uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Sadek Hamdan
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical and Computer Engineering)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Context Aware Layout Manager for Video Conferencing Systems
TITRE DE LA THÈSE / TITLE OF THESIS

Dr. A.El Saddik
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Dorina Petruil

S.Shirmohammadil

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

CONTEXT AWARE LAYOUT MANAGER FOR VIDEO CONFERENCING SYSTEMS

BY

Sadek Hamdan

A THESIS SUBMITTED TO THE
FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ELECTRICAL ENGINEERING

OTTAWA-CARLETON INSTITUTE FOR ELECTRICAL ENGINEERING
SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING
UNIVERSITY OF OTTAWA

Your file Votre référence
ISBN: 978-0-494-79679-5
Our file Notre référence
ISBN: 978-0-494-79679-5

Abstract

In videoconferencing systems, the views of remote participants need to be laid out on the display according to context/importance of each view, and to the existing display setup. Current approaches require the user to be involved in the control process of videoconferencing systems, and consequently cause an increase of time and cost when using this technology. We propose a novel approach based on the real estate market theory to automatically manage the layout of any type of display setup in videoconferencing systems while being aware of the context of the videoconference. We analyze the suitability of the proposed approach by comparing it to another layout manager approach, namely rule-based approach. The major advancement of this approach is that it gets the videoconferencing technology closer to its goal: to make participants able to communicate as if they were located in the same room.

Acknowledgements

This thesis is based on research conducted during my years at the Multimedia Communications Research Laboratory (MCRLab), University of Ottawa, and at Magor Communications Corporation. The work presented here is the result of highly collaborative efforts and I owe thanks to the people I have worked with.

First and foremost, I am thankful to my supervisor, Dr. Abdulmotaleb El Saddik, for the support he provided during this work. His enthusiasm and eagerness have always been a driving force for me.

I would like to convey my sincere gratitude and indebtedness to Dr. Mojtaba Hosseini from Magor Corp. for helping me organizing, planning and pursuing the research in this field and move forward along the path of innovation and novelty. I thank him for the many discussions, sometimes demanding and frustrating, but eventually problem-solving. His invaluable research ideas and experience kept me motivated all the way towards the progress and completion of this research work.

I would like to give my sincere thanks to my parents and my fiancée for their support throughout my graduate studies. I am also grateful to all my friends and fellow MCRLab students for their support, suggestions and constructive discussions. My special thanks go to Mohamad Eid for helping in the thesis writing, and to Anwar, Fawaz, Faki, and Jongeun.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

Videoconferencing is the use of telecommunication and multimedia technologies to bring multiple parties together from remote locations and to "allow them to communicate with each other as if they were seated in the same room." [25]. The main advantages of using this technology are to decrease traveling costs and time, and to increase flexibility by bringing team members spread across multiple sites together [2]. The advancements of the solutions supporting videoconferencing and the need of this technology have created a massive demand on this technology both in the industry and academia.

Video streams received from multiple sites during a videoconference are displayed in desktop windows. Many techniques are used to layout the windows (or views) of the remote sites on the local display. The simplest of these techniques is the "continuous presence" where only the presenting site is displayed; the disadvantage is that the interaction is restricted between the presenter and the site watching him/her, whereas in videoconferences all conferees can interact with each other at any point of time.

Another simple technique is the "Hollywood squares" where each site is displayed on a separate screen or window as seen in figure 1.1.

