Navigation by Query in Virtual Worlds

Alex van Ballegooij[1] and Anton Eliëns[2]

[1]Center for Mathematics and Computer Science(CWI)
Data Mining and Knowledge Discovery (INS1)
P.O.Box 94079, 1090GB Amsterdam, The Netherlands
alex.van.ballegooij@cwi.nl

[2]Vrije Universiteit
Department of Mathematics and Computer Science
De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
eliens@cs.vu.nl

Abstract

Web-based 3D virtual environments suffer from the classic problem of users getting 'lost-in-cyberspace'. Apart from getting lost, it is hard for an average user to discover all that a specific world has to offer without spending considerable time exploring that world. In order to remedy this situation we propose *navigation by query*.

The user interface for 3D virtual environments usually allows users to 'walk' around in the virtual world. *Navigation by query* augments this interface by allowing users to navigate a virtual world by means of querying its content.

We present the concept of *navigation by query* and discuss the requirements for an implementation for a VRML based environment. Additionally we illustrate problems an implementation must solve by presenting a prototype implementation.

1 Introduction

In the RIF project we are concerned with the retrieval of information from virtual worlds. [1] [1] We direct our research at finding, indexing and querying information contained in 3D virtual environments. To narrow our field of research we limit the term '*3D virtual environment*' to '*web-based 3D virtual communities*'. More specifically, we are experimenting with online VRML based virtual communities using blaxxun server technology. [5][3]

A 3D virtual community, usually consists of a number of places or sub-worlds where users can walk around a 3D environment and chat with each other. As these virtual worlds become larger and more complex users are more likely to get disorientated. According to Vinson there are three reasons to supply navigational support: "many VEs require the user to navigate, navigation in VEs is difficult and disorientation is upsetting". [18] Apart from the disorientation, it is hard for an average user to discover all that a specific world has to offer without spending considerable time exploring that world. In order to remedy this situation we propose *navigation by query*.

The user interface for 3D virtual environments usually allows users to 'walk' around in the virtual world. *Navigation by query* augments this interface by allowing users to navigate a virtual world by means of querying its content.

Apart from the obvious use of web-based virtual worlds for entertainment, web-based virtual worlds are also used for (e-)commercial, educational and cultural-heritage applications. [11][10][9]

In section 2 we present the RIF virtual community and illustrate how we apply *navigation by query* in this context. Section 3 discusses different types of information in virtual worlds and how this information might be discovered in a VRML world. Navigation and route-planning through a virtual world is discussed in section 4. We present our prototype in section 5 and discuss how the assumptions underlying the prototype can be relaxed in section 6. Section 7 discusses related work. Finally in section 8 we present our conclusions and discuss options for future work.

2 The RIF virtual community

As a test-bed for our Information Retrieval (IR) experiments we are constructing an online virtual community. [4][2] The development of *navigation by query* is done in the context of this community.

Technological environment of the RIF virtual community The RIF virtual community is being developed using blaxxun Community Server technology. The blaxxun Community Server is a state-of-the-art platform for creating web-based virtual environments. Apart from maintaining information about the state of the community and the whereabouts of its visitors, the server also supports the use of VRML in a multi user context.

VRML is a modeling language that allows for a flexible definition of 3D worlds and objects. Apart from 3D geometric models, a VRML world can contain (scripted) programming. Programming in VRML can either be done through

[1] RIF stands for *R*etrieval of *I*nformation in virtual worlds using *F*eature detectors.

the use of 'Script'-nodes or by using the so called External Authoring Interface, in both cases the programming model of VRML is event based.

The multi-user support the blaxxun architecture offers on top of plain VRML is amongst other things, support for avatars and shared-events. An avatar is the virtual embodiment of a user in the virtual world. Shared events are VRML events that are distributed amongst all simultaneous visitors of a specific VRML world.

Application of *navigation by query* in the RIF virtual community Our *navigation by query* prototype is built for one specific VRML world in our virtual community, a model of the third floor of the CWI building. This model has been constructed based upon a 2D map, see figures 1A and 1B.

