3D Annotations in Immersive Environments

Francisco J. Pinzón L.

Universidad de los Andes

Departamento de Ingenería de Sistemas y Computación

Ecole Nationale Superieure d'Arts et Métiers. Paris Tech.

Institut Image

July 2011

0

Brief

The requirement of a 3D Annotation System for Immersive Environments without concurrent telecoperation capabilities was a basic and unfulfilled need present at the Institute Image when I arrived. For this reason I was given the task of developing such system as my undergraduate thesis. The use of the immersive facilities present at the Institute in the provided solution was a requirement. The provided application succeed in the fulfillment of the original requirement. The following document describes this solution as well as the considerations taken in account for developing it.

1 Introduction

For a long time now I've wanted to visit France. Actually, thinking about it, I had that desire before I started to study Systems and Computer Engineering at Universidad de los Andes. Someday I just asked Tiberio about the possibilities of making my thesis abroad. He gave me certain options and one of them was the Institut Image at Chalon-sur-Saône (in which I am writing this Introduction). Since then a few months passed before the bureaucratic nightmare started.

From the university demanding information they already had, and for which I had to pay a considerable amount of money; to the French Embassy and their Kafkesque process to give me a 5 minutes appointment after weeks of collecting innumerable documents, without taking in account my travel to Brazil to participate in Interactivos? 2010 BH, Christmas and New Year's eve, I thought I wouldn't be able to fly to France in time. At the end everything came out alright.

The subject of this work was given to me by Frédéric two weeks after my arrival at Chalon. In standard circumstances I prefer my projects to be product of my ideas and not someone else's, but the opportunity to come here for a whole semester easily took over that. Looking backwards, this project has been a great learning opportunity, both personal and professionally.

The instructions for developing my project were simple and precise: A system for making annotations on 3D models using the cave facilities present at the institute. The idea is to be able to comment 3D model using immersive facilities. The annotations would be simply a plain text comment, with author, priority and its creation date attached.

So, as you can see, rather than the traditional way to choosing a thesis I was given a determinate topic and a narrow and precise expected solution. For this reason, in this project, I will work on the possibilities, contexts in which it might be used, constraints and potential scopes of the given solution and not on the process to propose the problem, as there was none.

Some of the basic design choices of the solution were given by the facilities present at the Institute. Windows XP running in the cave computers, infrared tracking cameras, passive stereo projections and the Android powered tablet are some examples of these 'design choices'. The software was developed using Microsoft Visual C++ for the part running in the servers and the Eclipse IDE for the tablet application; thus C++ and Java were used to build the solution. OpenSceneGraph and VRPN were also used.

The project was successfully developed and delivered within the expected time. Though there were significant changes in it's design and there are functionalities not fully implemented, it is still a usable product capable of solving the needs that gave place to it. There is a considerable amount of future work that could be done for improving the project, in case anyone is interested. The core of the application is almost fully developed but the improvements on other aspects would impact deeply the user experience. All the code for this project, and the latex files used to generate this document can be found at *github.com/fpinzn/Thesis*.

The structure of this work was given by the standard format used in the Department of Systems and Computer Engineering of the Universidad de los Andes. For this reason, and like with any other document written following such a generic structure, this document features several times the same principles, arguments and ideas with slight changes in redaction, focus and presentation. The reader should go directly to the sections of interest.

First of all I thank my mother and my sister for supporting me, I know it has been hard. I also thank Tiberio and Frédéric for supervising this work, and giving me this opportunity. David for his friendship, the discussions, links and quality content.

2

General Description

2.1 Goals

The objective of the project is to create a 3D Annotation System for Immersive Environments without concurrent tele-cooperation capabilities. The project is developed for fulfill the need of such system at the facilities of the Institut Image and at the COLIVRI Laboratory at the Universidad de los Andes; without being restricted to these two contexts.

The final solution must be capable of performing this task in models of a variety of 3D formats, associating with each annotation its type and urgency to be managed. On the other hand the system meet the ever-present needs of scalability, flexibility, performance and specially usability. The project must be developed using C++ and OpenSceneGraph and must run in Windows XP.

2.2 Prior Work

The most similar prior work I found while making the research for this one is presented by Kadobayaby et al. in [10]. It's similarity consists in the use of annotations in 3D immersive environments to deal with the inherent complexity of 3D models. The work is nevertheless, restricted by it's use of Croquet. The Croquet Project was a very promising project, now fallen into disregard and oblivion because of the sudden drop of its development. It also provides interesting ideas behind user defined paths inside the VR space that will be addressed later.

This work makes a good case behind the reasons of using a 3D annotation system to move beyond the "eye-candy" phase of the technology. It also provides a nice definition of annotation, as follows: "Annotations may be thought of as author- attributable content placed within a Croquet scene in association with (or reference to) a particular element of that scene.". On the other hand it also provides a rather bizarre visualization of these annotations, "...we have developed basic conventions by which annotations can be created independent of the form of media. We allow objects to become annotation even if the object was not originally designed to be an annotation". These ideas might be interesting but it's complexity make them hard to work well in a broad context.

The work by *Sonnet et al.* [13] integrates a 3d probe with annotations, giving meaningful information and an exploded view of the 3D model depending of the position of the 3D probe. It uses an automated algorithm to select the best position to display the annotations given both the position of the probe and the centroid of the objet of interest. It also uses handy techniques to navigate the 3D space and isolate important parts of the 3D model. On the other hand it uses complex ways of annotating objects and rare shapes to work as containers of the annotations.

