# Guiding Visitors of Web3D Worlds through Automatically Generated Tours

Luca Chittaro  Roberto Ranon  Lucio Ieronutti
HCI Lab, Department of Math and Computer Science
University of Udine
via delle Scienze 206, 33100 Udine, Italy
chittaro@dimi.uniud.it  ranon@dimi.uniud.it  ieronutt@dimi.uniud.it

## ABSTRACT

*Many Web3D sites do not offer sufficient assistance to (especially novice) users in navigating the virtual world, find objects/places of interests, and learn how to interact with them. This paper aims at helping the Web3D content creator to face this problem by: (i) proposing the adoption of guided tours of virtual worlds as an effective user aid and (ii) describing a novel tool that provides automatic code generation for adding such guided tours to VRML worlds. Finally, we will show how the tool has been used in the development of an application concerning a 3D computer science museum.*

## Categories and Subject Descriptors

I.3.6 [**Computer Graphics**]: Methodology and Techniques – *Interaction techniques*. I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism – *Interaction techniques* H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems – *Artificial, augmented, and virtual realities.*

## General Terms

Design, Human Factors, Algorithms.

## Keywords

Navigation Aids, Animated Characters, H-Anim, Virtual Museums, VRML

## 1. INTRODUCTION

A well-known factor limiting the diffusion and popularity of Web3D sites on the Internet is their scarce usability (especially from the point of view of novice users). While developers of traditional Web sites are aware that usability is one of the key issues for the success of their sites and rely on guidelines and tools that support them in this direction, Web3D content creators have generally neither Web3D-specific guidelines nor specific tools available to help them.

In our research, we are concentrating on finding ways to increase the usability of Web3D sites. In this paper, we consider a specific usability problem that affects many Web3D worlds, i.e. insufficient user assistance during world exploration, and propose an approach aimed at helping the Web3D content creator to face the problem with little effort.

The considered solution is based on exploiting a humanoid animated character (more specifically, an H-Anim character) making it able to lead the user on a *guided tour* of the world. This solution is at the same time a navigation aid (since it helps users in finding places of interest) and an information aid (since the character is also able to provide information about the encountered places and objects). Moreover, the introduction of an animated character has the additional advantage of making the virtual world more lively and attractive for the user.

Unfortunately, developing guided tours led by H-Anim characters is not an easy task for the Web3D content creator, since it currently has to be done mostly by hand (e.g., coding a suitable path for the virtual guide avoiding obstacles). Moreover, the code written for one world can be very limitedly reused for other ones.

The aim of this paper is to aid the Web3D content creator by proposing an automatic approach to solve the considered problem. More specifically, given a VRML world, an H-Anim character, and a high-level description of the desired destinations (basically, the ordered list of objects/places in the world that must be in the tour, and possibly their textual descriptions), the proposed tool is able to automatically generate the necessary VRML code to make the H-Anim character act as a virtual guide that: (i) leads the visitor through the virtual world on a tour that includes all the specified objects/places in the given order, and (ii) stops at each object/place and possibly presents it.

*Path planning* is one of the main capabilities an animated character must include to be able to autonomously travel any environment. *Path planning algorithms* have been originally developed by the artificial intelligence community for autonomous robot applications [12], and later successfully exploited also by the game industry [16][17] and virtual environments research [2][3][7][8][11]. However, in the Web3D context, there are neither general-purpose solutions targeted to the Web3D content creator, nor approaches to the problem that exploit the H-Anim standard.

The paper is structured as follows. First, in Section 2, we advocate the exploitation of virtual tours with embodied guides as an effective user aid in Web3D worlds, and compare it with other existing user aids (e.g. maps, viewpoint tours). In Section 3, we illustrate our approach to building guided tours with H-Anim characters, and show how a Web3D content creator can use it as a stand-alone tool for automatic code generation. In Section 4, we show how the approach can be also exploited within a dynamic Web3D site to generate guided tours that are *personalized* to the needs of the current visitor. In Section 5, we provide a practical example of how we exploited our approach in a specific application, by describing how we added personalized guided tours to a 3D Computer Science museum based on the virtual reconstruction of a computing center of the 70's. Finally, in Section 6 we discuss the current limitations of the proposed approach and outline how we plan to overcome them.

## 2. MOTIVATIONS

Many Web3D worlds, whether representing existing places (e.g., virtual cities) or imaginary ones, typically leave the user alone and partially or totally unassisted in navigating the environment, discover points of interest, and interact with objects. Although in games this can be a desirable situation, it is definitely not in Web sites devoted to other purposes (such as virtual tourism, training, museums, e-commerce,...). Leaving the user unassisted can lead to a number of usability problems, ranging from navigation issues (e.g., wayfinding) to difficulties in figuring out which operations can be performed on the objects in the world. In the following, we analyze these issues in detail.

### 2.1 Navigation Issues

The lack of proper navigation support causes the user to suffer from well-known navigation problems (e.g., wayfinding, disorientation, ...). As a result, visitors of a virtual world can: (i) become rapidly frustrated and leave the world; (ii) miss interesting parts of the world (especially in large environments), and (iii) complete the visit with the feeling of not having adequately explored the world.

This is particularly true for novice users of a virtual world, who should be helped as much as possible to navigate the world by offering them proper navigation aids (see, e.g., the survey on navigation and wayfinding issues in virtual environments provided by [20]). To this purpose, a first category of solutions can be derived from the design of real-world environments. For example, in the design of buildings, architects aim at reducing wayfinding problems for the people traveling the building by increasing visual access (i.e. the number of parts of the environment which can be seen by a person from her position in space) or including navigational cues (e.g., room numbers, names of buildings, landmarks). Landmarks are distinctive environmental features (e.g., a statue, a river, a town square, …) functioning as reference points [23]. However, this kind of solutions can be limitedly applied to Web3D sites that are virtual reconstructions of existing places, where the architecture of the environment cannot be modified.

A second category of solutions aims at providing the user with electronic navigation aids to augment her capabilities to explore and learn. A well known example in this category are the various types of electronic maps of the environment that help users orient themselves (see, e.g., [6][8]). However, electronic maps adopt a third-person perspective, which can require a considerable cognitive mapping effort to be correctly interpreted (e.g., consider the typical real-world situation where someone is trying to find her way in a city by using a map and has to translate the exocentric view of the map into her egocentric view). This claim is supported by psychological studies, e.g. Thorndyke and Hayes-Roth [21] compared spatial judgment abilities of subjects who learned an environment from personal exploration or from a map, highlighting the difficulty of changing perspective (e.g., subjects who acquired knowledge from the exocentric map perspective were most error prone in tasks that required to translate their knowledge into a response within the environment).

The two above mentioned categories of solutions are complementary and can be synergically exploited by the Web3D content creator. In this paper, we focus specifically on a solution belonging to the second category.

### 2.2 Getting Information about the World

Another difficulty that can be experienced by the visitor of a virtual environment is to determine what she can do with objects and how to get information about them. Instructions to the user are typically provided by introductory pages, which describe the contents of the world, and how to interact with objects. Many users typically skip these introductory pages and later fail to explore parts of the world because they are not aware of all the possible actions (e.g., to notice that some actions on an object are possibile, one could have to try moving the pointer over the object to check if the pointer icon changes).

