GLAZE

VISUALIZATION FRAMEWORK FOR MOBILE DEVICES

By

ROBERTO NUNO SILVA SOUSA

University of Madeira

Funchal, Madeira

2009

2/July/2009

ACKNOWLEDGMENTS

Page one of acknowledgments.

Page 2 of acknowledgments if needed

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

1.1 Everywhere and powerful

Mobile devices have become important artifacts supporting today's lifestyles each passing day their importance grows larger. People are always on the move and computing is reflecting this in its transition from the desktop and into people's day to day lives. Now more than ever mobile devices are taking the role of the desktop computer; people can play games, browse the internet, listen to music, watch television shows and more and each new generation of mobile devices include additional power, features and complexity.

This growing focus on increasingly multipurpose devices, has led to the inclusion of a multitude of sensor technologies like cameras and accelerometers, bigger and better screens and a general increase in performance. They have reached a point where it is indeed easier to just think of them as truly personal computers than just simple phones.

The application areas opened by such increases in processing and sensing capabilities are broad. Navigation as been extensively explored and many map based commercial systems exist (e.g. Nokia Maps). The research community is exploring other metaphors such as augmented reality (Narzt, et al., 2003) or non-visual directional queues (Erp, Veen, Jansen, & Dobbins, 2005). Other sensors are more focused on user interaction. For example, the sensing of pose to control the orientation of graphical displays (Hinckley, Pierce, Horvitz, & Sinclair, 2005), motion as input to dynamic simulations (Cho, Choi, Sung, Lee, Kim, & Murray-Smith, 2007) and camera input to perform cursor-like navigation and selection (Rohs, 2004).

1.2 The power of visualizations

One area at the convergence of these new technologies is that of information visualization. People have always relied on visual tools to better understand problems or situations around them and take advantage of the massively parallel visual nervous system to very quickly solve these problems. In computer science, visualization research is concerned with finding algorithms that make use of the stable structure of the human visual system to present data. Card et al. (Card, Mackinlay, & Shneiderman, 1999) defined visualization as "the use of computer-supported, interactive, visual representations of data to amplify cognition". Cognition is the acquisition or use of knowledge and the main objective of visualizations are insight not pictures and the goal of insight is discovery, decision making and explanation. Visualizations are important when there is a need to present massive amounts of information very quickly.

Visual displays provide the highest bandwidth channel from the computer to the human, we acquire more information through vision than all our other senses combined (Ware, 2004). Furthermore, a substantial portion of the human cortex is devoted to analyzing visual information. This is one of the greatest advantages of visualizations, provided we present it well huge amounts of data can be rapidly interpreted and processed by the user. Other advantages include allowing the perception of unanticipated emergent properties, as these can form the basis for new insight. Problems with the data become immediately apparent, like when one looks at a white wall if there is a black spot anywhere we find it almost immediately. It facilitates understanding of both large and small scale features of data. It facilitates hypothesis formation, finding new features in the visualization can lead to new insight of the data and that can lead to testable hypothesis (Ware, 2004). An example of this can be seen in Figure 1.

Further advantages are presented by Card et al. (Card, Mackinlay, & Shneiderman, 1999) by proposing six major ways in which visualization amplifies cognition:

- By increasing the memory and processing resources available to users. Visualizations can expand processing capability by using the resources of the visual system directly. Many attributed of visualizations can be processed in parallel. Visualizations can be used to store massive amounts of information in a quickly accessible way.
- By reducing the search for information. Visualizations can visually group or relate information, they can compact it and allow hierarchical search by using overviews to locate areas for more detailed search. They allow zooming and popping up details on demand and index data spatially by location with landmarks to provide rapid access.
- By using visual representations to enhance the detection of patterns. Visualizations can allow patterns in the data to reveal themselves. Aggregations of data can reveal themselves through clustering or common visual properties.
- By enabling perceptual inference operations. Visualizations allow some inferences to be made very easily that would be complicated otherwise. This is why we see areas like physics making extensive use of visualizations.
- By using perceptual attention mechanisms for monitoring. Visualizations allow for the monitoring of potential events if the display is organized so that these stand out by appearance or motion.
- By encoding information in a manipulable medium. Visualizations allow exploration of a space of parameter values and can amplify user operations.

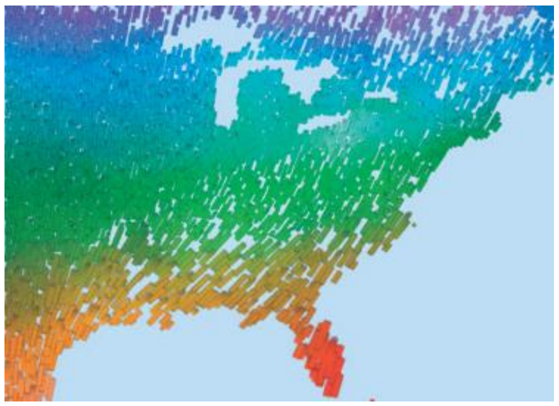


Figure 1 - Weather conditions over the eastern US visualized with simulated paintbrush strokes. Color represents temperature (purple/blue for cold to orange/red for hot), density represents wind speed (denser for stronger), orientation represents precipitation (tilted for heavier), size represents pressure (larger for higher), and luminance represents clouds (brighter for heavier). (Dennis, Kocherlakota, Sawant, Tateosian, & Healey, 2005)

1.3 The challenges of mobile visualizations

Information visualization is a very fertile field of investigation especially with the proliferation of powerful and cheap computers but a distinction needs to be made between developing visualizations for desktop computers and mobile devices. Even with improvements on processing and graphic power in mobile devices allowing more researchers to use a wide range of visualization techniques in various research areas they are still faced with many hurdles. Compared to desktop computers the main problems faced when developing visualizations for current mobile devices are (Chitarro, 2006):

- Display: screens are limited in size, with low resolution, fewer colors and non standard screen aspect ratio.
- Reduced processing capabilities.
- Interaction: Input peripherals are inadequate for complex tasks, different input techniques like stylus or thumb based input.
- Slower connectivity that affects interactivity when information is stored remotely.
- Vendor specific form-factor, performance and input peripherals.
- Tool available for development are very limited and overly complicated to use properly.

These are just a few points that differentiate visualization development between the desktop space and the mobile space and it shows that we simply can't trivially port visualizations from the PC space onto mobile devices. While many of these problems are slowly but steadily being solved, others such as screen size are more fundamental and will require paradigmatic shifts before significant improvements are made. However mobile devices also present possibilities not available in desktop computers for example they support interesting visualizations relying on integrated sensors (accelerometers, compasses, etc), cameras, GPS devices, etc. Especially the camera has been used to create many interesting visualizations, for example Meiguins et al (Meiguins, et al., 2006) use the camera to overlay multidimensional information in an augmented reality environment.

Another great advantage and one of the obvious ways of using visualizations on mobile devices would be for mobility, people wouldn’t need to be sitting in front of a desktop computer to view the visualizations used in their field, they could see it on the move and anytime they want. Furthermore, recently there has been an increase in research into mobile visualizations and the creation of various methods of visualization creation and exploration discussed in chapter III.

1.4 Frameworks help

Software frameworks are extensively used in software development, we can think of them as a premade solution to a specific problem. They provide a base that allows for the development of applications for that specific problem quickly and without worrying about low level details and how everything fits together.

Fortunately developing visualizations fits neatly into this description; it is a process with very specific stages as seen in the information visualization reference model detailed in Chapter II section 2.5 This allows for the development of a framework that hides this process from developers and lets them focus on each particular step, how they impact their problem and how to use them to achieve their objectives.

1.5 Objective

Visualization development is a growing area of research in mobile devices, their potential for this medium is undeniable, with a growing number of imbedded sensors and growing processing capabilities it is an attractive platform for this kind of research. The objective of this project is to develop a general and flexible framework based on the information visualization reference model detailed at chapter II section 2.5 to ease development of visualizations directed at mobile devices, but also to serve as a base for further extendibility to focus and ease development for particular problems.

1.6 Organization

This document is organized as follows, chapter II will explore current visualization frameworks for both mobile and desktop platforms; chapter III will describe the development and design of Glaze, a framework for mobile visualizations; chapter IV will describe and explore current research in all visualization techniques and respective solutions; chapter V will describe test applications developed using the framework, their requirement elicitation, design, development and evaluation; chapter VI will describe the potential of Glaze and future directions.

CHAPTER II

STATE OF THE ART – TECHNOLOGY AND FRAMEWORKS

This chapter presents current technology and frameworks for both mobile devices and desktops. The focus of this chapter is in particular technologies applied by each framework and how it affects visualization development. Those are scripting, components, remote or offloaded visualizations and raw data treatment with advantages and disadvantages of each approach. It presents and describes each stage of the information visualization reference model. Also a case study of a possible extension to Glaze to create a framework that is more focused on a specific type of visualization is reviewed.

2.1 Scripting

Scripting languages have gained a lot of use in visualization development, it allows for simple development by being simpler to write for, and without the need to compile. One major flaw is they are limited by the functions the base application is able to provide and those are usually directed at solving a specific problem or creating a specific visualization.

Meyer et al. (Meyer, Gîrba, & Lungu, 2006) created Madrian a framework that works directly with raw data and uses a scripting language to allow developers to directly create mappings in the script for that data and easily execute it. The script is also simple enough to be built on top of any graphical framework. It also allows for the creation of new visualizations by combining basic shapes. The ability to work directly with raw information and in a declarative way with scripting conveys more benefits, as not only does it remove an important step from the visualization process but it also simplifies visual mapping of elements in the raw data to visual elements in the visualization. Unfortunately it also brings disadvantages; mapping although simple is limited to whatever is supported by the visualization engine at the time the developer is writing the script, so generality is decreased. This does not invalidate the use of scripting to define this sort of mapping, just that some simplicity needs to be sacrificed for generality. The ability to create visualization by aggregating basic shapes is also an interesting feature for it allows one to very easily join disparate shapes to create something new with minimal effort. This type of aggregation is very simple, by looking at all visual components as a visualization on its own it is possible to go one step further and offer the same functionality at a higher level allowing for visualization aggregation.

Piskuliyski and Kumar (Piskuliyski & Kumar, 2007) propose an overlay visualization framework directed at learning with two objectives, to separate the application from the visualization and to separate the specification from the rendering. The separation of the visualization from the application was needed to maximize flexibility and minimize overhead. To separate the visualization from the application they identified two layers a layer of graphical primitives like boxes and arrows, and visual primitives like highlighting connecting two graphical primitives. Visual specifications are created declaratively allowing them to be automated as well as specified by users, or simply written by hand. This framework makes extensive use of scripting to define visual specifications; its creation is very simple so much so it can be automated, this is an advantage of such languages their ability to be automated and created at runtime. But we again see that its use is closely linked to the problem the framework is trying to solve, it can only provide a list of specifications for the creation of visual primitives.

Scripting languages although widely used are not a good fit when one is looking for generality, they mostly possess a supportive role to the main application and are used as a means to quickly create a visualization the framework in question was created to address. For them to have a more active role there is a need to find a more visualization agnostic use. One possibility is for visualization mash-up and combination, by using visualizations as a whole as a function scripts can be created to create new visualizations based on existing ones by using one or more as the input of another. Or more advanced uses if visualizations where if divided in stages the script could contain the order those stages are executed. To surmise, to guarantee both generality and flexibility, the objective for script use should not as a simple supporting mechanism for frameworks, but as a tool to manipulate visualizations as a whole.

2.2 Components

Component use is an interesting approach to development; they maximize reuse and simplify development.

Behzadan et al. (Behzadan, Timm, & Kamat, 2008) have developed a reusable, general-purpose, mobile augmented reality (AR) framework, for outdoor AR. It is built using a modular approach to both hardware and software allowing for pluggable hardware and software and thus remove the need to re-implement low-level communication interfaces making it readily shared and reused for a multitude of problems.

The framework is not aimed at mobile devices, but at the desktop space, but its ideas can easily be adapted. They use a laptop in conjunction with a GPS sensor, head mount display, camera, head tracker and a touch pad for hardware, such a multitude of devices warrants their need for a modular approach to hardware, an advantage of mobile devices since most have the needed sensors imbedded in the device itself.

They again take a very modular approach to software, to increase reusability each module is exported as a dynamic link library (DLL). This maximizes portability by allowing any potential user of their system to import and apply any module to their application with minimal effort.

Their approach to modularity is very positive, with such a multitude of sensors to work with the ability to simply use a component developed to communicate with a specific sensor is an important feature. For mobile devices we do not need their modular approach because all of them are imbedded in the device itself, but not all have the same sensors. By adopting a model where we can develop a module to communicate with a specific sensor, when faced with mobile devices that offer a different set of sensors we need only change the software components to match the difference with minimal or no change to visualizations.

Itoh and Tanaka (Itoh & Tanaka, 2006) propose a component based framework to generate visualization programs in a 3D environment to help users use Web services without the need for programming. Components in their framework take a special meaning; they are 3D shapes able to do a specific task, like a 3D textbox. Their framework has the ability to create compound components by connecting other visual components allowing users to create 3D applications using web services through combining 3D components. The framework possesses a central component they call a WebServiceProxy made to hide communication detail from the users and it is central to their system, it has a set of functions specifically made to handle web services enabling users to operate each by just using interactive manipulation, once all the information has been set it creates a 3D form with input interfaces for each argument of each function, that 3D form is comprised of the WebServiceProxy and other components. Once everything is done it creates a visualization of the retrieved results from the web service and arranges them geometrically in 3D virtual space.

Components here change from what they were previously; they are a static set of software components defined within the application. This approach works fine when the framework is aimed at a specific problem or visualization, but it fails when we need generality. It is impossible to create a set of static components and how they relate to each other for every type of visualization in existence, this has to be addressed in a problem by problem basis.

Component use is an interesting approach to handle unpredictability while minimizing change; especially for mobile devices with very different capabilities it is a good way to handle changing sensor configurations between different models.

2.3 Visualization offloading

Offloading most of the work to remote server is a widely used technique with mobile devices. Even with their growth in processing power some visualizations are still too intensive to be handled by mobile devices. The solution for this problem comes with offloading the rendering process to a remote server and having it send the final frame to the client for display.

Ehret et al. (Ehret, Ebert, Schuchardt, Steinmetz, & Hagen, 2004) present a framework for scalable information visualization for both mobile and desktop systems. It uses a server to generate the visualizations to the clients, both desktop and mobile. The framework consists of a data and presentation layer, the presentation layer is realized as a relational database containing information and references to external media files. The presentation layer dynamically generates pages that fit the screen of the client device; these pages are encoded in HTML to allow any standard compliant browser to render the visualization.

Their use of a data base to keep information about device capability presents interesting possibilities, most notably the fact that there is no need for configuration on client. Also the use of HTML to present the visualization guaranties broad compatibility.

The problem with this sort of system comes with the use of interactive visualizations where latency is an extremely important concern that is not addressed.

Zhou et al. (Zhou, Qu, Wu, & Chan, 2006) present a client oblivious framework for volume visualizations. Mobile devices as well as desktop computers have very different capabilities and performance, their framework present a client oblivious data model used to structure information about 3D models in such a way that it can be readable by any device and rendered according to its own capabilities while maintaining the maximum amount of information possible. This allows the server to not need to know any information about the device connected to it, it needs only to prepare the model and send it leaving to the client to render it according to its own capabilities. The server also encodes the most important information first so that clients can achieve meaningful results as soon as possible.

This is very similar to the previous framework, but it possesses the advantage of not needing any information about any client, this forces the clients to have an application capable of reading the data provided by the server. This approach simplifies communication immensely because of the current dissimilarity between processing capabilities of mobile devices and desktops. Another advantage is the creation of a standard with a separation between detail and performance, this guaranties device and platform independence. However this does not work if there is a need to manipulate the visualization, in this case the same restrictions from the previous work apply and latency would become an issue.

Remote or offloaded visualizations is an area with great research potential, but many hurdles still need to be overcome, to assure compatibility a standard communication format must be created or mass adoption of an existing one must take place. Also latency is a very important issue when we consider interaction one that is a much more difficult to resolve.

2.4 Raw Data

One of the most important steps when doing any kind of visualization is reading and classifying raw data so visualizations systems can understand and use them properly.

Kreuseler et al. (Kreuseler, Lopez, & Schumann, 2000) suggest a scalable visualization framework to address the problems of reducing information and obtaining structure, their framework implements a scalable preprocessing pipeline implementing several algorithms for analyzing unstructured information spaces as visual clutter arises or due to screen space the number of objects exceeds a limit it becomes indispensable to apply some preprocessing to gain structures, extract relevant subsets of the information and for reduce the active data size to a processable level.

By providing a very flexible preprocessing pipeline several algorithms can be applied for structuring unorganized data and forming manageable subsets of complex information spaces and facilitates visualization development, this has an interesting use in providing a way of abstracting all types of raw information to a unique preprocessed format to simplify visualization creation and communication between separate components. But there is also a need for many preprocessing algorithms depending on what kind of data we have to deal with. This is a very risky proposition especially for a general system; one type of untreatable data would render the system useless, and as noted by the authors there is no general mathematical framework or paradigm on how to build those groups or subsets.

Meyer et al. (Meyer, Gîrba, & Lungu, 2006) propose a different approach, instead of moving the data to be visualized to the tool they argue for moving the visualization tool to the data. By use of an interface, through which developers can script in a declarative fashion the visualization they can create their solution working directly with the data to prevent duplication (where data will be in their raw format and duplicated in the format used by the visualization).

The ability to work directly with raw data conveys has a major benefit of removing an important step from the visualization process that is transforming raw data into something to be used in the visualization. But it also brings many disadvantages; mapping although simple is limited to whatever is supported by the visualization engine at the time the developer is writing the scripts, so generality is decreased. Also this approach will eventually reach a significant performance wall. The main reason to duplicate data within the visualization is to process it and not have to worry about the original data. With this approach, unless the raw data can be changed, there is a need to repeat the data treatment process every time there is a change.

Kim et al. (Kim, Hsu, Moon, Kumar, & Helmy, 2008) present a framework leveraging MySQL databases and Google Earth for users to quickly visualize mobile networks and usage patterns. By using a set of always present information like MAC ID, access point, association start and end time they can save information about each access point in the data base and allow users to query it. It then visualizes the results using a Google Earth map.

Organizing raw information in a database is an interesting approach, developers can create tables to organize the data they are interested in, relate and also query them. This also has the advantage of allowing data mining to discover new facts and help the visualization process further. The flaw with this approach comes with multimedia content, there are many level which one can save this type of data, for example we can simply save the name or location of a video, or we can save each frame of even each pixel, this eventually leads to a lot of overhead and wasted processing power if for example we want to see the video and it is saved as pixel information in the database.

Treating raw data is an important subject in visualizations with no clear solution. The ability to offer a way to implement the ideal way of raw data processing in a problem by problem way seems to be the ideal approach to maintain generality.

2.5 General visualization model

In order to build a general visualization framework we need to first know if the visualization creation process can indeed be divided into discreet parts and if those parts communicate in a predictable way at all times for all types of visualizations. The answer for this is the information visualization reference model shown in Figure 2. Chi (Chi, 1999) developed the information visualization reference model in his Ph.D. thesis under the name of data state model. In it Chi showed that this model successfully modeled a wide array of visualization applications.

Card et al. (Card, Mackinlay, & Shneiderman, 1999) presented their own interpretation of that model and dubbed it the information visualization reference model shown in Figure 2. The information visualization reference model has been successfully used no a number of visualizations. In its essence the model presents the phases needed in

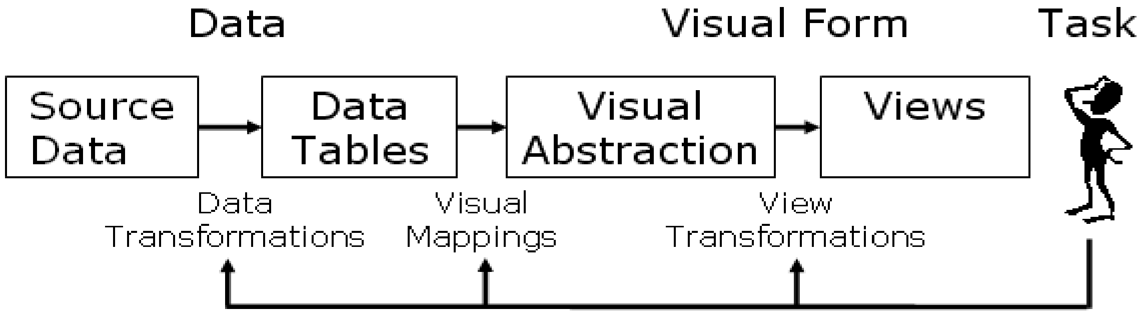


Figure 2 – Information Visualization Reference Model

order to map raw data into visual information. One can find similarities between it and the popular MVC (Model View Controller).

We must first transform the raw data into data tables based on a series of data transformations, these tables are a set of structured mathematical relations to make it easier to map during the next step, and they also use metadata to present data as mathematical relations. This metadata is especially important in choosing visualizations for they can change various aspects in the tables like for example the order of rows, it is also important to note that these transformations involve loss or gain of information. These data tables also serve as a base line data model that can back any number of visualizations.

The next step is visual mapping. The purpose is to transform data tables into visual structures. This can be done in various ways but it is essential that the visual structure preserves data without bringing out unwanted features. Visual structure is used to define the use of space, how one uses that space is the most dominant factor of visualizations. The visual structure includes the visual features such as spatial layout, color, size, etc. it is responsible for containing all the information needed to draw a visual representation of the data. One can have many visual structures serving as a visualization-specific model, each one being fed by one data table and each creating separate visualizations for the same data. Space has a structure that can be expressed with different axis, like for example a map with orthogonal axis. If something visible happens in the defined space it is called a mark, these have graphical propertied like color or type and these can be used to encode information.