The work by *Duval et al.* [5] works the possibilities around collaborative explorations of 3D scientific data. The most remarkable parts of it concern the definition of annotations as 3D viewpoints rather than textual information freely located in the space as shown in [10] and [13]. This idea will be used later in the development of this project[1]. It also uses a minimap to locate the users in the scene similar to the ones featured in the game industry.

The work on annotating and sketching on 3D models via a web interface developed by *Jung et al.* [9] draws interesting lines between the needs behind a post-it metaphor driven annotation system and a more free 3D sketching system. It also raises interesting points around the possible advantages of an asynchronous collaborative system and it's support for a semantic history of the evolution of the model, versus a real-time voice based discussion system.

The Chameleon System featured in [3] and explained deeply in [14] explores a similar window metaphor to the one used in this project to allow the user to select his objects of interest to associate to an annotation. However, there are important differences between their work and the one presented in this project, being the most important that the tablet used here is not attached to a mechanical arm to track the movement; it uses instead an infrared tracking system.

Other sources consulted did not work as prior work but as excellent ref-

[1] It actually ended being the selected annotation paradigm.

erences for a more specific part of the project. The work by *Gonçalves et al.* in their software Tag Around [7], concerned with correct tagging and meta-data enriching of photos, proposes an interesting approach to usability tests focused in non conventional interaction. The work by *Burton et al.*[4] and by *Bernheim et al.*[2] presents a really insightful conclusion around collaborative annotation systems, which as summarized in [9] should "be unobtrusive but accessible, inform without overwhelming, separate higher and lower priority information for different actors at different times", even when their work concerns the annotation of text documents the principles are extrapolable.

2.3 The Problem

The root of the problem is the inherent complexity of 3D models. The need to reduce these inherent complexity and expand the models through the use of textual information is the primary concern that gave light to this project. Textual information of a general kind would serve to cast clarity and allow a deeper understanding of the modeled scene. As said in [10] 'To be generally useful for collaborative research and learning, immersive virtual 3D spaces must include intuitive content creation and annotation tools'.

The uses of such technology are rather diverse. From enabling educational experiences not possible before, to reducing the time to market new products, via reducing the time to design in a diverse range of manufacturing companies; passing through semantically augmenting architectonic heritage, the potential uses are innumerable.

These potential uses and its associated potential shareholders are still restricted to the access of a cave-like facility given it's high costs. On the other hand, a personal computer version could be used by anyone with access to a standard computer. However, this scenario provides a rather poor experience and similar solutions are already available in the market.

The context of the problem is nevertheless a rather specific one, the facilities of the Institut Image, which devotes most of its attention to educational and research ends. These facilities count with two immersive systems relevant to this work. The first one is the MoVe, a cave-like system with three faces and the floor projecting passive-stereo images and a IR tracking system with 4 cameras located on the top of each corner. The second immersive system is the Spidar, a single screen projected as well with passive-stereo images and 4 IR cameras located on the four corners of its single screen and in which the application run at the end.

As part of the problem definition the requirements of using the C++ programming language, along with OpenSceneGraph, running in various Windows XP machines, were given.

3

Design and Specifications

3.1 The Problem's Definition

Even thought in previous sections the problem to solve has been discussed considerably, this section focuses in provide a short and precise definition of it.

This project proposes a solution to Annotate 3D models in a immersive cave-like system using an android tablet to interact with the scene. The part of the software that will be running in the cave servers and is aimed to render the scene will be developed using C++ and OpenSceneGraph. The part of the software that runs in the tablet will be developed using Java.

The annotations must be saved using an XML archive, so it can be easily read by any other software. The annotations must show the information in the best displayable way, maximizing readability and relevance. This annotations must be textual comments with priority and author.

The solution also may be able to run in a standard PC running Windows

XP.

3.2 Specifications

3.2.1 Functional and Non Functional Requirements

Functional Requirement 1 (FR1)
The user must be able to create annotations associated to the part of interest in the model.
Input:
The part of interest, the content of the annotation, its priority and its author.
Output:
The annotation is successfully linked to the model.

Functional Requirement 2 (FR2)
The user must be able to retrieve the annotations made in the model.
Input:
None.
Output:
The user can see a list of the annotations associated with de model.

Functional Requirement 3 (FR3)
The user must be able to delete annotations in the model.
Input:
The annotation the user wishes to delete.
Output:
The annotation is deleted.

Functional Requirement 4 (FR4)
The user must save the annotations made in the model to a file.
Input:
The annotations.
Output:
The annotations are saved to an external XML file.

Functional Requirement 5 (FR5)
The application must load automatically the annotations made in the model from a file.
Input:
The archive containing the annotations.
Output:
The user is now capable of seeing the annotations previously made.

Non Functional Requirement 1 (NFR1)
The application must work in the cave-like systems available at the facilities of the Institut Image.
Origin:
Context of the development of the project.
Level of Impact:
Sets the bases of the architecture of the system.

Non Functional Requirement 2 (NFR2)
The application should work as well in a standard PC running Windows XP.
Origin:
The need to use the system in other locations with different facilities.
Level of Impact:
Creates another branch of development inside the project.

