Interactive Mixed-Media Virtual Environment Prototyping

Jonathan Household
jhouseha@cs.uct.ac.za

David le Roux
dleroux@cs.uct.ac.za

ABSTRACT

This paper details a tool for designing and simulating virtual environments in two dimensions. The system consists of computer controlled agents projected onto a whiteboard. Markings (which can be altered dynamically) on the whiteboard represent the obstructions (walls). A camera captures new images of the whiteboard constantly. The image is then processed by the image-processing component to determine where the walls are and this information is fed into the artificial intelligence component so that the agents move about realistically and do not move through walls.

Natural movement throughout the environment is successfully implemented via a robust collision detection system as well as bounded path finding techniques. The AI system is efficient enough to allow for relatively large agent numbers, as well as operation at acceptable frame rate.

The image processing first transforms the input images so that the projected area forms a rectangle. The image is then segmented by dividing the image into regions and calculating a threshold for each region based on an offset from the mean.

General Terms

Algorithms, Performance, Design, Experimentation, Human Factors.

Keywords

Virtual Environment Prototyping, VE, Path Finding, Agents, Behaviour, Interaction, Segmentation, Thresholds, 2D Perspective Transforms.

1. INTRODUCTION

This paper details a system be used to simulate and design a virtual environment (VE) which is more advanced than the traditional use of cardboard pieces (representing agents) on a board. Specifically, the artificially intelligent agents are controlled by a computer and move and interact with a virtual environment.

VE prototyping is usually done during the design phase of a virtual environment or game. The aim is to see the layout of the VE floor-plan, what sort of interactions take place within the VE, and in what order. It is important to have a precise idea of what a VE will look like, as well as how it will function, before development. This is the case due to the fact that changes to the VE in terms of layout whilst in the implementation phase are costly. A VE prototyping tool which is cheap and provides a test bed for agent interactions and movements within a VE is therefore needed.

The following design was implemented: A web-cam was set up above a whiteboard. The camera feeds images into an image processing system which extracts the features of the virtual environment (such as where the walls are); this information is used by the artificial intelligence code to determine where to project the agents, using a standard data projector.

The advantage of using a physical whiteboard instead of a virtual canvas is that, it is more collaborative since many people can easily sit around a whiteboard, and many people are more comfortable drawing with a pen as opposed to a mouse. The white board system allows for a more direct and tactile user experience. For example the agents can be herded around by a users hand movements as well as other physical objects. A further example of the tactile nature of the system is that the user can possibly use bits of paper as a bridge across obstructions.

Section 2 gives some background on the development and use of mixed media interfaces.

Section 3 outlines the system components.

Section 4 describes the artificial intelligence systems.

Section 5 describes the image processing component.

Section 6 gives results of the functioning and performance the system.

2. BACKGROUND

Mixed media interfaces involve the integration of standard computer interfaces (i.e. mouse, keyboard and screen) with more familiar, physical or non-computer related interfaces. The advantages of such interfaces are that, people are very familiar with tactile interfaces (i.e. pen and paper). Because of this no metaphor are used, and the need for user testing of the interface is reduced.

The paper BrightBoard: a video-augmented environment [6], details a technique for more interactive and versatile whiteboards. The technique involves the augmentation of a white board with the use of computer software and digital cameras. The system allows the user to effectively control a computer by making simple marks on a whiteboard. Whilst the system contained image processing of marks on a whiteboard captured via a digital camera device, the use of a projector and the complications it creates (i.e. lens flares and lighting changes), mean that the practical methods of implementation could not be used for the implementation of the system proposed in this paper.

The paper PENPETS: A Physical Environment for Virtual Animals [8] describes the product very similar to the system described in this paper. Techniques from the PENPETS system

have not been published; hence techniques regarding implementation from the PENPETS system could not be used.

3. SYSTEM OVERVIEW

A web camera is used to capture user markings and gestures on a whiteboard. The captured images are processed via an image processing component, which generates an environment representation. This representation is used by an artificial intelligence component to simulate movement through the environment. The end result is projected agents that react to markings on a whiteboard.