The last step converts the visual structures into actual views based on view transformations. These views are defined by graphical parameters like position and scaling. Rendering consists of drawing the contents of the visual structure into any number of views; these can provide varying perspectives onto the data. Lastly the user can control how the data is transformed, mapped to visual structures and how they want to view it. Those transformations are all the techniques that will be addressed in chapter IV, namely exploration, interaction and distortion techniques. During each of these steps the user can affect the creation of the visualization, like what kind of raw data is gathered, how it‘s transformed, what kind of properties marks the visual structure should have and what view point to use to display data (Aaltonen & Lehikoinen, Refining visualization reference model for context information, 2005).

Aaltonen and Lehikoinen (Aaltonen & Lehikoinen, Refining visualization reference model for context information, 2005) further refined the information visualization reference model for context information, this is especially useful for mobile devices because they possess a multitude of sensors and are used a lot in the development of context-aware applications.

2.6 Case study

Dennis et al. (Dennis, Kocherlakota, Sawant, Tateosian, & Healey, 2005) present an end to end framework of the information visualization reference model to promote a more comprehensive visualization framework describing how to apply expertise from human psychophysics, databases, rational logic, and artificial intelligence to visualization; and illustrate the benefits of a more complete framework. They go one step further from the concern of algorithms to visualize certain domains of data and ask themselves what to do before displaying the information and after the user views the data, by introducing data management into visualization, metadata generation and management, techniques to preprocess data to extract and display critical details, and intelligent systems that help users design effective visualizations. They provide a new view over the information visualization reference model with new objectives as seen in Figure 3; this also makes this framework a good example of a possible extension for Glaze.

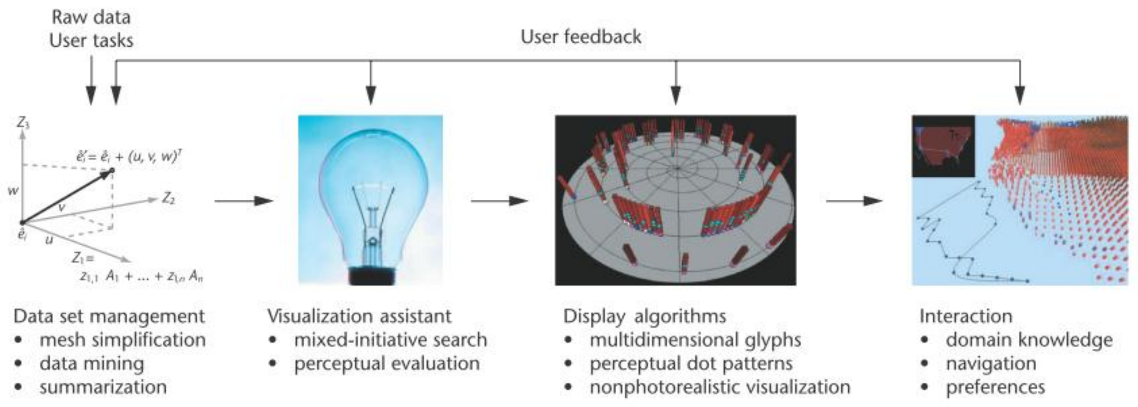


Figure 3 – Proposed framework including data set management, assisted visualization design, display and interaction (Dennis, Kocherlakota, Sawant, Tateosian, & Healey, 2005)

2.6.1 First stage

The first stage of the model is the data set management (data tables and transformations); managing raw data is a very important problem, without good management of data even the most advanced visualization can be overwhelmed by a sufficiently large data set. There is a need to for proper data structures, data base management and network protocols. Real world data also has properties that need to be taken into account and properly handled, like missing information, noise, uncertainty and duplication.

They suggest two approaches for this problem, data mining and mesh simplification. Data mining algorithms analyze a training set to build rules to subdivide multidimensional data into different categories; they can also apply the rules to assign membership to unclassified elements. This allows users to select subsets of the data by category to reduce the amount of elements in the visualization; the rules also help compress dimensionality by for example combining dependent attributes into a single composite value. Mesh simplification converts multidimensional data into a triangular mesh where each vertex corresponds to a data element. Attribute values are encoded as surface features stored at each vertex. The system can apply feature-preserving simplification algorithms to reduce the mesh's size. This reduces regions with little variation to a few vertices, while areas with high variation remain intact.

These solutions are both very interesting, the use of data mining for the automatic creation of organizing rules is of great interest, but data mining is notoriously memory and processor intensive which does not make it ideal for mobile devices. Also interactive visualizations would present problems, if one interaction changes any of the raw data the whole process needs to be executed again, the same goes for the second solution.

2.6.2 Second stage

The second stage is critical to generate effective, high-quality and perceptually salient visualizations. This is a non trivial process; we must consider the data to be visualized, the user's analysis tasks and the strengths and limitations of different display techniques in order to arrive at a satisfactory result. Moreover sometimes there is a need to build multiple visualizations for the same data set in order to try and see different aspects of its structure.

Their solution is a mixed initiative tool called ViA (Visualization Assistant) that collaborates with users to iterate a final set of high-quality, perceptually salient visualizations. ViA uses rules of perception to build visualizations that map individual data attributes to different visual features like hue or size. It uses AI search algorithms to rapidly identify high quality data-feature mappings and then designs and evaluates each mapping based on application independent input about the data (like spatial frequency and if it is continuous or discreet), the viewer’s analysis needs and guidelines from human perception like how we define color.

By testing how people perceive color, textures, shapes etc results in guidelines for each of those features, ViA stores each feature's strengths and weaknesses in an evaluation engine. A data-feature mapping is reduced to attribute-feature pairs like one attribute assign to hue, for each evaluation engine to determine how well this feature supports the attribute mapped to returning a normalized evaluation weight to summarize the analysis. Suggestions are provided for poor pairings suggesting how that attribute could be better visualized. ViA's search algorithm stores results from each attribute-feature mapping, non conflicting mappings are chained together to produce strategies on how to improve the current mapping, for each chain the algorithm assigns an expected improvement weight and places it in a priority queue, the one with the highest value is removed and applied to produce a new mapping, and is similarly evaluated. This search continues until a predefined stopping condition at that point the top n mappings are presented allowing users to select different mappings to visualize their data in different ways. Users can modify input conditions and ask ViA to continue searching, also they can lock attribute-feature pairs in a mapping and ask for new results based on these constraints.

Their use of AI to generate mapping from data to visual elements using guidelines from the human perception is interesting, especially considering how this process is done with the aid of the user. This has the advantage of iteratively improving the visualization with the aid of the user and to allow for simple multiple visualizations on the same data using the same process.

The disadvantage is the lack of flexibility of this system, there is no mention of a way to define arbitrary visual elements to create original visualizations. Even with that capability there would be a need to insert new perception rules for ViA to be able to correctly use that feature.

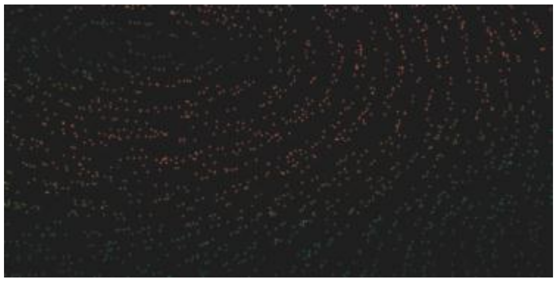


Figure 4 – visualizing flow in a simulated supernova collapse, relative dot position and color represents magnitudes (blue for low to red for high) (Dennis, Kocherlakota, Sawant, Tateosian, & Healey, 2005)

2.6.3 Third stage

The third stage is the display algorithms here the focus is on perception either through suggestions from ViA or from other models in how we perceive information. The result is a visualization optimized for user's data and tasks that also harnesses the capabilities of a low-level visual system for rapid and accurate visual analysis, data exploration tasks are usually completed in 200 milliseconds or less per image.

They present three examples demonstrating how they are applying this knowledge to build effective visualizations. The first algorithm is the use of 2D glyphs or 3D objects that can vary in appearance to encode information. Properties like color, texture and motion are very useful to visualize physical and abstract data an example can be seen in Figure 1. The second uses the perception of orientation to visualize 2D fields. A collection of dots is placed throughout a flow field with careful positioning so their positions relative to each other generate perceived orientations matching the flow direction of the flow field an example can be seen in Figure 4. In the third they designed a correspondence model to match visual features to glyph-based visualization to brushstroke properties in a non-photorealistic image, this allows for the use of data-feature mapping to render a set of brushstrokes that produce a painting of the data elements stored in a multidimensional data set. The motivation for this is to determine if aesthetic visualization improve effectiveness. Their interest is based on investigations of orientation and engagement in perception, orientation leads the viewer’s focus of attention to a particular location, with guidelines on how to use low-level vision they can control that orientation to direct the viewer's gaze to important areas. They hypothesize that such a representation will increase engagement, encouraging the viewer to remain at the chosen focus of attention and increase the amount of detail they are able to remember and recall an example can be seen in Figure 1.

The ability to develop display algorithms that harness the low-level visual system with the aid of ViA or perception models is interesting in maximizing the amount of information and velocity people can read it. Care must be taken with some more specific visualizations like AR that do not focus exclusively in presenting information but also in aiding users in real world tasks, the models used must take this into account and so does ViA.

2.6.4 Fourth stage

The fourth and last is the interaction, most visualization systems only allow users very simple types of interaction, like changing camera position and changing data-feature mapping in real time. Their approach allows for collaboration between the computer and user to combine their specific strengths, computers can rapidly analyze, search, apply rules and render data while the user can easily interpret patterns and relationships, understand external details about the data being studied, and apply specialized domain knowledge. Another issue is the need to include application context in a visualization, ViA tries to support this by allowing the user to lock subsets of data-feature mappings, and identify penalties by reporting evaluation weights between the best mappings with and without those user constraints allowing users to make an informed decision on how they want to proceed. Sometimes users are interested in dynamically changing subset of elements, by identifying which with Boolean logic rules spatially neighboring elements of interest are clustered into local groups and connected with a graph. A global spanning tree is constructed to bind the local graphs together with the final graph structure being presented in a map, allowing users to decide where in the data set they want to explore. The ability to create rules although effective has drawbacks, they are time consuming and static, if a user's interest shifts during visualization they must be updated by hand, it might also be hard for the user to define exactly what the properties of interest are.

This framework is very interesting not only because it improves upon the information visualization reference model, by automating some steps, but it also provides suggestions on how to effectively help users in their exploration process and how to facilitate visualization development for developers by having guidelines for low-level human visual systems. Because if follows the information visualization reference model it can be also be used as a possible extension of Glaze.

2.7 Conclusion

Many of the frameworks viewed here, although powerful, are aimed at particular problems and cannot be used as a basis for development of arbitrary types of visualizations. Many types of technologies are also used within each framework like components, scripting, raw data handling and remote rendering or calculation.

Components are the most interesting of those technologies, it allows for a separation between elements of a visualization like the raw data handling and the visualization itself, this brings many advantages like the ability to change which type of information the visualization is using with minimal to no change in its source code. Another advantage comes by allowing and encouraging reuse thus greatly facilitating development.

Scripting is also very interesting, but their use is mostly as a support technology for specific objectives. More general and flexible uses must be found for the use of this technology to be justified other than in a project by project basis.

Visualization offloading or rendering is useful for mobile devices especially when faced with a computationally complicated visualization. But its use is diminished when faced with interactive visualizations where latency becomes the definite issue of this technology.

Raw data handling is very important when it comes to visualizations, the more organized they are the faster and easier the visualization can access and use them. The problem again comes at the lack of flexibility current technologies offer, no single solution can be said to be superior to all others. So the best solution is to leave this process to the visualization developer where they can be free to choose the ideal method for their current problem.

Another problem is the lack of mobile device oriented frameworks, which is not surprising given how recently these achieved an acceptable performance for this kind of research. Glaze is an attempt to fill this void by serving as a general, flexible and base for future extension for any and all kinds of visualizations in the mobile space.

This chapter presented current technology and frameworks for both mobile devices and desktops with a focus on particular technologies applied by each framework and how it affects visualization development like scripting, components, remote or offloaded visualizations and raw data treatment with advantages and disadvantages of each approach. It presented and described each stage of the information visualization reference model. Also a case study of a possible extension to Glaze to create a framework that is more focused on a specific type of visualization was reviewed.

CHAPTER III

GLAZE FRAMEWORK

This chapter presents the Glaze framework. It begins by describing the requirements and objectives of the project. It then describes Glaze's overall architecture, individual components how they function and relate to each other and support technologies and libraries. It ends with advantages and disadvantages of the current approach.

3.1 Project requirements

The original goals of this project were to develop a mobile device application which overlays graphical information on a live video feed from the device's camera. That information should include the following features:

- Static bitmap images
- Text
- Simple Vector graphics (lines, circles, arcs)
- Transparency
- Animations
- Simple 3D graphics

The ultimate objective for this system was to be capable of being used to create visualizations for mobile devices overlaid on live camera feed.

By organizing these features the next set of requirements were reached:

1. The application must support static bitmap images.
2. The application must able to render text.
3. The application must render simple vector graphics.
4. The application must have an animation system.
5. The application must render simple 3D graphics.
6. The application must be capable of rendering any content on top of live camera feed.
7. The application must be able to read information from all sensors present on the phone, be it camera, accelerometers and more.
8. The application must be capable of handling a wide variety of visualizations.
9. The application must be as platform independent as possible.

Given these requirement the need for a graphics oriented API was obvious, as well as a very powerful mobile device, also because the application needed to be capable of being used for any number of visualizations the need for generality was important as well.

Firstly graphic rendering is provided by OpenGL ES it has texture support loaded from static bitmap images, simple vector graphics with the use of wireframes and a very complete 3D rendering pipeline. The only missing functionality was the ability to render text and an animation library, those are implemented in Glaze as a bitmap font renderer in both 3D and 2D and a time based animation engine.

The camera feed can be seen as a texture in Open GL ES context, there is the need to capture it and convert it to a format the graphics API can handle.

Other sensors are easily used; each possesses native APIs to access them without much trouble.

Secondly platform independence cannot be achieved by simply writing the application in a common language like JAVA, each platform possesses specific and proprietary libraries, especially for sensors, that made this an impossibility, this is made harder with the lack of hardware to test multiplatform code. Because of this multiplatform is not directly achieved instead the component approach to Glaze makes it as simple as possible to port to other platforms by only having the core application very bound to the platform it is being developed in, the other components are much more standard by minimizing the use of platform specific libraries. Also native support for XML gives developers a platform independent tool to offload aspects of their visualizations and more easily port them to another platform.

Finally the last and most important requirement is the ability for the application to be capable of handling as many types of visualizations as possible. Visualizations are very different from each other and by providing support to some one type we inevitable break support for another. The solution was to create a framework that provides supports for the common elements to all visualizations as well as libraries to support all the other requirements of the project.

3.2 Glaze's overall architecture

Generality and flexibility were identified as important aspects of the framework, not only because the project would be used to create a multitude of unspecified visualizations, but also because this system was to be developed for mobile devices possessing very different types of hardware and capabilities. For these reasons Glaze is a plug-in based architecture, (Figure 5) this allows for a great deal of flexibility when designing a visualization, each component can be tailored towards a specific type of hardware or capability. This approach simplifies visualization development to achieve the most general system possible by dividing the information visualization reference model into discreet components each with its own responsibilities. The system is divided into 4 parts, the core system, inputs component, visualization component and front-end component. All components can be configured using XML with further configuration personalization possible, although required configuration in the framework is very simple and functions as just a way for the core system to know which components to use for the visualization.

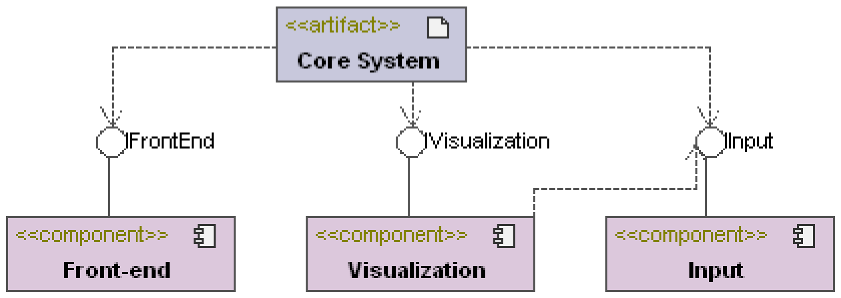


Figure 5 – Component diagram representing glaze's overall architecture

All this enables the visualization developer to focus only on the visualization creation process and not on presentation and raw data handling code assuming the needed components needed for their work already exist. Another advantage of this approach is the complete separation of the input, visualization and front-end components from each other; this enables designers to use any one or multiple input libraries as their data sources, and also separates them from graphic library detail. That separation however is not complete, since graphic API's and hardware capability is always changing creating a front-end wrapper to hide all graphic commands from the visualizations will be very time consuming and in constant need of maintenance. With this concern in mind the front-end will simply be responsible for initializing the graphics subsystem, refreshing and rendering the scene to the screen. This will make the visualizations dependent on which front-end they will use, but allow them to directly and freely use the chosen API. This relationship between the front-end and visualization components allows for the simultaneous rendering of an arbitrary amount of different visualizations simultaneously.

The components closely follow the reference model in that the input component is responsible for reading and organizing source or raw data into data tables (or how ever the input developer wishes), how it does that can be called data transformations and each input component will have its own and in this case it will itself be that data table. The visualization component will be responsible for querying the input component and ask it for whatever data it needs, further treat that data if needed and perform a visual mapping, how that is done will determine the visual abstraction or what visual element the data will be connected to, view transformation and views will too be determined by the developer in the visualization component. Finally the front end will be responsible for drawing everything as specified by the developer onto the screen. Because all components are DLLs there is an interface class shared between the core system and all the other components defining the base services (methods) provided by each type of component and that must be implemented. A sequence diagram describing how everything communicates with each other to execute a visualization is shown on Figure 6.

This completes the overall picture of the system, so this thesis now turns to each individual component, describing what their purposes are and how they interact with all the others.