Providing information about objects can be crucial, for example, in virtual museums, virtual training and e-commerce sites, where getting additional information is a fundamental part of the user experience. Some sites face this problem by providing the information during the visit in a separate HTML frame. However, this solution has the disadvantage of decreasing user immersion, and is not viable when the user wants to display the Web3D world in full screen mode (this is a possibility offered by recent VRML players, but does not allow one to simultaneously display other windows and HTML frames).

### 2.3 Using Animated Characters as Guides

One of the solutions that can take into account all the previous concerns is to offer *tours* of the Web3D world. Moreover, from the Web3D content creator perspective, the possibility of offering tours allows one to naturally suggest preferable ways to visit the world. This can be important in many kinds of virtual worlds, such as virtual exhibitions, museums, and cities.

Simple forms of (unguided) tours are already offered by some Web3D sites by providing a set of viewpoints through which the user has to cycle to visit the world in a certain order. However, while viewpoints can be useful for quickly navigating the world, they are not easy to use for learning navigational knowledge, such as paths, relating the different parts of the world (if teleporting is used, they also break the continuity of the experience, further contributing to user disorientation). Viewpoint control functions can even go unnoticed by novice users of VRML browsers.

The approach we adopt in this paper is to provide the user with *guided tours* based on an H-Anim character that acts as a *virtual guide*, i.e. it is able both to lead the user to the required places and to provide information through a semi-transparent *On Screen Display* (*OSD*) available inside the 3D world. In the following, we discuss in detail how this approach addresses the previously illustrated concerns, and the benefits of using humanoid characters as virtual guides.

First, virtual guides are navigation aids, leading users around and preventing them from becoming lost: the user has simply to follow the guide to visit the world. As pointed out by Rickel and Lewis Johnson [18], showing the user where relevant objects are and how to get to them is likely to be more effective than trying to tell users where objects are located. Moreover, the expert user has still the possibility to ignore the guide and follow her own tour, but it is very likely that novice users will instead appreciate and use the aid. For example, informal experiments with users in a virtual city showed that the environment was explored to a greater level and by an higher number of users as a result of tour guides explaining how to get the most out of the system [9]. From a more general point of view, while a virtual guide does not directly help the user in overcoming some typical 3D navigation problems (e.g., bumping into obstacles), a properly chosen tour can lower the probability that the user finds himself in positions where it can be difficult to find a way (e.g. corners).

Second, virtual guides can be also employed as a natural way to provide users with additional information (e.g. usage of a particular object) during the visit, instead of providing those information into separate Web pages. For example, a virtual museum guide could give an introduction at the beginning of the visit and later present specific items on display, possibly explaining how to interact with them. With respect to other forms of user guidance, an animated character can draw user's attention with the most common and natural methods, such as gaze and pointing gestures [18]. Moreover, the guide can also use its body orientation as a cue to the suggested attentional focus [18].

Third, introducing an animated character in the world can contribute to alleviate an additional problem, i.e. most virtual worlds where the user is left completely alone look like dead places (like just after the builders have left the building), that are less attractive to the user. Animated characters make instead the virtual place more lively, attractive, and less intimidating to the user. Results of empirical studies show that animated characters may have a strong motivational impact: users tend to experience presentations given by animated characters as lively and engaging [14] [22].

Finally, from a human-computer interaction point of view, the guide metaphor has the advantage of being consistent with the real-world experience of users and need not to be learned. From this point of view, we concentrated our attention on guides that travel around the world by simply walking (i.e., they do not fly, crawl, ...). Moreover, it would be very difficult for the user to follow paths that require movements other than walking, e.g., 6DOF flying in a 3D space (as VRML browsers typically allow) is known to be very difficult for novice users.

## 3. DERIVING VIRTUAL TOURS FOR WEB3D WORLDS WITH H-ANIM GUIDES

Achieving the previously illustrated goals requires to deal with a number of different problems, ranging from the derivation of suitable paths for the virtual guide, to choosing how to present objects/places of interest to the user. Writing ad-hoc code to solve these problems is not a viable solution, because the Web3D content creator would have to start from scratch with each virtual world and the process would be too costly.

Existing tools for Web3D content creators are beginning to offer some basic automation capabilities. For example, Internet Scene Assembler [15] provides some support for the creation of character paths: the Web3D content creator has to specify a set of *navigation checkpoints* (i.e. points along the desired path), and the tool derives an interpolation between them. However, this kind of solution leaves to the content creator the responsibility of: (i) guaranteeing that the animation will be collision-free with respect to other objects in the world, i.e., each line connecting two consecutive checkpoints must not intersect any obstacle; (ii) guaranteeing a smooth animation, with no unnatural character movements (e.g. too sudden turns or too rapid changes in speed), e.g., a sufficient number of checkpoints must be provided; (iii) guaranteeing that the animation will be compenetration-free, i.e., checkpoints must be positioned far away enough from obstacles to guarantee that no part of the humanoid (e.g. hands) will compenetrate obstacles. As a result, creating a suitable path for a virtual guide constitutes a time-consuming, trial-and-error activity for the Web3D content creator.

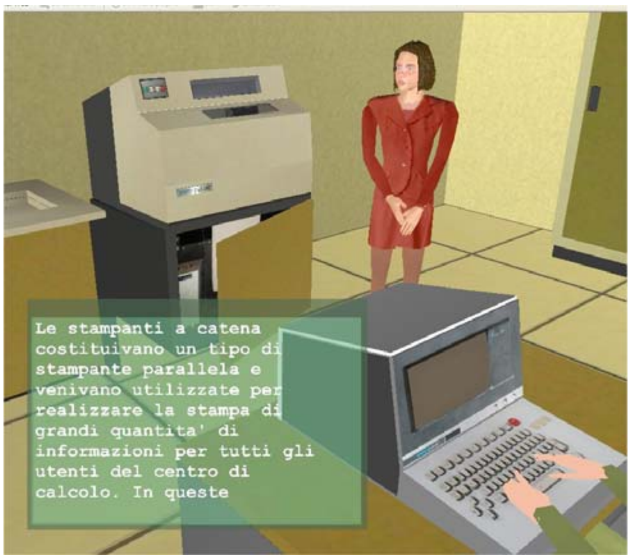
There is thus the need for solutions that provide increased levels of automation and are simpler and faster to use. In particular, we aim at providing fully automatic code generation in VRML, allowing the content creator to concentrate only on high-level aspects (e.g., which objects/places of the world need to be presented, what could be the text for their presentation,...). More specifically, our tool, called *VRML Tour Creator*, requires the following inputs:

- ❑ The VRML file of the world for which the guided tour needs to be developed;
- ❑ The H-Anim model for the virtual guide and the height and radius of the bounding cylinder of the H-Anim model;
- ❑ The ordered list (called *TourList*) of k objects/places to be presented. Each object/place is specified by a triple;

  *(GuideLocation, PresentationText, GuideGesture)*

  where *GuideLocation* is the location and orientation in the VRML world where the guide must stop (near the corresponding object/place) and possibly start the presentation, *PresentationText* is the text that describes the object/place, and *GuideGesture* is the identifier of the gesture the guide has to perform to point the object/place. *GuideGesture* is chosen from a list of available gestures, whose code for H-Anim characters is stored in a database (called *gesture DB*). The order of objects/places in the list will be followed by the guide during the tour;
- ❑ The desired maximum speed of walking for the virtual guide (in VRML units/second).