Figure 2 shows the interface for our *navigation by query* prototype. The interface uses a standard user interface component developed for the RIF virtual community. The window shown in the bottom-left corner of figure 2, takes input that the user types (bottom part) and shows possible answers for the query in a list (top part). This example shows a query for '*coffee machine*' that yields three results. Note that this scene is explicitly annotated.[2] When a list of answers is shown, a user can select one of the answers by clicking with the mouse which instructs the system to take the user to the location in the world that specific answer refers to.

In essence *navigation by query* takes the user from one place to another using a suitable viewpoint transition. An unsuitable viewpoint transition can cause disorientation as figure 3 illustrates. In our case, where we are working with a VRML model of a buildings interior, a 'walking' paradigm is an option. When the movement follows a route that the user could also 'walk' along using the normal user interface, there is much less chance of disorientation.

3 Information in virtual worlds

The basic concept of *navigation by query* is to augment the standard user interface in a virtual environment with a system that has knowledge about the content of the virtual world and lets users find objects and locations through querying.

3.1 Types of information

What kind of information is relevant in virtual worlds? At first sight geometric objects that represent real-life objects in the scene appear to be the most interesting aspect of a world. However, in many cases it is not the geometric data that a VRML world consists of that is interesting for a user, but the interaction facilities provided by that world.

We distinguish between the following forms of potentially interesting information:

- Viewpoints
- Areas of interest
- Objects
- Persons
- Text

Viewpoints can simply be considered as 'interesting views' in the world, just as the real world has its touristic sights. An *area of interest* refers to a spatial area in the world, that has interesting characteristics. For instance a virtual room is an area of interest and the information that could be useful about such an area might be the room number, its inhabitants and other information that is applicable to that specific area. As mentioned, *objects* are the most obvious element of a virtual world to store information about. The multi-user aspect of virtual worlds implies that (virtual) persons other than the user itself can be present in the world at any time. These persons can be virtual representations of other users, but we can also consider the embodiment of an (intelligent) agent as a person. Finally, a VRML world can contain *text*. Text can be present simply as text, for instance the text displayed on billboards. Another kind of text that is present in multi-user virtual worlds is the chat text, people can interact with each other by means of a text-based chat, these virtual conversations are usually the most interesting aspect of these worlds to users.

The time and context in which information is available in virtual worlds is another issue. We distinguish between the following possibilities:

- Static, always present
- Shared, may or may not be present based upon user action
- Dynamic, generated at runtime, for instance by scripts
- Temporal, only present at certain times
- Hidden, available, but only after specific user action

For obvious reasons *static* data is the easiest to deal with. Unfortunately it makes up only a fraction of the interesting parts of a virtual world. As mentioned above, often the interaction opportunities are that which make a world interesting to users. *Shared* objects which are influenced by users are potentially the most interesting category of elements in

[2] For instance all the coffee machines are annotated with 'coffee machine'. This makes the example query posed in figure 2 possible and makes that this simple approach works fine for this proof of concept. In general though, simple keyword matching on annotations can not be expected to work well on annotated (multi-) media data. A discussion on this problem in the context of image retrieval can be found in [12].