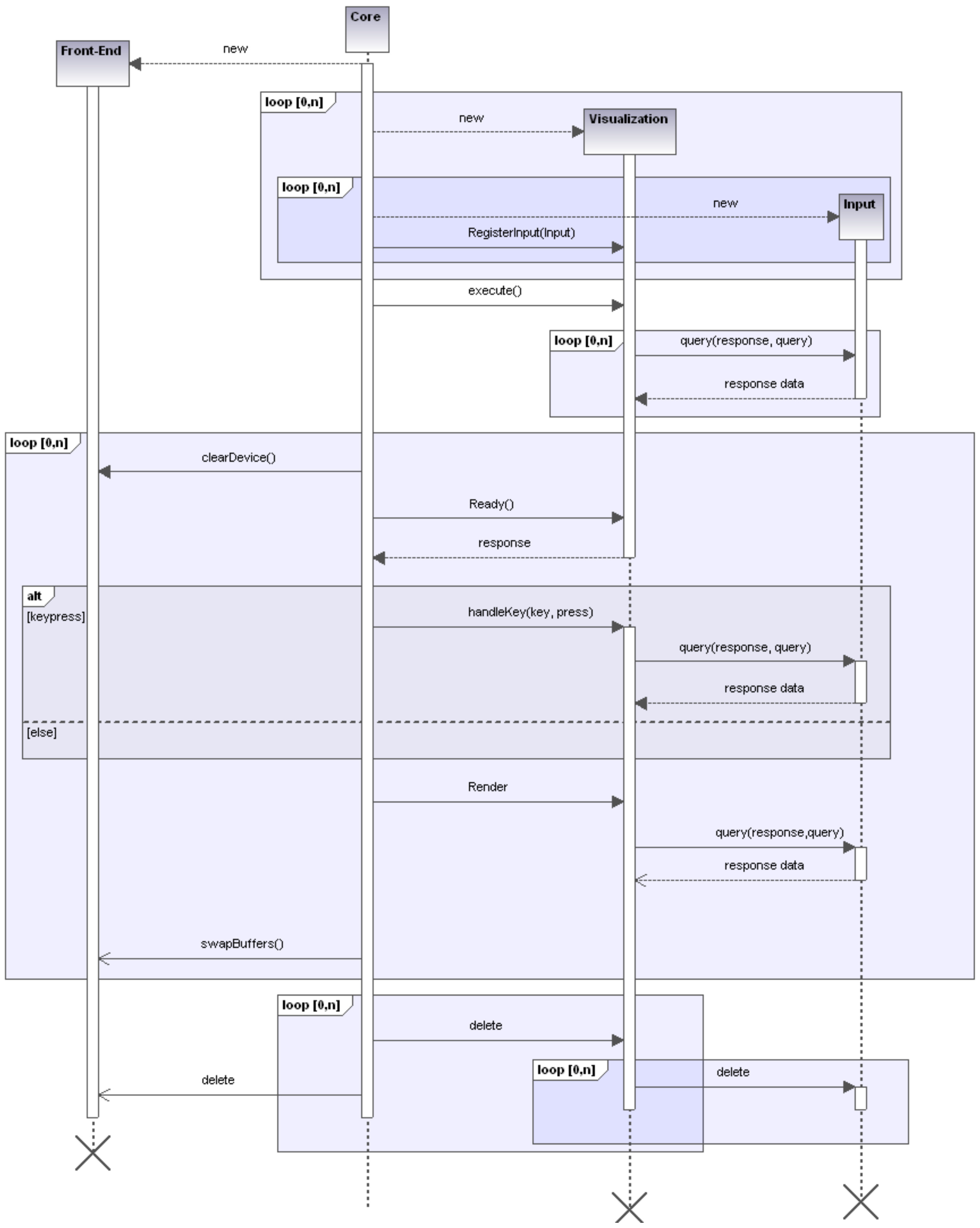


Figure 6 – visualization execution sequence model

3.3 Core system

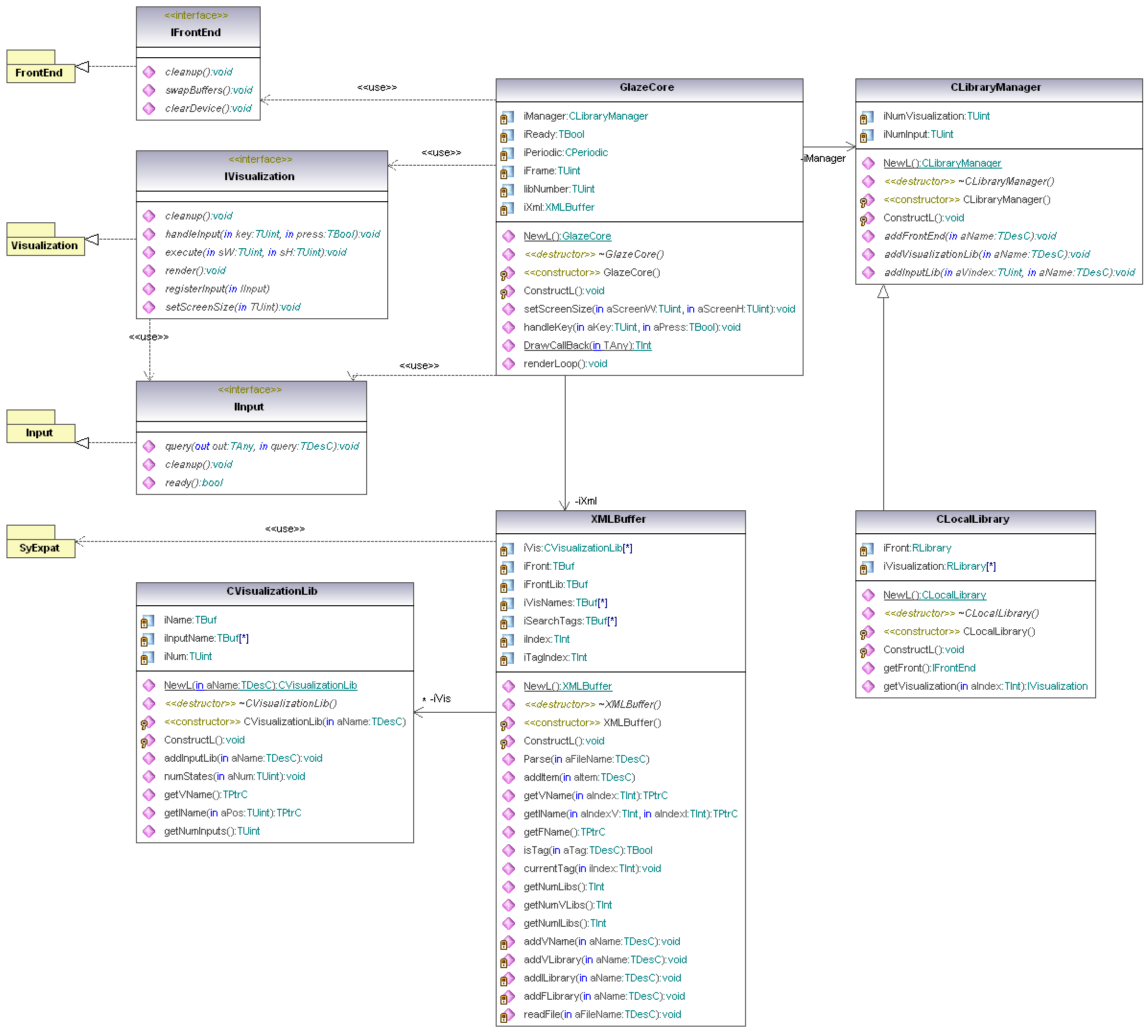

FrontEnd
<<interface>>
IFrontEnd
cleanup():void
swapBuffers():void
clearDevice():void
<<use>>
GlazeCore
iManager:CLibraryManager
iReady:TBool
iPeriodic:CPeriodic
iFrame:TUint
libNumber:TUint
iXml:XMLBuffer
NewL():GlazeCore
<<destructor>> ~GlazeCore()
<<constructor>> GlazeCore()
ConstructL():void
setScreenSize(in aScreenW:TUint, in aScreenH:TUint):void
handleKey(in aKey:TUint, in aPress:TBool):void
DrawCallBack(in TAny):TInt
renderLoop():void
-iManager
CLibraryManager
iNumVisualization:TUint
iNumInput:TUint
NewL():CLibraryManager
<<destructor>> ~CLibraryManager()
<<constructor>> CLibraryManager()
ConstructL():void
addFrontEnd(in aName:TDesC):void
addVisualizationLib(in aName:TDesC):void
addInputLib(in aVindex:TUint, in aName:TDesC):void
Visualization
<<interface>>
IVisualization
cleanup():void
handleInput(in key:TUint, in press:TBool):void
execute(in sW:TUint, in sH:TUint):void
render():void
registerInput(in IInput)
setScreenSize(in TUint):void
<<use>>
<<use>>
Input
<<interface>>
IInput
query(out out:TAny, in query:TDesC):void
cleanup():void
ready():bool
-iXml
SyExpat
<<use>>
XMLBuffer
iVis:CVisualizationLib[*]
iFront:TBuf
iFrontLib:TBuf
iVisNames:TBuf[*]
iSearchTags:TBuf[*]
iIndex:TInt
iTagIndex:TInt
NewL():XMLBuffer
<<destructor>> ~XMLBuffer()
<<constructor>> XMLBuffer()
ConstructL():void
Parse(in aFileName:TDesC)
addItem(in aItem:TDesC)
getVName(in aIndex:TInt):TPtrC
getIName(in aIndexV:TInt, in aIndexI:TInt):TPtrC
getFName():TPtrC
isTag(in aTag:TDesC):TBool
currentTag(in iIndex:TInt):void
getNumLibs():TInt
getNumVLibs():TInt
getNumILibs():TInt
addVName(in aName:TDesC):void
addVLibrary(in aName:TDesC):void
addILibrary(in aName:TDesC):void
addFLibrary(in aName:TDesC):void
readFile(in aFileName:TDesC):void
CVisualizationLib
iName:TBuf
iInputName:TBuf[*]
iNum:TUint
NewL(in aName:TDesC):CVisualizationLib
<<destructor>> ~CVisualizationLib()
<<constructor>> CVisualizationLib(in aName:TDesC)
ConstructL():void
addInputLib(in aName:TDesC):void
numStates(in aNum:TUint):void
getVName():TPtrC
getIName(in aPos:TUint):TPtrC
getNumInputs():TUint
* -iVis
CLocalLibrary
iFront:RLibrary
iVisualization:RLibrary[*]
NewL():CLocalLibrary
<<destructor>> ~CLocalLibrary()
<<constructor>> CLocalLibrary()
ConstructL():void
getFront():IFrontEnd
getVisualization(in aIndex:TInt):IVisualization

Figure 7 – glaze's core system class diagram

Glaze is centered around the core system, it is an application developed for a specific platform that uses components that provide what is needed to achieve a visualization. Configuration is automatic, provided the correct XML files are present. Its responsibilities include initializing, managing and receiving user input. It was kept simple in order to facilitate cross platform development; it simply reads an XML containing the name of secondary XML files for front-end and visualization components that are going to be used. The secondary XML for the front-end only needs the name of the DLL implementing that front-end. As for the visualizations they must possess the name of the DLL that implements it and the name of the input component DLL they will use.

With that information it then first initializes the front-end component followed by the visualization component then by the input components ending by registering them in the visualization component so it can use them directly without communicating with the core. It is also responsible for rerouting user input to the visualization component for processing. Its final responsibility is in calling the front-end component's refresh and screen render when all the visualizations are done with their respective rendering cycles. We can easily see its main objective is to control and manage the process described in the information visualization reference model.

3.4 Input component

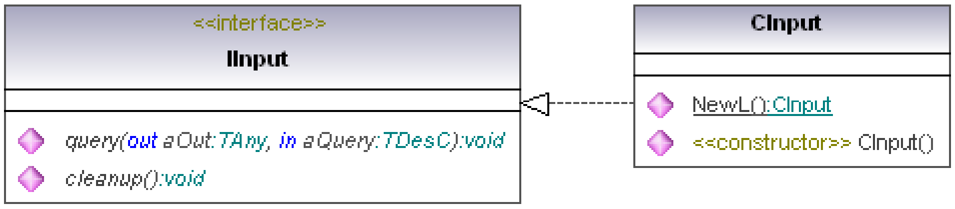


Figure 8 – glaze's input component base class diagram

The input component is responsible for reading information from a specific source of raw data (or untreated data) be it from a file, a sensor or even the internet, that will be decided by the creator of the component.

Details of how it reads and organizes the data also remain hidden from all the other components and are decided by its creator. An example of this is a component to read a file, one might create a component to read a specific file type and when it's initialized it immediately reads and stores the content of that file in memory, or it can wait for a query and access the file directly thus saving memory in the process. Each one of these approaches has advantages and disadvantages depending on what we want to use them for.

Queries are an essential aspect of this component, in order for a visualization to access data from an input it needs to query it, we can look at this the same way we look at SQL, the input component will have the data organized how the creator wants it to be and the visualization only needs to ask for specific items they want from that data. To make a parallel to the visualization reference model the input component is the data table, its variables are the metadata and the data on the variable are the data in said table. We can see that an SQL like language for the visualization to access the input would be ideal, but outside of the scope of the project.

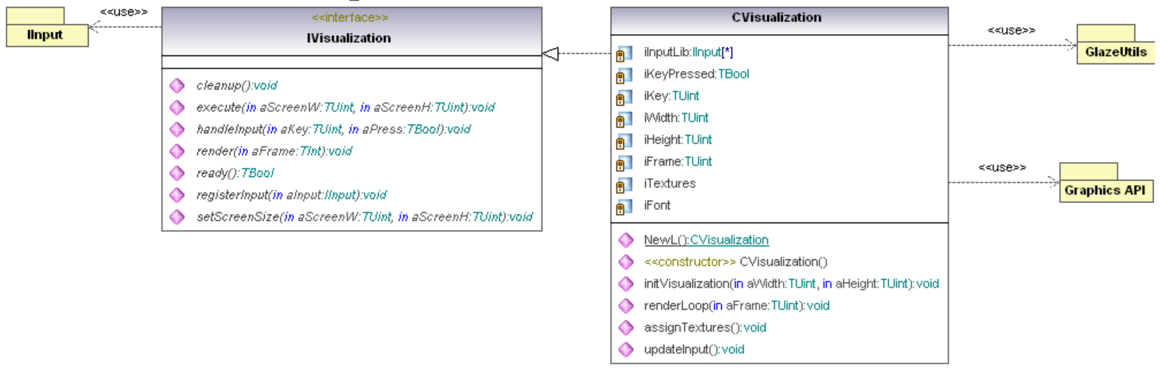
So far we have seen that there is a lot of freedom when creating and organizing input components, the last thing left to clarify is how is the visualization creators are going to know what information is available from the input components. It is not enough to have that information in the input component itself because developers need to know beforehand what is available to use and what is not so they can make proper queries. This type of information must accompany the input component itself in whatever manner the developer feels is appropriate, within that information at least what variables or metadata can be queried and their respective type must be provided. This allows for visualization developers to know what information is available from that input component and its specific type. Unfortunately the use of types is a necessity and cannot be avoided this forces the data being communicated between visualization and input to be explicitly cast every time a query is performed.

The input components' main class is not abstract, due to the nature of DLL creation all of the next methods are overloaded from the input component interface class and must be implemented, those are:

1. A query method to handle visualization queries about any of the data in the component.
2. A cleanup method, this works very much like a destructor in a class and shares a similar behavior.

These are the basic and minimum methods needed to create an input component, of course developers can include any other classes, custom made or not to the visualizations without any fear of incompatibilities.

3.5 Visualization component



IInput
<<use>>
<<interface>>
IVisualization
cleanup():void
execute(in aScreenW:TUint, in aScreenH:TUint):void
handleInput(in aKey:TUint, in aPress:TBool):void
render(in aFrame:TInt):void
ready():TBool
registerInput(in aInput:IInput):void
setScreenSize(in aScreenW:TUint, in aScreenH:TUint):void
CVisualization
iInputLib:IInput[*]
iKeyPressed:TBool
iKey:TUint
iWidth:TUint
iHeight:TUint
iFrame:TUint
iTextures
iFont
NewL():CVisualization
<<constructor>> CVisualization()
initVisualization(in aWidth:TUint, in aHeight:TUint):void
renderLoop(in aFrame:TUint):void
assignTextures():void
updateInput():void
<<use>>
GlazeUtils
<<use>>
Graphics API

Figure 9 – glaze's visualization component base class diagram

The visualization component is the most important and largest of the four. This is where the developer will create the visualization. We can look at this component as encapsulating the rest of the visualization reference model. In it the developer will proceed to create the abstract table and the views. There are no concrete methods and structures to transform the data and organize it that must be provided by the developers. This increases the difficulty of developing a visualization, but what is lost in ease of use is made up by generality for it provides freedom to approach any problem in anyway the developers want. Even without implicit methods for transforming and organizing data the visualization reference model is still being followed, they must find a way to map the data they collect from the input to visual elements and organize them for the user to see, the difference is that it is not a set process defined by the framework but by the developer.

The visualization components' main class is not abstract, due to the nature of DLL creation all of the next methods are overloaded from the visualization component interface class and must be implemented, those are:

1. An input component register method, this block could remain the same for all visualizations, but in this early version exploration in how to save references to input components is encouraged.
2. A method to set the screen to used by the visualization, this again can and most likely will be the same for most full screen visualizations and when

changes occur in screen format (like size), but given the need for flexibility this function can be altered.

3. A user input handler, this too will most likely remain the same for most visualizations, but defining it is simple and allows more control over user input, although an intermediate input handling class could facilitate this process.
4. An execution method, this is where all visualization related initializations take place including graphical elements, this is also where all pre-rendering queries to the inputs and visual mapping of those happens.
5. A method to inform the core system when the visualization is ready to start rendering, this is especially important in non threaded visualizations, if the rendering process begins prematurely everything else will be considerably slowed down.
6. A render method; this is where all the code needed to render the visualization into the screen will be, including the last part of the visualization reference model, the views.
7. The final method is the cleanup, this works very much like a destructor in a class and shares a similar behavior.

These are the basic and minimum methods needed to create a visualization component, of course developers can include any other classes custom made or not to the visualizations without any fear of incompatibilities.

3.6 Front-end component

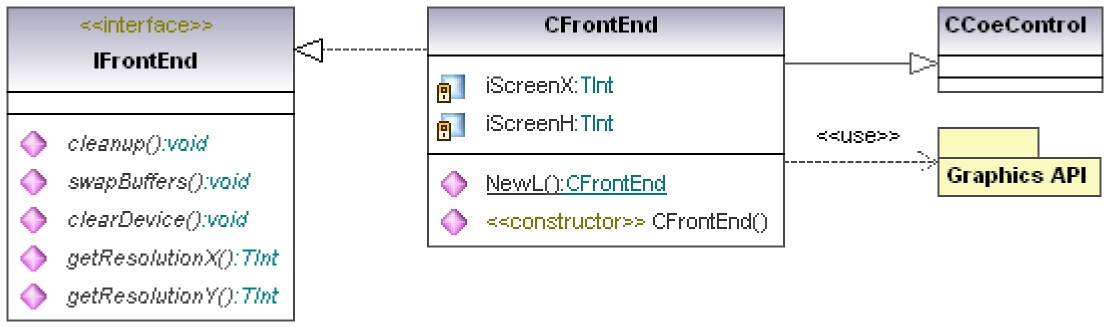


Figure 10 – glaze's front-end component base class diagram

The front-end component is responsible for initializing and refreshing the graphic side of the visualization and also of rendering the scene to the screen. The advantage of this is to remove the need to initialize and upkeep the graphic side of the visualization thus simplifying the development process. With this there is only a worry about the specific needs of the visualization and nothing else. Besides reducing the complexity of development this approach also frees developers to choose the API that is most suitable to their work. It can also be seen as a way to future proof the application as mobile devices become more powerful the more graphic API's or graphic systems become available this will simplify their introduction and use by developers which only need to re-implement the front-end with the new API. Another great advantage of this approach is that it can be used to aggregate visualizations, by removing the need to refresh and output the scene to the screen from the visualizations we can execute the rendering loop of all selected visualizations and end up outputting to the screen all of them at the same time. This all works to significantly increase the expressiveness of the entire system.

The front-end's main class is not abstract, due to the nature of DLL creation all of the next methods are overloaded from the front-end's interface class and must be implemented, those are:

1. A method about how to render everything onto the screen.
2. A method about how to clear the screen.
3. Two methods to send the rendering area or screen resolution.
4. The final method is the cleanup, this works very much like a destructor in a class and shares a similar behavior.

These are the basic and minimum methods needed to create a front-end component, of course developers can include any other classes custom made or not to the visualizations without any fear of incompatibilities.

The decision on which front-end to use is very important because the same API must be used in both the front-end and the visualization, until an intermediate wrapper or abstraction is provided.

3.7 XML

Glaze makes extensive use of XML. It's most basic use is for configurations, although a difference between configurations types must be established. There are two kinds of configurations possible in Glaze, framework configurations, those are the minimum obligatory configurations that must be present when executing the application and were discussed in section 3.3. The other is visualization configurations; those are created by developers as they need it. This separation allows for the introduction of framework specific features without them interfering with visualizations and vice versa. Only the core, visualization and front-end component must have an XML with minimum configuration that are part of the framework configuration. Except for the core XML developers are free to extend all the others for their visualizations in any way they wish. They are free also to create any number of new files for their visualizations, for any component.

Platform independence is also one of the reasons for the use of XML; by offloading as much of the visualization's particularities it makes the porting process that much simpler and faster, if the visualization developers uses the XML file as way of creating templates for their users where they can use their own data and configure the output the porting process will be confined to just their engine and no change will be noticed by the user when they use Glaze in another platform.

The DTD and schema of the XML used for framework configuration can be found in the appendix section 6 and 7.

3.8 Support Libraries

Glaze also possesses a set of libraries providing support for all other requested and useful extra features. Libraries present in this current version are an Open GL ES utility

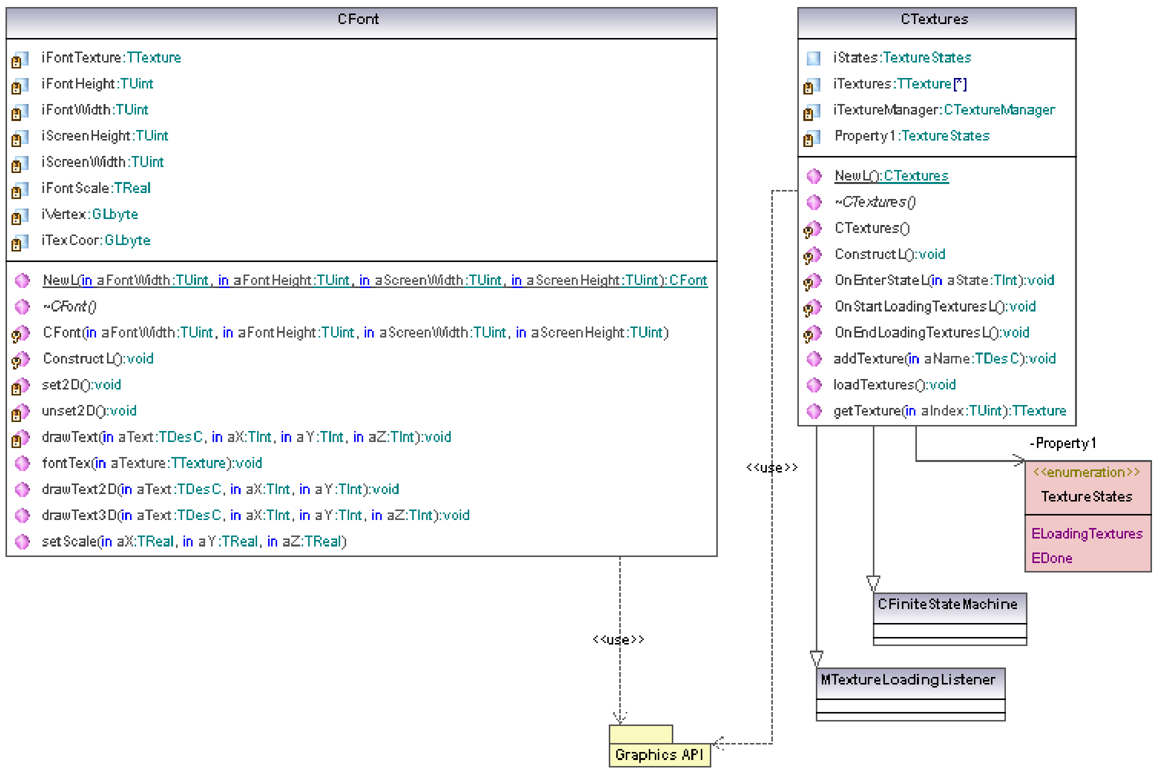

CFont
iFontTexture:TTexture
iFontHeight:TUint
iFontWidth:TUint
iScreenHeight:TUint
iScreenWidth:TUint
iFontScale:TReal
iVertex:GLbyte
iTexCoor:GLbyte
NewL(in aFontWidth:TUint, in aFontHeight:TUint, in aScreenWidth:TUint, in aScreenHeight:TUint):CFont
~CFont()
CFont(in aFontWidth:TUint, in aFontHeight:TUint, in aScreenWidth:TUint, in aScreenHeight:TUint)
ConstructL():void
set2D():void
unset2D():void
drawText(in aText:TDesC, in aX:TInt, in aY:TInt, in aZ:TInt):void
fontTex(in aTexture:TTexture):void
drawText2D(in aText:TDesC, in aX:TInt, in aY:TInt):void
drawText3D(in aText:TDesC, in aX:TInt, in aY:TInt, in aZ:TInt):void
setScale(in aX:TReal, in aY:TReal, in aZ:TReal)
CTextures
iStates:TextureStates
iTextures:TTexture[*]
iTextureManager:CTextureManager
Property1:TextureStates
NewL():CTextures
~CTextures()
CTextures()
ConstructL():void
OnEnterStateL(in aState:TInt):void
OnStartLoadingTexturesL():void
OnEndLoadingTexturesL():void
addTexture(in aName:TDesC):void
loadTextures():void
getTexture(in aIndex:TUint):TTexture
-Property1
<<enumeration>>
TextureStates
ELoadingTextures
EDone
CFiniteStateMachine
MTextureLoadingListener
<<use>>
<<use>>
Graphics API

Figure 11 – glaze's support class diagrams for the font and texture manager

library for texture support, extra math and 3D functions. An open source XML SAX parser library used to parse XML files called SyExpat, the reason for this is that memory is a big concern with mobile devices and a SAX parser minimizes memory usage considerably compared to DOM parsers. A Bluetooth library providing a base engine to be used in the visualization, it provides methods for device and service discovery, connecting and listening to other devices, and service advertizing. A text library providing bitmap font support with the ability to render ASCII text anywhere in 2D or 3D space. Finally an animation manager library providing time based animation support to arbitrary visual elements.

