6th International Conference on Applied Informatics
Eger, Hungary, January 27–31, 2004.

Development of a virtual reality navigation test

Cecília Sik Lányi, Ádám Tilinger, Tamás Umenhoffer

University of Veszprém H-8200 Veszprém,
Egyetem u. 10.
e-mail: lanyi@almos.vein.hu, tilinger@vision.vein.hu,
umitomi@axelero.hu

Abstract

The goal of our recent experiments at the University of Veszprém was to find the characteristics and differences of left and right handed people in motion and behaviour in virtual worlds. To do this we needed to create a program to display virtual worlds. In this paper we will describe the method we used to solve the most difficult problem we faced while creating this program. This is collision detection. We'll also speak about the experiments, and our results.

1. Introduction

Most people are right handed, only 10 percent are left handed. We can well observe the differences between left and right handed people's motion and in the use of utensils in our everyday life. Due to the countless experiments made in this topic, we can describe right- and left-handedness quite well. It is the consequence of the asymmetry of the human brain. The two hemispheres have different characteristics and are responsible for different functionalities [2]. In most people's case the left hemisphere is dominant, they are right handed. Up to now many utensils (scissors, can openers) and working tools (keyboard, mouse) are made especially for left-handers to undo the difficulties in their daily lives. With our experiments we would like to help the designers to create user-friendly virtual worlds both for left- and right-handers.

2. The Virtual Environment

To examine people's motion in virtual reality we need a program that displays this three dimensional world. This program has many other tasks (i.e.: treat sounds

and movement, store the movement).

To build our virtual worlds we used a well-known three-dimensional modelling package MAYA. The constructed world is exported to a text file with a script using MAYA's script language. This text file includes the objects to be displayed, full information about their structure (three-dimensional coordinates of vertexes and faces), the material properties and textures used to display the objects and the light sources appearing in our world. This text file will be loaded by our program, which will display this world using OpenGL instructions.

3. Collision Detection

The virtual world and the moving observer influence each other. The simplest case of this influence is that we shouldn't walk through the objects of the world. To avoid this we need collision detection. Among the several collision detection techniques our program uses a method which is called binary space partitioning (BSP) and binary space partitioning trees. BSP-trees are used in many areas of computer graphics including visibility orderings, culling, speeding up ray tracing and collision detection [1][2][4].

4. Binary Space Partitioning

To create a BSP-tree we divide the space into two subspaces and repeat this for the subspaces until each subspace has only one objects (or faces) in it. In the nodes of the tree we store the dividing planes and in the leaves we will have the partitioned objects. Using this technique we get a structure like on Figure 1.In our case we didn't need to partition according to three dimensions because the observer couldn't move up and down. So we could use a two dimensional form of the BSP trees. In this case we partition the two-dimensional space with lines and we will partition line segments. An important part of partitioning is the question that how to choose the dividing line. For collision detection the proper choice is to take one of the dividable line segments to define the dividing line. In this way the dividable line segments will be found in the nodes of the tree and each leaf will describe a subspace. These subspaces are disjoint subspaces of the two dimensional space (Figure 1).

This can be used in collision detection. If the partitioned object is solid than these subspaces have two types: the ones, which are inside the object, and the ones, which are outside. To detect collision we only have to examine the point where we stand and the point we are heading for and if they are on opposite sides than we detected a collision, and don't allow the movement. For every point we can easily find the subspace it is located in by walking through our tree and in each node examining that on which side of the dividing line does our point lie and continue examining - for that child only - ,until we reach a leaf.

With this method we get a fast and elegant way to solve the collision detection problem, which works on objects with various shapes.

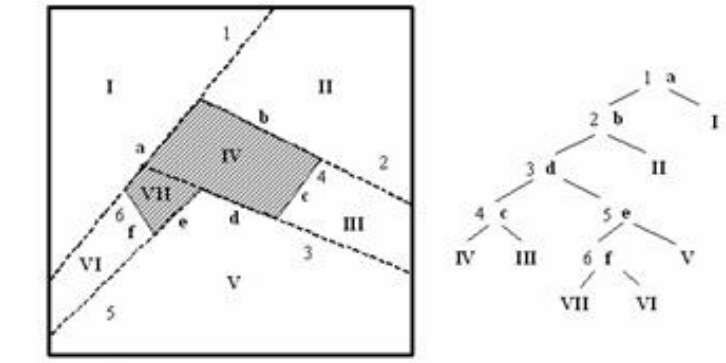

Figure 1: Partitioned object

5. Our Virtual World

The virtual world we built is considerably symmetric in structure, in which we placed paintings of Dutch landscape painters and a statue in the middle of the room (Figure 2). The user can move freely to examine the exhibited works of art while a relaxing classical music is played.

Figure 2 : Screenshots of the picture gallery

Pressing the ESCAPE key the main menu of the program appears. Here we can find the interface tools for recording and playing back the motion, for stopping the playback and record. The user's properties can also be set here. The properties of the user are: name (or ID), age, sex, dominant hand and additional information, which could be useful. We tried to make the user interface as handy as possible.

The program can import and process any kind of virtual world, which was properly built in Maya and was properly exported. In modelling our worlds we should consider the complexity of the objects (polygon count and texture size), our world should suit the PC's performance we are going to use to run the program. The worlds we made will run on today's average PC.