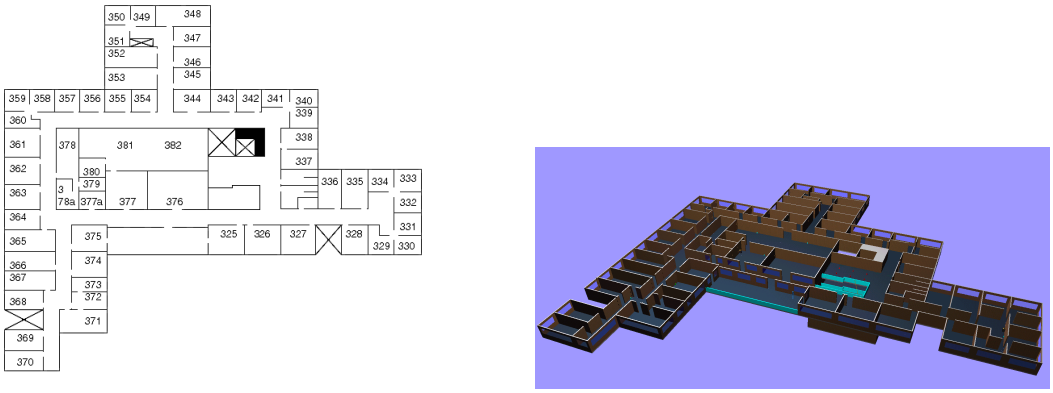

Figure 1:
(A) The 2D map of the 3rd-floor at CWI. (B) The 3rd-floor VRML world.

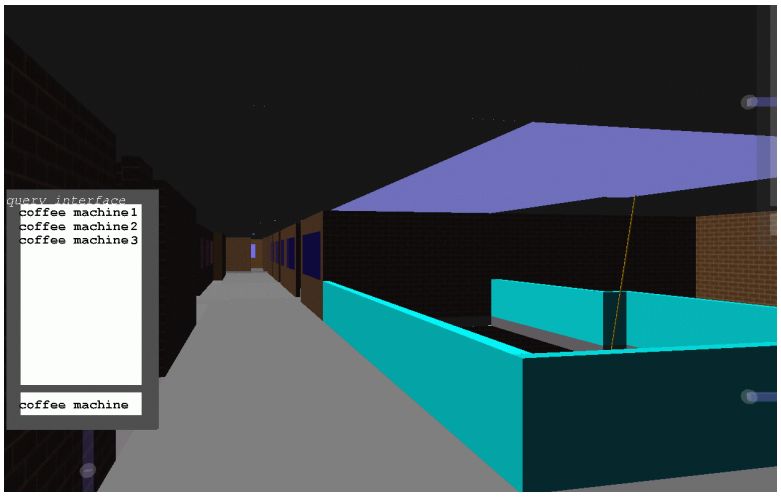

Figure 2: A simple example query resulting in 3 answers.

a virtual world, since category contains the user-user interaction possibilities. Some examples are the 'chat' and the avatar-embodiment of the users themselves.

Temporal data has many of the same problems that *shared* and *dynamic* data have, as these categories must be discovered at query-time and for the largest part cannot be determined beforehand. *Hidden* data might for instance be something that is only visible to a user after pressing some button or some equivalent user action. This is a category that should either not be available through queries at all, because information may be hidden for a valid reason. Or its location should be shown to the user with an indication of the action needed to make it visible, possibly through textual explanation or by showing an animation of that specific action. A 'visual' reward for a user after solving a puzzle is just one example of an element that should not be available through shortcuts such as *navigation by query*.

3.2 Finding information in VRML

In order to find information in a virtual world it is necessary to scan the VRML scene for potentially interesting features. However the nature of multi user VRML worlds presents us with a number of problems.

The time at which to scan a scene There are two options for when to scan a VRML scene for information. One is to interact with the VRML-browser as the scene is active and thus scan the scene runtime. The other option is to download the static VRML file and scan it off-line.

VRML, as many 3D scene description languages, is structured as a so-called scene-graph. It is to be expected that 3D graphics systems developed in the forseable future will be based upon a similar structure, see [6]. The VRML scene-graph is a *directed acyclic graph* and thus does not contain any cycles. The scene-graph consists (among other things) of transformations and geometry definitions. [5] A VRML