3.9 Advantages and Disadvantages

Glaze's approach to visualization development bring along a lot of advantages, being based on the information visualization reference model allows it to be used in any problem faced by developers. The minimalistic approach means the coding process is largely unconstrained and developers have considerable flexibility in how to approach and solve their problems. It also increases the ability for future expansion to focus on a specific type of problem or visualization. The plug-in architecture allows not only component reuse but also flexibility by not locking a visualization to a particular input, both developers and users can change components without the need to recompile the visualizations. Development is also simplified by this architecture, developers need only focus on one particular component, provided all the other components they need have already been developed. This architecture indirectly forces developers to think about the problem of visualization development as a set of discreet problems by separating responsibilities between all components, it also facilitates future portability by having significantly simpler parts do develop instead of one massive monolithic application.

The component approach also brings future advantages, it is the first step in realizing distributed components and with it distributed visualizations, even visualization services over the internet. By effectively separating the input components from the visualizations as they are now we can have visualization from remote input data, or local input data visualized remotely then downloaded to our device, all including mash-up capabilities as well.

As for disadvantages, Glaze's generality and minimalistic approach means a lot of code can be repeated between visualizations, the most notable examples are with registering input components and setting the screen area for the visualization. Communication between visualization and front-end depends heavily on the language components are being developed in, as it is there is a necessity to constantly cast return variables in the input component, this weakens the ability to change inputs without the need to recompile the visualization, this however can be solved by sending expected types in the query so the input can adapt itself, the downside of this is that it introduces unneeded complexity.

Another disadvantage in the present version is the fact that while the front-end and visualization components are separate (they're not even aware of each other) they share the same pool of memory, as in Open GL ES commands from either of them affects the overall visualization, although this does bring interesting side effects an intermediate layer separating the two more effectively would be a better solution and would allow developers to change front-end with no penalty, where now by using an Open GL ES front-end developers must use the same API commands in their visualizations.

Looking back at the main goals of the project, initial features are fully supported, by the use of Open GL ES and included custom libraries. Requirements are also fulfilled by a combination of present APIs and Glaze's architecture.

This chapter presented the Glaze framework. It described the requirements and objectives of the project, Glaze's the overall architecture, individual components how they function and relate to each other and support technologies and libraries. And also advantages and disadvantages of the current approach.

CHAPTER IV

STATE OF THE ART – VISUALIZATIONS

This chapter presents and describes visualization techniques divided according to a classification scheme of, exploration, interaction and distortion; it also includes corresponding methods implementing them. Finally it discusses problems with mobile visualization development and the different ways of visualizing different types of information.

4.1 Visualization Techniques

When designing a visualization system one must keep in mind visualization exploration, interaction and distortion techniques. In order to have an effective visualization it must support effective user exploration, how fast or efficiently will users perceive what the visualization is trying to convey is achieved through efficient mapping facilitating exploration by the user. It must also possess interaction to allow for effective exploration, like the ability to change visualization angle to a new view to look for new features of the data, this allows for users to manipulate the visualization and redefine their goals when new insight into the data has been gained. Finally distortion techniques are important as a means to filter the data, visualizations with large sets of data become hard to work with and the ability to for example zoom into a particular group while maintaining an overall view of the entire set is an important characteristic. In short visualization exploration allows the visual representation of the data and its exploration. Interaction techniques allow the user to directly manipulate a given visualization, it allows one to change the visualizations dynamically and according to an exploration objective, they also allow relations and combining multiple independent visualizations. Distortion techniques provide a way to focus on something specific while maintaining the overall view of the data. Its objective is to show more detailed information about a portion of the data and maintain the rest with much less information (Keim, Designing Pixel-Oriented Visualization Techniques: Theory and Applications, 2000).

4.2 Interaction Techniques

Interaction techniques include mapping, projection, filtering, zooming, interactive linking and brushing. These techniques allow direct interaction with the visualization and allow it to be directly and dynamically changed according to the exploration objectives (Son & Calitz, 2004), they also make it possible to relate and combine multiple independent visualizations (Keim, Information Visualization and Visual Data Mining, 2002). Each of those techniques can be mapped to a type of interaction and executed at the request of the user.

This technique focuses on allowing users to change what they are looking at to either achieve a new vantage point over current data, or to introduce or change that data to enable new types of visual exploration. An example of this could be when users looks at a map they can move the map around to view a new area and they can zoom in or out depending on the detail they want. Another example can be seen is software like Microsoft's Photosynth (Microsoft Corporation, 2006) that allows users to visualize any number of photos. Users can interact by moving images and zooming in or out.

4.3 Distortion Techniques

Distortion techniques can be divided into two sets, Overview + Detail and Focus + Context being the most researched. The objective with distortion techniques is to aid the data exploration process by providing a way to focus on detail while preserving the overview of the data, it generally does this by displaying portions of the data with a high level of detail while showing the rest with a lower level of detail. There are also two types of distortion techniques, dynamic and interactive depending on whether the changes to the visualization are made automatically or manually by the user (Keim, Information Visualization and Visual Data Mining, 2002).

4.3.1 Overview + Detail

Distortion techniques revolve around the need to present detailed information while maintaining the overall view intact. The way researchers have solved this is to include specific distortion techniques like scroll and zoom functions to let users navigate and detail specific parts of the visualization, but to do this is complex and difficult and they also lose the global context when examining details and vice versa. One example (Capin, Pulli, & Akenine-Möller, 2008) shown in Figure 5, this approach separates the visualization into two windows, one with a detailed view and the other with a general (context) view. It fails to work in mobile devices because of the lack of screen real estate makes it difficult to see the two views.

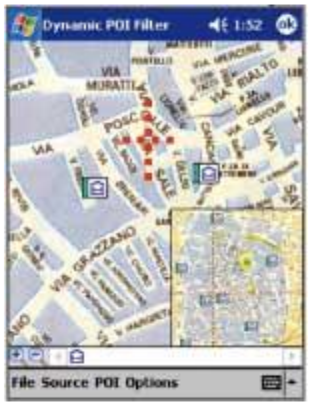


Figure 12 – a traditional Overview + Detail visualization (Chitarro, 2006)

4.3.2 Focus + Context

Another solution is Focus + Context (Capin, Pulli, & Akenine-Möller, 2008) unlike the previous one this provides context and detailed information simultaneously in one view, the most famous and used example of this is the fisheye view that magnifies objects in the users focal attention and decreases the size of the rest according to their distance from the focal point. These solutions are always based in this principle the view-point is manipulated in such a way as to enable seeing important items in detail while pre-serving the broader context in which the object belongs.

Many implementations of the fisheye view exist for mobile devices although it is not a panacea for visualization distortion other solutions have appeared using the same concept like the rubber sheet (Sarkar, Snibbe, Tversky, & Reiss, 1993) seen in Figure 16, where the user could select an area of interest and stretch it to the desired size.

Mackinlay et al. (Mackinlay, Robertson, & Card, 1991), proposed a perspective wall seen in Figure 13; it displays information items on a band with a focus region in the center and with context regions in both sides, with a band set at an angle. Robertson and Mackinlay (Robertson & Mackinlay, 1993) improved on the previous and the fisheye view by introducing the lens metaphor to visualize large documents. That lens works very much like a magnifying glass, the information items in the center are in focus with a gradual reduction in magnification closer to the edge, the lens could be moved and the magnification changed by moving the lends in the Z plane. Rao and Card developed the lens table (Rao & Card, 1994) it is similar to the rubber sheet metaphor but using discrete regions of focus delimited by rows and columns to allow users to view information from very large tables in context.

The Halo technique (Baudisch & Rosenholtz, 2003) seen in Figure 17, uses circles to introduce objects into the display region even if they are outside of it.

Magic Eye View (Kreuseler, Lopez, & Schumann, 2000) seen in Figure 14, is specifically tailored for hierarchies and works by laying a hierarchy on a hemisphere with the root at the pole the rest are then projected down into the plane below. Focus on a particular region can be accomplished by shifting the viewpoint around on the plane. Colors are used to distinguish between levels in the hierarchy. Ghinea et al. (Ghinea, Heigum, & Fongen, 2008) proposed an improvement to the original Magic Eye View seen in Figure 15, particularly suited for mobile devices that take into account the different aspect ratio of these devices.

Other approaches are visual references to off-screen areas or entities; this augments the detailed view with interactive visual references to off-screen entities related to the same context as what is in the detailed view. With this solution even without showing the general view, users are aware of its existence. Good examples of this are arrows or Halos mentioned above, while the user looks at a particular detailed view circles can indicate other related views.

Intuitive navigation techniques is another approach, although the name is misleading, it is based around the traditional scroll and zoom functions but takes into account the particularities of mobile devices. Examples of this are ZoneZoom (Robbins, Cutrell, Sarin, & Horvitz, 2004) seen in Figure 18, it automatically divides the screen into portions each connected to a button on the devices keyboard, when the user presses one of the buttons an animated zoom and pan of the corresponding portion is made. The main advantage of this approach is the use of the users' familiarity with mobile phone's keyboards where the main flaw is the limited number of portions on the screen, a possible fix to this would be to use this to navigate an ontology, they would need to create a hierarchical nodes or a tree of what needed to be represented and each portion would serve as an intermediate note of that ontology showing general data of that node until the user reaches one of the leafs for a more detailed view.

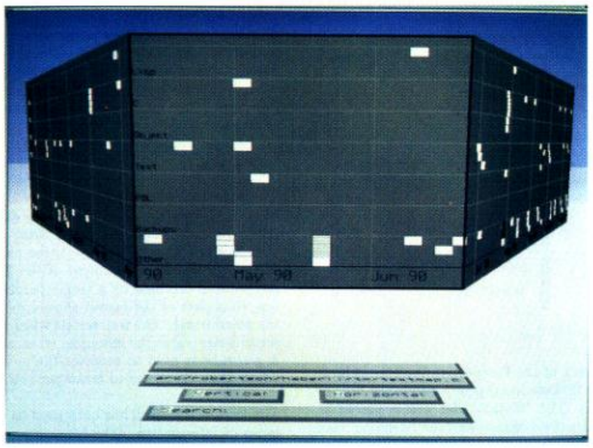


Figure 14 – perspective wall visualization example

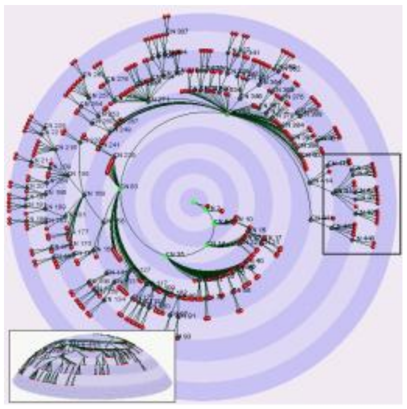


Figure 13 – Magic eye view visualization example

Figure 15 – Mobile Magic eye view example

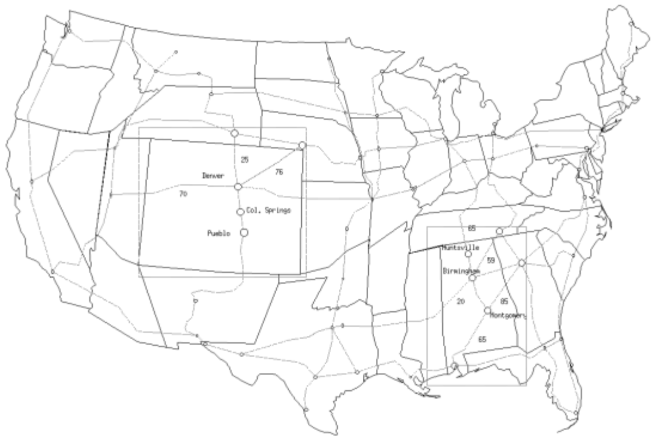


Figure 18 – Rubber sheet visualization example

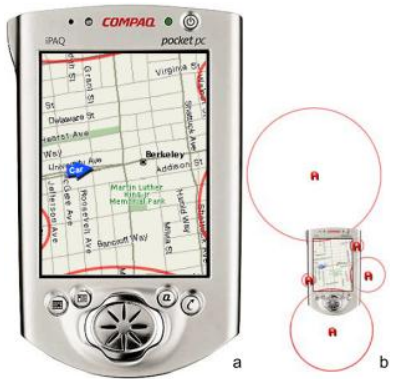


Figure 17 – Halo visualization example

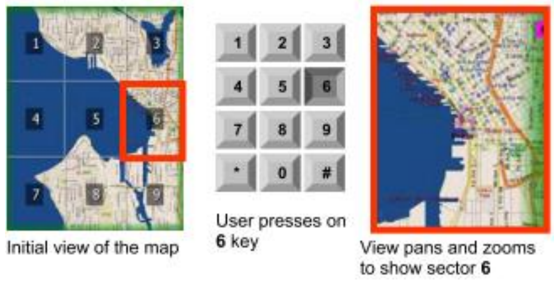


Figure 16 – ZoneZoom visualization example

4.4 Exploration Techniques

Many techniques are available in visualization exploration; those techniques can be divided into five groups, geometric projection, icon-based, pixel-based, hierarchical and graph-based (Keim, Designing Pixel-Oriented Visualization Techniques: Theory and Applications, 2000).

4.4.1 Geometric projection

Geometric projection includes scatterplot matrices, coplots, landscapes, prosection views, hyperslice and parallel coordinates. Geometric projections are used to visualize multidimensional data sets; this technique also includes many statistical methods like component analysis, factor analysis and multidimensional scaling. They aim at finding interesting transformations of multidimensional data sets. Hyperslice (Wijk & Liere, 1993) seen in Figure 19, is one of the most well known geometric projection, it maps n dimensional data by using n equidistance axes corresponding to each dimension and are linearly scaled from the minimum to the maximum value of the corresponding dimension. This type of visualization isn't used much in mobile device spaces, one possible cause may be hardware limitations.

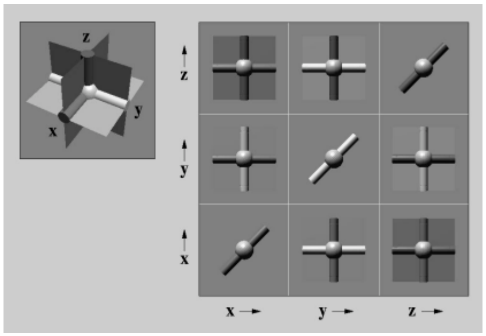


Figure 19 - Hyperslice visualization example

4.4.2 Icon-based

Icon-based include stick figures seen in figure 20, shape coding and color icons. This technique relies in the use of arbitrarily created icons and mapping the value of an attribute with a feature of that icon, like color, shape orientation, etc (Mazza, 2004). The icons are plotted in graphs or matrices so that the analyst can extract information by viewing the data objects as a whole (Sachinopoulou, 2001).

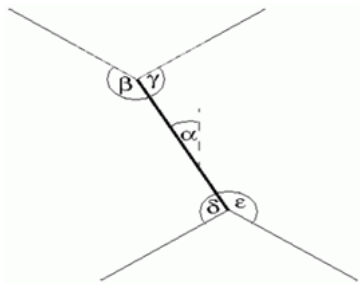


Figure 20 – Stick figure visualization example

4.4.3 Pixel-based

Pixel-based like seen in figure 21 include spiral, recursive pattern, circle segment. These have obvious uses in mobile devices. Keim (Keim, Designing Pixel-Oriented Visualization Techniques: Theory and Applications, 2000) details this technique to some depth. The idea of pixel-based techniques is to map each value to a colored pixel and present the data values belonging to attributes in a separate sub window. The advantages of this technique are that since each pixel is mapped to one value the designer can maximize the amount of information presented to the user and in low resolution small screens that is vitally important. There exists a class of special pixel-oriented techniques, the query-dependent techniques; the idea of these is to visualize only the data which is relevant in the context of a specific query. Pixel oriented techniques divide the screen into multiple sub windows each for one attribute or dimensions of the data except for query-dependent techniques where an extra sub window is needed for the queries. In each sub window the values are arranged according to an overall general sorting those can be data-driven or if the data is query-dependent it can be query-driven. Dependencies, relationships and patterns between dimensions can be seen by relating corresponding regions in the multiple windows.

There are various concerns when developing this kind of exploration technique:

- The use of color, each value must be mapped to a color and that choice is very important, especially in mobile devices that could have a limited color palette are subject to various lighting conditions that can influence how people perceive colors (Ware, 2004). Also the fact that different devices can have different color ranges, a problem addressed by Stone et al. (Stone, Cowan, & Beatty, 1988) where they describe a process of gamut mapping designed to maintain color appearance between multiple devices. Another problem with the use of color are color blind people, one solution for this is to use color sequences that vary mainly on a black-to-white scale (this is also helps detect many more noticeable differences in both color blind and non color blind people (Ware, 2004)) or on a yellow-to-blue, these will still be clear even for color blind people (Ware, 2004).
- How to organize or arrange the pixels in the sub windows; this is very dependent on the data and task of the visualization so it has to be created to suit a particular purpose.
- The shape of the sub windows this is important because attributes or dimensions can have different amounts of data and simply making the sub windows a rectangle can lead to wasted space and the data can become quite distance from each other thus making it difficult to find interesting relationships between them.
- How to order the sub windows for their attributes or dimensions, in some applications that is not a problem for there is no natural ordering of the dimension, but to detect dependencies between dimensions it is always best to place related dimensions next to each other.

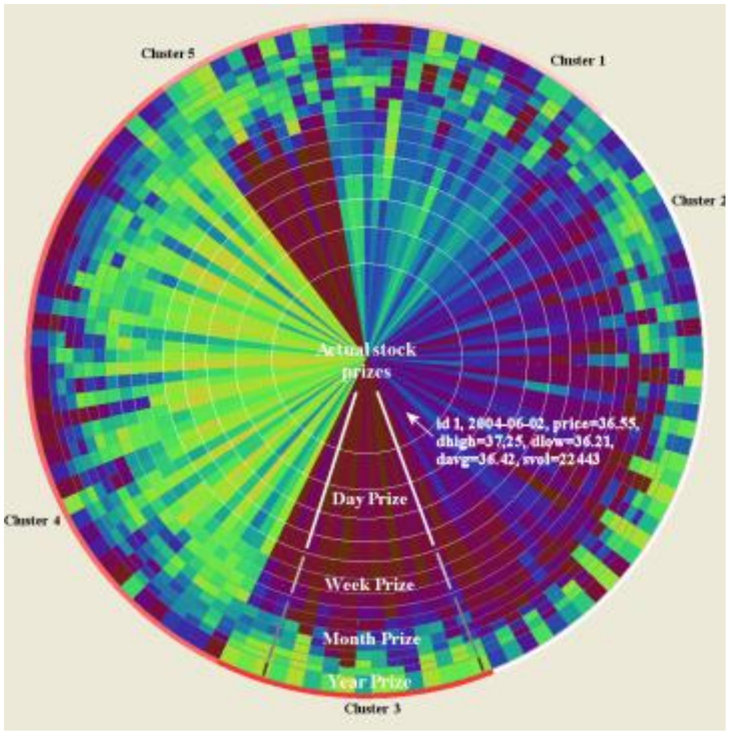


Figure 21 – Pixel based visualization example. (Keim, Schneidewind, & Sips, Scalable Pixel based Visual Data Exploration, 2006)

4.4.4 Hierarchical visualizations

Hierarchical visualizations include dimensional stacking, treemap and cone-trees seen in Figure 23. These can also be viewed as a graph, which will be detailed in the next section. A lot of solutions and problems affect both types of visualization exploration techniques, but there is still a very valid reason to use hierarchical visualizations and that is to represent an ontology. Ontology is an "explicit specification of a conceptualization" where conceptualization is an abstract simplified view of the world. Like graphs it contains objects, concepts and other entities of interest and their relationships (Katifori, Halatsis, Lepouras, Vassilakis, & Giannopoulou, 2007). In order to represent an ontology the visualization has to support classes or entities, relations, instances and properties (Katifori, Halatsis, Lepouras, Vassilakis, & Giannopoulou, 2007). There are several examples of these kinds of visualizations in mobile devices, Hau et al. (Hao & Zhang, 2007) created a method they called Radial Edgeless Tree (RELT) seen in Figure 22, to visualize hierarchical data on mobile devices. RELT recursively partitions the display to maximize screen usage. They use adjacency and direction to represent relationships between nodes. The root node starts at the upper left corner of the screen and owns the entire display, he can divide his space accordingly to each child and they will in turn own their sections to further divide for their children and so on. The result is the whole screen in divided into regions in somewhat of a spider web pattern the nodes in the same level are in the same distance to the root (upper left corner) lower level nodes are progressively closer to the bottom right corner. Yoo et al. (Yoo & Cheon, 2006) propose a method of maximizing resources and speed of recognition; they used a fisheye view on a radial layout, placing the information the user needs most in the center and the rest around it according to the degree of interest. One particularity of their approach is the use of animation to improve user's recognition of information.