The *VRML Tour Creator* then outputs a new version of the VRML world that includes the code generated for the guided tour. As a result, when the user enters the world, she will see the H-Anim character in the location specified by the first item of the *TourList*, and will be able to read the *PresentationText* in the semi-transparent *OSD* (see Figure 1). While the text is displayed, if no *GuideGesture* has been instead specified, the guide only mimics a talking person. If a *GuideGesture* has been specified, the guide periodically switches its attention between the user and the object/place: when the guide performs the specified pointing gesture, its orientation is turned towards the pointed object; in other moments, the



```
Le stampanti a catena
costituivano un tipo di
stampante parallela e
venivano utilizzate per
realizzare la stampa di
grandi quantita' di
informazioni per tutti gli
utenti del centro di
calcolo. In queste
```

**Figure 1.** A guide presenting an object. The required presentation text is displayed on a semi-transparent OSD.

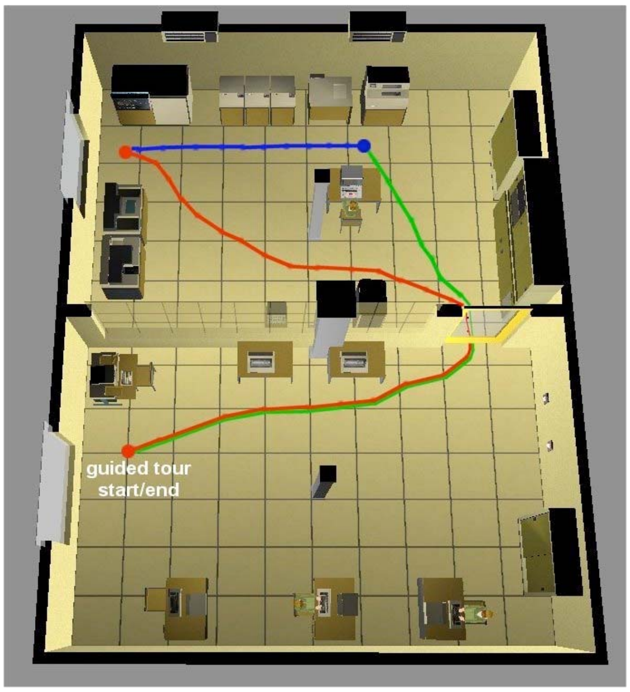


**Figure 2.** A Tour path derived by our tool in the virtual museum application.

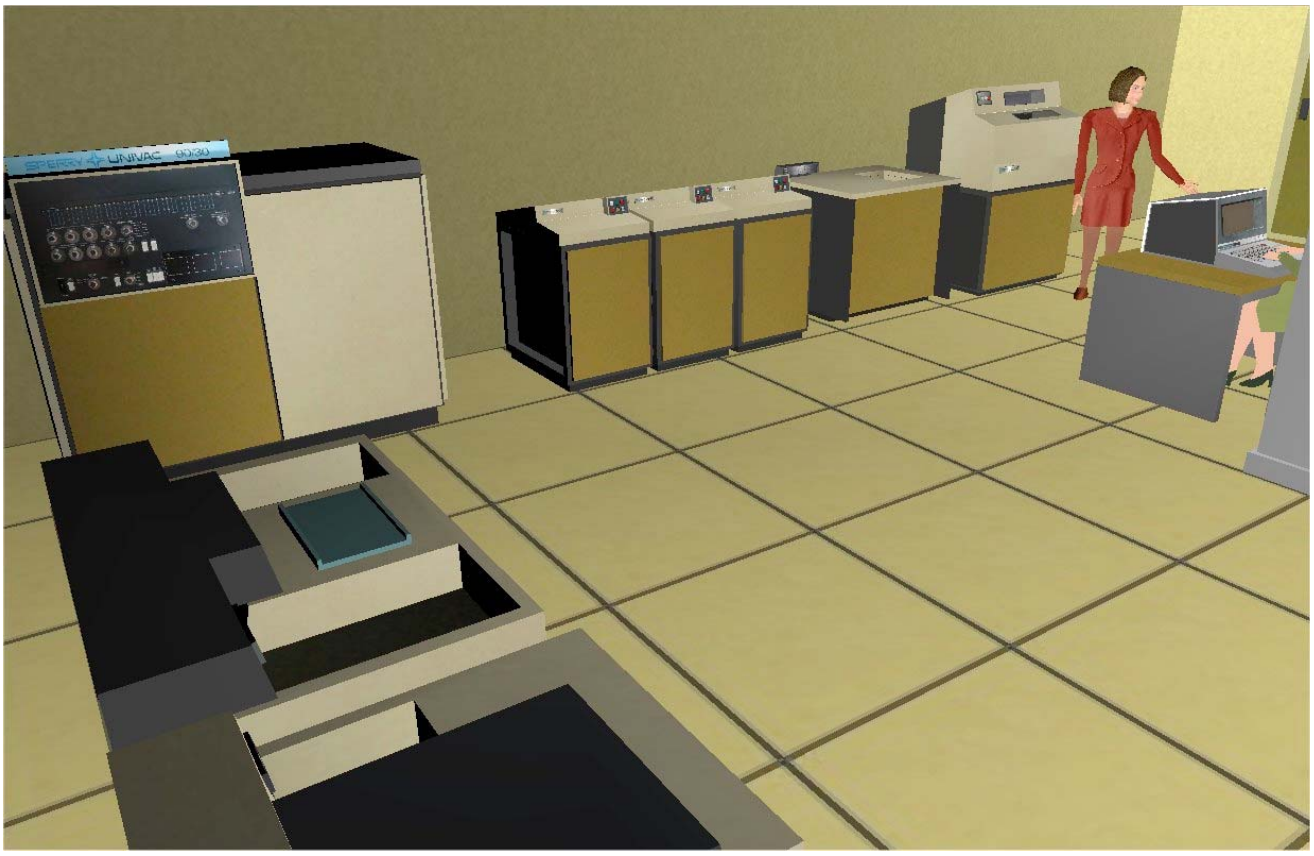


**Figure 3.** A screenshot of the virtual museum application.

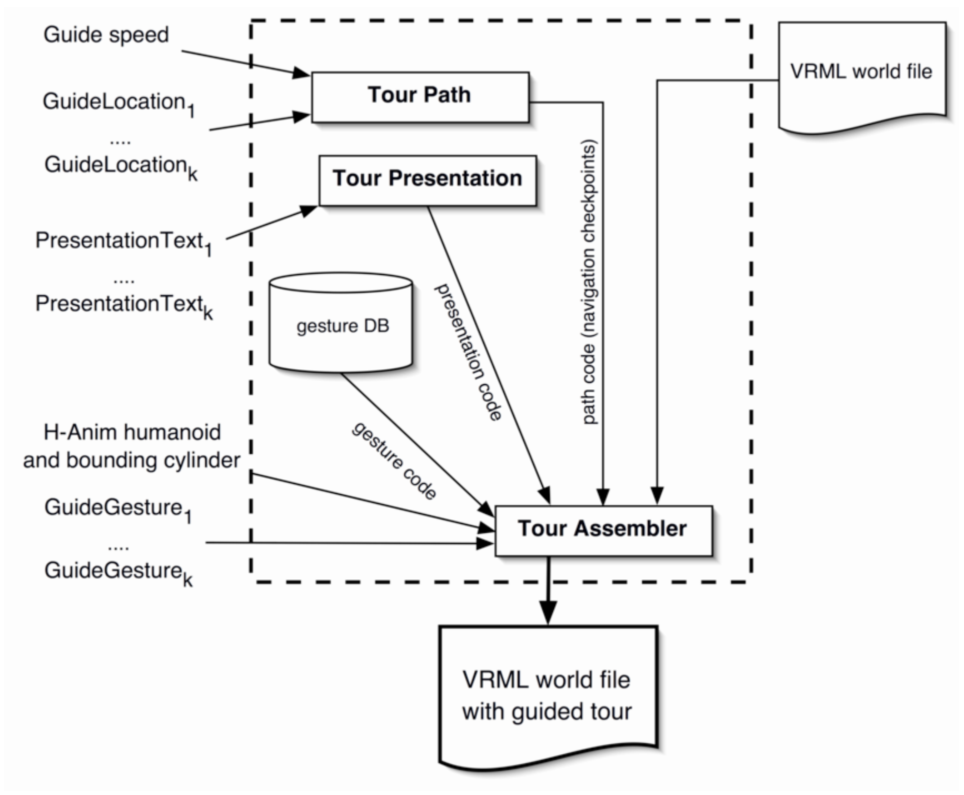


**Figure 4.** Architecture of the VRML Tour Creator.

guide is oriented towards the position of the user (dynamically detected through a proximity sensor), and moves its arms and hands to mimic a talking person. This choice has been taken to achieve two goals: on one hand, giving the user the feeling that the guide is addressing her; on the other hand, ensuring that the attention towards the required object/place is clearly directed.

After the presentation of an object, the guide starts walking towards the following item in the *TourList*, following the shortest path towards it that prevents collisions and compenetrations with any object in the world. Following the shortest path reduces travel time, while avoiding collisions and compenetrations makes the guide behavior closer to the real-world.

The general architecture of the *VRML Tour Creator* (which has been implemented in Java) is depicted in Figure 4. It is mainly composed by three modules: a *Tour Path* module that derives a list of appropriate navigation checkpoints for the required tour, a *Tour Presentation* module that generates the code for presenting objects/places, and a *Tour Assembler* module that combines the output of the other two modules to produce the VRML code for the guided tour. In the following, we consider separately each module, describing the main technical issues we had to deal with and illustrating the adopted solutions.

## 3.1 The Tour Path module

The function of the *Tour Path* module is to derive an appropriate sequence of navigation checkpoints that will be used by VRML position and rotation interpolators to move the guide from each object/place to the subsequent one in the desired tour. Two kinds of navigation checkpoints are used: (i) *translation checkpoints*, that contain a VRML translation value (i.e., a guide position in the world) and a time value (i.e., seconds elapsed with respect to the starting time of the current path), and (ii) *rotation checkpoints* that contain a VRML rotation value (i.e., a guide rotation along its vertical axis) and a time value (i.e., seconds elapsed with respect to the starting time of the current path).

The approach we follow to deal with the problem of deriving a suitable set of navigation checkpoints for the guided tour is based on using a *path planning* algorithm. *Path planning* is the problem of finding an optimal path from one place to another (the goal), avoiding obstacles [11]. The problem has been studied in artificial intelligence and robotics (in particular, for mobile robot applications) and the proposed solutions have been later employed by the videogame industry (e.g.[16][17]) and in the context of virtual environments (e.g. [2][3][11]). Many approaches to path planning have been proposed to deal with different needs and conditions: some of the possible choices are whether or not the environment is known in advance (and how accurately the environment can be possibly sensed), whether the terrain is plain or not, whether obstacles can move or not, how the cost of traveling a region is calculated (e.g. in strategy games, roads are easier to travel than mountains), and other application-specific constraints (e.g. planning a path for a military unit that must hide itself from enemies). There are two main classes of path planning approaches: one focuses on *complete* methods (i.e., those that are guaranteed to find a solution or determine that none exists, as the method we employed in this work), while the other focuses on *probabilistic* methods, which trade completeness for a substantial reduction in the complexity of the problem. For example, Amato and Yan [1] propose a randomized method for efficient path planning in complex configurations (e.g., the space is very cluttered, and there are 6DOF for movement). Probabilistic solutions looks very promising when dealing with strong requirements such as real-time constraints, moving obstacles, planning with more than 3DOF (e.g. flying), or large environments.