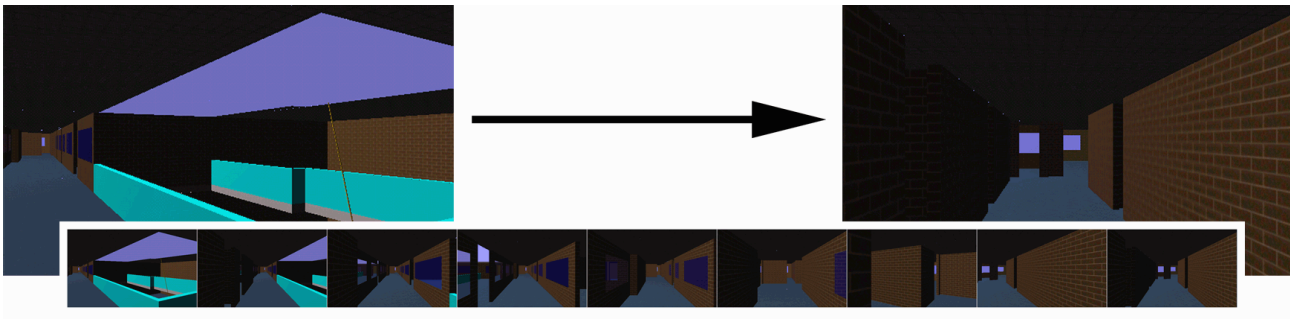

Figure 3: A change in location by jump(top) and a smooth 'walking' transition(bottom).

scene-graph consists of *nodes* that contain fields, these fields may or may not be accessible at run-time. One of the most notable field types that is not accessible runtime are the fields of *indexed face sets*, basically the geometry definition of free-form shapes. This is why a runtime-only scanning is not sufficient.

As indicated in section 3.1 some categories of information can only be determined at runtime so off-line-only scanning is not sufficient as well. At least a combination of both approaches is needed.

Scanning for information When scanning a scene-graph there are a number of techniques that might be applied to detect 'interesting data'. There are features of a scene that are easily deduced from the scene-graph, such as:

- Annotations
- Node-types
- Textual content

One possibility is to assume, as we did in our example, that a scene is annotated and simply scan the entire hierarchy for annotation nodes. Another option is to recognize that certain interesting objects are always contained in special types of nodes, for instance in the blaxxun architecture avatars are always contained in an 'Avatar' node. To allow the use of text in a world, VRML contains a so-called 'Text' node, 'Text' nodes directly contain the ASCII strings that their geometric shapes represent. Additionally, nodes in a VRML file can be tagged by defining a name for that node by means of a 'DEF' command. This node-name may describe what the node represents and is thus potentially useful.

Furthermore the properties of the geometric data that makes up the scene can be examined more thoroughly to attempt to determine high-level meaning based on low-level features. Unfortunately such techniques usually rely on complex analysis and heuristics. Elements to consider for analysis could be:

- Materials
- Textures
- Geometric shapes

Material definitions, that indicate how certain parts of the world should look, could be used as an indication of what the geometry that the material is applied to represents. As an extension to the normal color properties that can be defined through the use of materials VRML allows the use of *texture-maps*. Texture maps are 2D images that are projected onto a polygon in 3D space. Texture maps are used in two different situations. Either as stand in for a shape that is too complex to be effectively represented as 3D geometry, for instance a tree which would normally contain too many branches and leaves for an average capacity rendering system. Or as an enriched way to assign a color to a polygon, where the texture-map is an image of the structure of a material, for instance bricks in a brick wall. [3] Analyzing the shape and spatial layout of geometry in a world is also an option, but determining meaning from a shape is a complex analysis problem.

4 Navigating the virtual world

Once a (possible) answer to a query has been found, the concept of *navigation by query* is that the system takes care of navigating the user from its current location, through the world, to the location of the answer. The design of motion control techniques raises some issues [19].

4.1 The navigation metaphor

It is important to provide a suitable navigation metaphor for a world to the user. As indicated in section 2, a sudden

[3] Both types of texture-maps are used for the same reason, to limit the number of polygons needed to model an object. In the case of 'object-stand-in' this is obvious. For 'material-textures' this is because the projection of a texture-map onto a single polygon approximates a structure that would otherwise need to be modeled using many separate polygons. However, the implied 'meaning' of both types of texture is different.

jump from one viewpoint to another can cause disorientation. Smoothly traveling through the world gives the user more awareness of other sights the world has to offer, eventually perhaps learning the user how to find certain object without querying. The idea is based on the real-world analog of a person walking along and showing how to reach the destination. [21]

In wide open space a simple viewpoint interpolation might be sufficient, while such a direct transition from one viewpoint to another can cause undesirable effects in more dense environments. To avoid flying through walls, another approach for more dense outdoor scenes, one might think of a fly-over approach, where the user is first lifted above all obstacles and then simply moved to the target location by flying. But for highly dense environments, such as indoor scenes, not passing through solid objects is a problem. The only sensible approach in these cases is to follow a walking metaphor, where the navigation is restricted to the same movements that would be possible using the normal navigational interface provided by the browser.