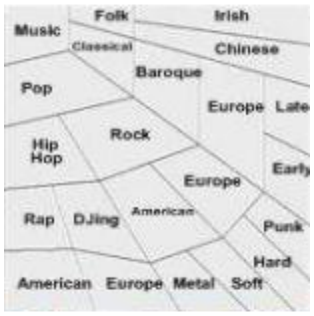


Figure 22 – RELT visualization example (Hao & Zhang, 2007)

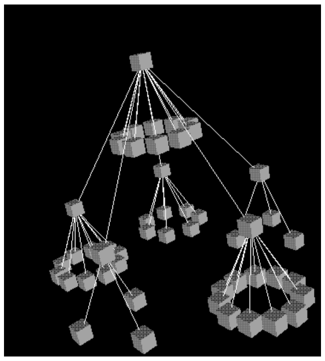


Figure 23 – Cone tree visualization example

4.4.5 Graph-based

Graph-based visualizations include cluster as seen in Figure 28, symmetry optimized, and of course all the hierarchical techniques. These can be used to visualize an immensely broad range of data; anything that possesses some sort of relation can be visualized using these methods. The main issues with this technique is size, this is a kind of visualization that naturally needs a lot of space and since it is used on a lot of situations in which the node count can reach thousands, a large number of elements can compromise performance and in the case of mobile device displays quickly reach the screen's limit. Even with current methods of layout and display of elements it would be impossible to discern between nodes and edges. When dealing with visualizations with a lot of elements displaying the entire graph might give an idea of the overall structure or a specific location in it, but it doesn't have many other uses besides that (Herman, Melançon, & Marshall, 2000). One solution for this is to restrict the quantity of information on the screen, one of the these solutions is called Magic Lenses (Bier, Stone, Pier, Buxton, & DeRose, 1993) seen in Figure 26, it filters the amount of data displayed on the screen, another is using zoom, arranging the information in 3D space and zooming on the node we are looking for, that might present context problems if the user doesn't know how the graph is structured or if it has many elements the search could be long and frustrating. Kamba et al (Kamba, Elson, Harpold, Stamper, & Sukaviriya, 1996) presented a way of saving screen space regardless of visualization technique used. They suggest using pop up components that remain hidden until needed at which point they become partially transparent to preserve context. Furthermore each interaction element can also be arranged on the screen to retain maximum work area.

The issue of size can be further divided into two aspects, layout and complexity, these are especially important in mobile devices, if the designers doesn't plan his layout properly they can waste a lot of space that could be otherwise used to show more information. Several graph layout approaches exist like spanning trees (Herman, Melançon, & Marshall, 2000) seen in Figure 24, those could be a solution to gain predictability of the layout (Herman, Melançon, & Marshall, 2000), 3D layout (Herman, Melançon, & Marshall, 2000) seen in Figure 25, where the extra dimension can provide more space to draw the visualization and could ease the problem of displaying large structures, however this is not without its problems, objects can occlude one another (although this could probably be solved with transparencies) and it is also harder to select the best view from a 3D space. Hyperbolic layout (Herman, Melançon, & Marshall, 2000) seen in Figure 27, has been developed with graph visualization and interaction in mind, and can be implemented in both 2D and 3D and provide a distorted view of a tree that resembles a fisheye view, this view makes it possible to interact with it, even on very large graphs, the downside is that it’s much harder to understand how it is created than with simpler techniques.

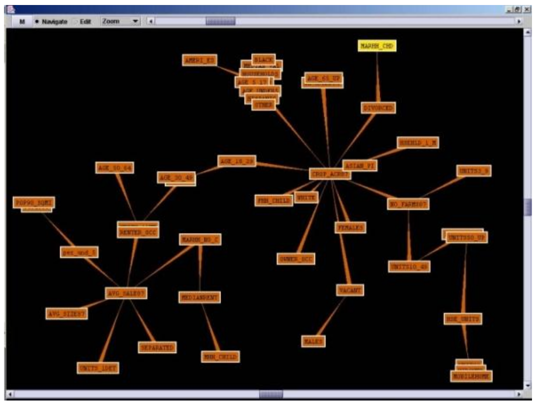


Figure 26 – spanning tree layout example (Herman, Melançon, & Marshall, 2000)

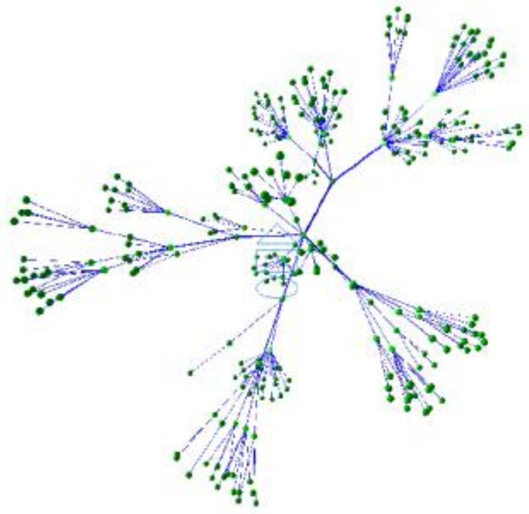


Figure 28 – 3D layout example (Herman, Melançon, & Marshall, 2000)

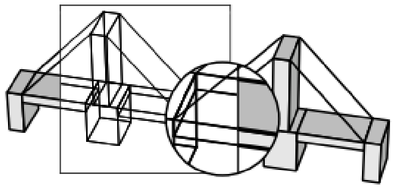


Figure 25 – Magic lenses example, 3D shaded bridge showing a 3D wireframe lens and a 2D magnifier (Bier, Stone, Pier, Buxton, & DeRose, 1993)

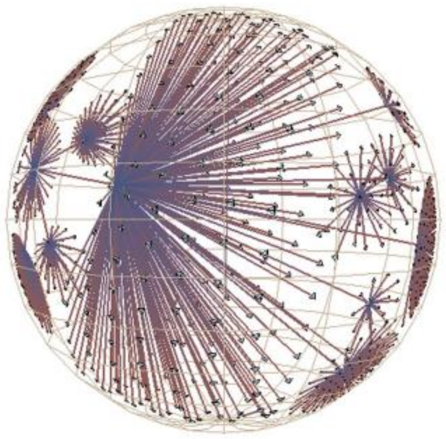


Figure 27 – Hyperbolic layout example (Herman, Melançon, & Marshall, 2000)

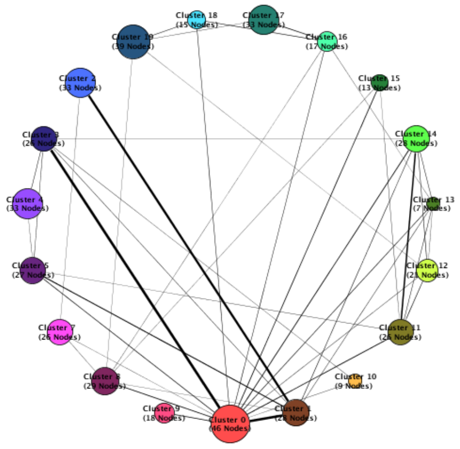


Figure 24 – Cluster graph visualization example

4.4.6 Exploration Process

Exploration techniques have another aspect that deserves attention. Most of the research done in exploration techniques is about how to represent the data, which it obvious given that the main focus of this technique is to help extract information from a visual depiction of said data. But there is another aspect often overlooked the exploratory aspect of visualization, the process itself, the information content of the visualization process, the results, history and relationships between those results. Jankun-Kelly et al (Jankun-Kelly, Gertz, & Ma, 2007) developed a general model to capture the salient aspects of the exploration, such as how the results were generated and how they were used to generate new results, a representation to document the contents of the visualization session for later analysis and dissemination and a general framework to manage instances of the model. There are several advantages to capturing the visualization process, the user can record their sessions at a much higher level and create graphical representations or analyze the process. This also allows designers to share results and process information between different visualization interfaces. Also because the process is represented in a formal model, it is open to a multitude of analyses. Because both the model and framework are general they are very powerful and can be applied to a wide range of visualization problems. One advantage of this, especially for mobile devices, is that this will assist users to explore, communicate and understand their results, this is very important because of the attention problems inherent within mobile devices as discussed in the next section. Also because people use mobile devices mostly to communicate between themselves and having a way to add visualizations to that communication process can be very advantageous. Their model is what they call a “P-set of visualization exploration” that encapsulates the interactions a user can have with a visualization system and how these interactions are part of the greater exploration session. Note that P-sets are nothing more than parameter sets. When the user explores the visualization he repeatedly forms a set of parameters to apply to the visualization and when applied create a result. The model is used to know how the new result is related to the previous one. It basically encapsulates the way a user can interact with the visualization and how that interaction is part of their exploration session. To be able to model the exploration session they found four key elements for a visualization session.

- The visualization transform says how to bind the data to visual elements and how to render them.
- The visualization parameters are all the arguments for the previous element
- The visualization results are is the outcome of applying the previous parameters to the visualization transform and are uniquely identified by them.
- The derivations these are the most important as they say how the results were created, how the user created new parameters from the old and how they were applied the old p-set model to create the new and which p-sets were combined with which transform to create the new one.

Their next step was to create a way to share the visualizations captured by the model, for that they used XML, for its extensibility, effectiveness and because it is easily sharable, all of the previous elements like visualization transform, parameters, results and derivations are saved along with supporting information into the file.

4.5 Problems with mobile visualization

Unfortunately some of these techniques can’t be used on mobile devices, when one designs a visualization one must take into account the environment and the level of attention. The environment is particularly important mostly because of lighting conditions, with a desktop it can be easily controlled, but that is not true for mobile devices (Ware, 2004) (Chitarro, 2006). Mobile devices are meant to be used on the move and in a variety of environments and lighting conditions; those have a prominent impact on the visualization. Most screens have problems working in a very bright environment, that makes colors easily mistakable with one another (Ware, 2004). This is a very important aspect to be mindful of, many visualizations use color to distinguish visual elements and if the user cannot distinguish between colors effectively the effectiveness of the visualization is considerably diminished. In extreme cases it might even lead the user to false conclusion about the data. A possible solution for this problem can be OLED screens.

Attention is another important aspect, users can use the mobile device outside their working environment and they might not be able to devote the attention needed to the visualization or otherwise might be interrupted by an external event requiring their attention (Chitarro, 2006). This lack of attention is very important, users could for example interact with the visualization in a way they were not aware of and become confused by the result, this leads to unpredictability and eventually frustration.

With the above a designer doing a particular visualization for a mobile device must also pay attention to the following (Chitarro, 2006):

- Mapping: the way the information is visually mapped. A good mapping between objects and their relations is very important; they must be defined and consistent throughout the application.
- Selection: the presented information must be relevant and sufficient for the task. Showing too little will lead to wrong conclusions, too much and it will be difficult to make sense of all the information, this is especially critical in mobile devices because of their small screens and the fact that they are used in open and hard to predict or control environments (such as the streets for example).
- Presentation: they must decide how to present the information on the screen. Even with a good visual map of the data the application can be ineffective when the screen is small.
- Interactivity: they must decide on how and with what tools to navigate the visualization, this is important to increase user engagement and increase exploration abilities.
- Human factors: they need to take into account interference with human perceptual and cognitive capabilities, certain visualizations can create a visual illusion and the user reads data that is not there. User must be able to quickly identify and evaluate a visualization.
- Evaluation: they need to test the visualization’s effectiveness on users.

These are only a few of the issues that designers must take into consideration when designing a visualization for mobile devices.

Presentation is of special importance for a mobile device (Chitarro, 2006). With their small screens mobile devices can very quickly have too much information and this is a danger in any non trivial visualization for which distortion techniques play a vital role in solving.

There are many concerns facing designers when developing a visualization directed at mobile devices, this is one reason supporting the development of a framework, by removing most of the low level complexity of creating a mobile visualization developers and designers can focus more time on developing an effective visualization taking care of the pitfalls listed above.

4.6 Visualizing different types of information

Over the years there has been a lot of research on how to visualize certain types of information such as text, maps and abstract data in mobile devices. This literature is briefly reviewed in the subsequent sections.

4.6.1 Text

Visualizing text is one of the most important visualizations known, almost all information in existence is represented in text form and a lot of research into it has been made over the years. The leading presentation of text is horizontally in one line, this is the most obvious and trivial to create and is also the most familiar and accepted by users even though it is not particularly efficient in mobile devices with their small screens. Another is rapid serial visual presentation (RSVP) that presents small chunks or a single word of text appearing in a sequence at a single location as seen in Figure 30. Another approach is a variation of the fisheye view using horizontal text and focusing on the line the user is reading expanding it while at the same time gradually pushing back the rest, this keeps the general view of the text present at all times and focuses on what the line the user wants to read and it's immediate neighbors (Chitarro, 2006) an example of this can be seen in Figure 29.

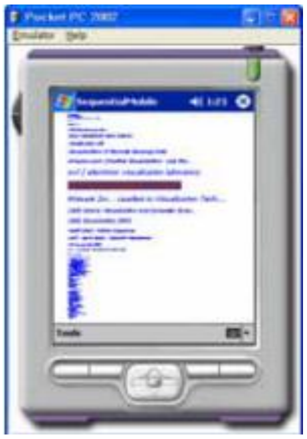
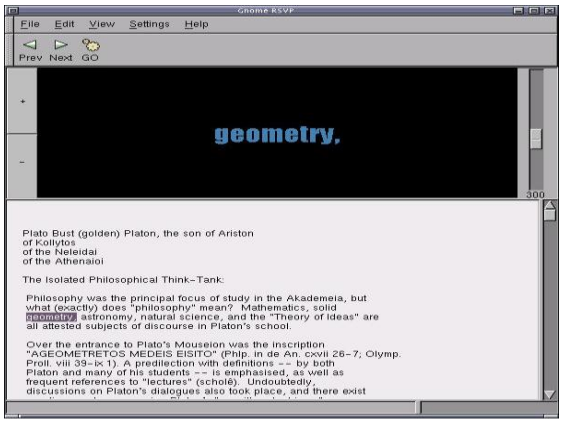


Figure 29 – Fisheye visualization applied to text



geometry,

Plato Bust (golden) Platon, the son of Ariston
of Kollytos
of the Neleidai
of the Athenaioi

The Isolated Philosophical Think-Tank:

Philosophy was the principal focus of study in the Akademeia, but
what (exactly) does "philosophy" mean? Mathematics, solid
geometry, astronomy, natural science, and the "Theory of Ideas" are
all attested subjects of discourse in Platon's school.

Over the entrance to Plato's Mouseion was the inscription
"AGEOMETRETOS MEDEIS EISITO" (Phlp. in de An. cxvii 26-7; Olymp.
Proll. viii 39-ix 1). A predilection with definitions -- by both
Platon and many of his students -- is emphasised, as well as
frequent references to "lectures" (scholê). Undoubtedly,
discussions on Platon's dialogues also took place, and there exist

Figure 30 – RSVP visualization example

4.6.2 Pictures

Visualizing pictures is growing in interest especially on the internet. Hao Liu et al (Liu, Xie, Ma, & Zhang, 2003) propose an adaptation of the rapid serial visual presentation seen in Figure 31. A browser for images that uses an image-processing algorithm to identify possible interests the user might have like faces. Microsoft's Photosynth is also a good example of a picture visualization. It allows users to interact with their pictures and form virtual scenes composed of a multitude of images related to each other.

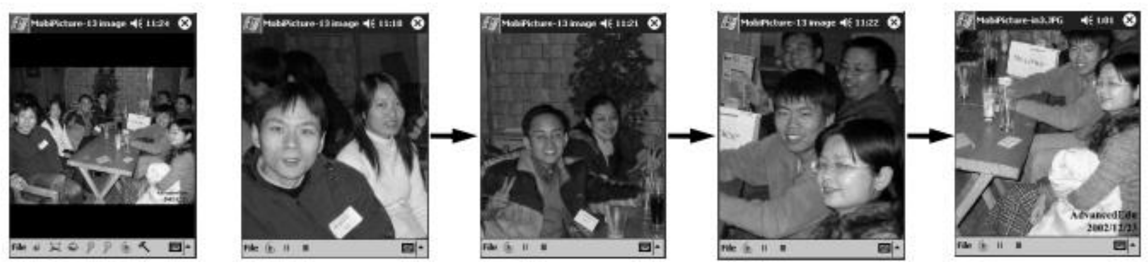


Figure 31 – RSVP adapted for images

4.6.3 Maps

Visualizing maps has been a very busy area of research for the past few years with the advent of GPS enabled mobile devices; there is a lot of potential in these kinds of visualizations from presenting maps highlighting potential areas of interests for the user to orienting them in an unknown city. Lee et al. (Lee, Forlizzi, & Hudson, 2005) propose a contextually optimized in-vehicle navigation system called MOVE seen in Figure 32. The objective was to present a map to help users navigate while driving while minimizing the amount of attention needed to the navigational interface. They reached this objective by abstracting the route users needed to take and dividing it to a set of objectives; they eliminated needless information and presented meaningful landmarks. They finally applied some distortion techniques to present detailed information on the user's current location as well as an overview of the whole route.

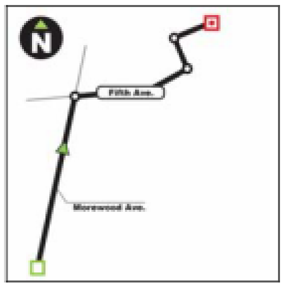


Figure 32 – MOVE route map visualization (Lee, Forlizzi, & Hudson, 2005)

4.6.4 Physical objects

Visualizing physical objects is also an area of research with great interest especially for augmented reality applications where through the camera users can see super imposed on the world a virtual representation of an object to make it look like it's actually there, Burigat and Chittaro (Burigat & Chittaro, 2005) developed an interesting system seen in

Figure 33, using this idea where they use a device with a GPS to track the current location of the user to show virtual monuments on the street like they were real. While interesting this idea is limited by the processing power of the devices, another approach is seen in (Knödel, 2008) by Knödel where they offload the rendering work to a more powerful computer and transmit the animation frames to the mobile device resulting in a visualization with a considerably more quality, the problem they face is that they are dependent on a network be it personal or the internet and if the quality of service is not guaranteed the visualization suffers a lot. Aaltonen and Lehikoinen (Aaltonen & Lehikoinen, Exploring augmented reality visualizations, 2006) took it one step further and developed an augmented reality visualization that applies three popular techniques, pan and focus, overview and detail and focus+context, they also provide examples of these techniques using AR.

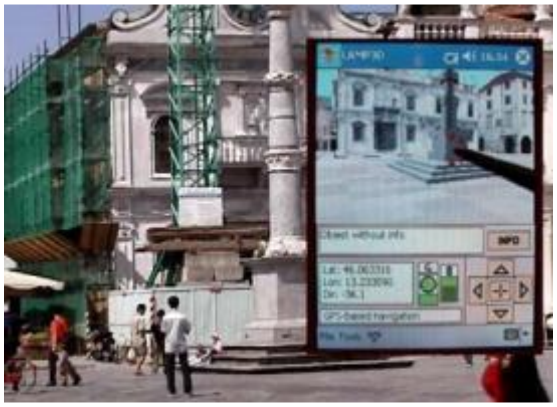


Figure 33 – visualizing real monuments with a virtual representation (Burigat & Chittaro, 2005)

4.6.5 Abstract data

Visualizing abstract data is also an area with a lot of research, all the rest that doesn't fit in the previous categories tends to go here and this is especially important in scientific visualization. Here solutions tend to involve using multiple levels of detail and fisheye views. An interesting visualization for abstract data that can potentially solve the problem of lack of screen space in mobile devices is one created by Meiguins et al. (Meiguins, et al., 2006) seen in Figure 34, where they use augmented reality to represent multidimensional information and create a visualization tool capable of zoom, filter, detail on demand and relationships to create a visualization for abstract data superimposed on the feed of a camera. The way the previous solution could help in solving the screen space problem is by attaching and relating data to real world objects and performing zoom, filter, detail and relationship functions by walking and interacting with objects in the real world, in another words giving meaning to an artifact that is shared by all those participating in the visualization, this approach is very familiar to those who are know the embodied interaction paradigm introduced a few years ago by Dourish (Dourish, 2001).

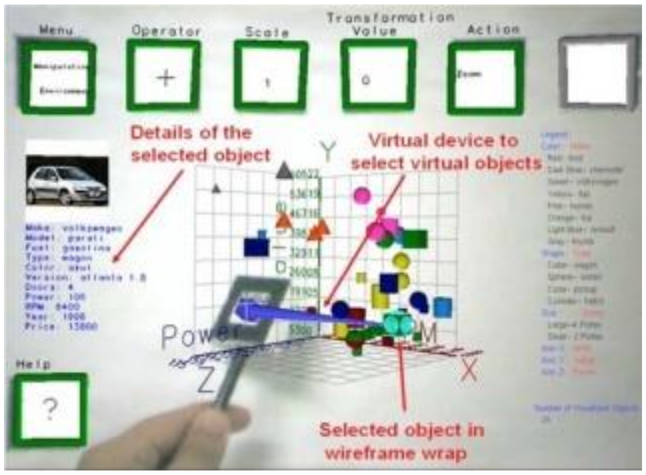

Menu
Operator
Scale
Transformation Value
Action
Details of the selected object
Virtual device to select virtual objects
Help
Power
Y
X
Z
Selected object in wireframe wrap

Figure 34 – Visualizing abstract information using AR (Meiguins, et al., 2006)

4.7 Conclusion

When developing for a mobile platform developers and designers need to take into account more than if they were developing for more traditional platforms. By providing developers and designers with a general, flexible and extendable base to develop visualizations they can focus most of their attention to the problem at hand and how to apply the three visualization techniques of exploration, interaction and distortion to build an effective visualization for any type of data, be it text, images, maps, physical or abstract. Glaze aims to provide a general framework where they can apply any method of any technique to the visualization they wish to create. This flexibility allows for simple experimentation and reuse. And by having an extendable base, to allow for customization to better fit a specific problem.

This chapter described all three visualization techniques, exploration, interaction and distortion, with corresponding methods and implementations. It talked about problems facing mobile visualization development how different types of information are visualized.

CHAPTER V

GLAZE VISUALIZATIONS

This chapter presents and describes test visualizations made with Glaze called Night vision and Information Overlay. These where developed using observations and semi-structured interviews had been used to elicit requirements for them.