Non Functional Requirement 3 (NFR3)
The part of the application running in the cave servers must be developed using C++ and OpenSceneGraph.
Origin:
These default technologies used for high demanding applications and the *de-facto* standard inside the laboratories related to the project.
Level of Impact:
Sets the technologies to be used in the development.

Non Functional Requirement 4 (NFR4)
The part of the application running in the cave servers must run in Windows XP.
Origin:
The OS of the cave is Windows XP.
Level of Impact:
Sets the environment of development. Given the use of OpenSceneGraph as the base layer the impact is not as deep as it might have been.

Non Functional Requirement 5 (NFR5)
The application must save the annotations in XML format.
Origin:
Asked by Frédéric, taking in account the advantages of the highly standardized format.
Level of Impact:
Creates another module in the software to save and load annotations from an XML file.

Non Functional Requirement 6 (NFR6)
The application will make use of the VRPN technology to interchange information between devices.
Origin:
Enhances the modifiability of the system and is a standard practice in both laboratories.
Level of Impact:
Sets architectural constraints.

Initially the parts of interest were exclusively polygons, I thought during the last stage of the development to broad the definition and include any point of view with a marquee highlighting the important sector, and drop the polygon selection due to it's hard of use and impact in the application workflow. The creation and associations of the marquees wasn't developed, so the application is capable of commenting points of view as the work presented by Duval [5].

3.2.2 Acceptance Level

The minimal accepted solution would be the one who fulfills all the requirements (functional and nonfunctional) mentioned before, except the Non Functional Requirement 2. This is, a solution that runs and is fully functional in cave like systems but doesn't work in a standard PC running windows XP.

The optimal solution in the available time is then, the one described below on the requirements. The description of better global solutions will be reserved for the *Future Work* part of this document as they involve a significantly major development work, timeline functionalities upon the models and their textual information, that are beyond the reach of this project.

An incomplete solution would be developed in the case that the calendar designed for developing the solution falls short due to unforeseen serious technical difficulties during the implementation. In this scenario the part of the project that would suffer the most, is without doubt the related to the reviewing of the annotations. This, because the first part that will be implemented is the one concerned with creating the annotations given it's need to afterwards see the annotations made.[1]

On the other hand, another part of the project that could present non-optimal quality is the one concerned with the user experience. The project is thought to be a clean, minimal but satisfying solution to the problem it aims to solve, nevertheless a huge effort is being invested in improving the

[1] Although the application was not fully developed, it's still an usable and reliable solution.

user experience and the quality attributes related to it.

Taking in account the ideas presented in the famous book by the creators of the successful company 37signals [6] and the Apple Development Guidelines [1], a nice user experience involves reducing the span of functional requirements to the minimum necessary to successfully solve the problem, while focusing in less evident goods of the software. In the particular case of the project, the rendering power, the net throughput and the responsivity of the front-end (the Android powered tablet) are the most relevant factors in providing a satisfactory user experience. Is yet to be seen how easy or hard is to tune them right.

3.3 Restrictions

The most significant restrictions at the time of developing the project is, with complete certainty, both the inexperience of the author with the technologies used and the time to deliver it. There wasn't any other technological restriction which could affect the project. Given the nature of the project there weren't any economic, environmental, social, political, ethical, legal, health or security related restrictions [2].

On the other hand, a significant restriction to broad the contexts, in which the application is a relevant solution to a similar problem, is the amount of money needed to build similar immersive facilities to the ones present at the Institut Image or the COLIVRI laboratory at the Universidad de los Andes.

[2] This remark is made taking into account the ABET directives for such documents.

4

Design Process

4.1 Information Gathering

The information sources used to develop the design and the architectural choices where basically the online databases provided by the Universidad de los Andes library site. Being the most used and consulted the ACM Digital Library and IEEE Computer Society Digital Library online libraries . Given the precise instructions to develop the project there wasn't a particular need to search information about the origin of the problem to solve, it was just a need present at the Institute Image, as discussed in previous sections of this document.

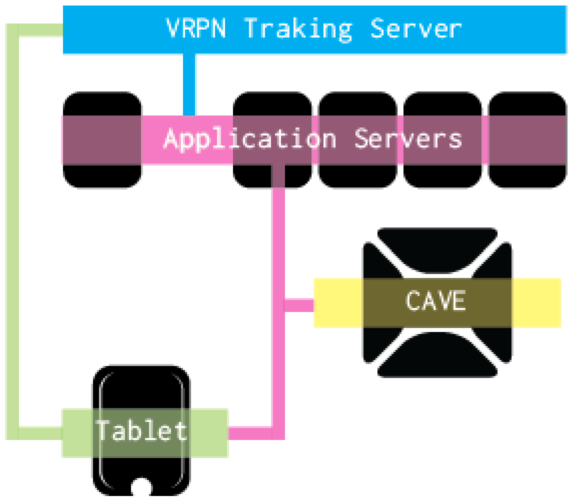
The reviosion of the bibliography and references used in articles describing similar projects [10], [13], [5] and [9] pointed other useful sources of information; even when they weren't so closely related to the project's subject. These references were used basically to evaluate potential design alternatives, taking in account the reduced number of iterations possible to develop the application, such as [2], [8]. Other sources of information that influenced the work were known by the author before, such as [1], [6] and [11].