4.2 Determining a suitable viewpoint

In order to present an answer to the user we need to determine a viewpoint in the virtual world where the user can see the answer. There are a number of information-types that require a different approach:

- Viewpoints
- Areas of interest
- Objects and persons

Viewpoints directly represent a view of the world, thus in this case the viewpoint itself is already given. For *areas of interest* we should generate a viewpoint inside the area looking in an arbitrary direction, the only thing that has to be taken into account is that the location itself is not contained in a solid object. *Objects* and *persons* are similar, they are both solid objects that represent something of interest. In this case we need to generate a viewpoint that is near the specified object and faces it, so the user can see the object. The problem is that we need to make sure that the line of sight of the user is not obstructed by any other object.

As explained in section 3.2, it is not possible to obtain detailed geometric data from a scene at runtime. This is why it is impossible to construct a sufficiently detailed model of the virtual world containing both all static and all dynamically generated geometry that can serve as a reference to check whether a line of sight is obstructed or not. Without explicit VRML browser support all that can be done is to build a reference model based upon the static VRML file. Dynamically generated geometry can either be ignored, or certain spatial properties have to be assumed. Luckily the blaxxun VRML browser offers built in support to check the intersection of a line with the geometry present in a scene. This feature, intended to allow users to 'pick' elements of a scene with the mouse, can be used to check the obstruction of a line of sight.

4.3 Route planning

Even for dense interior environments route-planning itself is not too hard a problem once the navigational metaphor and layout of the world are known. A convenient way to store a map of a 3D world is to use a graph representation. A shortest path algorithm can be applied to plan a route through such a graph. The problem in planning a route through a dense virtual world is to know the layout of the world.

In some cases the 3D world is modeled after a 2D map and directly analyzing the 2D map is far easier than reconstructing such a map from the 3D model. But just as annotation is not widely available, we cannot simply assume that such maps are available.

Inspiration may come from the gaming world. Notably the computer controlled characters for the Quake-2 multiplayer game variants, called bots. [13] One option to construct maps of 3D worlds is to analyze the geometric content of a world and build a map that way. This approach is taken by the SoarBot, a bot implementation based upon the SOAR agent architecture. [14][15] These bots build a map of the game world in a preprocessing stage, this is however a more complex task for more generic worlds. Another computationally less complex method is to monitor user navigation and build maps by analyzing routes that real users have followed through the world in the past. This second approach is used by for instance the Eraser bot and is very effective given the assumption that users 'donate' routes to all relevant locations in a world that way.[16]

4.4 Answering a query

Answering a query is done by determining a suitable viewpoint for the object of interest and planning a route through the scene from the current location to that viewpoint following a suitable navigation metaphor.

Once a route has been planned, navigating such a route is merely a case of following a predefined path. This is accomplished by generating a series of VRML viewpoint animations, alternatively application of more high level path following logic may be useful. [17]

5 Prototype implementation

As a proof of concept and a vehicle for further research we have constructed a prototype implementation that offers *navigation by query* functionality. As mentioned in section

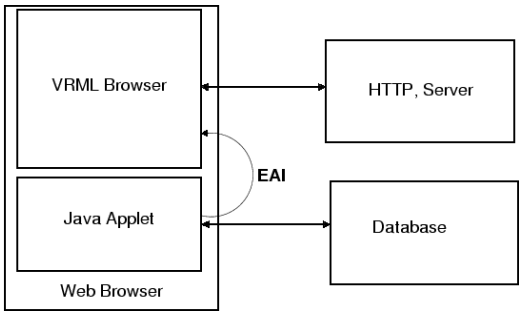

Figure 4: The structure of the prototype.