However, considering the Web3D context, there are neither approaches to the problem that exploit the H-Anim standard, nor general-purpose solutions targeted at the Web3D content creator.

The solutions developed in other contexts are not suited to the Web3D context for various reasons. In the approaches to path-planning developed in robotics, for example, the robot can typically sense only a part of the environment. Moreover, robotic applications present also sensor accuracy problems (e.g., one may need to drastically change a planned path because sensors previously gave an inaccurate representation of the environment). On the contrary, in Web3D worlds one has an accurate representation of the full world available (which could in principle be derived from the VRML file of the world itself).

Approaches to path planning developed for 3D videogames typically suffer from the fact that character paths must be generated while the game is running and limited CPU time can be assigned to the task. As a result, they tend to offer scarce automation abilities and require a considerable work for each level of a game. For example, the path planning algorithm developed for computer-controlled characters in the *Unreal Tournament* game requires the developer to carefully position a number of navigation checkpoints that must be both sufficient but not too large, such that they cover each intersection or turn, are in line of sight with each other, and sufficiently far from walls and obstacles [16].

The approaches to path planning developed for virtual environments often focus on obtaining an extremely realistic behavior of the walking humanoid. For example, Granieri et al. [10] propose a multiprocessing system for the real-time execution of behaviors and motion of an interactive human-like agent based on an underlying model of continuous behavior coupled with a discrete scheduling mechanism. In [7], the output of a path planning algorithm is used to plan each walking step of a humanoid (i.e. planning occurs at the footprint level), and then a motion control system using inverse kinematics is used to control the motion of the rest of the body after each leg movement. However, these approaches are not meant for the VRML and H-Anim standards, they do not propose stand-alone tools for content developers, and they are sometimes based on high-end hardware such as multi-processors.

In our approach, we combine ideas taken from different approaches, integrating them inside a stand-alone tool that aims at providing a good compromise between simplicity (the final tool is targeted at the average Web3D content creator, and should also be run on a common PC), generality (support of the H-Anim standard and applicability to many VRML worlds), and realism.

### *3.1.1 Path planning for the guided tour*

We adopt a modified version of a well-known *grid-based* path planning approach [12]. In the following, we give a concise overview of this approach and we discuss in detail how we applied it to the Web3D context.

We classify areas of the virtual world in two categories: areas that the guide can freely navigate, and areas that it cannot (because they contain obstacles). From this perspective, the optimal path is simply the one with minimum length among those that avoid obstacles.

We first represent the 3D world as a *2D occupancy grid* [12], i.e. a two dimensional matrix. Each cell of the grid corresponds to a (small) area of world floor (i.e. the plane on which the guide walks). The grid indicates which cells can be traveled by the guide, and which cannot. The grid can be obtained by ideally cutting the virtual world with a plane that is parallel to the floor and positioned just above the guide's virtual head (obstacles above the head do not need to be considered), and then projecting the objects that are below the cutting plane on the floor. Those cells that contain (a part of) a projected obstacle are marked as NOT-FREE, while the others are marked as FREE. Moreover, to avoid compenetration between extremities of the guide (e.g., hands) and obstacles (this can happen when the guide goes very close to the border of a cell that contains an obstacle), we mark as NOT-FREE also those cells that do not contain obstacles but are within a circle of R from each original NOT-FREE cell, where R is the radius of the bounding cylinder of the humanoid. In this way, it is guaranteed that the guide will always be in positions that avoid compenetration. As an example, consider the trivial virtual world shown in Figure 5 and its occupancy grid shown in Figure 6: NOT-FREE cells that contain obstacles are highlighted in black, NOT-FREE cells that do not contain obstacles in grey, and FREE cells in white.