For both visualizations a description of the development process including requirements, development and user testing is provided as well as a detailed description of how those visualizations were made with the Glaze framework.

5.1 Observations

The first stage of visualization development was dedicated to observation of people using their mobile phones in their daily lives in order to gain a better understanding of the purpose and uses people have for their mobile devices.

The observation was conducted at about lunch time, in a mall. The reason for this was to observe people when they are relaxing and off work hours, also the mall is a place with a high concentration of people and also a meeting place, this increases chances of finding someone using the phone. The objective of this phase was not much on trying to elicit potential requirements for the project, but to get an idea of how people use their phones in their day-to-day lives.

The most significant observations included:

- Some people walking around with their phones in their hands even though they have pockets.
- No one walks and uses their phones simultaneously except in a call.
- Some people use it as a watch.
- Some people use it to play a game and writing SMSs, none of them were moving, they were either sitting down or standing.
- Every time someone has to look at the phone they always look down to it, never in an angle where they could probably see what's beyond the phone.
- Some people were extremely fast typers, those tend to use both hands and are predominantly in their teens.
- Those using the phone for something that required the keyboard didn't show signs of difficulty.
- Saw some people in the middle of a call looking at the screen of the phone for a second to see if the call was disconnected and then immediately putting the phone next to their ears.
- Camera phones where predominant especially with younger people.

This preliminary data showed people have embraced the mobile device as more than just a phone, using it for entertainment, as watches and as a means to communicate other than with voice. Also those that take advantage of those features are predominantly young and extremely skilled with their use. Independently of age people are very hesitant to move and look at the screen simultaneously, with good reason our eyes are the main source of information about our surroundings and focusing on a screen when walking around on the street diminishes our ability to focus on what is around us.

These observations suggest that the projects focus should be maintained on young and experienced mobile device users for the reason that they are the most open to new technologies and ideas, and that visualizations that aid our visual system in visualizing our surroundings in more efficient ways can be a good general area for developing test applications.

5.2 Interviews

In order to test the framework a set of test visualizations that tried to use as much of all features offered was developed. For these visualizations to be of potential value to users a set of 18 semi-structured interviews, targeted at people with experience handling mobile devices were planned with the objective of being an ideation process for potential interesting visualizations. Very simple and deliberately open ended questions were asked with the objective of eliciting device features and applications unrelated to current technological capabilities and limitations and to also allow the use of the interviewees imagination. Answers can be seen in the appendix section 1.

The questions were:

- What do you love about your cell phone?
- What do you hate about your cell phone?
- What is your dream feature for a cell phone?

In order to quickly find a common trend with the data the answers were prepared by removing common words and isolated letters and finally visualized using a Tag-Cloud visualization seen in Figure 35 to present an overview of the most asked for features and ideas. Results focused a lot in hardware like battery, keyboard and reception, but a closer look at the results also revealed words that allowed for an interesting visualization, like camera, information, everywhere among other more concealed ones that inspired Information Overlay detailed in section 5.4.

A detailed analysis of the answers was also made in search of original and unique ideas. One was found the Night vision detailed in the next section. These candidates were selected for visualization development based on their novelty, potential utility and the frequency with which they were requested.

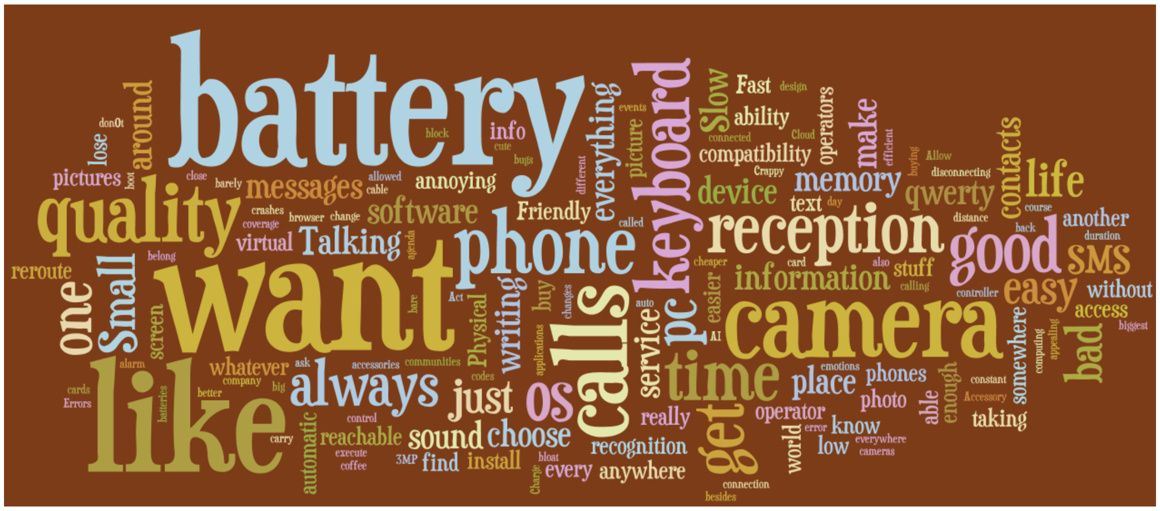


Figure 35 – Tag-cloud visualization of filtered answers

5.3 Night Vision

The night vision visualization can be thought of in many different ways. The classic night-vision example is the green or gray tinted view we often see in films, that cannot be achieved by means of software alone, we need to view infra-red light sources (all cameras capture infra-red) that are filtered by the device's own camera before giving developers access to the frame. To realistically achieve this without the aid of an external infra-red sensor would be with a simple hardware mod of putting a film negative in front of the internal lens to block visible light and allow infra-red to pass, but this would remove the need for a software component seeing as the video source would effectively be classic night-vision. What we are adopting is one of surrounding awareness, the objective of night-vision is essentially to allow people to see at night what is around them so they can navigate their environment as they would during the day, our solution is to present a virtual replica of what is around the user at any given time and replicate the user's movement and orientation in real space in virtual space to maintain coherency between the two. This allows users to navigate in any environment independently of ambient luminosity by simply looking at the phone.

The hardware used for this visualization, besides the phone, is a GPS and a digital compass. The GPS allows the visualization to track the user's position in the real world while the digital compass tracks where the user is looking at. By mapping position and orientation to a virtual equivalent we can replicate what the user sees in the real world in the virtual world. As for the model the more detailed it is the more it will look like its real counterpart and allow for easily identifiable landmarks.

Because we use the GPS the visualization does not work indoors, although this can be easily fixed, by changing the GPS input component to another that works indoors, like beaconing radio as seen in Pirates! (Björk, Falk, Hansson, & Ljungstrand, 2001). Another way would be to use a component that supported both GPS and beaconing radio that was capable of switching between both according to their availability.

5.3.1 Requirements

A second batch of semi structured interviews was conducted with a smaller number of young expert phone users to better understand what they would like to use these visualizations for and what they would like them to have, the questions were again very open and directed at a specific example in an attempt give the interviewees a mental picture of the visualization and also to elicit a set of requirements. Answers can be seen in the appendix section 2.

The questions were:

- Imagine an application that lets you see your surroundings at night when you have trouble identifying what is around you. What would be an interesting feature for this application?
- What would you like to do with an application like this?

In the first question interviewees focused on many popular ways of seeing at night, like night vision lenses, sonar, maps and just plain lights. But also mentioned were some interesting alternatives like the ability to view the environment in wireframe mode and the ability to see it like it was day time.

For the second question answers focused mostly on navigation, like finding their way home and avoiding objects.

Most of what was said can be easily solved by providing a 3D model of where they are with the only problem being making it as complete as possible. But by just representing buildings and streets users gain significant knowledge of their surroundings they can identify dangerous zones like roads; they can locate exits and businesses with minimal effort. By having their movements reproduced in the virtual scene they can also know at all times where to go in any situation.

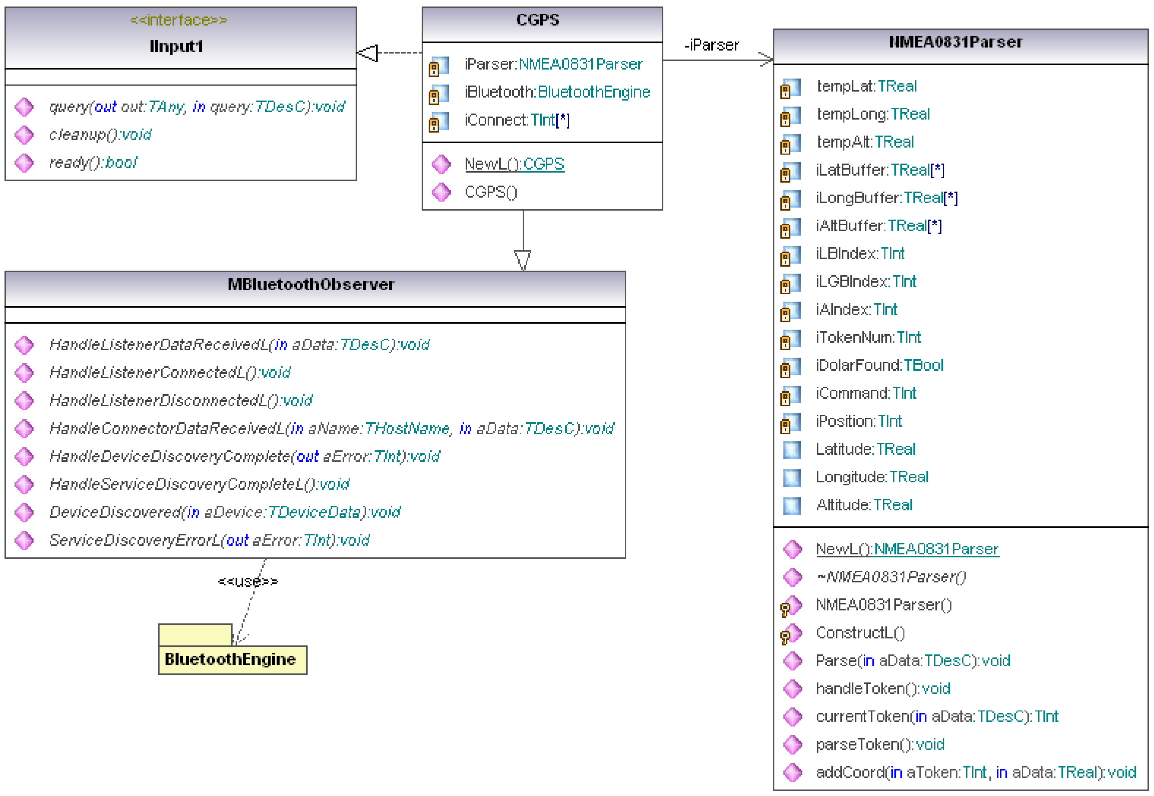
Taking into account the answers provided and the objectives of the visualizations the next set of requirements were created:

1. Accurate representation of buildings and objects of interest.
2. Virtual movement must match real world movement

5.3.2 Development

Development of the visualization started with development of the input components needed for the final application in this case the GPS, the digital compass and the Open GL ES front-end.

5.3.3 GPS input component



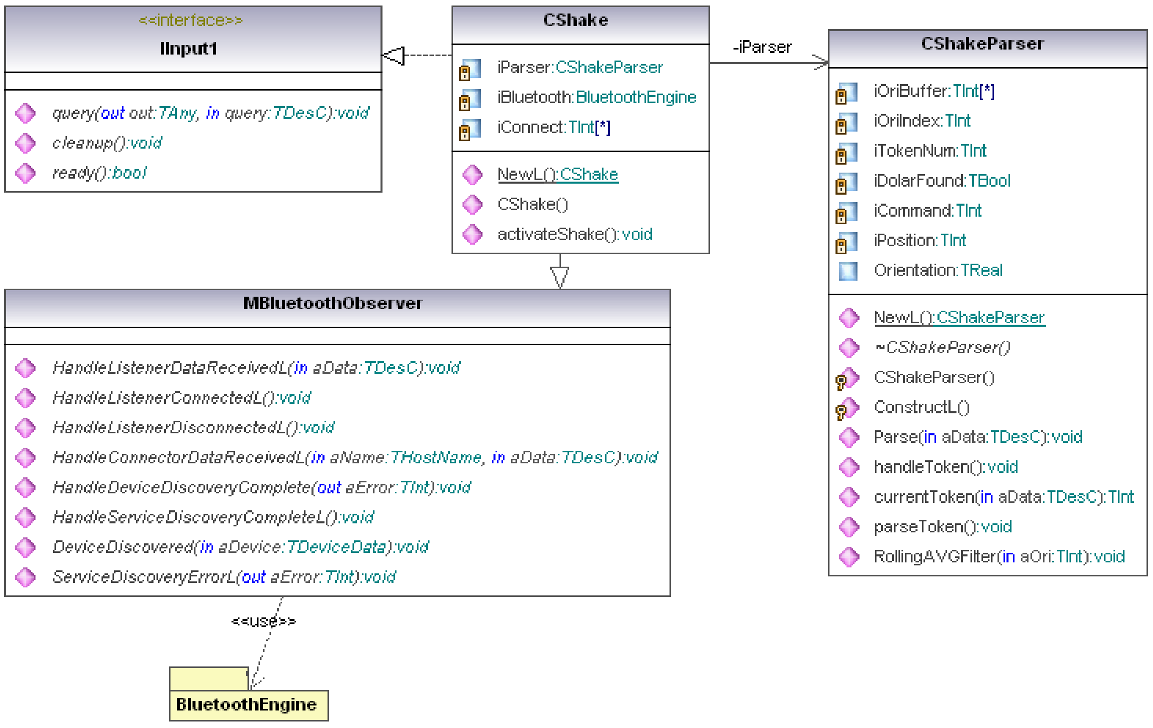
<<interface>>
IInput1
query(out out:TAny, in query:TDesC):void
cleanup():void
ready():bool
CGPS
iParser:NMEA0831Parser
iBluetooth:BluetoothEngine
iConnect:TInt[*]
NewL():CGPS
CGPS()
-iParser
NMEA0831Parser
tempLat:TReal
tempLong:TReal
tempAlt:TReal
iLatBuffer:TReal[*]
iLongBuffer:TReal[*]
iAltBuffer:TReal[*]
iLBIndex:TInt
iLGBIndex:TInt
iAIndex:TInt
iTokenNum:TInt
iDolarFound:TBool
iCommand:TInt
iPosition:TInt
Latitude:TReal
Longitude:TReal
Altitude:TReal
NewL():NMEA0831Parser
~NMEA0831Parser()
NMEA0831Parser()
ConstructL()
Parse(in aData:TDesC):void
handleToken():void
currentToken(in aData:TDesC):TInt
parseToken():void
addCoord(in aToken:TInt, in aData:TReal):void
MBluetoothObserver
HandleListenerDataReceivedL(in aData:TDesC):void
HandleListenerConnectedL():void
HandleListenerDisconnectedL():void
HandleConnectorDataReceivedL(in aName:THostName, in aData:TDesC):void
HandleDeviceDiscoveryComplete(out aError:TInt):void
HandleServiceDiscoveryCompleteL():void
DeviceDiscovered(in aDevice:TDeviceData):void
ServiceDiscoveryErrorL(out aError:TInt):void
<<use>>
BluetoothEngine

Figure 36 – GPS input component class diagram

Due to the relatively poor performance of the test device's internal GPS an external one was used that communicated using Bluetooth and sends NMEA sentences. Due to restrictions in the use of the native GPS library in the test phone one had to be built to parse the minimum amount of NMEA sentences in order to get global positioning coordinates (longitude, latitude and altitude above sea level), altitude above sea level being the hardest number to get an accurate reading was ignored in favor of geoid altitude that is constant for most of the test site.

Special care was taken on parsing NMEA sentences, a great number of sentences is sent by the device and quickly finding which are interesting and from those extract the values rapidly to save as much performance as possible was a concern.

5.3.4 Digital compass input component



<<interface>>
IInput1
query(out out:TAny, in query:TDesC):void
cleanup():void
ready():bool
CShake
iParser:CShakeParser
iBluetooth:BluetoothEngine
iConnect:TInt[*]
NewL():CShake
CShake()
activateShake():void
-iParser
CShakeParser
iOriBuffer:TInt[*]
iOriIndex:TInt
iTokenNum:TInt
iDolarFound:TBool
iCommand:TInt
iPosition:TInt
Orientation:TReal
NewL():CShakeParser
~CShakeParser()
CShakeParser()
ConstructL()
Parse(in aData:TDesC):void
handleToken():void
currentToken(in aData:TDesC):TInt
parseToken():void
RollingAVGFilter(in aOri:TInt):void
MBluetoothObserver
HandleListenerDataReceivedL(in aData:TDesC):void
HandleListenerConnectedL():void
HandleListenerDisconnectedL():void
HandleConnectorDataReceivedL(in aName:THostName, in aData:TDesC):void
HandleDeviceDiscoveryComplete(out aError:TInt):void
HandleServiceDiscoveryCompleteL():void
DeviceDiscovered(in aDevice:TDeviceData):void
ServiceDiscoveryErrorL(out aError:TInt):void
<<use>>
BluetoothEngine

Figure 37 – SHAKE input component class diagram

The test mobile device does not possess a digital compass, to solve this, a sensor pack called SHAKE described in (Williamson, Murray-Smith, & Hughes, 2007) was used and it provides not only a digital compass but a multitude of other imbedded sensors. It too communicates with the mobile device through Bluetooth using very similar sentences to NMEA allowing for reuse of the parsing class used in the GPS component. One difference between the two is SHAKES's need for configuration before proper use meaning the phone needed to send specific commands to activate and deactivate sensors as well as sending frequency.

5.3.5 Open GL ES front-end

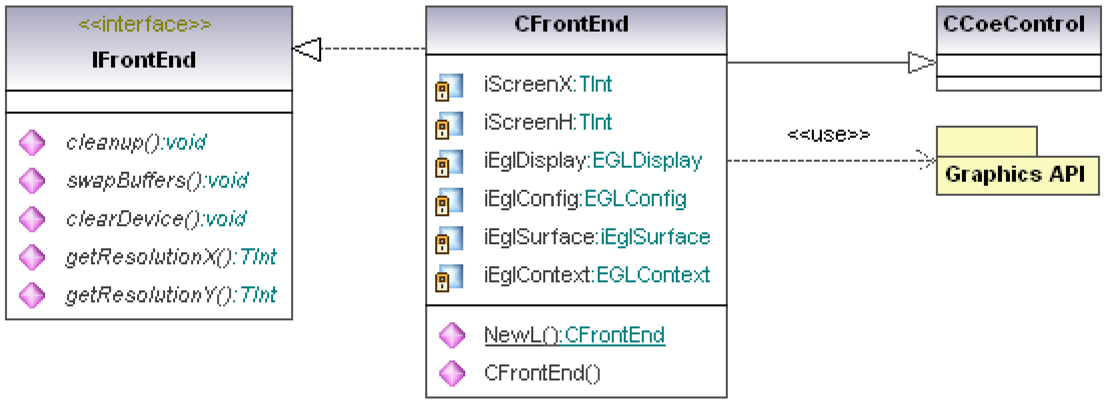


Figure 38 – Open GL ES front-end component class diagram

The Open GL ES API was selected to handle graphics specific detail. This choice has implications as was explained in chapter III section 3.6, the same rendering API must be used in both the front-end and the visualization.

How the system as a whole handles scene creation and screen rendering is defined within this component.

5.3.6 Visualization component

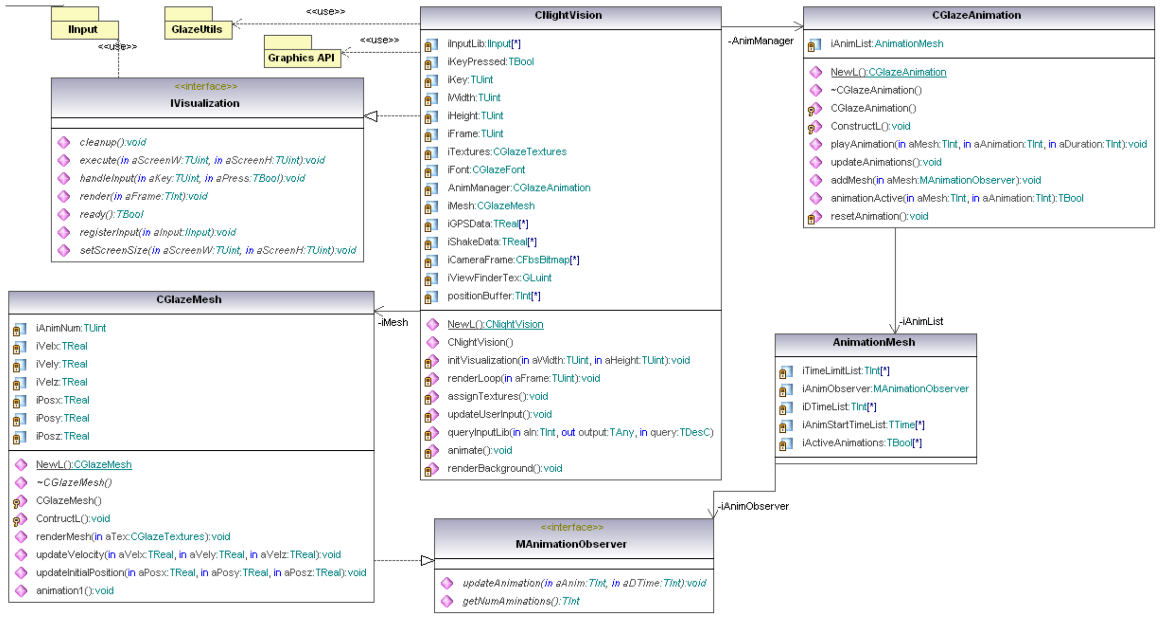


Figure 39 – Night vision visualization class diagram

The visualization component was the focus of the development. The main objective of this component is to apply visual mappings on the information provided by the input component to create visual structures or visual elements on the screen and to provide view transformations to change the view over the data. Here the data is the information from the GPS (latitude, longitude, and geoid altitude) and SHAKE (heading). The objective of the visualization is to map movement and rotation in the real world in a virtual world design to look similar to the real counterpart, with this in mind latitude is mapped to the x coordinate in virtual space, longitude is mapped to z and altitude is mapped to y, these coordinates where chosen to match Open GL ES convention, SHAKE's orientation is mapped to world rotation. It was noted in Chapter III section 3.5 that there were no concrete methods and structures to transform and organize the data coming from input-component that must be provided by the developers. Here we see this principle in action, there is an implicit mapping of information coming from raw data to visual structures even with no concrete methods to do this, it is made with or without a conscious decision from developers. In order for the visualization to visualize the data from the input-components this simply has to be done. As for view transformation just by mapping real world position and orientation to virtual position and rotation we are applying this, by just moving the user can get a new view of the information we wish to convey, the user’s surroundings, this again is applied implicitly by the developer consciously or not without the need for specific methods.