2 the prototype is built around a single virtual world that is a part of the RIF online virtual community.

5.1 Assumptions

Before discussing the requirements for the prototype we should note a number of assumptions we made about the environment of the prototype:

1. the environment is explicitly annotated
2. we have a map to use in the route-planning and navigation tasks
3. we know how certain objects are annotated and can therefore use simple keyword matching to find exactly those objects that we want to find

Since annotation is present in our test environment, our system only has to scan the VRML scene-graph for meta-information. Furthermore as this prototype is a proof of concept we are satisfied with retrieval capabilities for static data. One dynamic feature we want to support however is the ability to find other users.

5.2 Requirements

In addition to the assumptions, we formulated a number of basic requirements for the prototype. Primarily because we wanted to provide possibilities for further extension of the prototype as well as integration into our online community.

The main requirement is that meta-data storage and querying should be managed in the form of a database. Firstly a database is needed to support incremental information gathering. And secondly once filled with sufficient information, the database can also serve as a knowledge base on virtual worlds for other applications. Additionally a separate database allows for the storage and retrieval of information not otherwise available at run-time.

A subjective requirement is that we want the *navigation by query* interface to use one of our standard 3D interface widgets. Although virtual environments using standard desktop technology are not immersive we feel that care must be taken to involve the user as much as possible into the world. Because of this, we do not want to disrupt the interaction with the 3D environment by (re-)introducing separate 2D interface elements such as Java windows.

5.3 Prototype structure

The prototype structure is illustrated in figure 4. Most of the functionality is concentrated in this Java-applet that communicates with the users VRML browser through the external authoring interface(EAI). [8] [7]

The prototype is built around a Java-applet that connects runtime to the VRML browser, because this is the only straightforward way to be able to retrieve dynamic data. The applet takes care of scanning the scene-graph for meta-information, a connection to a database is used to store this information. Additionally the applet also monitors the presence and location of avatars in the world to allow users to find other users through querying.

The database contains meta-information, which comes from a variety of source. Primarily the applet stores all information it has discovered by scanning the scene-graph and information that was added directly into the database. Additionally there may be any number of (off-line) tools that extract and store information. The tool that generated the map used for navigation is one example of such a tool. As the applet does not contain all meta-information stored on a virtual world, queries are answered by the database. The results from a query are interpreted and used by the applet to steer the navigation through the virtual world.

As figure 2 illustrates, the user interface for the applet is inserted into the actual VRML scene itself. Additionally a number of VRML-scripts are inserted to VRML browser to support the applet with functionality that is not possible to achieve directly through the EAI.

Route planning is based upon a map that is encoded as a graph structure that we constructed based upon the available 2D map and is stored in the database. Firstly a path in the graph is calculated from the graph-node nearest to the user to the graph-node nearest to the destination, this can be determined since the graph-nodes refer to actual locations in 3D space. Since the graph-nodes directly refer to a location in the scene following the route can be accomplished by generating a viewpoint animation through the locations referred to the graph-nodes in the path from source to destination. This technique is used because we have chosen to use a 'walking' paradigm for navigation, as indicated in section 4 other paradigms will require other solutions.

6 Relaxation of the assumptions

As explained in section 5.1 the current prototype is based on a number of assumptions. These assumptions restrict

the usability of the current prototype. By relaxing these assumptions it may be possible to use the prototype on a wider variety of virtual environments.

The first assumption is that the world is sufficiently annotated to facilitate searching. Since in general annotation cannot be expected to be supplied by the author(s) of virtual wolds this is a serious restriction. Thus we should either allow the incremental addition of annotations by users or rely on automatic annotation.

Research on user-annotation shows that users are generally not reliable sources for annotations. However, given that even bad annotation at least provides some information to work with, user-annotation appears to be the most viable option to solve the problem of annotation-less worlds seems to be user-annotation.
⟶ [**Need a reference for this ?**]

The second assumption is that a map is available for navigational purposes. As indicated in section 4, different kinds of environments will need different kinds of navigational metaphors. Maps will not be needed for certain forms of navigation, such as flying. For cases in which a map is needed we indicated in section 4.3 that both automatic exploration of a world by an agent and map construction based upon past real-user navigation have been used with success in the context of games. Both techniques could be extended to more generic virtual worlds. Automatic exploration of a world however cannot be expected to work equally well in all cases, since not all worlds need to conform to the same constraints. Because of this we feel that the use of user navigation to construct a map of possible routes through a world is a more generic solution.