Now suppose that we want to find a path for the guide from its actual position (at the top left corner) to the goal position that has been highlighted with a sphere. Given the occupancy grid and the goal position, a *cost grid* for the current path planning problem is determined. The cost grid is a matrix of the same size as the occupancy grid (there is a one-to-one correspondence between the cells in the occupancy grid and those in the cost grid) where each cell contains the minimum cost needed for traveling from the cell itself to the goal position. Each value of the cost grid is calculated as follows: first, the cost of NOT-FREE cells in the occupancy grid is set to the largest representable number; then, the cost of FREE cells is calculated starting from the goal cell (whose cost is zero), and then recursively considering adjacent cells that do not contain obstacles (i.e., using a floodfill algorithm [13]). The cost of adjacent cells is estimated as the cost of the current cell plus 1 (for laterally adjacent cells) or plus $\sqrt{2}$ (for diagonally adjacent cells). As an example, Figure 7 shows the cost grid for the path planning problem depicted in Figure 5.

Using the cost grid, we then calculate the navigation checkpoints along the shortest path in the following way (note that a cell can contain multiple checkpoints, because the guide can make several steps in that cell).

First, given the actual position of the guide, we determine the direction of movement (i.e., an angle between 0 and $2\pi$) along the shortest path as the cost gradient [12] at the current guide position: if the cost grid is visualized as a 3D surface where

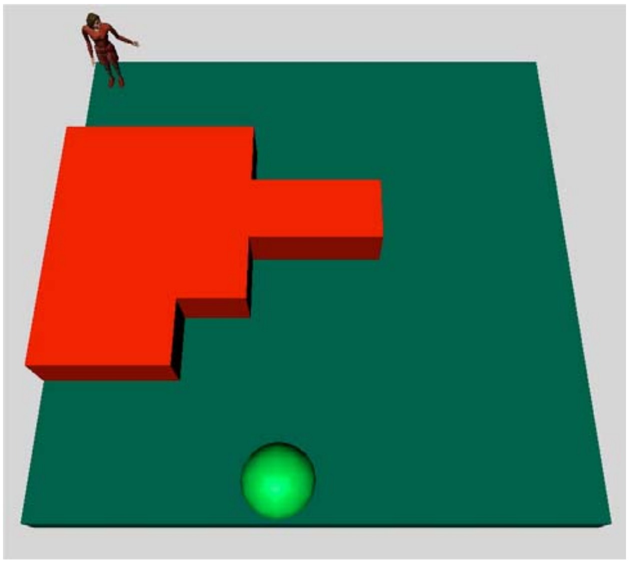


**Figure 5.** A simple VRML world with a virtual guide starting at the top left corner, and its goal position (the sphere).

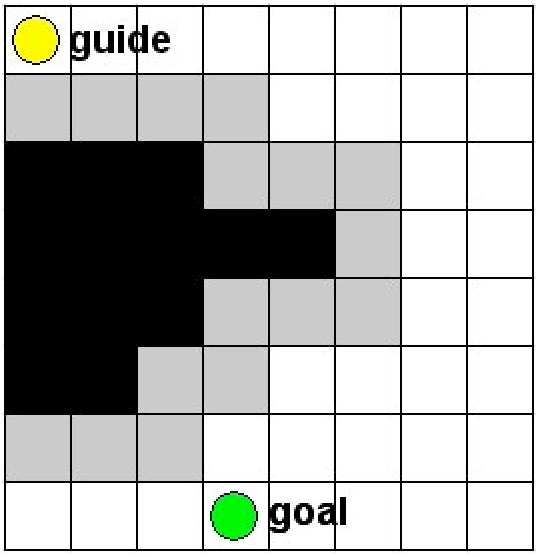
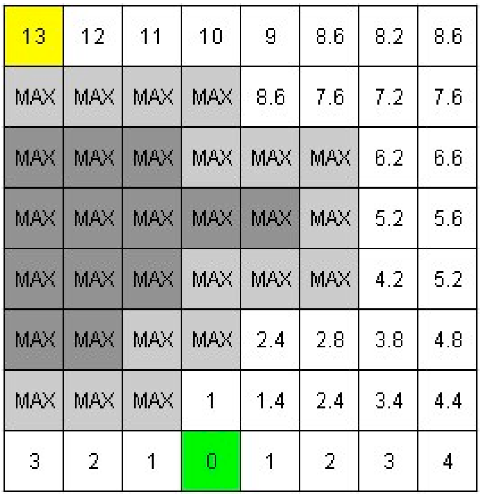


**Figure 6.** Occupancy grid for the path planning problem depicted in Figure 5.

heights corresponds to costs (i.e. the goal is at the lowest level), then the gradient determines the downhill direction along the shortest path. We separately calculate the gradients along the X and Z axis, and then we convert them into a direction between 0 and $2\pi$.

Second, we derive both a new translation checkpoint with a translation value equal to the position where the guide will be after taking a step in the calculated direction, and a new rotation checkpoint with rotation value equal to the calculated direction.

These two steps are then repeated (considering the newly derived checkpoints as the actual location and orientation of



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 13 | 12 | 11 | 10 | 9 | 8.6 | 8.2 | 8.6 |
| MAX | MAX | MAX | MAX | 8.6 | 7.6 | 7.2 | 7.6 |
| MAX | MAX | MAX | MAX | MAX | MAX | 6.2 | 6.6 |
| MAX | MAX | MAX | MAX | MAX | MAX | 5.2 | 5.6 |
| MAX | MAX | MAX | MAX | MAX | MAX | 4.2 | 5.2 |
| MAX | MAX | MAX | MAX | 2.4 | 2.8 | 3.8 | 4.8 |
| MAX | MAX | MAX | 1 | 1.4 | 2.4 | 3.4 | 4.4 |
| 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 |

**Figure 7.** Cost grid for the path planning problem depicted in Figure 5.

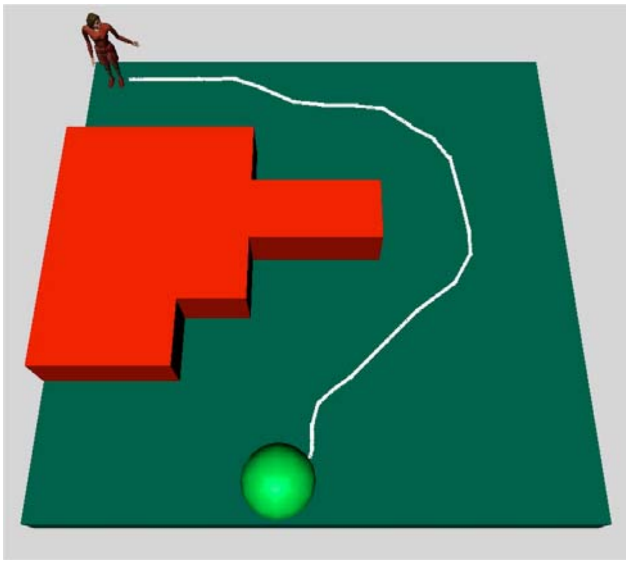


**Figure 8.** Calculated path for the path planning problem depicted in Figure 5.

the guide) until the humanoid is sufficiently near to the goal position. Since when the guide is near an obstacle (i.e. it is in a cell which is adjacent to NOT-FREE cells) the gradient method can cause unrealistic (jerky) movements, in such cases the algorithm uses the direction towards the lowest cost neighbor instead of calculating the gradient.