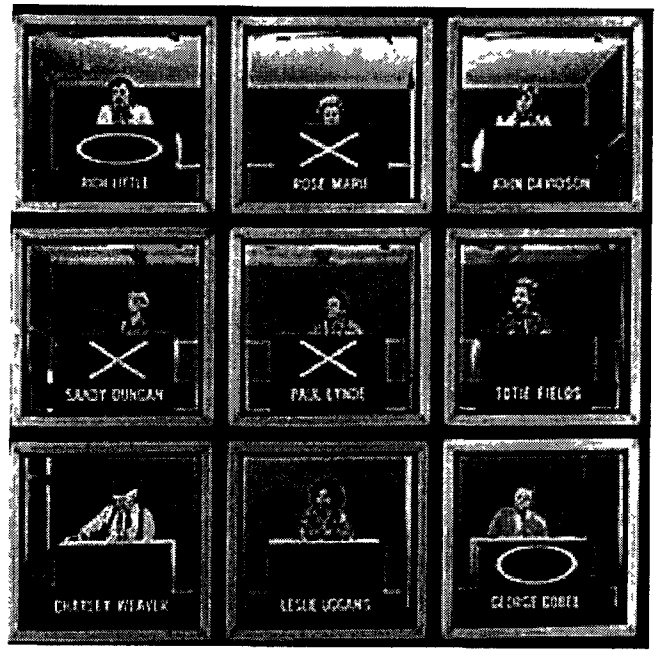

Figure 1.1 Hollywood squares [41]

Tele-cubicles system [9, 10, 11] is another form of Hollywood squares, however the way the remote sites are is different, each site is displayed in one side of the user as seen in figure 1.2.

Figure 1.2 tele-cubicles system [11]

The main disadvantage here is scalability; that is the number of conferees cannot scale. Moreover this approach does not reflect the context of the conference because all views are treated equally and they are displayed statically with the same size.

Another more advanced technique is the rule-based approach where the user has to specify the rules or constraints of how to layout the views on the display. Such technique uses finite state machine (FSM) triggered by timers or events, similar to systems in [22] and [23]. Figure 1.3 shows three state finite state machine to determine at any given moment which camera is selected as the output camera [42].

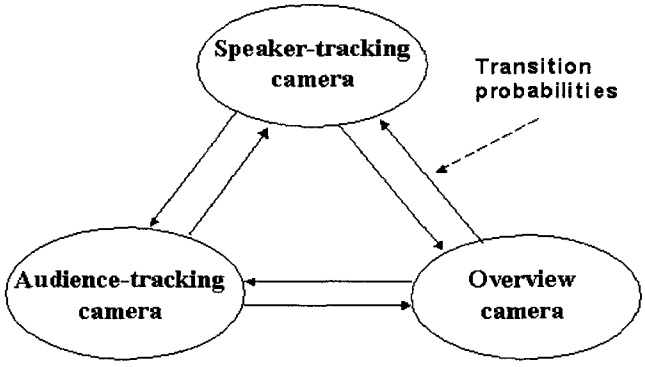

Figure 1.3 Finite state machine for view selection [42]

The disadvantage of this technique is that, in each state of the FSM, the user has code the layout rules for all possible scenarios that might happen during the videoconference, and the result will be a long list of if/else statements, therefore this technique is not scalable. Moreover if the display setup changes, the user has to code new layout rules, so this technique is not adaptable to new display setups.

During a videoconference, the context parameters including the number of participants, their status, the numbers of views, and the video quality of the views, are all subject to change at any time. Moreover, the hardware display setup may differ among multiple sites participating in the videoconference. Therefore the need of a generic layout management system that adapts to both the videoconferencing context and the different hardware setups, without involving the human user, is the driving force for us to conduct research in the area of automated layout management.

1.2 The Existing Problem

The research in the field of Human-Computer Interaction (HCI) aims to study and facilitate interaction between humans and technology [12]. Thus the optimal goal is to make the interface between them fully transparent and to free the user from the burden of controlling the technology rather than using it. However in videoconferencing systems this goal is still far beyond reach, because the human is still involved in the control process when using this technology. The aspects of this involvement are described below.

First, with the advancements in the production of higher quality displays (high definition) and lower costs, the use of large screens is becoming more common in videoconferencing systems; an example is the system developed at Magor Communication Inc [2]. This opens up the possibility to have different hardware setup on every site and thus creates the need to configure videoconferencing systems by specifying new layout rules for every new setup. For example the setup made of two high definition (HD) screens has different layout rules than a video wall made of 3*3 HD screens. This task needs to be done by expert users who specify the rules to layout the views on this new setup.