5.3.7 User Testing

A simple unstructured think-aloud method was chosen for testing this visualization. It consisted of asking the user to simply interact normally with the system and determine their own tasks, while still communicating their thought process. The test ended with a questionnaire about user opinions. Since this is a very new type of application with minimal interaction, it was more important to know how they would use it, if they would understand its use and their reactions. Also being a mobile application with a specific use context (outside) it was important to make the experience as close as possible to real life for potential users. User testing was made with 8 potential users. Each participant was allowed as much time as they needed to adapt and use the visualization as they felt necessary; as they used the visualizations observations on how they used the system were also taken. The questionnaire was kept to a minimum and with 4 direct questions. Answers can be seen in the appendix section 5.

The questions were:

- What’s the best thing about this visualization?
- What’s the worst thing about this visualization?
- How can it be improved?
- In what other things could this be useful for?

The purpose of these questions was to know what should be kept, what should be removed, what should be improved and what other possible uses could a visualizations of this type have.

The answers to the first question were very positive to the mapping of real movement to virtual movement and of recognizable objects in from real to virtual world, users also appreciated its ease of use, interaction was just a simple matter of moving and looking. Other positive answers included the fact that it allowed them to have a clearer vision of what was around them and being able to see even without any natural light. Finally one of the interviewees noted how the technology could very easily have other applications besides visualizations, like helping visually impaired people navigate the environment.

The answers to the second question focused a lot on the hardware itself, with many not liking the ability to only look around in one axis (SHAKE one has a one dimensional compass), the strange configuration of the whole system seen in Figure 39 (SHAKE’s digital compass is very sensitive to electromagnetic interference so it had to be separated from the other devices by a somewhat long antenna like structure) and the reliability of the GPS. Another complaint was the lack of information, the current version shows only building and not other dynamic elements like people or cars and also there is no information on the buildings themselves. Others complained about how their hands were tired from the position they needed to hold the equipment.

As for improvements the most asked for was multi axis orientation and putting all devices internally in the phone. Also more information like dynamic elements, more accuracy on the GPS, faster start up, more realistic models and the ability to go somewhere without actually moving there.

Other suggestions focused a lot on AR and games, with some suggesting learning, area tracking, diving, exploration and to help lost children find their parents and vice-versa.

Figure 41 – Night vision visualization demo

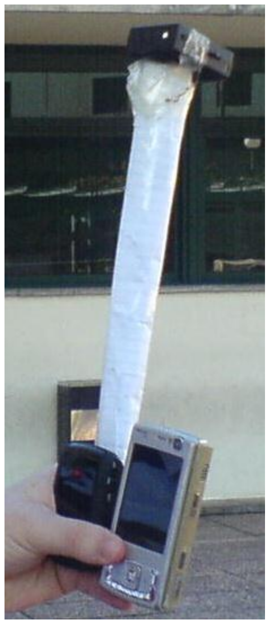


Figure 40 – Hardware setup including the phone, GPS and SHAKE

5.3.8 Conclusions from the test

Observations taken during testing suggest this is a very distracting application (with two users almost getting run over by a car), also some tended to look around to see if what they were seeing in the screen matched the real world and only one noticed the small lag present when moving around.

With this data the areas of improvement are very clear, more information especially in regards of dynamic elements like people and cars, more realistic models with constantly up to date textures and helpful extra information to help in navigation. Finally all the hardware in should be included in one device, like what we see in more recent mobile devices possessing integrated GPS, camera and digital compass.

5.4 Information Overlay

The information overlay visualization is the classic example to realize all the requirements asked for the project. It consists of overlaying on top of the camera feed information about what we are currently seeing, from general knowledge, related events, geographical information to comments from other people that have also seen it. The objective of this visualization is to present relevant or interesting information about what is around the user. There are obvious uses for tourism as a way of exploration and discovery of foreign places.

This visualization uses the same hardware as the night vision allowing for complete component reuse without any change within them. Like the night vision visualization there is a mapping of real world coordinates to virtual coordinates that coupled with information on positions of real world objects allows for the calculation of their relative position to the user in virtual space. This makes it possible to have arbitrary representations of whatever object is around the user with the visualization showing where the target is on the overlaid image. A side effect of the visualization being rendered in virtual 3D space is the presentation of an extra information element of distance between the user and the target.

As with the night vision visualization this system cannot currently be used indoors.

5.4.1 Requirements

This visualization has a much greater focus on interaction than the night-vision that had no interaction elements other than simple natural movements; here there was a need for an interface, since there is so much information that can be presented about an object it cannot simply be shown on the screen without saturating it, especially with small mobile screens. There is also another issue that needed attention, this being a way to explore the environment for targets of interest what is the best way to present the user that there are potential targets around him.

To answer these questions and to try to elicit a set of requirements another batch of interviews were conducted with a small group of 9 young expert phone users. Answers can be seen in the appendix section 3.

The questions were:

- How would you like the application to let you know about possible targets of interest?
- What kind of information would you like to see?
- Would you like to immediately see information related to the object (or objects) as they come into view or would you like them to be hidden for you to activate later?
- What other functions would you like this application to be able to do?
- What other uses could this type of application have?

In the first question almost all interviewees expressed a preference for arrows and icons, other suggestions included text and outlines.

For question two answers diversify according to what type of structure they imagine seeing, for objects like statues of buildings, historical information and events related to them were the preferred choice, for shops they would like to see promotions and available products. Other preferred a set of preset choices for them to configure manually as needed.

For the third question only one of the interviewees wanted to see the information as the object came into view, the rest preferred to activate it as needed.

The fourth question varied a lot from AR, virtual GPS, showing information about related objects, extra information like trivia and information about where the user is, saving all the information with a picture, area tracking, creating a list of objects close to the user, caching what was seen so that later on the user can look at the information again.

Answers in the final question were very similar to the ones in night vision, AR and games were predominant, further suggestion included saying bus stop times, finding lost items and helping post office workers and firemen.

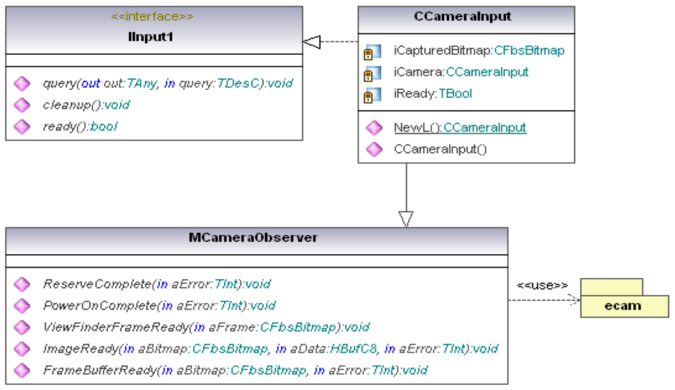
Taking into account the answers provided and the objectives of the visualizations the next set of requirements were created:

1. The system must let the user know of targets of interest in the vicinity.
2. The system must present information related to an object only when asked for it.
3. The system must provide at least historical information, events.
4. The system must overlay all the information on top of the camera's live stream.

5.4.2 Development

Development of this visualization focused almost immediately on the visualization component. All the input components from the Night vision visualization were simply reused leaving only the camera input and visualization components.

5.4.3 Camera input component



<<interface>>
IInput1
query(out out:TAny, in query:TDesC):void
cleanup():void
ready():bool
CCameraInput
iCapturedBitmap:CFbsBitmap
iCamera:CCameraInput
iReady:TBool
NewL():CCameraInput
CCameraInput()
MCameraObserver
ReserveComplete(in aError:TInt):void
PowerOnComplete(in aError:TInt):void
ViewFinderFrameReady(in aFrame:CFbsBitmap):void
ImageReady(in aBitmap:CFbsBitmap, in aData:HBufC8, in aError:TInt):void
FrameBufferReady(in aBitmap:CFbsBitmap, in aError:TInt):void
<<use>>
ecam

Figure 42 – Camera input component class diagram

This input component was extremely important not only for this visualization but also in the wider context of the project's requirements. The SDK in the test phone provides a capable and simple to use camera API allowing for frame-by-frame capture of the camera feed. However the format in the camera's feed frame is not compatible with Open GL ES and so there was a need to convert it to a suitable one. The conversion process is the most performance intensive process in the visualization, but on the whole the performance penalty was acceptable in that it did not make the visualization unusable.

5.4.4 Visualization component

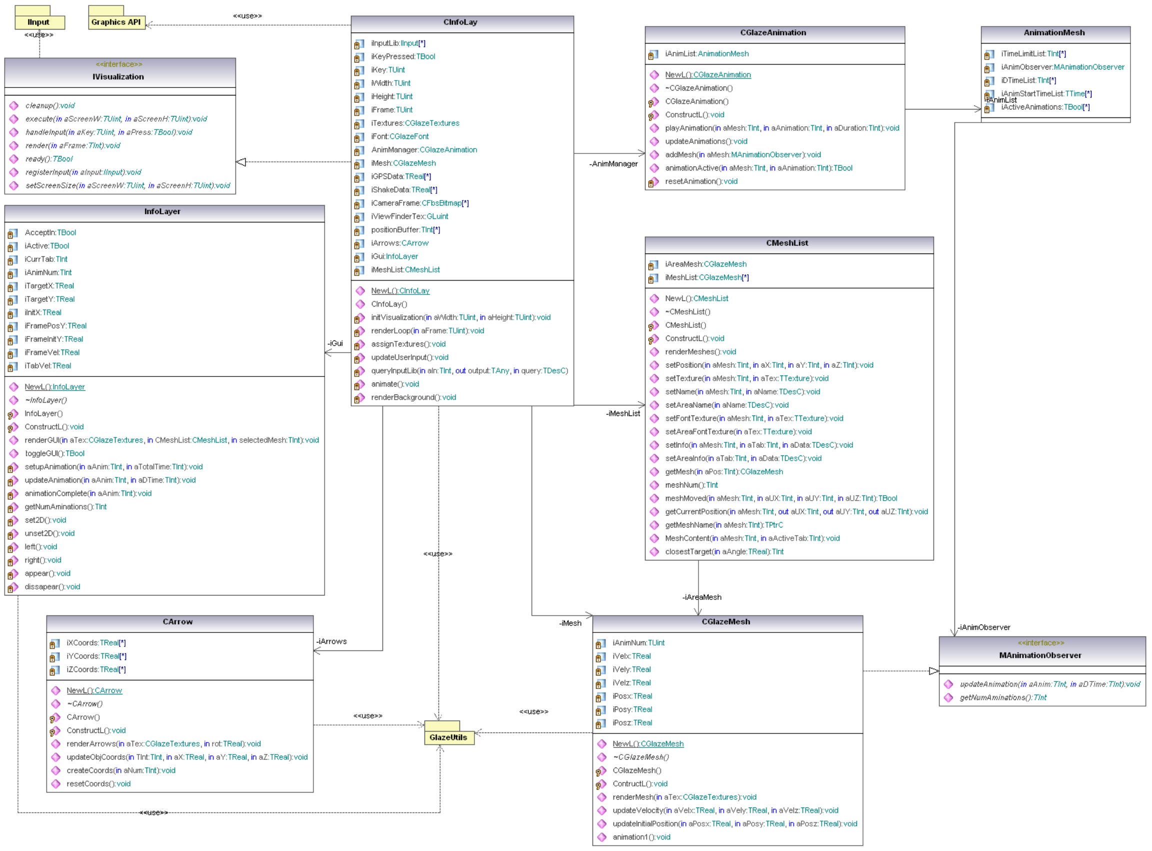


Figure 43 - Information Layer visualization component class diagram

As was the case with the previous visualization the main objective of this one was to map position and orientation in real life to position and rotation to virtual space with the added step of providing a set of information about the object currently in front of the user.

It follows very closely the mapping of sensor data to visual elements, the position in latitude and longitude of all objects of interest are provided to the application (they have been hard coded in the test version) and their relative position is calculated based on the user's current position. The icons representing each object of interest is placed in their relative position around the user, this guaranties they are always well placed no matter where the user is, assuming a good reading from the GPS. When the users change their orientation a simple rotation is applied to the world matching the angle the user turned. Camera feed is also mapped to a texture serving as background of the visualization, the only major problem was the size of each camera frame, Open GL ES only reads power of two textures (or textures with powers of two for height and width size) forcing each frame to be cropped from 256x192 to 256x128 a fairly large drop in height and to avoid image distortion and keep a reasonable quality the texture could not fill the screen as was the initial objective.

As there was a need to show users potential targets of interests around them something of a compass was built using arrows with the same origin but pointing to where the targets were around the user. They also possessed different lengths according to how far the target was from the user, so longer arrows mean the target is farther away.

The final requirement was to present information about the target only when asked; a rudimentary interface built with two main areas of interest the content area and the tab area. When seeing a target of interest the visualization lets the user know the current focus by presenting the name of the target on the top left side of the screen the user can proceed to press a button to call the interface presenting all the information about the current target. That information was organized in themes separated by tabs, the user could navigate those tabs by just pressing left or right on the phone and could similarly dismiss the interface to focus on searching their surroundings again. To maintain the interface as general as possible each target's class can implement their particular way of presenting information in that area on all tabs, giving freedom for each target to differentiate themselves from the others.

5.4.5 User Testing

Considering the similarities in the use of this visualization to the previous one the same approach to user testing was taken.

Testing was made with 8 potential users followed by a simple questionnaire to know what they thought of the visualization and its potential use. Each participant was allowed as much time as they needed to adapt and use the visualization anyway they felt necessary; as they used the visualizations observations on how they used the system were also taken. The questionnaire was kept to a minimum and with direct questions, answers can be seen in the appendix section 5.

The questions were:

- What's the best thing about this visualization?
- What's the worst thing about this visualization?
- How can it be improved?
- In what other things could this be useful for?

The purpose of these questions was to know what should be kept, what should be removed, what should be improved and what other possible uses could a visualizations of this type have.

For the first question interviewees liked the fact that information was easily accessible and concise, its ease of use, the use of live feed to know where the target is, showing the name of the target on the top of the screen to know immediately what target their pointing at, and the ability to see information about targets that are possibly not in view, like behind a building.

The answers to the second question focused again on the equipment, the need to carry all this around was not welcomed, other complaints were about confusion if there are too many targets, not locking on targets (if the user moves while the information interface is on to another target the interface is refreshed automatically to the information of the new target), interface is not semi transparent and covers the picture.

As for improvements many asked for the menu to lock on a target to allow users to move around without it changing the current information, focus on target within a certain distance from the user, do the same but contacts on the phone and save walks, both the video and information.

Other uses were mainly to act as a guide in a new area, education, retail and urban combat.

Figure 44 – Information Overlay demo visualization on overlay mode

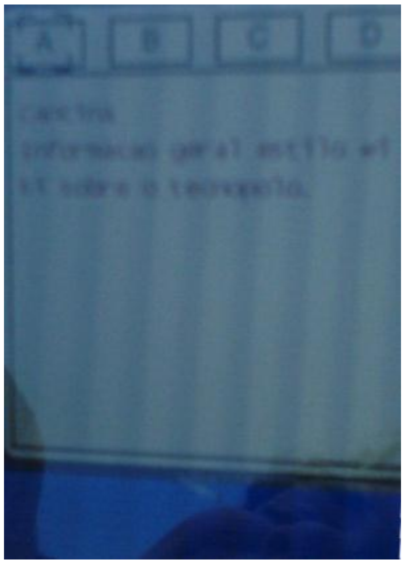


Figure 45 – Information Overlay demo visualization in information mode

5.4.6 Conclusion from the test

Observations taken during testing showed that showing the camera feed significantly increased user awareness of their surroundings. Many users did not understand the arrows and queried about what they were, but when explained of their purpose they immediately used them and agreed with their functionality. Because the visualization automatically updates the information area in the interface when the current target changes users were forced to focus a lot on maintaining the phone as still as possible to access the information. The last thing was they were constantly toggling the interface in an attempt to see if the target overlaid on the camera feed was the one in focus to make sense of the information.

With this data the areas of improvement are very clear, implementing a target system where the interface maintains the info of the target indefinitely is a major area of improvement. A new way of representing the compass pointing to targets is also required to allow users to more quickly identify it, finally making the interface element semitransparent to allow users to look at both the information, the feed and the target icon.

This chapter presented and described test visualizations made with Glaze called Night vision and Information Overlay using observations and semi-structured interviews to elicit requirements for them.

For both visualizations a description of the development process including requirements, development and user testing was provided as well as a detailed look at how those visualizations were made with the Glaze framework.

CHAPTER VI

CONCLUSION

6.1 Comparison

Glaze addresses a number of limitations present in other visualization frameworks. These include their limitation to one specific problem or visualization even when they posses some flexibility by the use of components those were designed to help only with the domain the framework was designed to address. On the other hand Glaze was designed to be highly general and flexible as to be used for development for as many visualizations as possible. Another point of difference to other solutions is that Glaze provides a minimal software structure so that porting is kept simple and fast, this less is more approach brings an added advantage of it being easily extended and used as base for more complex or visualization oriented frameworks, an example of this is something similar to the chapter II section 2.6 case study, where they specialized the information visualization reference model to create a system where both users and the computer help each other create more effective visualizations.

Another important characteristic of Glaze is its modularity. Unlike other frameworks Glaze divided the visualization creation process itself and turned each of those divisions into a concrete component that could be reused and mixed with any other. This helps constrain a visualization developer's task to a systematic set of modular problems and eases visualization development by actively encouraging component reuse.

The inclusion of XML for framework and individual component configuration provides a mechanism for developers to introduce visualization configuration and user customization, which is also extendable at the developers' discretion. Users can further customize the system by combining on-screen visualization components by use of a framework configuration. This provides the ability to "mash-up" different visualizations to render them simultaneously. This separation between framework and component specific configuration allows for the introduction of new internal features to the framework without it affecting visualization development and vice-versa.

However, Glaze could also be improved. One important missing element is scripting language support, this is very widely supported by many visualization frameworks with the main objective of facilitating development. The main reason it is not featured in Glaze is mainly because the use of scripting often comes with the need to sacrifice generality. The nature of these languages is to serve as support for the main application by either have the application offloading some work to them or by having scripts call functions defined within the main application. The problem of having a general system is the inexistence of such functions so the goal of scripts would be merely as a support tool for trivial operations or for use in a project by project basis. However Glaze could introduce scripting languages with a different form, instead of having them provide support to the main application, a Glaze library could be developed for scripting languages. This would allow visualizations developed with those languages to be standalone and more importantly they could possibly achieve platform independence.

The use of remote technologies, like offloading processing to a remote server is also of interest, a lot of frameworks especially those aiming at cross platform and device compatibility offer this feature, currently Glaze does not offer such capabilities. But with the growing connectivity of mobile devices to the internet this is becoming an important aspect, not only to render more computationally intensive applications but also in creating new and innovative visualizations. Glaze's component architecture allows the framework to be easily augmented for remote communications without breaking inter component support, not only that but a communication standard using XML can also be built without breaking compatibility with any visualizations. Glaze's natural separation of visualization and raw data could be improved and extended to enable visualizations to use remote data, for example one user could be seeing a visualization mapping accelerometer data to a bar graph and that data could be coming from another phone anywhere in the world. The ability to "mash-up" could also be improved by joining visualizations rendered in other phones with others in the native device, for example two users could be seeing a visualization from the same remote source while one applies a fish-eye to it and the other a zoom.

6.2 Future

Glaze has a lot of room to grow; the ease of extending it can be improved by further abstracting key classes. More framework specific configurations like the ability to control the execution of visualizations, by creating visualizations in stages users could potentially control how those stages are executed and how they relate to each other enabling them to create even more varied visualizations by simply using XML or scripting.

Another area of future work is that of distributed applications, this holds the most promise, Glaze provides a natural separation of the visualization to the raw data, this could be improved and extended to enable visualizations to use remote data. Another potential capability provided by the use of visualization side XML configuration and remote visualization is the ability to create visualization services much like what we see in Many Eyes. In order to have a visualization users need to have all the components needed by that visualization on their device, a visualization service would eliminate that, if another user has the visualization I want to use and provides it as a service, it would simply be a need of sending the configurations needed, the remote device would render it and send the result back.

Future work will also include further user studies to capture both additional requirements for concept generation and more scientific studies to measure empirical performance.

6.3 Conclusion

Visualizations are essential to present the maximum amount of information in the smallest space possible, given the increase in computational power and the limited screen size present in these devices information visualization was identified as an important area of investigation. A lack of general frameworks designed to help developers and designers in the development of visualizations in general was identified. Given the need for special care, like user attention and usage environment when developing visualizations aimed at mobile devices a framework that removed most of the low level complexities and allowed developers and designers to focus on their end product was an obvious need. Opening such functionalities to developers with a general, flexible and extendable framework directed at mobile devices is a first stage to enabling a new breed of rich, expressive mobile applications capable of saliently presenting the complexities of real world information.

REFERENCES

Aaltonen, A., & Lehikoinen, J. (2006). Exploring augmented reality visualizations. *Advanced visual interfaces* (pp. 453 - 456). Venezia: ACM.