4. ARTIFICIAL INTELLIGENCE

4.1 Base AI, Obstacle Detection and Avoidance

The base structure of the AI system consists of the following:

- Agent movement is random.
- Agents cannot walk through obstacle.
- If agents are located on an obstacle they can walk off it.

Random movement is implemented selecting a location on a grid, (240*320 array of square objects) and then pushing the location (square object) onto the front of the path-storing list attribute of the agent. If the path-storing list is empty, or an obstacle blocks the agent, a new random location.

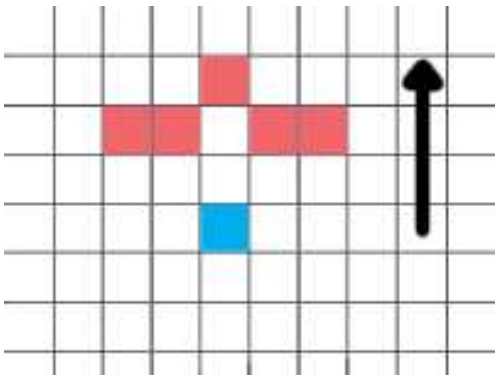

Figure 4.1 Squares Check for Obstacles

Figure 4.1 is showing an example of the layout of squares checked for obstacles (walls, etc). The blue square represents the position of the agent, the red squares represent the squares checked for obstacles and the arrow indicates the direction of agent movement.

This enables one to customize the shape of the squares scanned. This is useful if the agents are different in shape. Due to inaccuracies of the images obtained via the web camera device, the location of the obstacles fluctuates (captured image vary). The pixel bank allows the agents to detect obstacle from further away, minimizing the risk of agents walking through or getting trapped within obstacles (more robust collision detection).

4.2 Scan Areas

A mechanism for the detection of locations with various attributes is implemented. Examples scanned for attributes are food or danger areas. These locations are detected so the agent can take the appropriate action.

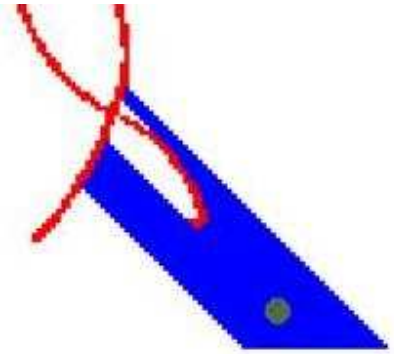

Figure 4.2 Predator Type Scan Field

The current implementation of the agent scan field consists of a line shape, which is radiated outwards from the agent position. The direction, in which the line is radiated, depends on the direction the agent is moving. Figure 4.2 shows an agent moving in a North Westerly direction. The blue section represents the scan area for this agent. One can see that the scanned area of the agent doesn't pierce walls and that the field doesn't proceed round obstacles (i.e. the agent cannot see round corners). This is achieved by not projecting the part line outward, if that part of the line has made contact with an obstacle.

The way in which the scan fields radiate from the agent position, as well as the shape of the scan field are customisable. For instance one may want to have a long, narrow, forward facing scan field (such as the field in Figure 3.2) for a predator type agent, or a shorter, wider, forward and backward scanning field for a prey type agent.

4.3 Path Finding

Concerns with respect to path finding are:

- That it should be fast enough to calculate paths, without noticeable slowdown of the system.

- The path finding algorithm should be suitable for the calculation of paths in a dynamic environment.
- Agents should move though out the environment in a natural way.

The following path finding algorithms and techniques were investigated:

The **Breadth first searching algorithm** [2], which is an uninformed search, that functions by scanning uniformly outward from the starting node (position) in the search graph.

Dijkstra's shortest path algorithm [3] this algorithm scans adjacent grid cells for the lowest cost block based on the G cost heuristic (the cost of moving from the start point of the path to the scanned grid cell). It is a best first search algorithm. This search technique is undirected because the heuristics used are not goal specific.

Potential fields involve the pre-processing of the grid and calculating a potential value for each of the grid cells, the path is then calculated to be that which the follows the high to low potential path (goal has zero or very low potential and obstacles have a very high potential making it impossible to walk onto them).

Non Uniform Grid Cells can be used. Pre-processing of the search grid based on the rectilinear obstacles can lead to fewer search nodes (adjoining obstacle locations become one).