4.2 Design Alternatives

The goal of this section is to explain the design alternatives and the reasons behind the selection of the alternative that was ultimately implemented. Even in the case that there weren't alternatives at all, like in the system architecture, this section will still explain the pros and cons of the default design.

4.2.1 System Architecture

The system running the application is a three-layered system, as can bee seen in the image below . The top level, the VRPN Tracking servers receives tracking information from the tablet VRPN device, which design will be discussed later, and from the IR tracking cameras following the flat passive markers attached to the back of the tablet.



The VRPN server centralizes and retransmits the information from all the interaction devices that interact with the immersive system in which are used, in this case the tablet and the passive stereo glasses. The use of VRPN was a must for the project, as shown by the Non Functional Requirements in the previous chapter. VRPN stands for *Virtual Reality Peripheral Network* and is "... a set of classes within a library and a set of servers that implement a device-independent, network-transparent interface between application programs and the set of physical devices (trackers, buttons, etc.) used in a virtual reality (VR) system" as defined by its creators in [12]. The use of VRPN is a development standard at the Institute Image as well as at the COLIVRI laboratory because of the flexibility provided by an abstraction layer that separates front-end devices from the application input.

The second layer is composed by the Application Servers in charge of rendering the scene to project it in the CAVE or the Spidar. One of these application servers render the scene from the tablet point of view, and stream the scene in real time, to allow the camera metaphor that will be explained in the User Interaction section of this chapter. Is important to note, that there isn't communication between those servers; each one, renders the scene from a particular perspective calculating the adequate frustrum for it's designated screen. Each of the Application Servers renders the images for both eyes of it's screen, but each image is projected by a different projector in order to work as a passive 3D system.

4.2.2 User Interaction

Front-end Devices

There were several initial options for the user interaction design at the start of the project due to the devices already at use at the Institute. Here I'll discuss the reason for choosing an Android tablet over tracking gloves, flysticks and speech recognition. The general principles I took in account to take the decision were the *c*ompatibility and ease of use of the devices and, when possible, the reduction of the number of front-end devices to the minimum necessary to successfully perform the operations needed to use the application.

The definition of *compatibility between devices* is defined, when the use needs to use two or more devices to interact with the application, as the capability to perform any operation in the system without the need to change a device from hand, or any other action that would increase the interaction complexity. So, two or more devices are incompatible if they can't be used and carried by the user in the same way during the use of the application. For example, if the application uses a flystick to select 3D objects and a glove to allow the user to interact with the application in a gestural way; and the user needs to change the flystick from his left hand to his right hand, because he's right-handed and can't perform precision operations with his left hand [1], the devices are mutually incompatible under the *i*nteraction paradigm provided.

The *interaction paradigm* is composed by the set of input actions expected to be made by the user to be able to use the application and the interaction metaphors used to enhance and simplify this interaction.

In this order of ideas, the relevant basic actions are selecting objects (whether they are polygons or menu objects) and writing texts [2]. The following analysis explains how the selection of front end devices was made. The term ID implies that there's incompatibility between the proposed devices, OK means there's not.

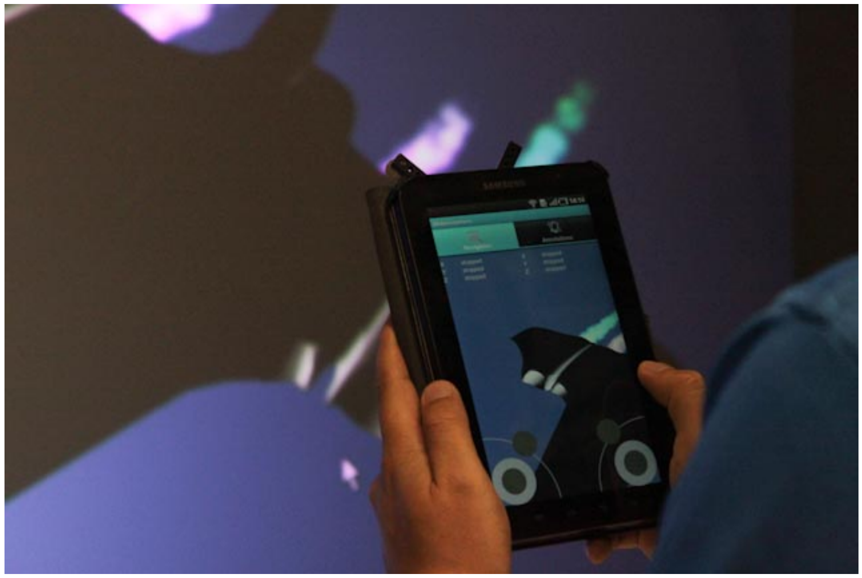
		Text Input Device	
		Speech Recognition	Tablet
Selection Device	Flystick	OK	ID
	Glove	OK	ID
	Tablet	OK	OK

As can be seen the Tablet is the only device that can be used both as a pointer device and as a text input device. There are several ways to use

[1]These factors vary greatly between users, so to broad the validity and applicability of the concept for the purposes of this work, I suppose every user has a dominant hand and can't perform precision operations with his other hand.

[2]Is interesting that even a basic pointing device can be used as a text input device, as shown by the screen-keyboard applications available in Mac OS and Windows. Nevertheless, using a non touchscreen pointing device is incredibly slow and would work as a bottleneck in the application workflow.

the Tablet as a pointing device, the one selected in this project due to it's usability and *woah factor* is the window, or blank frame, metaphor.



