2: Collision Detection

Clearly the walls of the labyrinth are an important component of the game because they constrain the user into travelling certain paths. In this part of the assignment you need to implement a basic collision detection algorithm. Remember that the walls are static, therefore you can use the maps in order to determine whether you can travel in a particular direction. There are other ways you can implement this and it’s up to you to decide.

Task3: Some Treasure and The Enemy!!!

In order to make your game play stand out a little bit more (and because almost all games need to follow some sort of objective) at this point you should introduce both treasure and enemies. Collecting treasure and avoiding enemies are both objectives.

```
################
#........y#$..@#
#....####.####.#
#....#$.#......#
#..###..#.######
#..X#...#.#@...#
#......@#...##$#
################
```

In the map above @ indicates a stationary enemy whereas $ indicates a stationary treasure chest. It would be really cool if you implement moving enemies, i.e. you place an enemy in a room and you implement a very basic algorithm whereby the enemy simply moves through the maze ala pacman ghosts.

Task4: Lighting your World

Illumination is an important aspect of your world. For your implementation you need to make use of shader programs which implement different types of illumination. Light sources would then need to be integrated into the virtual world in order to illuminate its different parts. Clearly the positioning of all these light sources in the world is very important. Again you can use a symbol such as * to indicate the position of point lights in you map file. Remember to use an ambient component in your implementation of the lighting equation so that every part of the labyrinth is visible.

```
################
#........y#$..@#
#.*..####.####.#
#....#$.#*....*#
#..###.*#.######
#.*X#...#.#@..*#
#......@#*..##$#
################
```

Task5: Assignment Write-up

This task is extremely important as you are expected to clearly describe how you architected and built your virtual world components, outlining any important design decisions taken. You are encouraged to use diagrams to explain the architecture where necessary; moreover, feel free to use pseudo-code to illustrate the algorithms implemented. If the deliverable has short-comings or bugs or even lacks some of the expected functionality, you must put this down in the write-up.

Final Notes

Remember that the first step in achieving a solid, stable and efficient deliverable is that of sitting down as a group and designing the system before anything else. Identify how everything is supposed to work at a high-level, and then start by singling out the various components of your virtual world. Once you identify the components, start categorising the entities and their relationships, how they interact and work within their sub-systems, and how the sub-systems communicate together within the system. Outline all this design, and make sure you are able to describe to yourself precisely what each component is responsible of and how it would be carrying out its duties. Make sure you do this before doing any programming as this will save you time discarding broken designs or trying to patch badly designed components to make them talk, and will let you concentrate on the actual problem at hand. Good luck, feel free to experiment and try to enjoy this assignment. The effort put into this assignment will be duly noted during marking.