•**The A* algorithm** [2]

This algorithm is the same as Dijkstra's algorithm with the addition of a heuristic (H cost), which is the predicted cost of moving from the current grid cell to the goal. The A* algorithm is therefore a directional search reducing the amount of nodes in the search graph.

The implemented system uses the A* algorithm due to its superior performance in terms of both iterations and time taken for path calculations. The A* implementation was optimised via the use of a binary heap data structure [10].

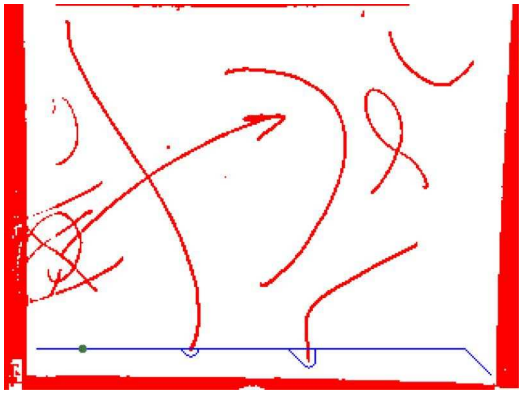

Figure 4.3 Bounded A* Calculated Path

A bounded version of the A* algorithm is implemented, where after the search algorithm performs more than a threshold value of iterations, obstacles are no longer taken into account. This facilitates exploration down dead ends as well as performance optimisation (see Figure 4.3).

4.4 Extended Features

Variables can be changed that affect the behaviour of agents. These include the speed of the agent, the amount of obstacle collisions that occur before agents calculate paths as well as the types of attributes the agents scan for and how the agent reacts to them.

Path storage for agents is maintained in a standard list structure allowing for the paths of any length as well as dynamic changing of an agent's path at run time.

5. IMAGE PROCESSING

5.1 Segmentation

The two aims of the image processing component are that it:

- Gives a qualitatively correct segmentation
- Is fast enough to segment several frames per second on the average current PC.

There are several approaches to segmenting images:

Global Threshold

A single threshold is used for the whole image. The problem is that wall and background values may overlap, but not in the same region of the image. For example, the background of one side might be darker than the other, but then the walls on this region will also

be darker, but a single global threshold will not segment the image correctly.

Moving average

The image is scanned one pixel at a time. Each pixel gets its own threshold based on the average grey-scale value of the last *n* pixels. This handles variations in intensity better than the global threshold. A horizontal and a vertical pass can be done and the thresholds can be averaged.

Regional Threshold

The image is split up into various regions and each region gets its own threshold.

Offset from mean

The threshold for each region is simply set to the mean value of the region minus *x*. The reason this is feasible is that the input images consist mainly of background.

Histogram analysis

The pixel values within a region are plotted in a histogram. Ideally the histogram should have a bimodal distribution, one representing the wall and the other representing the background. Unfortunately, since the walls form such a small part of the whole region, the wall data doesn't influence the histogram much. If a crude edge detection is done on the region first, e.g. by convolving the image with a Laplacian type matrix, and then only putting the values of the pixels that were identified as part of the edge into the histogram[9], two peaks can be obtained. There are various ways of determining where the two peaks are, such as the *two peaks* method [5] which takes the first peak as the highest bar and finds the second peak by taking into account the height of the bar and the distance from the first peak (favouring those peaks that are further away). Another way of determining where to place the threshold using histogram data is to use *iterative selection* [7] which uses consecutive passes through the histogram to refine the threshold. The new threshold becomes the average of: the mean of the pixels above the threshold, and the mean of the pixels below the threshold.

5.2 2D Perspective Transformation

Since the projector may not project onto a surface that is perfectly perpendicular to it, the image obtained by the camera will not necessarily be a rectangle. A 2D perspective transformation can convert the image obtained into one where the corners are at the corners of a 320 x 240 rectangle. The required matrix can be numerically calculated using Gaussian reduction. This transformation preserves lines but not angles. For each point in the 320 x 240 rectangle the transform is done and then rounded to give the closest pixel in the input image. [1] Presents some theory concerning homogeneous co-ordinates and perspective transformations.