Aaltonen, A., & Lehikoinen, J. (2005). Refining visualization reference model for context information. *Personal and Ubiquitous Computing , 9* (6), 381-394.

Baudisch, P., & Rosenholtz, R. (2003). Halo: a technique for visualizing off-screen objects. *Conference on Human Factors in Computing Systems* (pp. 481 - 488). Ft. Lauderdale: ACM.

Behzadan, A., Timm, B., & Kamat, V. (2008). General-purpose modular hardware and software framework for mobile outdoor augmented reality applications in engineering. *Advanced Engineering Informatics , 22* (1), 90-105.

Bier, E., Stone, M., Pier, K., Buxton, W., & DeRose, T. (1993). Toolglass and Magic Lenses: The See-Through Interface. *International Conference on Computer Graphics and Interactive Techniques* (pp. 73 - 80). Anaheim: ACM.

Björk, S., Falk, J., Hansson, R., & Ljungstrand, P. (2001). Pirates! Using the physical world as a game board. *Proceedings of Interact 2001 Conference on Human-Computer Interaction*, (pp. 9-13).

Burigat, S., & Chittaro, L. (2005). Location-aware visualization of VRML models in GPS-based mobile guides. *3D technologies for the World Wide Web* (pp. 57 - 64). Bangor: ACM.

Capin, T., Pulli, K., & Akenine-Möller, T. (2008). The State of the Art in Mobile Graphics Research. *IEEE Computer Graphics and Applications , 29* (4), 74-84.

Card, S., Mackinlay, J., & Shneiderman, B. (1999). *Readings in Information Visualization: Using Vision to Think.* Morgan Kaufmann Publishers.

Chi, E. H.-h. (1999). *A Framework for Information Visualization Spreadsheets.* Phd Thesis, University of Minnesota.

Chitarro, L. (2006). Visualizing Information on Mobile Devices. *39* (3), 40 - 45.

Cho, S.-J., Choi, C., Sung, Y., Lee, K., Kim, Y.-B., & Murray-Smith, R. (2007). Dynamics of tilt-based browsing on mobile devices. *Conference on Human Factors in Computing Systems* (pp. 1947-1952). ACM.

Dennis, B., Kocherlakota, S., Sawant, A., Tateosian, L., & Healey, C. (2005). Designing a Visualization Framework for Multidimensional Data. *IEEE Computer Graphics and Applications , 25* (6), 10-15.

Dourish, P. (2001). *Where the Action Is The Foundations of Embodied Interaction.* Cambridge: The MIT Press.

Ehret, J., Ebert, A., Schuchardt, L., Steinmetz, H., & Hagen, H. (2004). Context-Adaptive Mobile Visualization and Information Management. *Visualization '04.* IEEE Computer Society.

Erp, J., Veen, H., Jansen, C., & Dobbins, T. (2005). Waypoint navigation with a vibrotactile waist belt. *ACM Transactions on Applied Perception , 2* (2), 106-117.

Ghinea, G., Heigum, J., & Fongen, A. (2008). Information visualization for mobile devices: A novel approach based on the MagicEyeView. *International Symposium on Wireless Pervasive Computing*, (pp. 566 - 570). Santorini.

Gröhn, M. (2008). Visualization.

Hao, J., & Zhang, K. (2007). A Mobile Interface for Hierarchical Information Visualization and Navigation. *IEEE symposium on consumer electronics.*

Herman, I., Melançon, G., & Marshall, M. S. (2000). Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics , 6* (1), pp. 24 - 43.

Hinckley, K., Pierce, J., Horvitz, E., & Sinclair, M. (2005). Foreground and background interaction with sensor-enhanced mobile devices. *ACM Transactions on Computer-Human Interaction , 12* (1), 31-52.

Itoh, M., & Tanaka, Y. (2006). 3D Component-Based Visualization Framework for Generating Simple 3D Applications Using Web Services. *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence* (pp. 823-830). IEEE Computer Society.

Jankun-Kelly, T., Gertz, M., & Ma, K.-L. (2007). A Model and Framework for Visualization Exploration. *13* (2), 357-369.

Kamba, T., Elson, S., Harpold, T., Stamper, T., & Sukaviriya, P. (1996). Using small screen space more efficiently. *Conference on Human Factors in Computing Systems* (pp. 383 - 390). Vancouver: ACM.

Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., & Giannopoulou, E. (2007). Ontology visualization methods—a survey. *39* (4).

Keim, D. (2000). Designing Pixel-Oriented Visualization Techniques: Theory and Applications. *IEEE Transactions on Visualization and Computer Graphics , 6* (1), pp. 59 - 78.

Keim, D. (2002). Information Visualization and Visual Data Mining. *IEEE Transactions on Visualization and Computer Graphics , 8* (1), 1 - 8.

Keim, D., Schneidewind, J., & Sips, M. (2006). Scalable Pixel based Visual Data Exploration. *Visual Information Expert Workshop* (pp. 12-24). Paris: Springer Berlin / Heidelberg.

Kim, J., Hsu, W.-J., Moon, S., Kumar, U., & Helmy, A. (2008). Visualization and Representation of Mobile Network Users. *Sensor, Mesh and Ad Hoc Communications and Networks* (pp. 588-590). IEEE Computer Society.

Knödel, S. (2008). Visualization and interaction with mobile technology. *Human computer interaction with mobile devices and services* (pp. 557-558). Amsterdam: ACM.

Kreuseler, M., Lopez, N., & Schumann, H. (2000). A Scalable Framework for Information Visualization. *INFOVIS.* IEEE Computer Society.

Lee, J., Forlizzi, J., & Hudson, S. (2005). Studying the effectiveness of MOVE: a contextually optimized in-vehicle navigation system. *Conference on Human Factors in Computing Systems* (pp. 571-580). Portland: ACM.

Liu, H., Xie, X., Ma, W.-Y., & Zhang, H.-J. (2003). Automatic browsing of large pictures on mobile devices. *International Multimedia Conference* (pp. 148 - 155). Berkeley: ACM.

Mackinlay, J., Robertson, G., & Card, S. (1991). The perspective wall: detail and context smoothly integrated. *Conference on Human Factors in Computing Systems* (pp. 173 - 176). New Orleans: ACM.

Mazza, R. (2004). *Introduction to Information Visualization.* University of Lugano, Faculty of Communication Sciences.

Meiguins, B. S., Carmo, R., Almeida, L., Gonçalves, A., Pinheiro, S., Garcia, M., et al. (2006). Multidimensional information visualization using augmented reality. (pp. 391 - 394). New York: ACM.

Meyer, M., Gîrba, T., & Lungu, M. (2006). Mondrian: an agile information visualization framework. In M. Meyer, T. Gîrba, & M. Lungu (Ed.), *Proceedings of the 2006 ACM symposium on Software visualization* (pp. 135-144). Brighton: ACM.

Microsoft Corporation. (2006, July 26). *Photosynth*. (Microsoft Corporation) Retrieved June 29, 2009, from Photosynth: Your photos, automatically in 3D.: http://photosynth.net/

Narzt, W., Pomberger, G., Ferscha, A., Kolb, D., Müller, R., Wieghardt, J., et al. (2003). Pervasive Information Acquisition for Mobile AR-Navigation Systems. *Mobile Computing Systems and Applications* (pp. 13-20). IEEE Publishing.

Piskuliyski, T., & Kumar, A. (2007). A General Framework for Overlay Visualization. *Electronic Notes in Theoretical Computer Science. 178*, pp. 161-169. Elsevier Science Publishers.

Rao, R., & Card, S. (1994). The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. *Conference on Human Factors in Computing Systems* (pp. 318 - 322). Boston: ACM.

Robbins, D., Cutrell, E., Sarin, R., & Horvitz, E. (2004). ZoneZoom: map navigation for smartphones with recursive view segmentation. *Advanced visual interfaces* (pp. 231 - 234). Gallipoli: ACM.

Robertson, G., & Mackinlay, J. (1993). The document lens. *Symposium on User Interface Software and Technology* (pp. 101 - 108). Atlanta: ACM.

Rohs, M. (2004). Real-World Interaction with Camera-Phones. *International Symposium on Ubiquitous Computing Systems* (pp. 74-89). Springer.

Sachinopoulou, A. (2001). *Multidimensional Visualization.* Technical Research Centre of Finland. Espoo.

Sarkar, M., Snibbe, S., Tversky, O., & Reiss, S. (1993). *Stretching the Rubber Sheet: A Metaphor for Viewing Large Layouts on Small Screens.* Providence: Brown University.

Son, L., & Calitz, A. (2004). *The Visualisation of Internet Usage.* Masters Dissertation, University of Port Elizabeth, Department of Computer Science and Information Systems, Port Elizabeth.

Stone, M., Cowan, W., & Beatty, J. (1988). Color gamut mapping and the printing of digital color images. *ACM Transactions on Graphics , 7* (4), 249 - 292.

Ware, C. (2004). *Information Visualization Perception for Design.* San Francisco: Morgan Kaufmann Publishers.

Wijk, J., & Liere, R. (1993). HyperSlice: visualization of scalar functions of many variables. *Conference on Visualization '93*, (pp. 119 - 125). San Jose.

Williamson, J., Murray-Smith, R., & Hughes, S. (2007). Shoogle: excitatory multimodal interaction on mobile devices. *Conference on Human Factors in Computing Systems* (pp. 121-124). San Jose: ACM.

Yoo, H. Y., & Cheon, S. H. (2006). Visualization by information type on mobile device. *2006 Asia-Pacific Symposium on Information Visualisation. 60*, pp. 143 - 146 . Darlinghurst: Australian Computer Society, Inc.

Zhou, H., Qu, H., Wu, Y., & Chan, M.-Y. (2006). Volume visualization on mobile devices. *14th Pacific Conference on Computer Graphics and Applications* (pp. 76-84). Taipe: National Taiwan University Press

APPENDICES

1 Preliminary interview

What do you love about cell phone?

- Physical qwerty keyboard
- Light, good battery, easy to make calls or text messages and a camera for those unexpected moments, rotating cameras or one in front and another in the back, quality isn't that important as long as it's the bare minimum
- Camera besides calling, I want to record interesting stuff and it's easier with a phone
- Good reception, battery longevity, good camera and memory, picture quality
- Simple and robust
- the ability to remain connected to the world, no matter where I go, I really like taking pictures, but it's not really enough to justify having a camera with me at all times, phones today have a very reasonable picture quality, they are more personal than a pc, I like compatibility we should always be able to choose whatever service we want and not with platform offers a service.
- Not having to install more than one program to synchronize everything with my PCs, I like being able to install applications without having to carry around a cable.
- Fast and efficient
- Small, not having to recharge the battery every day, having whatever song I want as a ring, using it as an alarm, I can turn it off whenever I want
- Style and easy to navigate
- I like what it's for, making calls
- Friendly software, good interface, memory, quick to learn
- Speed in which I search and find contacts
- Java, wireless, GPS, qwerty keyboard, writing recognition, agenda, sound quality and 3MP camera
- Good sound quality, small, easy to do what I want to do, easy to access specific functionalities of the phone
- Storage, camera, good keyboard
- Portable, always reachable, cute
- Talking with distance people, sending text messages at any time

What do you hate about cell phone?

- Slow boot time
- The fact that you're always reachable at any time, we lose our privacy, there is also health issues, they say it's bad to spend too much time talking on the phone, some keyboards are too soft, it loses sensibility, it would be better to have a physical and virtual keyboard, automatic writing is annoying
- Too big
- Calls disconnecting every ten seconds, bad reception, writing SMS without a qwerty keyboard, I don0t like having to press the same key to reach the letter I want
- Errors, crashes and incompatibilities
- No reception in tunnels, bad reception just with my operator, bugs, locking, rebooting because of an error
- Accessory and service standardization, with this we wouldn't have compatibility problems and that is my biggest problem, each company tries to implement their own standard, when we buy a phone we are not just buying a device we buy all the accessories needed to make the phone work like we want it to.
- Can't change battery, can't have WIFI always on, low battery life
- Ruined batteries, slow OS, symbian, Motorola
- Paid calls, having to be always on the lookout for the battery and reception.
- Can't block calls and SMS we don't want unless we ask the operator, can't reroute SMS's to another number.
- Don't like bloat (features and functionality), low lifetime battery, different operators, never enough battery, sometimes they just stop working, constant design changes
- Annoying software
- Can't reroute SMS, doesn't register my last calls
- Bad sound quality, slow OS, small battery life
- Battery life
- Crappy menus, battery duration, lack of network coverage
- Free internet everywhere, easier connection to the PC, get information of a place by a photo, auto-process the photo
- When we lose reception, when battery runs out
- Price

What is your dream feature for a cell phone?

- Small as an ipod nano and a more user friendly UI.
- Voice recognition to make calls and to write messages, fast browser, more memory, pointing the camera somewhere and get info about that place like restaurants close by, shops that sell stuff I like, not everything at once I want to choose what information I want to see
- Get information about a place you're taking photos, like history and events
- Nightvision

- Projector for a keyboard and screen, professional level camera and an OS that allowed me to run office and matlab, panoramic HD pictures an all in one device(no cards, no wallets)
- Cloud computing, I want phones to be just a modem with a screen and barely needs a processor, that processing should be made in the PC, that way we choose an OS the software and everything they want and don't get restricted to one device, of course battery life should not suffer because of that
- A virtual card so I could have n operators at the same time, some AI too if I'm sleeping and someone calls I don't want it to wake me but I want to know latter who called, ability to read codes, automatic translations, shopping info.
- Get my coffee
- Allow me to control my pc from anywhere, video quality like a normal camera, more functional and cheaper.
- Charge the battery with solar power, transmit smells or emotions through the phone
- Import all my contacts from all the communities I belong to
- Show how to get somewhere and what is around me.
- Know the geographical location of my contacts, talking to more than one person at the same time
- Act as a universal controller
- Access and execute programs or files in a pc anywhere in the world
- Teletransport, find services around me
- A phone that doesn't look like a machine, something more appealing and interactive

2 Night vision interview

Imagine an application that lets you see your surroundings at night when you have trouble identifying what is around you. What would be an interesting feature for this application?

- A flashlight
- Infra red camera
- A map
- Something to take the clothes off girls around you.
- A sonar
- Look at the environment like it was day time.
- A wireframe of the whole environment.
- Make the screen completely white to serve as a flash light.

What would you like to do with an application like this?

- A map with your position to help me know where to go
- Information identifying stuff in the environment, like dangerous objects and obstacles.

- Something to look for dangerous stuff in my pictures and videos and also let me know about exits.
- Let me know exactly where I am and how to go to a familiar place.
- Tell me which way was home or a place to stay, and tell me where I am.
- Tell me how to go to a place with light.

3 Information Overlay interview

How would you like the application to let you know about possible targets of interest?

- An icon informing me something is there, as many icons as there are targets.
- A blinking button to call out my attention with a list of all points of interest, and it points me to the one I have chosen.
- An arrow or a line.
- Sound signal, vibration, arrows and text.
- An icon pointing to it.
- Highlight buildings I might find useful with an arrow
- Make a colored outline appear around the targets, and make use of different colors to indicate different levels of interest. Make them become semi-transparent after a few seconds. Put an arrow on the edge of the screen with a small label saying which direction to turn if I'm interested.
- An icon and arrow with some variation according to what I like.

What kind of information would you like to see?

- Historical info, tourist info, directions.
- Historical info, schedules, services.
- Depends on what I am seeing, events, history, location, products, and promotions.
- Information selected from the start, I want the ability to filter (select) from everything.
- I want to choose the targets of interest.
- What services I can find in the buildings
- Two different levels of information, one very simple and on by default, the other with more detail, with wiki like levels of info.
- Zones of interest.

Would you like to immediately see information related to the object (or objects) as they come into view or would you like them to be hidden for you to activate later?

- When I point at something an icon appears, when I press a button the camera frame disappears and a page appears, I then read the information like it was a phantom page (magnifying glass) by moving the phone and zooming in or out.
- Show immediately to call my attention.
- Hidden but with an option to automatically show the ones I have previously chosen.
- I want to select it as I want it.
- I want to choose the info related to that target.

- As they come into view
- Like some sort of clickable non-obstructing label
- Just an indication that info exists when the object appears on screen.

What other functions would you like this application to be able to do?

- AR, going back in time by moving the phone, virtual 3D GPS.
- Inform about things or places related to what I'm looking at.
- More information, trivia, things related to what I am looking at, information about where I am.
- Suggestions about similar things to what I am seeing, how far I am from somewhere, virtual 3D GPS.
- Save the information with a photo, historic information.
- Check playbacks of my "walks"
- Picture snapshot, video recording, object bookmarking, custom label creation and position. List of objects in # meters. Show related objects to object X. Show map. Cache seen objects so that later on you can "Search for object" and know where it was last seen
- See detailed services provided by a store, previews of museums and such.

What other uses could this type of application have?

- Games.
- Learning.
- Identify people.
- Tell me where the cops are.
- Check what busses pass through the buildings
- Finding lost items, games
- Helping post office workers finding the right house to put mail. Helping the fire department have a better idea of where the emergency is by simply scanning around them.

4 Night vision user text interview and observations

What's the best thing about this?

- Intuitive, simple, being in a small device is good.
- The model was a fair representation of the real world; movement was pretty much real time with nice precision.
- It's cool to be able to see we can't see without natural light
- Moving and seeing like we do in real life.
- The sync between the virtual and real world, mobility and technological novelty
- Going towards and away from thing and see where I am going.
- That it can be scalable to have other functionalities other than just visualizations like vibrating when we are close to wall could help blind people.
- it allows me to have a clearer vision of where I am and what is around me.

What's the worst thing about this?

- Distracting, not that much information
- Can't look up or down, only shows buildings, too shaky, antenna is too weird to walk around in.
- My hand got tired from holding it up, not much information
- Textures are not up to date, that can lead to errors.
- Orientation only in one axis, we can go through walls.
- Low resolution, poor alignment, confusion and latency.
- For the purpose of being a guide at night it needs to be a little more reliable.
- Models and textures are not up to date (there could be some sort of construction), one direction orientation.

How could it be improved?

- Nothing much besides the devices
- Get rid of the shaking; make it detect variable elements instead of only static ones
- Better modeling and more information.
- Zoom, choose a path.
- Multi axis orientation, AR during the day, virtual world during the night
- Reduce latency, better models and a bigger area.
- Faster start up, more accuracy when calculating a person's position.
- Multi axis orientation, see what is around use without having to necessarily go there, know where streets end up going to.

What other things could this be useful for?

- Area exploration, terrain simulation, diving.
- Helping blind people, games (AR).
- AR, learning, virtual guides
- Games.
- Guides, learning and games
- Movement/area tracking.
- Get information from buildings around us.
- To help lost children know where they are and where to find their parents and vice versa.

Observations:

- Distracting (two were almost ran over by a car)
- They looked around a lot
- Noticed lag
- Needed a hand understanding he could turn around immediately.
- Initial confusion

5 Information overlay user test interview and observations

What's the best thing about this visualization?

- It showing the direction of local places of interest with the arrows. We can get relevant information about what we are seeing.

- Ease of identifying places of interest, concise information.
- Immediate information about what I am seeing, just look at it and get the information, immediate information change.
- Get the information I want about something and also information about the place, not just what I am pointing at.
- Information appears straight away over what I am seeing, the menu (pressing a button and it appears), integration of the information about what I am seeing, we quickly get the information when looking at something.
- Visualizing information about areas and monuments that are not visible.
- Having the live feed and recognizing the buildings, it's scalable and we can put a lot of information, works in a phone that is not iPhone, very few interaction needed to get the information.
- Easy to use

What's the worst thing about this visualization?

- Should stand out more that there is a place of interest around.
- It becomes confusing with many places
- Too much equipment, menu is not transparent
- Doesn't lock on the target, it can have too many arrows
- Too much equipment, eats too much battery
- Menu covers the image
- Should not change the information on the menu if the target changes in the background, bigger and more appealing arrows with names.
- Information changes even with the menu on, it gets confusing if there are many places around.

How can it be improved?

- Only show targets within a certain area.
- Keep the information on the menu while it is on even when pointing at someplace else.
- More visible arrows, keep the menu from changing while it is on, transparent menu, freeze the camera feed when I turn the menu on.
- Freeze the information on a place, save information about what we point for later (history)
- Represent how long we took to get some place, do the same to the contacts on your phone, filter information, be a guide, save walks and keep info of what we looked at, define a distance e show only information within that radius.
- Improve the UI and the data base.

In what other things could this be useful for?

- Orientation games
- As a guide, show information about people
- Education and training.
- Education, digital guide, retail.
- Bus schedules
- Helping blind people, daily route planer, social networks, parenting
- Urban warfare logistics.

Observations:

- Many users ignored the arrows in the beginning and asked their purpose
- Were very careful to keep the phone as still as possible when reading information to keep it from changing
- Toggled the interface to make sure the target matched what they were reading.

6 XML DTD

Core DTD

```
<!ELEMENT core (FE, V)>
<!ELEMENT FE (#PCDATA)>
<!ELEMENT V (#PCDATA)>
```

Front-end DTD

```
<!ELEMENT frontend (fname)>
<!ELEMENT fname (#PCDATA)>
```

Visualization DTD

```
<!ELEMENT visualization (vname)>
<!ELEMENT vname (inputs)>
<!ELEMENT inputs (inputlib)>
<!ELEMENT inputlib (#PCDATA)>
```

7 XML Schema

Core system XML schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="core">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="FE" type="xs:string"/>
      <xs:element name="V" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
</xs:schema>
```

Front-end component XML schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="frontend">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="fname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Visualization component XML schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="visualization">
            <xs:complexType>
                        <xs:sequence>
                                    <xs:element name="vname">
                                    <xs:complexType>
                                    <xs:sequence>
                                                <xs:element name="inputs">
                                                <xs:complexType>
                                                <xs:sequence>
                                                            <xs:element name="inputlib" type="xs:string"/>
                                                </xs:sequence>
                                                <xs:complexType>
                                    </xs:sequence>
                                    <xs:complexType>
                        </xs:sequence>
            </xs:complexType>
</xs:element>

</xs:schema>
```