Second, there are many factors that contribute to providing the context of a videoconference and thus affect the layout of views, such as the number of participants, the role of each of them, his/her status, and the quality of the video stream. Some of

these factors, such as the video quality, can be automatically determined by the system, however for the other factors the user intervention is needed to provide the necessary information, similarly to system [24] where the user is needed to describe the features of the content of the captured video.

Moreover, if the user wants to add more parameters that the system or the user can detect, he/she needs to change the implementation of the system according to his/her needs. As an example if a user wants to evaluate the views and to layout them according to the volume level of a talking participant, he/she needs to manipulate the system to take into consideration this parameter. In this case the task of the user is even harder, that is manipulating the implementation of the system.

1.3 Objectives and contributions

1.3.1 Objectives

The engagement of the human in videoconferencing technology, and the resulting time and cost of his/her engagement are the main motivations to develop a scalable and automated layout management system that manages the layout in videoconferencing systems.

This system should work on different types of display setups, without any intervention from the user in this process. In other words this system should work for different

display setups of different number and size of screens and/or cameras, contrary to the rule-based approach where the user has to code new layout rules for new display setups. Also this system should be aware of the context during the videoconference by detecting the changes in the context, and adapting to them by changing the layout accordingly.

Therefore the name of our approach is *"context aware layout manager for videoconferencing systems"*.

1.3.2 Contributions

The contributions of this thesis can be summarized in the following:

- Design and development of an automated layout management algorithm based on the real estate metaphor in order to manage the display of diverse hardware setups: PDA, laptop, wall size displays, etc, by using a real estate market metaphor.
- Design of context aware component capable of evaluating views of multiple sites according to their context.
- Using the results of the evaluation in the automated layout management algorithm to layout the views according to their context.
- Design of functions to enable a user to add/remove the features that he/she thinks are relevant/irrelevant to the context of a videoconference.

The above developed algorithm was also integrated into an application called LayoutViewer developed at Magor Communications Corporation [2].

1.4 Thesis organization

The remainder of the thesis is organized as follows:

Chapter 2 presents an overview of background literature and related studies. Background concepts on human computer interaction, context awareness and automated layout are provided first. Next, the main techniques for automated layout are described in general, and then the following systems are described with details: DesignComposer [21] a context-based automated layout generator for web pages, and then the iCam [22] system used in automated videography, and finally two applications where the user can customize the display according to his/her preference.

Chapter 3 presents the proposed idea and elaborates on the system design phase. The design phase includes the system overall architecture, the design of context awareness component, and the design of layout management component.

Chapter 4 focuses on the implementation of the system. This includes the technology and deployment, the tool used to simulate the layout manager component, and the implementation software architecture of the system.

Chapter 5 presents the layout and performance results for rule-based and real estate layout managers using the viewer tool described in chapter 4. Then the results of a usability test are presented and described, followed by a discussion of all the previous results.

Finally the work is summarized in **Chapter 6**, which also includes a conclusion and a discussion of potential future work.

Chapter 2

Literature Background and Related Work

2.1 Literature Background

2.1.1 Human-Computer Interaction

"Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them." [12]. Here what is meant by computer systems is not only the computer hardware, but also any user interface application used to facilitate the interaction between human and computers. From this definition it is understood that the human, specifically the computer system user, is the major concern for computer systems developers. This is referred to as "User-centered design" which is rooted in the idea that users must take center-stage in the design of any computer system [13].

The reason that makes Human-Computer interaction an important field of study is the fact that people should not have to change the way they use a system in order to adapt with it, instead, the system should be designed to match and adapt their to requirements[14].

Thus the goals of Human-Computer interaction are to:

- Understand the factors that determine how people use technology i.e. their needs and vulnerabilities.
- Develop tools and techniques to enable building systems with good usability to achieve efficient, effective, and safe interaction between users and computers.