6. RESULTS

6.1 Path Finding Performance

The use of potential fields and non-uniform grids, require large amounts of pre-processing of the search grid (space). Because of this such methods are infeasible in a dynamically changing environment. Hence they were not considered for implementation.

The breadth first search produces a search graph containing an unacceptably large number of nodes. This Results in a high number of algorithm iterations. The relative performance of this algorithm can be seen in figures 6.1 and 6.2.

The A* algorithm yields optimal performance, in terms of both algorithm iterations as well as the total time taken to calculate paths (again see Figures 6.1 and 6.2.).

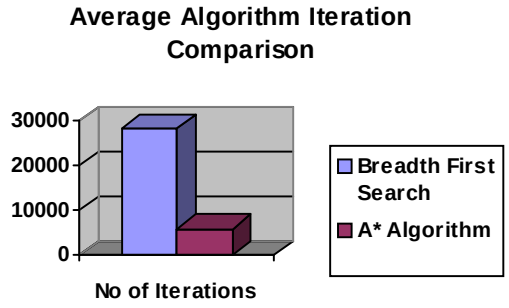

Figure 6.1

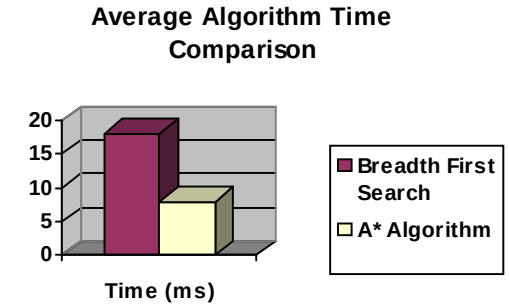

Figure 6.2

6.2 Qualitative Performance of Agent AI

The agents successfully avoid obstacles and are prevented from walking through walls. Agents can navigate a maze without getting stuck in any one part of the maze if an exit to the area exists. Agents are capable of successfully detecting entities of interest in the environment and reacting to these entities, once detected. These elements of the system were tested by observation and program use.

6.3 AI General Performance

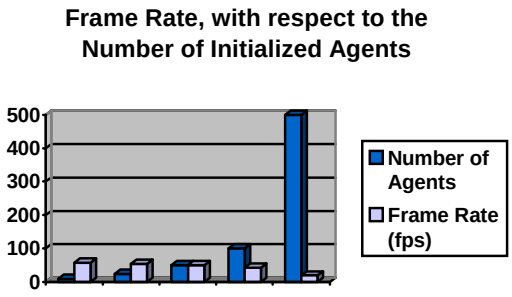

Figure 6.3

The system is capable of operation at acceptable frame rates whilst large amount of agents are initialised. Figure 6.3 illustrates that acceptable frame rates are achieved with the initialisation of up to 100 agents. This illustrates the acceptable functioning of the AI system in terms of speed and efficiency. The results in Figure 6.3 were obtained with the image processing system deactivated.

6.4 Segmentation Results

No single global threshold gives a satisfactory segmentation as can be seen in Figure 6.4

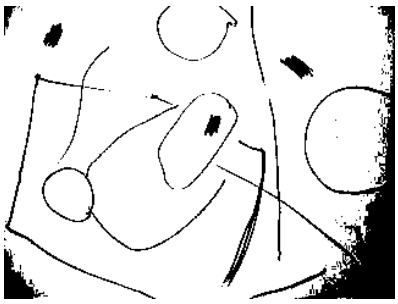

Figure 6.4 – Segmentation using a global threshold

The moving average algorithm gives satisfactory results although it gives best results when the traversal alternates between left-to-right and right-to-left and when vertical information is also taken into account. Vertical information can be taken into account by first doing a horizontal traversal and then averaging the threshold obtained with the thresholds obtained from a vertical traversal. Figure 6.5 shows the segmentation obtained using this method:

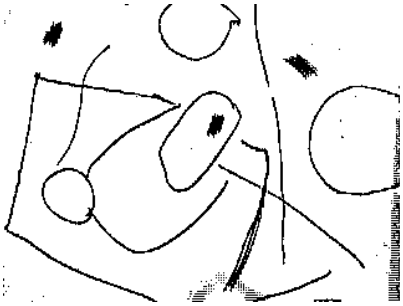

Figure 6.5 – Segmentation using a moving average

The regional segmentation based on thresholds calculated from offsets from the mean gives a satisfactory segmentation (as can be seen in Figure 6.6) and is fast. This is the segmentation technique used in the implementation.