Finally the third assumption was that simple keyword-matching was enough to formulate queries. Given that annotations are subjective and word-choice is personal this is generally not true.
⟶ [**Need a reference for this ?**]
Standard text retrieval techniques, such as stemming and query-expansion, can be incorporated into the database engine and possibly solve some of the problems related to standard keyword-matching.
⟶ [**Need a reference for this ?**]

Another approach might be to avoid these problems by introducing Content Based Retrieval (CBR) functionality. We briefly experimented with 3D shape-matching and although the development of a suitable metric is difficult, we feel that query-methods other than text may be possible. Query by example may become an option given a suitable similarity function.

7 Related work

Our approach to developing *navigation by query* has primarily been driven from IR point of view. If we address the issue as an user interface problem, there is a wealth of work that might contribute to improvements of the prototype.

From a user-interface perspective we can identify two important problems. Firstly there is the method in which a user can formulate a query, secondly the way in which a query-result is visualized.

As described in section 5 we use a 2D interface widget embedded in the virtual world to allow users to interact with the system. Several researchers have proposed the use of 2D interaction techniques in 3D environments, since it seems to be the easiest way to provide effective interaction possibilities for users. Unfortunately, little study has been done to determine the effectiveness of such interfaces. [20]

Another approach to allow user interaction is to use a speech interface. Speech and natural language interfaces for virtual reality might solve the problem of how to enter textual queries in an immersive environment. [22]

Presenting the query results to the user is in our case a simple list of answers, once again displayed on the 2D interface widget. Once the user selects one single answer from the answer set the system takes care of navigation and takes the user to the location in the world that that answer refers to.

A more elegant method, especially in the context of immersive environments might be the use of 'intelligent' agents. The functionality of the *navigation by query* system can easily be incorporated into an avatar-embodied agent who takes the user on a guided tour around the world visiting possibly relevant location until either the user has found what he is looking for or there are no more answers in the answer set. This is one of the things currently under investigation in the WASP project.
⟶ [**'Plug' WASP ?**]

Finally there is a partial problem with the realization of suitable viewpoints to show the result of a query. The approach we use now, simply store the viewpoint in the database is a viable one if we decide to use user-annotation. But if we want to pursue the automatic-annotation goal we will need to generate these viewpoints automatically as well. Quite possibly the work on automatic cinematographic camera positioning in VEs can serve as inspiration to solve this problem more elegantly. [23][24]

8 Conclusions and future work

Our experiments indicate that *navigation by query* is feasible and may help users to find locations and objects, that would otherwise be hard to find without prior knowledge of

the layout of the world. The most important issue in the effectiveness of *navigation by query* is using the proper navigational paradigm, an unsuitable navigation only helps users to get lost.

However it is hard to obtain knowledge about a world without explicit annotation. Annotation can either be added by the scene author or it could potentially be incrementally added and rated by users. Also, determining suitable navigational routes without an explicitly defined map is hard. Maps can be supplied by the author of a scene, or it might be build by analyzing the navigational history of real users.

Future work One of the more challenging options we want to look into is to develop a method to build maps dynamically based on geometric information and/or real-user navigation. This would greatly improve the usability of our system on existing worlds because maps are very likely not to be available for existing scenes.

Also the absence of annotation in existing worlds creates the need to examine the possibility for automatic annotation. However as we expect this to be an unsolvable problem, we might first experiment with user annotation and rating. Another approach we are currently examining is to apply a form of 'query by example' to realize a Content Based Retrieval (CBR) prototype.

The way in which we allow querying needs to be improved, as basic boolean keyword matching is clearly insufficient. Implementation of more 'intelligent' query solving techniques might be interesting.