The best of the user interfaces are the ones that are most transparent to the user and that give users a sense of mastery of their environment [26], meaning that the user interact with the system without noticing the barriers between himself/herself and the system, thus he/she feels immersed in the environment he/she is interacting with as if they are interacting with real environment.

2.1.2 Context Awareness

2.1.2.1 Definition of Context in Computing

Schilit and Theimer [15] refer to context as location, identities of nearby people and objects, and changes to those objects. The authors claim that the important aspects of context are: where you are, who you are with, and what resources are nearby. They also divided context into three categories [16]:

• Computing context: which includes network connectivity, communication costs, and bandwidth, and nearby resources such as printers, displays, and workstations.

• User context: including user's profile, location, people nearby, and current social situation and relationships.

• Physical context: such as lighting, noise levels, traffic conditions, and temperature.

Dey defines context in [17] as "any information that can be used to characterize the situation of an entity". According to this definition, an entity is a person, place, or object that is relevant to the interaction between a user and an application, including the user and applications.

Chen and Kotz define context in [18] as the group of environmental states and settings that either determines the behavior of an application or in which an application event happens and is interesting to the user. Here they defined the first kind of context as active context that influences the behaviors of an application, and the second kind of context as passive context that is relevant but not critical to an application. They also considered that "time" is an important context for many applications, and that when recording the user and physical context across a time span, we obtain a context history that can be useful for several applications.

Starting from the definition of Chen and Kotz which is more practical than the previous definitions we think that the context used in our system falls under the category of active

context that affects the application's behavior. To prove this classification we define the context of our system and we show how it affects its behavior. The context in our system is the environment of the videoconference including on each site: the conferees identities, their status (speaking or silent), the history of the conferees, the importance of the site (CEO office, regular meeting room, etc), the manipulation of a shared desktop between conferees, and the hardware display setup at this site. The context of each site affects how the view of that site is allocated on the display of other sites with the proper position and size. Thus according to the definition of Chen and Kotz, the set of environmental states and settings on each site determines the application's behavior.

Time is considered a part of the videoconferencing context. When the history of a conferee is recorded across a time interval, it affects the allocation of his view on the display (position and resolution). For example, a conferee active only during the last hour is displayed on a less important location and size (side of the screen) relative to an active conferee during the last two meetings (displayed on the middle of the screen.)

2.1.2.2 Context-aware Applications

Schilit defines context-aware computing by categorizing context-aware applications as follows [16]:

1. *Proximate selection*: is a user interface technique where the objects located nearby are emphasized.

2. *Automatic contextual reconfiguration*: is a process of adding new components, removing presented components, or altering the connections between components due to context changes.
3. *Contextual information and commands*: can produce different results according to the context in which they are used.
4. *Context-triggered actions*: are simple if-else rules used to specify how context-aware systems should behave. The category of context-aware software is similar to contextual information and commands; the difference is that context-triggered action commands are invoked automatically according to the if-else rules.

Pascoe [19] proposes taxonomy of context-aware features, including contextual sensing, contextual adaptation, contextual resource discovery and contextual augmentation

Dey lists three categories of context-aware features that applications may support: presentation of information and services to a user, automatic execution of a service, and tagging of context to information for later retrieval [17].

Chen and Kotz [18] distinguish between two ways to use context: automatically adapt the behaviors according to discovered context, using active context; or present the context to the user on the fly and/or store the context for the user to retrieve later, using passive context. They have two definitions of context-aware computing:

1. Active context awareness: an application automatically adapts to discovered context, by changing the application's behavior.
2. Passive context awareness: an application presents the new or updated context to an interested user or makes the context persistent for the user to retrieve later.

In our system, the active context-aware computing is of our interest, because it helps making videoconferencing systems less controlled by the user and thus eliminate the burden of the tasks that consume his time, such as allocating the views on the proper location on the display. So by making videoconferencing application actively context-aware we are helping this technology to be automated, consequently we are facilitating the interaction between humans and this technology.