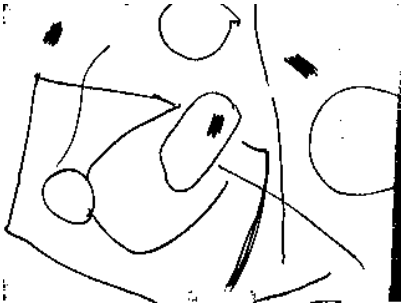

Figure 6.6 – Segmentation using interpolated regional offset from the mean thresholds

Two peaks and iterative selection give similar results and neither give consistently good results, even when both classes of pixels are present in a region. Figure 6.7 shows the segmentation obtained using iterative selection:

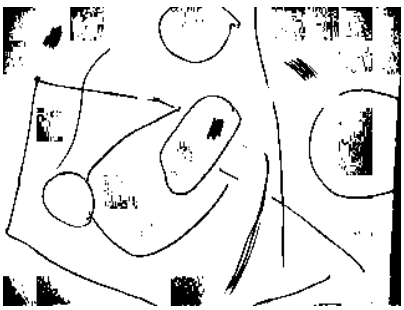

Figure 6.7 – Segmentation using iterative selection on Laplacian generated histograms

6.5 2D Perspective Transformation Results

The 2D perspective transformation is necessary because otherwise the agents may seem to walk through walls because the position of the walls relative to the camera is not the same as the position of the walls relative to the projector. Agents seeming to walk through walls is not acceptable from a usability point of view. An example of this transformation is shown in Figure 6.8

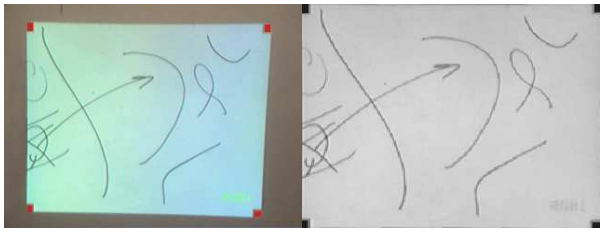

Figure 6.8 – 2D perspective transformation

7. CONCLUSIONS

The AI system successfully simulates natural movement through the environment represented on the whiteboard. Path finding as well as agents collision detection is both robust and adequate in terms of performance.

The interpolated regional offset from the mean segmentation algorithm is both qualitatively good and fast. The 2D perspective transformation is also necessary for usability reasons. The image processing component is capable of processing (transform and segmentation) 13 frames per second on current low-end hardware.

The system provides a low cost method for the prototyping of a VE.

8. REFERENCES

[1] Edward Angel, Interactive Computer Graphics: A Top down Approach Using OpenGL Third Edition, 2003.

[2] [Nilsson, 1998] Artificial Intelligence: A New Synthesis, Nils J. Nilsson, 1998.

[3] P.E. Hart, N.J. Nilsson and B. Raphael, a Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Trans. On Systems Science and Cybernetics, 1968.

[4] Nils J. Nilsson, Artificial Intelligence: A New Synthesis, 1998.

[5] Parker J., “Algorithms for Image Processing and Computer Vision”, Wiley & Sons Inc., 1997.

[6] Quentin Stafford-Fraser, Peter Robinson, Bright Board: A Video-Augmented Environment, 1996.

[7] Ridler T., Calvard S. Picture, Thresholding Using an Iterative Selection Method, IEEE Transactions on Systems, Man, and Cybernetics. Vol. SMC-8. 8:630-632, 1978.

[8] Shaun O’Mahony and John A. Robinson, PENPETS: A Physical Environment for Virtual Animals, ACM, 2003.

[9] [Weszka J., Nagel R., Rosenfeld A., A Threshold Selection Technique, IEEE Trans. on computer. Vol. 23:1322-1326, 1974.

[10] Mark Allen Wiess, Data Structures and Problem Solving Using C++: Second Edition, 2000.