Finally, given sufficient dynamic map-building and annotation capabilities, we would like to make our implementation more generic so that we can apply it to any (VRML based) virtual world on the Internet.

References

[1] A. Eliëns. RIF Project page. *http://www.cs.vu.nl/∼eliens/projects/rif/*. 1999.

[2] A. Eliëns A.R. van Ballegooij. RIF Virtual Community. *http://www.cwi.nl/∼blaxxun*. 1999.

[3] Blaxxun Interactive. Blaxxun Interactive website. *http://www.blaxxun.com/*. 1999.

[4] B. Ribeiro-Neto R. Baeza-Yates. *Modern Information Retrieval*. ISBN 0-201-39829-X. Addison-Wesley 1999.

[5] VRML97 ISO/IEC 14772-1:1997. The Virtual Reality Modeling Language. *http://www.web3d.org/technicalinfo/specifications/vrml97/index.htm*. 1997.

[6] H. Sowizral. Scenegraphs in the new millenium. *IEEE, Computer Graphics and Applications*. vol 20/1, Jan/Feb 2000, pg 56.

[7] EAI Working Group. VRML External Authoring Interface. *http://www.web3d.org/WorkingGroups/vrml-eai/Specification/*. 1999.

[8] Sun Microsystems. Java website. *http://java.sun.com/*. 2000.

[9] A. Eliëns L. Rutledge, A.R. van Ballegooij. Virtual Context, relating paintings to their subject. Presented at the Culture Track of WWW9. *http://www.cwi.nl/∼lloyd/Papers/WWW9-Culture/*. Tuesday, May 16th, 2000.

[10] Learnatix. Learnatix website. *http://www.learnatix.de/*. 1999.

[11] MCC Swisscom, Fokus/Hexmac. VR-Shop. *http://www.vr-shop.iao.fhg.de/*. 1999.

[12] P.G.B Enser. Progress in documentation: pictoral information retrieval. Journal of documentation vol 51, 1995, pg 126–170.

[13] iD software. Quake 2. *http://www.idsoftware.com/quake2/index.html*. 1999.

[14] J.E. Laird A. Newell P.S. Rosenbloom SOAR: An Architecture for General Intelligence. Artificial Intelligence vol 33/1, 1987, pg 1–64.

[15] S. Houchard J. Hartford. Soarbot Interface System. *http://www.eecs.umich.edu/∼soarbot*. 1998.

[16] Impact Develpment Team. Impact Develpment Team Website. *http://impact.frag.com/*. 1999.

[17] Craig W. Reynolds. Steering Behaviors for Autonomous Characters. Presented at the 1999 Game Developers Conference. *http://www.red3d.com/cwr/steer/gdc99/*. March 19, 1999.

[18] N.G. Vinson Design Guidelines for Landmarks to Support Navigation in Virtual Environments. Proceedings, ACM CHI'99 Conference on Human Factors in Computing Systems 1999, pg 278-285.

[19] L.F. Hodges D.A. Bowman, D. Koller. Travel in Immersive Virtual Environments: An Evaluation of Viewpoint Motion Control Techniques. IEEE, Proceedings, 1997 Virtual Reality Annual International Symposium.

[20] R.W. Lindeman J.L. Sibert J.K. Hahn Towards Usable VR: An Empirical Study of User Interfaces for Immersive Virtual Environments. Proceedings, ACM CHI'99 Conference on Human Factors in Computing Systems 1999, pg 64-71.

[21] J.L. Sibert R.P. Darken. A Toolset for Navigation in Virtual Environments. Proceedings, ACM User Interface Software & Technology. 1993, pg 157-165.

[22] S. McGlashan T. Axling Talking to Agents in Virtual Worlds ??? ???

[23] W.H. Bares J.C. Lester Cinematographic User Models for Automated Realtime Camera Control in Dynamic 3D Environments Proceedings, User Modeling: Sixth International Conference, UM97 1997

[24] B. Tomlinson B. Blumberg D. Nain Expressive Autonomous Cinematography for Interactive Virtual Environments ??? ???