A first approximation to the window metaphor can be seen in the work by Jung [9]. It consists basically of the use of a screen, its position and rotation to show the scene as if the screen frame was blank. An important difference between the work by Jung and this project is that, in his work, he doesn't take in account the relative motion of the user's head and the screen.

Application Workflow

The application workflow is straightforward. It consists of the following steps, separated in two stages *Configuration* and *Execution*:

The Configuration steps are:

1. Update the Annotations XML and the 3D Model to their latests versions. [3].

[3]It's highly encouraged the use of a folder synchronization software such as Dropbox®

2. Start the ART Track Toolkit and start the tracking.

3. Start the VRPN Server.

4. Start the Python Server provided with the software, that will send the XML Annotations to the Tablet and start the application on the server side.

The Execution steps are:

1. Check the tablet is connected to the same LAN as the other computers used by the application.

2. Launch the 3D Annotations application from the tablet.

3. Follow the instructions.

The instructions during the Execution stage are quite simple, compared to the ones in the configuration stage. Even when the purpose of this project is to deliver a highly usable annotation system, the Configuration Stage can not be simplified. On the other hand these steps must be followed only once, while there's no need to shut down the system. The possibility of automatizing the configuration process will be examined in the future work section.

User Interface

There's no user interface for the software running in the CAVE servers, so the following content will explain the user interface of the tablet application.

---

or Ubuntu One®, this can automatize this whole step and render it invisible to the final user

A poster at the end of this document shows the information below graphically.

Once the user has executed the application the first screen asks for the user name, this name will be used to fill the author field of the annotations made in this execution session. After typing his name, the user will face a progress dialog while the application connects to the Python Server and retrieves the Annotation XML to be used in this session.

Once the system has load the model and the annotations, the user can start to read the annotations made by others or start annotating the model himself. The main screen of the application is a tabbed window with two tabs: Navigation and Annotations. In the Navigation tab the user will be able to navigate the model using two 3D joysticks located at the bottom of the screen, while in the background and filling almost all the screen space a real-time render of the model will be rendered following the window metaphor discussed above. This render is made by one of the computers running the application and send to the tablet wirelessly using the UDP protocol[4]. The CAVE will project the scene based on the calculated location information received by the tablet.

The 3D joysticks were thought to be used without capturing the user's gaze. Nevertheless, given that the scene can be seen in the tablet as well, the user can use the system as he want to; whether seeing the screen or the tablet. When the user reaches the place an annotation was made, he will notice so because a semitransparent sphere will be rendered on top of the

[4]Using TCP would slow down the streaming throughput in the sake of guaranteeing the reception of all the frames by the tablet. A missing frame is not a real problem in a streaming context.

model, he can tap the tablet screen where the sphere is shown to examine the annotation's contents.

In a different case, if the user wishes to crate a new annotation, the user can tap the screen. This will prompt the user to create a marquee to annotate. After the marquee is created he will click the *Annotate* button and a dialog will be shown. The dialog will contain a form to be filled with the annotation priority and it's contents. If the user confirm the annotation, tapping the button *Save*, the annotation will be saved by the CAVE computers in the Annotations XML and the user will return to the navigation Tab.

The other Tab is the Annotations Tab. Here the user can see a list of all the annotations made in the model organized by priority. In the sake of usability, a button called *Create New Annotation* is shown and it's activation will prompt the user to the navigation screen with the instructions of how to make an annotation.

If the user taps on any of the annotations shown, a dialog with it's details will be shown giving also the option to navigate to the annotation's location. The user can also delete the annotation or return to the Annotations List.

4.2.3 Software

Cave

The code used to load and display the model and communicate to the VRPN Server is largely based on the code used by Phillipe Crassous which is based on the code developed by Sébastien Gerin. This reuse of code lead to a considerable saving of time and also helped to avoid most of the problems associated with my lack of experience using C++ and OpenSecneGraph.

The application running in all the CAVE servers is the exactly the same. The expected differences on behavior and point of view are derived from a configuration file located in the same directory as the executable application. This configuration file facility was already developed in the code authored by Sébastien.

Tablet

The tablet application was developed for this project from 0. It uses a VRPN library developed by Phillipe as well. The application workflow is completely coded in the Android application, and the cave software works as a state-machine receiving signals from the tablet via the VRPN centralized server.

The relevant pieces of the application are: the Image Viewer enabled to receive the streaming output via UDP, generated by an instance of the main program running specially for this purpose and the 3D joysticks featured in the navigation screen. Two of these 3D joysticks enable the user to move with 6 degrees of freedom, so he can move without constrains trough the model of interest.

The Streaming Image Viewer is extends the standard Image Viewer featured by the Android API. It uses an extra thread to read the incoming images. The last incoming image is saved as the image the Image Viewer will display in the next redraw cycle as part of the interface redraw automatically performed by Android.

The 3D joysticks are largely based on the Mobile-Anarchy-Widgets google code project which features a 2D joystick widget, but modified to add an extra axis to allow 3D of freedom movement on each joystick. The next page features a sort of UML diagram that briefly explains the code structure followed.