By connecting the set of translation checkpoints derived for the path planning problem in Figure 5, we obtain the path shown in Figure 8.

Time values for all the derived checkpoints are then calculated as follows. First, time values for translation checkpoints are set according to the guide desired speed, avoiding sudden accelerations. As a consequence, the guide reaches its maximum speed only a few instants after starting its walk, and slows down just before reaching her goal position.

Since the path derived by our algorithm is a stepwise linear curve, we want to avoid situations where the guide is not oriented towards the walking direction (i.e., the guide “slides” laterally while walking). In principle, this could be achieved by simply having the guide stop at each turn, rotate towards the new direction, and then resume walking. However, while this strategy would work for a wheeled robot, it is unrealistic for a humanoid. Therefore, we adopt the following solution: we delay the time of each rotation checkpoint with respect to the corresponding translation checkpoint by a small amount (i.e., the time value of each rotation checkpoint is the average between the time values of the corresponding translation checkpoint and the subsequent translation checkpoint). As a consequence, given a translation checkpoint that corresponds to a turn in the path, the guide will start changing its orientation (at the previous rotation checkpoint) just before the turning point, and will then reach the required degree of rotation just after the turning point.

This solution is not optimal, because there are still situations in which the guide is not precisely oriented towards the walking direction. However, the result looks anyway quite realistic, and has the advantage of keeping the walking animation smooth (i.e., no stops are made for turning).

For a complete guided tour, given the occupancy grid and a sequence of *k* *GuideLocation*s, *k-1* path planning problems, that differ in the initial and final position of the guide, need to be solved on that grid. More specifically, each problem will use $GuideLocation_i$ as the initial position and $GuideLocation_{i+1}$ as the goal position. When all *k-1* path planning problems have been solved, the complete sequence of navigation checkpoints for moving the guide through the desired tour is available.

### *3.1.2 Building the occupancy grid*

The accuracy of the occupancy grid is critical for the success of path planning. For example, if a cell that is considered free from obstacles contains instead a part of some object, there is no guarantee that the derived guide animation will be completely free from collisions and compenetrations.

The resolution of the grid is also important: in principle, it would be best to use high resolutions, since this would result in the most accurate representation of obstacles (geometrically complex obstacles can be accurately represented using a proper resolution of the grid) and smoothest guide animation.

In general, since the algorithm that computes the path is computationally expensive (although polynomial and manageable for most practical uses, i.e. when the world and the grid resolution are not extremely large), one should generally avoid too high resolutions. In particular, given a $n*n$ grid, the procedure that calculates the cost grid has to perform $n^2$ cost evaluations. In order to limit this problem, one can (besides trivially limiting the size of the grid) adopt well-known approaches (such as the *A** algorithm [19]) that try to calculate only the cost for relevant cells (i.e., cells that are likely to contain the optimal path). Anyway, this complexity issue is not critical in our case, because our tool performs path-planning off-line. The issue would instead become more important if one needs to run the path-planning algorithm on the client-side, e.g. to have the possibility of dynamically changing the path of an animated character during the visit.

At present, we build the occupancy grid by hand, either by starting from a paper sketch of the world map (that the content creator often draws before building a world) or by starting from a computer image that has been obtained by putting a virtual camera in a proper position above the virtual world, and then taking a snapshot. However, we are currently experimenting with an automatic approach that, once the floor has been given a special color (i.e. not used for other objects in the world), uses low-level image processing algorithms on the 2D image given by the virtual camera to distinguish regions where the guide can travel from regions containing obstacles. A similar approach to occupancy grid derivation has been successfully employed by [11].

## 3.2 The Tour Presentation Module

The *Tour Presentation* module derives the VRML code for the presentation of each object/place in the tour. Each *PresentationText* is displayed in a semi-transparent *OSD* that appears at presentation time (see Figure 1). However, other methods could be employed for presentation. For example, we have easily developed a talking version of the virtual guide using a Microsoft Text to Speech ActiveX object: every time the guide has to present an object/place, the proper text for the presentation is loaded in a hidden HTML frame to be read by the ActiveX object. However, the *OSD* solution has the advantage of being compatible with various platforms and does not require the user to download and install a speech engine.

The code for presentations on the *OSD* is derived as follows. Each *PresentationText* is first divided by the *Tour Presentation* module into a suitable number of substrings, each one containing a portion of text that will fit into the width of the *OSD* area; then, the module creates a number of VRML Text nodes, such that each node contains a number of substrings to fill the height of the *OSD*. The *OSD* is a VRML PROTO, whose instantiation with the Text nodes as field values gives the VRML code for the presentations. The PROTO of the *OSD* includes a script that will select and display the correct *PresentationText* for the object/place that is being presented.

## 3.3 The Tour Assembler Module

The *Tour Assembler* module uses the outputs derived by the two previously described modules together with other inputs from the Web3D content creator (i.e., the VRML file of the world, the H-Anim model and the required *GuideGesture*s for the object/places in the tour) to assemble the VRML code for the guided tour and include it in the world.

Figure 9 shows an high-level schema of the VRML code for the virtual tour. A *Tour Controller* Java script sends the proper start/stop commands to animate the guide (walking, performing gestures, ...) and activates the *OSD* when needed.

For example, when the guide needs to walk towards a new object/place, a the *Tour Controller* activates the position and rotation interpolators that, following the sequence of navigation checkpoints, set the location and orientation of the guide over time such that it moves towards the destination on the calculated path. At the same time, the *Tour Controller* runs also the code for making the H-Anim humanoid change properly its posture over time (in this case, move its legs). Only the relevant code for making the guide perform the required gestures is retrieved from the *gesture DB* and inserted in the final VRML code instance.

The same method is used to make the OSD appear on the screen, and display the required text.