For collision detection we built a wall object in Maya (Figure 3). Viewing it from above gives us the line segments, which need to be partitioned with the method described above.

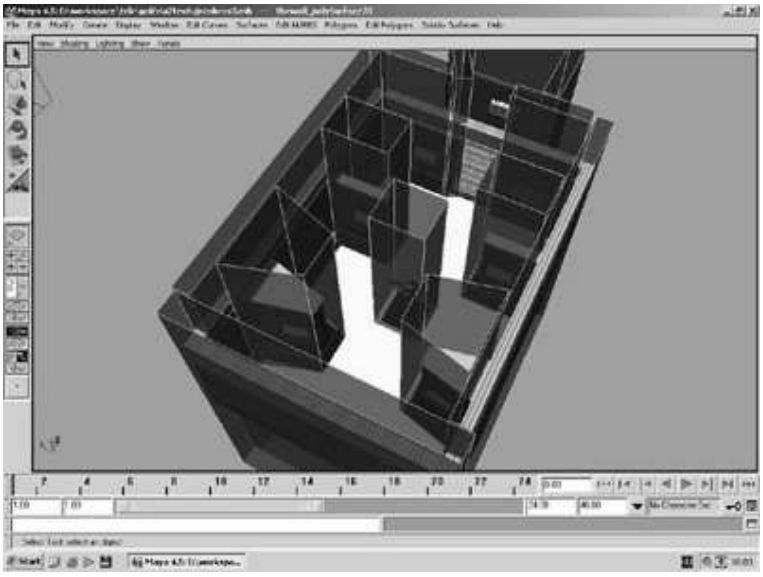

Figure 3: The world built in MAYA and the wall object

6. Examining the Movement

The program stores the viewer's position in every second to a text file. We analyse the data exported by the program in Microsoft Excel. We only need two dimensions from the stored coordinates, as we can't move up or down. These two dimensions can be represented in a point diagram. This way we can easily examine the path of the walk (Figure 4). We can examine more than one person's data on the same diagram to easily compare them.

7. Results

Up to now we examined twenty person's paths of walk, six of them were left-handed two of them were ambidextrous and the others were right-handed. We could well observe the differences between left- and right-handers. Most left-handers did a circle starting from the left while walking through the gallery. On the other hand right-handers usually don't do a circle, they go from corner to corner, sometimes turning right, sometimes turning left (Figure 5).

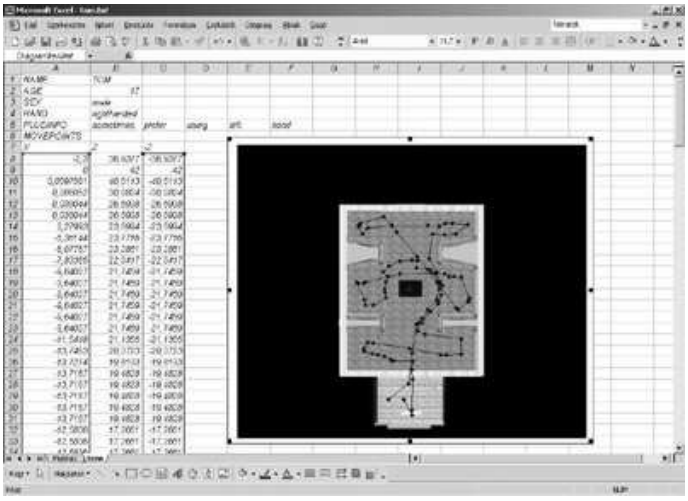

Figure 4: Examine the data in Excel

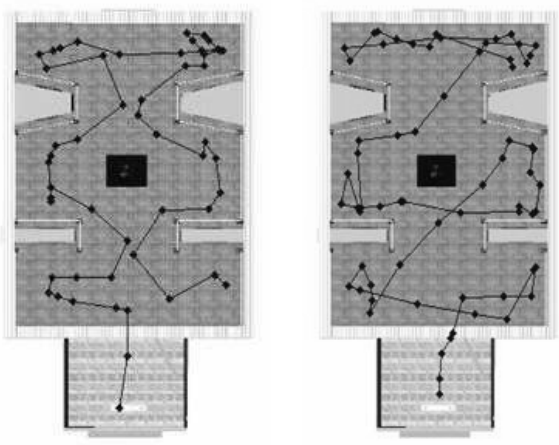

Figure 5: Left- and right-hander's path of walk

Our experiments will be continued with higher numbers of tests and with greater worlds which are more complex. According to our results we should take care of symmetrical designing, in case of interactive worlds the usable objects should be

placed equally on both left and right side. In this way virtual worlds will be user friendly for left-handers too.

References

[1] Michael Abrash: Graphics Programming Black Book, Chapter 59-60, Coriolis Group Books, 1997

[2] R.L.ATKINSON et al.:Hilgard's Introduction to Psychology, Twelfth edition, Harcourt Brace College Publishers, Fort Worth, 1996

[3] Bruce F. Naylor: A Tutorial on Binary Space Partitioning Trees, Computer Games Developer Conference, 433-457, 1998., www.cs.cmu.edu/afs/andrew/scs/cs/15-463/pub/www/notes/bsp_tutorial.pdf

[4] Szirmay-Kalos László: Számítógépes grafika (Computer Graphics), ComputerBooks Budapest, 2002