Login

MainActivity

NavigationTab AnnotationListTab

DualJoystickView ImageStreamView CreateAnnotation Dialog ReviewAnnotation Dialog *

Joystick 3DView

▲

Joystick3DMoveListener
Joystick3DClickedListener

Annotation

How to read this:

If a name is below another, without another third in the middle, it means the second (the one below) is a member of the first one. The one above is the father class, the one below is the member class.

If an asterisk (*) is between two names it means there might be n instances of the member class associated to the father class.

If a triangle (▲) is below a name, it means the rest of the names are from classes thar are implemented / extended by the class of the name above it.

4.2.4 Modularity

Following the need of flexibility, the application was structured in a highly modular way. The best example of this is the code for the camera/window metaphor. The code in the tablet to enable this metaphor is completely independent of the code running in the cave servers, beyond the network layer protocol used, which is simply UDP.

For the metaphor to work a VRPN tracking system is needed. The nature of the tracking system is completely up to the implementer. The only information needed is the location and orientation of the tablet inside the immersive space. A calibration process between the tracker coordinates and the tracking coordinates is needed, but most of the hard work is already done. A communication from the tablet to the computer via VRPN is completely optional, but highly recommended.

5

Implementation

5.1 Implementation Review

The implementation was separated in three large stages. The first one was the development of a test application which runs in Mac OS X and was developed using C++ and OpenFrameworks. The second stage consisted of the development of particular concept tests, regarding some possible technical difficulties I expected to have. The third stage was the actual development of the final application.

5.1.1 First Stage Report

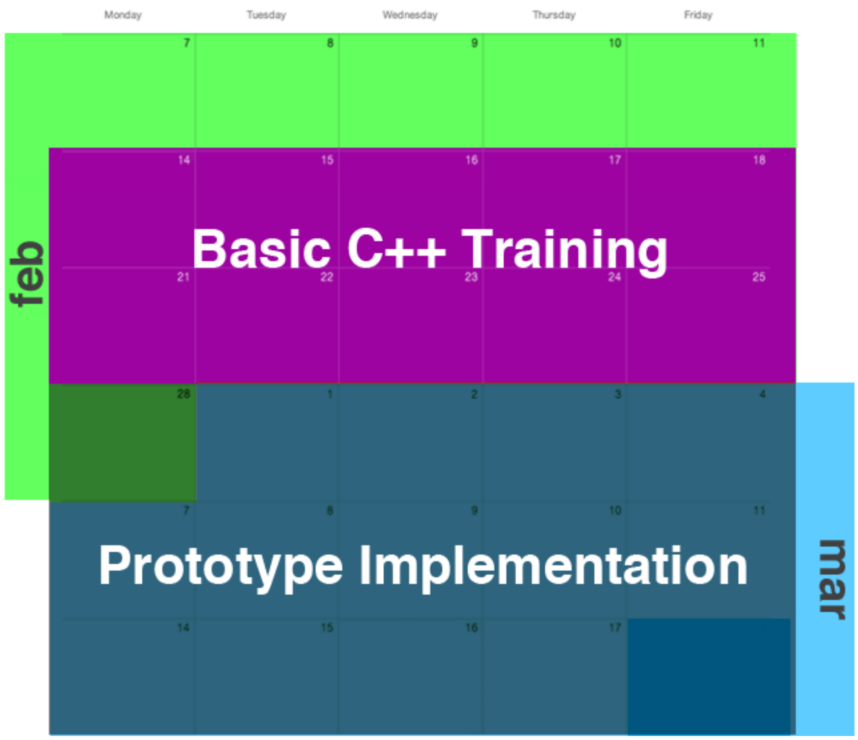
The text below is extracted as it was delivered to Frédéric and Tiberio at the end of the first stage. Keep this in mind while reading.

Process

The first part of my stance in Chalon was spent in training myself in C++. Although I've used it for some college assignments before I wasn't in the shape of starting a new project. I made a pair of experiments to get used to the differences between C++ and the languages I already used. This period took place from the monday january 14 to the friday january 25.

The second part of my stance was concerned with a quick prototype implementation of the 3D Annotation System. This prototype would run in a regular PC providing the basic functionality expected from the final system, but whose user interaction requirements where simplified in the sake of simplicity. As of tuesday, march 24 the prototype was able to load a .3DS model, display it with textures and topological information and allow the user to select vertices. This day took place a meeting with Frédéric and Sebastian to evaluate and comment the prototype and choose the direction of the next steps to take. Strictly speaking this part of the process is over, although I'll end the prototype with the planned requirements.

The calendar so far looks like this:



Prototype

Requirements:

Loads multi-meshed and textured 3DS models. Displays the textured model. Displays on the top of the model its topological information (vertices and edges). Allows the user to select a set of vertices using the mouse. Allows the user to add an annotation using the keyboard. Retrieves the annotations attached to the selected vertices. The requirements in italics are those which weren't fulfilled before the meeting.