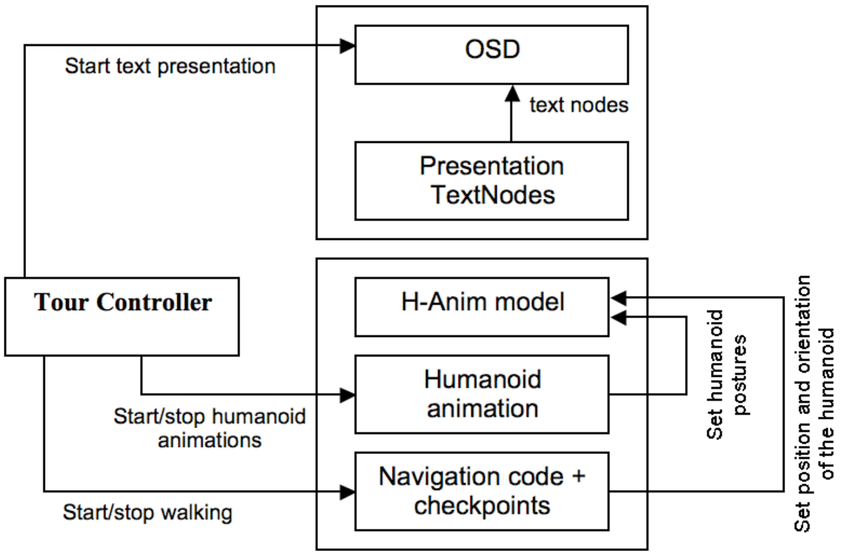


**Figure 9.** Schema of the VRML code for a generic guided tour.

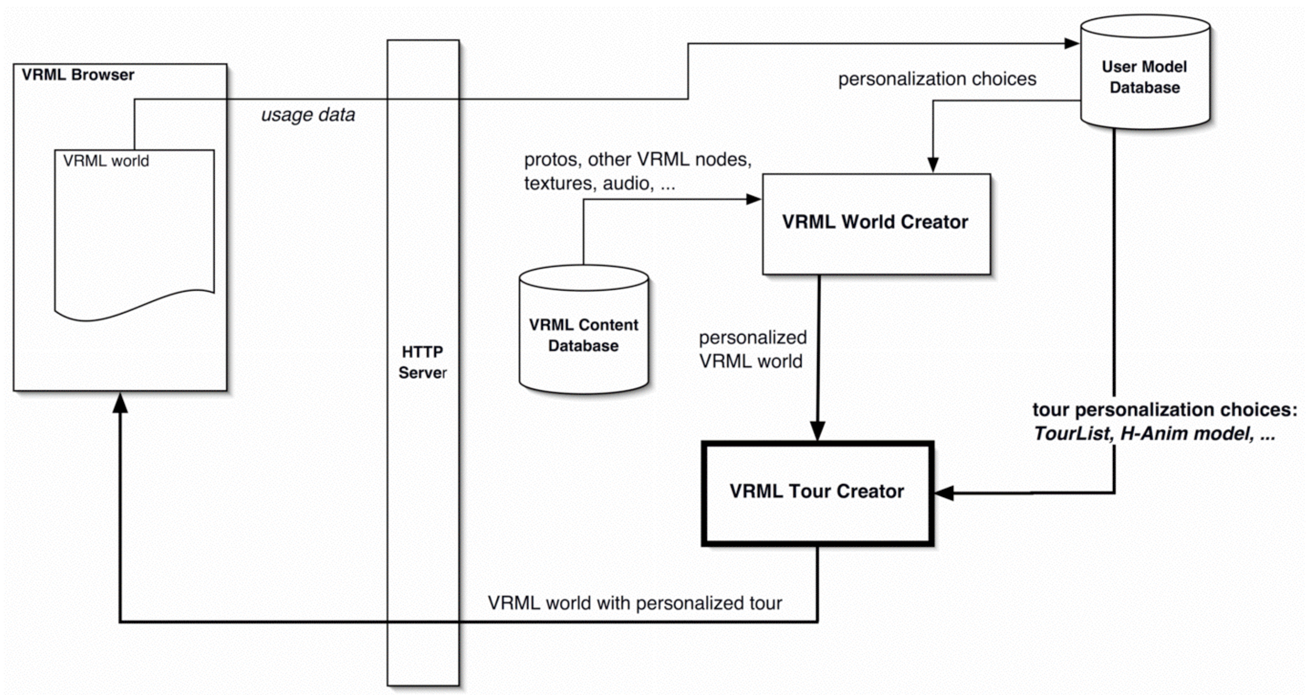


**Figure 10.** The integration of the *VRML World Creator* into the AWE3D architecture.

## 4. DERIVING PERSONALIZED GUIDED TOURS

Automating the process of deriving a guided tour for a VRML world opens up possibilities that go beyond helping the Web3D content creator in building pre-defined tours. Indeed, our tool can be included in server-side programs that dynamically generate different tours for different visitors, taking into account what is known about the user's preferences, interests, and needs. As a result, we can individually *personalize* the guided tour of the Web3D world, i.e. build a *user-adaptive* system [4].

Personalized tours can be more effective than pre-defined ones from the point of view of usability and/or user satisfaction. Simple adaptation examples concern the speed of the virtual guide that could be automatically set according to the estimated/measured user skill at 3D navigation, and the humanoid model could be automatically chosen according to the user age (a young user will probably prefer a more funny, cartoon-like character, while a more rational user will probably prefer a more serious character). More complex examples could involve the choice of which objects/places should be included in the tour and the order of their visit. For example, in a virtual city the system could derive specific tours that match user's preferences towards architecture or history or restaurants or a weighted combination of different interests.

Moreover, if the server records data about the user's visits, there is the possibility of deriving tours that take into account past interactions of each user with the 3D world. For example, if we record whether certain objects/places have not been presented to the user in previous visits (because the user had left the world before seeing them), we could derive a tour that focuses on them.

The approach we proposed can be easily integrated into any architecture that is able to dynamically generate VRML content, such as [5][24]. The architecture we have recently proposed, called AWE3D [5], explicitly supports the recording of usage data inside the VRML world and is thus suited for our personalization purposes. For this reason, we show how our approach is integrated into the AWE3D architecture.

Briefly, the AWE3D architecture can be divided in three main parts [5]:

- ❑ a client-side part, whose purpose is to monitor the user's behavior inside the virtual world by means of properly positioned VRML sensors and to send the usage data to the server;
- ❑ a server-side user model, stored in a database and updated using the data acquired from the client. Its purpose is to maintain an up-to-date description of the user's preferences, interests, and needs;
- ❑ a server-side module (*VRML World Creator*) that exploits the user model to adapt a set of predefined VRML PROTOs by properly instantiating them with field values that best match the user model. The set of instantiated PROTOs, possibly together with other VRML content, constitutes the personalized VRML world, which is sent to the user.

The integration of our tool into AWE3D to automatically derive personalized guided tours is highlighted in bold in Figure 10.

First, we add the parameters describing a tour (i.e. the *TourList* with its components, guide speed, H-Anim model, ...) to the user model. For each user, the value of those parameters can be then assigned differently to create a personalized tour. Once the *VRML World Creator* derives a personalized version of the VRML world, we give it as input to the *VRML Tour Creator*, together with the tour personalization choices. The final VRML world with the personalized tour is then made available to the user.

## 5. EXAMPLE: A VIRTUAL MUSEUM

Recently, we developed a 3D Computer Science museum based on a VRML world representing a data processing center of the 70's, reproducing hardware from the Univac/Sperry 90/30 line.

The virtual museum (partially shown in Figure 3) is organized into two rooms: a terminal room and a data processing room. Each room is filled with terminals, printers, disk drives, computers. The user has the possibility of interacting with the items on display, e.g. opening printers to look inside, extract disks from drives, etc. To further increase the realism of the experience, we included also the needed generic furniture (desks, chairs, …) and proper sounds associated to printers, keyboards, air conditioners, and so on.

We then produced textual explanations to give visitors information about the purpose and functioning of the devices shown in the museum, and the different kinds of professional skills (e.g. data entry, analyst).

When we built the first version of the world, visitors were supposed to explore it by themselves, but informal experiments with users showed that some form of guidance was needed during the visit. For example, some users wanted to know if there was a logical order to be followed, or, after visiting the museum, asked if they had seen everything. Some of them (who were not familiar with VRML worlds) had difficulties in traveling around, interacting with some objects (e.g. open a disk drive and extract the disk) or even did not notice some possibilities of interaction.

The tool proposed in this paper has been used to automatically virtual tours presenting the various kinds of computing equipment, and describing how to interact with them, without the need of changing the original world. For example, Figure 2 shows the path followed by the guide in a short tour with three objects.