2.1.3 Automated Layout

Layout, in the videoconferencing context, is the procedure of determining the position and size of visual objects on the display, and the result of that process [20].

Automated layout is the use of a computer program to automate the layout process. It is at the crossroads between artificial intelligence and human computer interaction. Automated layout is becoming more and more important because the amount of data needed to be presented quickly surpasses our ability to present it manually [20].

2.2 Related Work

In this section the two most adopted methods that address automated presentation layout are shown, afterwards we present a context-based automated layout generator for web pages, and then iCam system used in automated videography, and at the end we present two applications where the user can customize the display according to his preference.

2.2.1 Automated Layout Techniques

Lok and Feiner [20] distinguished between two methods that address automated presentation layout: the constraints-based method and the learning technique. The goal of the constraints-based method is to take the constraints between the material intended to be viewed (components), and make a constraint network to generate a set of positions and sizes for each of the components in the network. There are two types of constraints:

1. Abstract: the constraint describes a high-level relationship between two components that are to be included in the layout (e.g., "Caption x describes picture y").
2. Spatial: the constraint enforces position or size restrictions on the components (e.g., "Caption x below picture y").

Figure 2.1 shows and example of both spatial and abstract constraints. Figure 2.1 (a) shows a network of constraints that might be used in an automated layout system. Figure 2.1 (b) shows a set of spatial constraints that express the constraint network. Figure 2.1 (c) shows a set of abstract constraints that express the same relationships.

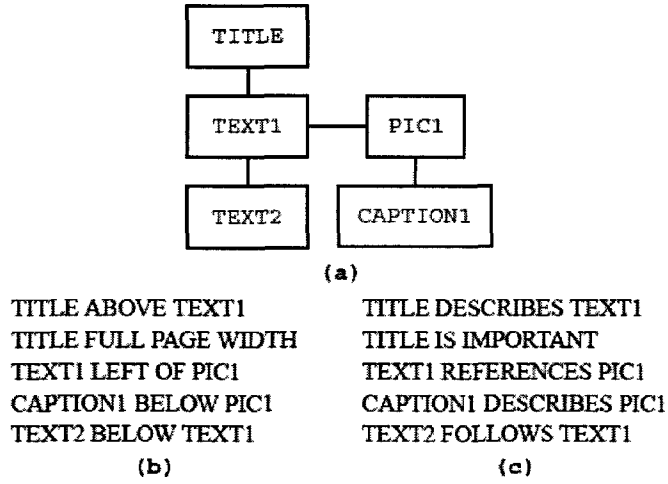

Figure 2.1 Spatial and abstract constraints [20]

The vast majority of research in automated layout to date focuses on constraint-based methods (e.g. [27- 38]).

In the learning technique, the system "learns" based on interaction with the user. For example the user sets the system into training mode where the relative locations of components are demonstrated. Spatial constraints that will be used to generate the layout are then extracted from the interactions with the user. Systems in [34], [39] and [40] have some form of learning used during the interactive specification of constraints.

In videoconferencing systems the constraints approach is not feasible. Since the content of a view affects its layout, and since the content features of a view are not limited, it will be a burden on the user to specify the constraint rules for each of these features i.e. the status (talking or silent) of each conferee, their roles, their locations, etc. Moreover

the user has to specify the constrains to deal with all possible number of views, for example a display with three views has constraints rules different from those with five views. Even more, the display setup can be different between videoconferencing sites, again the user has to specify the constraint rules for every possible display setup configuration, for example a display made of two 1920*1080 screens and an HD camera has constraints different than the display of 15.4" laptop with a webcam. To be more specific, if we use constraint based approach the number of rules needed in videoconferencing systems equals the number of all features of a view, times the number of the views, times the number of possible display setups that shows these view. Thus it is clear that the constraint-based approach is not feasible.