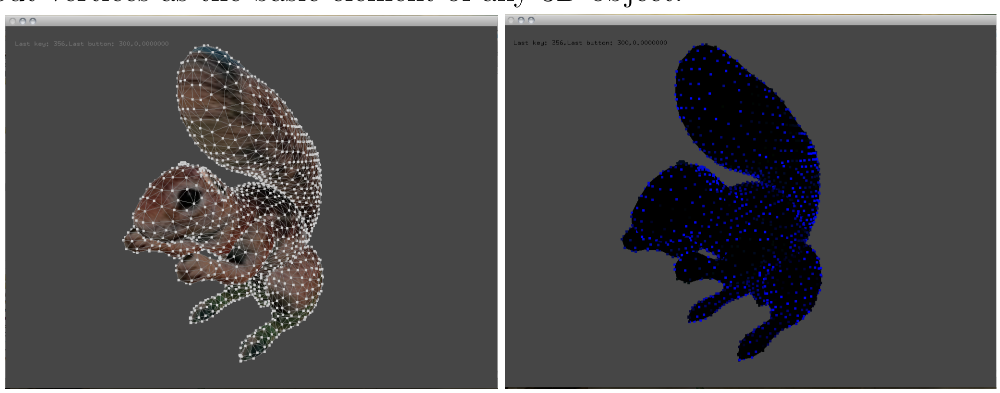
Tech specs:

The prototype was developed on Mac OS X 10.6, using XCode 3.2.4

as IDE and OpenFrameworks as an entry level framework which simplifies some of the tasks needed to fulfill the requirements. Most of the work on the prototype was made on the top of the ofx3DModelLoader OpenFrameworks add-on version.

Design decisions:

The selection of OpenFrameworks as the general purpose framework of the project modeled some of the decisions toked. The selection of ofx3DModelLoader as the base of the code to load and display defined that the archive type to load was 3Ds. My prior use of the Color Picking to select objects in 3D space through a 2D interface made it the best option to me at the time. The selection of vertices as the basic anchor of the annotations was made thinking about vertices as the basic element of any 3D object.



Conclusions

1. The selection of Color Picking wasn't the best, it took me most of the time of the development of the prototype (because of unseen difficulties) and isn't suitable to use in following versions of the product, given the

user interaction requirements[1].

2. The selection of vertices as the anchor points of the annotations was changed during the meeting. Although it's true that the vertices are the basic element of any 3D model, the polygons are a more natural and meaningful part to attach annotations to.

3. Another meaningful decision took from the first prototype was the selection of OpenSceneGraph and not Openframeworks as a more suitable framework to base the work on.

4. A talk with Tiberio exposed the need to evaluate the user interaction with the system before building the complete product. Somewhat like a prior iteration on the UX design. It's also needed to make a more formal definition of an annotation [2].

5. During the meeting with Frédéric and Sebastian, Frédéric pointed the possibility of using speech recognition to make the annotations as a more natural way to interact with the system (vs the option thought before, of using an Android powered tablet to write down the annotations)[3].

[1] I thought that at that point of the development, it turned that with the use of the tablet the color picking algorithm is not only suitable but presents meaningful advantages.

[2] A sudden cut in the expected development time rendered this impossible.

[3] The reasons behind not following this alternative have been already discussed in the previous chapter.

5.1.2 Second Stage

This period took place from the star of April to the second week of May. Definitively not my most productive time in Chalon, nevertheless I think it was a necessary period of time[4]. During the second stage I was deeply concerned about my inexperience with the tools used, so I experimented and tried tutorials in order to gain experience. I was familiar with the Android platform, the Java programming language and the Eclipse IDE, nevertheless I felt I should try to make some stuff before starting to work in the project. This way I could avoid beginner errors on the application. Those tests are not related to the subject of this document in any other way than the acquisition of technical abilities to proceed in the implementation of the project. I also used a code example made by Sébastien to display and navigate 3D scenes in the cave.

At the end of the second stage I was given the code made by Philippe that addressed similar objectives as this project.

5.1.3 Third Stage

The third stage begun with the design and programming of the tablet's software. I had a hard time trying to connect the tablet to a WiFi router provided by the Institute's Network Administrator. I decided to buy my own router and start experimenting with video streaming to the tablet, it took me nearly a week to make it work correctly. Another issue that took me more time than I expected was the programming of the 3D Joysticks.

[4] However, it could last a little less.

After some time tweaking the tablet app and the scene streaming a progress meeting was made with Frédéric and Sébastien. After this meeting I had only one week left to implement the parts of the software I avoided during most of the time. I worked hard that last week and ended with a complete solution, and almost the proposed one. As I've already commented, the difference between the application delivered and the one designed is that the delivered application is incapable of create and display the comments as marquees. So the user, to review the annotations must go to the list in the tablet app, and review one by one.

5.2 Expected Results

The expected results consisted of an application capable of solving the problem largely described in previous sections. But to summarize and follow the document format the summarized requirements of the application were:

1. Navigate a 3D model with 6D of freedom.
2. Allow a non intrusive and cognitively ergonomic navigation method.
3. Stream rendered scene to the tablet taking into consideration its position.
4. Select a relevant part of the model to associate the annotation to.
5. Introduce textual information such as author, contents and priority.
6. Use VRPN as the protocol between devices in runtime.

7. Feature a flexible solution that may run in other environments with minor tweaks.

8. Save and load the progress as an XML file that may be synchronized by a third party application.

9. Run in a standard Windows XP machine with mouse and keyboard.

It is relevant to comment that these requirements don't have anything to do with the Functional or Non-Functional requirements featured previously. These are summarized requirements that were considered as part of the development of the project, and were deduced trough the iterations of the process.

All the requirements mentioned above were met, with the notable exception of the last one: *Run in a standard Windows XP machine with mouse and keyboard.* The reason behind this is that, as noted in the list of Non-Functional Requirements the development of this part of the project implied a whole new branch in the software development.