By using the AWE3D architecture to dynamically generate VRML content, we had the ability to automatically personalize the derived tours according to what is known about the user. The needed user data are first acquired using an HTML form (filled by the user on its first visit) which asks about age, education, computer science knowledge, and how much time she intends to spend for the visit, and then continuously updated by recording which devices have been presented to the user in the different visits, whether the user looked inside them, and so on. This, for example, gives users the ability to stop a tour, and then, in the next visit, take a tour that includes only objects that have not been previously seen.

First informal tests of the museum with guided tours have shown favorable responses from users. More formal evaluation with users is planned, but has not been carried out yet.

## 6. CONCLUSIONS AND FUTURE WORK

This paper presented an approach to automatically derive and personalize guided tours for VRML worlds, and briefly showed one of its applications (a Web3D virtual museum). The described approach is characterized by two main limitations.

First, it assumes that the areas where the guide can navigate are plain. This can be a problem for those virtual worlds that contain stairs or hills. Consequently, if the world is a building with multiple floors, the proposed tool is able to derive a path for each floor, but it is not able to automatically connect two plans for different floors with suitable guide animations (e.g. climbing the stairs).

Second, the path planning procedure does not take into account possible moving objects or other animated characters. As a result, it is not guaranteed that the guide animation will be collision-free with respect to these obstacles. At present, to solve the problem without extending the tool, one has to consider as an obstacle each cell of the occupancy grid where other moving objects could be.

Future goals for this research are the following. First, we intend to extend the approach to more than one animated character. This would allow both to develop tours with multiple characters, and at the same time to solve one of the above mentioned limitations, by taking into account other moving objects when deriving collision-free paths.

Second, we plan to extend the possible functions and behaviors of the character. This ranges from the possibility of having users give high-level commands to characters (e.g. "tell me more about this object", "I'm not interested in this object") to extending the characters abilities in interacting with the world, e.g. the possibility of traveling into non-plain surfaces.

## 7. ACKNOWLEDGMENTS

Our research activities on Internet technologies are partially supported by the Friuli Venezia Giulia region, under Regional Law 3/98.

Paolo Giangrandi played an important role in the design of the virtual museum application.

The H-Anim humanoid used in the museum application has been implemented by our students Erik Burigo and Gregor de Pellegrin.

We would like to thank one of the anonymous reviewers for the valuable comments which helped us improve the paper.

## 8. REFERENCES

[1] Amato, N., and Yan, W., 1996. A Randomized Roadmap Method for Path and Manipulation Planning. In Proceedings of the IEEE Conference on Robotics and Automation (ICRA 1996), 113-120.

[2] Badler, N. I., Phillips, C. B., and Webber, B. L. 1993. Simulating Humans: Computer Graphics, Animation, and Control. Oxford University Press.

[3] Bandi, S., and Thalmann, D. 2000. Path finding for human motion in virtual environments. Computational Geometry: Theory and Applications, 15, 1-3, 103-127.

[4] Brusilovsky, P. 2001. Adaptive hypermedia. User Modeling and User Adapted Interaction, 11, 1-2, 87-110.

[5] Chittaro, L., and Ranon, R. 2002. Dynamic Generation of Personalized VRML Content: a General Approach and its Application to 3D E-Commerce. In Proceedings of Web3D 2002: 7th International Conference on 3D Web Technology, ACM Press, New York, 145-154.

[6] Chittaro, L., and Scagnetto, I. 2001. Is Semitransparency Useful for Navigating Virtual Environments? In Proceedings of VRST-2001: 8th ACM Symposium on Virtual Reality Software & Technology, ACM Press, New York, 159-166.

[7] Chung, S., and Hahn, J. K. 1999. Animation of Human Walking in Virtual Environments. In Proceedings of Computer Animation 1999, Geneva, Switzerland, 4-16.

[8] Darken, R.P., and Sibert, J.L. 1996. Wayfinding Strategies and Behaviors in Large Virtual Worlds. In Proceedings of CHI '96, ACM Press, New York, 142-149.

[9] Ennis, G., and Maver, T. 1999. Visit VR Glasgow. In Proceedings of ECAADE 1999 - Education for Computer Aided Architectural Design in Europe, http://www.hut.fi/events/ecaade/E2001presentations/15_04_ennis.pdf

[10] Granieri, J. P., Becket, W., Reich, B. D., Crabtree, J., and Badler, N. 1995. Behavioral Control for Real-Time Simulated Human Agents. In Proceedings of the Symposium on Interactive 3D Graphics, ACM Press, New York, 173-180.

[11] Kuffner, J. J. 1998. Goal-Directed Navigation for Animated Characters Using Real-Time Path Planning and Control. In Proceedings of CAPTECH '98: Workshop on Modelling and Motion Capture Techniques for Virtual Environments, Lecture Notes in Artificial Intelligence 1537, Springer-Verlag, Berlin, 171-187.

[12] Latombe, J.C. 1991. Robot Motion Planning. Kluwer Academic Publisher, Boston, MA.

[13] Lengyel J., Reichert M., Donald B. R., and Greenberg D. P. 1990. Real-Time Robot Motion Planning Using Rasterized Computer Graphics Hardware. In Proceedings of SIGGRAPH '90, ACM Press, New York, 327-336.

[14] Lester, J., Converse, S.A., Stone, B. A., and Kahler, S. E. 1997. Animated Pedagogical Agents and Problem-Solving Effectiveness: A Large-Scale Empirical Evaluation. In Proceedings of the Eigth World Conference on Artificial Intelligence in Education, 23-30.

[15] Parallelgraphics, Internet Scene Assembler, http://www.parallelgraphics.com/products/isa

[16] Polge, S. 1999. Unreal Tournament AI and Gameplay for Level Designers. http://unreal.epicgames.com/UT_AI.htm.

[17] Reynolds, C. W. 1999. Steering Behaviors For Autonomous Characters. In Proceedings of Game Developers Conference 1999, Miller Freeman Game Group, San Francisco, 763-782.

[18] Rickel, J., and Lewis Johnson, W. 2000. Task-Oriented Collaboration with Embodied Agents in Virtual Worlds. In J. Cassell, J. Sullivan, and S. Prevost (Eds.), Embodied Conversational Agents. MIT Press, Boston.

[19] Russell, S., and Norvig, P. 1995. Artificial Intelligence: A Modern Approach. Prentice Hall.

[20] Satalich, G.A. 1995. Navigation and Wayfinding in Virtual Reality: Finding the Proper Tools and Cues to Enhance Navigational Awareness. MS Thesis, http://www.hitl.washington.edu/publications/satalich

[21] Thorndyke, P.W., and Hayes-Roth, B. 1982. Differences in Spatial Knowledge Acquired from Maps and Navigation. Cognitive Psychology, 14, 560-589.

[22] van Mulken, S., André, E., and Muller, J. 1998. The Persona Effect: How Substantial is it? In Proceedings of HCI'98, Springer Verlag, Berlin, 53-66.

[23] Vinson, N.G. 1999. Design Guidelines for Landmarks to Support Navigation in Virtual Environments. In Proceedings of CHI '99, ACM Press, New York, 278-284.

[24] Walczak, K., 2002. Building Database Applications of Virtual Reality with X-VRML. In Proceedings of Web3D 2002: 7th International Conference on 3D Web Technology, ACM Press, New York, 111-120.