The solution presented is then, as specified in the Acceptance Levels, an optimal solution and as such it was accepted by the project supervisors.

6

Conclusions

6.1 Methods and Result Validation

This part of the document is a whole chapter in the ABET format. For my project, this part of the guidelines is particularly out of focus. I decline to waste space and time in such thing, so this small section is its replacement.

The method used to validate the application was a standard use test by Frédéric, Sébastien and other memebers of the team at the Institut Image that evidenced the correct behavior of the application and the fullfillment of the Functional and Non-Functional requirements.

As the deliverable of the project is an application with a straight forward architecture, and standard usability tests would take more time than we can afford.

6.2 Discussion

This work features a 3D annotation system to be used in immersive environments. The application make use of an Android powered Tablet, an IR tracking system, and computers running windows XP. The position and orientation of the tablet is calculated by the IR tracking system and based on this information a render scene is streamed to the tablet via a WiFi connection. This is thought to work as a window metaphor, so in the tablet screen is showed the rendered scene as it was a blank frame and being consistent with the image projected in the immersive system. This, enables the user to make use of its natural intuition to interact in the physical world to ease the learning curve of the application, and use it in a more natural way. The user can position its viewpoint in a his desired location and associate an annotation with this point of view. Other users might review the annotation and its associated viewpoint to enable an agile workflow with the associated model.

There were to important issues during the development. The first one is the lack of selecting an specific region in a screenshot to associate the annotation, it is a simple issue that might be solved with a little development time and more C++ expertise. The other important issue is the window/camera metaphor calibration. This calibration is a pecise process and even the slightest imperfection is evident to the user, again more time and expertise would solve the problem.

6.3 Future Work

The most important future work that might be developed is the improvement of the camera/window metaphor, as its correct implementation would imply a huge leap forward in virtual reality systems. This enhancement would be made by developing a tool that enables an easy way to calibrate the system to display correctly the metaphor.

Another important future work, would be the ability to follow specific user's annotations and track the progress in the annotations of the model in a similar fashion as the code progress is displayed in the github repositories.

Of course the modification of the application, so it would be capable of running in a standard PC would broad it's reach and potentialities.

Bibliography

[1] Apple Inc. Apple human interface guidelines.

[2] Bernheim Brush A.J., Bargeron D., G. A. C. J. Robust annotation positioning in digital documents. In *Proceedings of the SIG-CHI Conference on Human Factors in Computing Systems* (2001), pp. 285–292.

[3] Bowman, D. *3D user interfaces: theory and practice.* Pearson Education. Addison-Wesley, 2005.

[4] Burton N, Kilgour A C, T. H. A case study in the use of vrml 2.0 for marketing a product. In *Proceedings of From Desk-Top to Web-Top: Virtual Environments on the Internet, World Wide Web and Networks* (1997), pp. 1–24.

[5] Duval, T., Fleury, C., Nouailhas, B., and Aguerreche, L. Collaborative exploration of 3d scientific data. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2008), VRST '08, ACM, pp. 303–304.

[6] Fried, J., Hansson, H., and Linderman, M. *Getting Real: the Smarter, Faster, Easier Way to Build a Successful Web Application.* 37signals, 2009.

[7] Gonçalves, D., Jesus, R., Grangeiro, F., Romão, T., and Correia, N. Tag around: a 3d gesture game for image annotation. In

Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology (New York, NY, USA, 2008), ACE '08, ACM, pp. 259–262.

[8] Hong, L., Chi, E. H., and Card, S. K. Annotating 3d electronic books. In *CHI '05 extended abstracts on Human factors in computing systems* (New York, NY, USA, 2005), CHI EA '05, ACM, pp. 1463–1466.

[9] Jung, T., Gross, M. D., and Do, E. Y.-L. Annotating and sketching on 3d web models. In *Proceedings of the 7th international conference on Intelligent user interfaces* (New York, NY, USA, 2002), IUI '02, ACM, pp. 95–102.

[10] Kadobayashi, R., Lombardi, J., McCahill, M. P., Stearns, H., Tanaka, K., and Kay, A. Annotation authoring in collaborative 3d virtual environments. In *Proceedings of the 2005 international conference on Augmented tele-existence* (New York, NY, USA, 2005), ICAT '05, ACM, pp. 255–256.

[11] Lidwell, W., Holden, K., and Butler, J. *Universal principles of design: 100 ways to enhance usability, influence perception, increase appeal, make better design decisions, and teach through design*. Rockport, 2003.

[12] Russell M. Taylor II, Thomas C. Hudson, A. S. H. W. J. J. A. T. H. Vrpn: A device-independent, network-transparent vr peripheral system. *VRST* (2001).

[13] Sonnet, H., Carpendale, S., and Strothotte, T. Integrating expanding annotations with a 3d explosion probe. In *Proceedings of the working conference on Advanced visual interfaces* (New York, NY, USA, 2004), AVI '04, ACM, pp. 63–70.

[14] Tsang, M., Fitzmaurice, G. W., Kurtenbach, G., Khan, A., and Buxton, B. Boom chameleon: simultaneous capture of 3d viewpoint, voice and gesture annotations on a spatially-aware display. In *Proceedings of the 15th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2002), UIST '02, ACM, pp. 111–120.