In videoconferencing systems the learning technique is not feasible as well, because one of the advantages of videoconferencing systems is to reduce the travel time of conferees. So adding extra time to train the system about how to layout views in these systems before the start of the meeting will diminish this advantage and the conferees will not be satisfied with the system as a whole.

2.2.2 DesignComposer: A Context-based Automated Layout Generator for Web Pages

The rationale of DesignComposer [21] is that online presentations are often partially or fully dynamic. They are displayed in heterogeneous environments of various hardware and/or software and the contents might be generated during display time from a

knowledge source like a database. Using a static layout under these conditions may result in layout deficiencies which have a negative effect on readability and understandability of the offered information.

DesignComposer is created based on a hybrid approach, combining constraints, rules and pattern techniques. One of the main techniques used is the separation of layout from the presentation; this means that the constraints of how to present objects are separated from their content. The knowledge is bundled into a layout style, which inherits the following advantages:

• Support of document classes: one style can be applied to different presentations; a presentation can be shown in different styles.

• The author is freed from the often cumbersome and difficult layout task.

• Maintenance of layout knowledge is simplified.

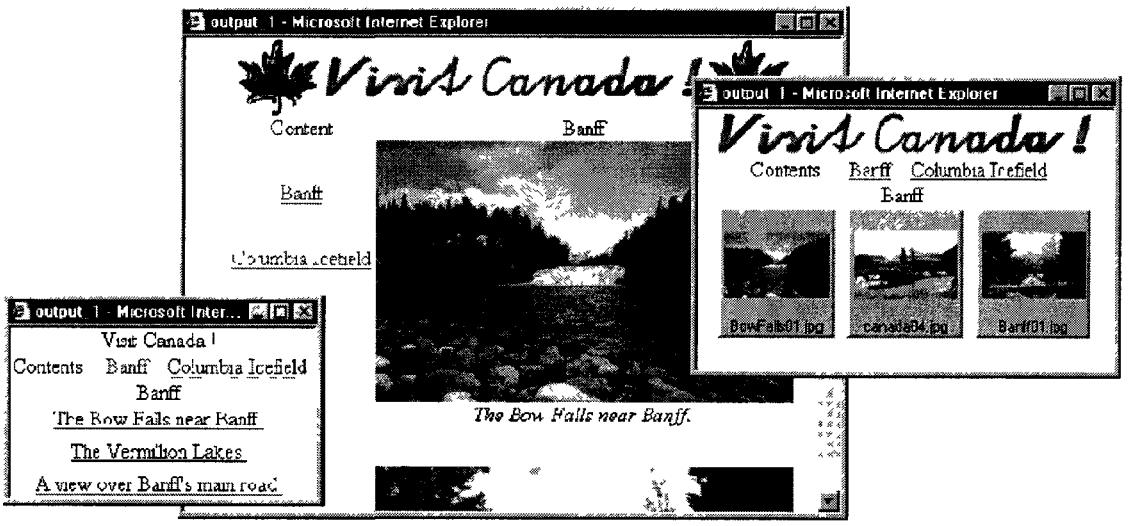

Figure 2.2 Different layouts for one presentation in DesignComposer [21]

DesignComposer is targeted to be used for webpage and uses XML or HTML scripts to interact with the elements of the Document Object Model (DOM), thus it cannot be used in the videoconferencing context. Also it does not consider heterogeneous hardware setups, which is the case in the videoconferencing technology.

2.2.3 iCam

In iCam [22] system there are multiple software modules called Virtual Cameramen (VC). Each VC controls a camera, automatically tracking a lecturer or a talking audience member. The VCs communicate with another software module called Virtual Director (VD), which is responsible for selecting the best camera shot given the available camera shots from all VCs by using a timed automaton (finite state machine with timers). For example, when a student asks a question, the VD will automatically switch to the camera that shows the student; when a camera has been on air for a certain period, the VD will switch to another camera to improve the output. This system is a timer-driven switch.

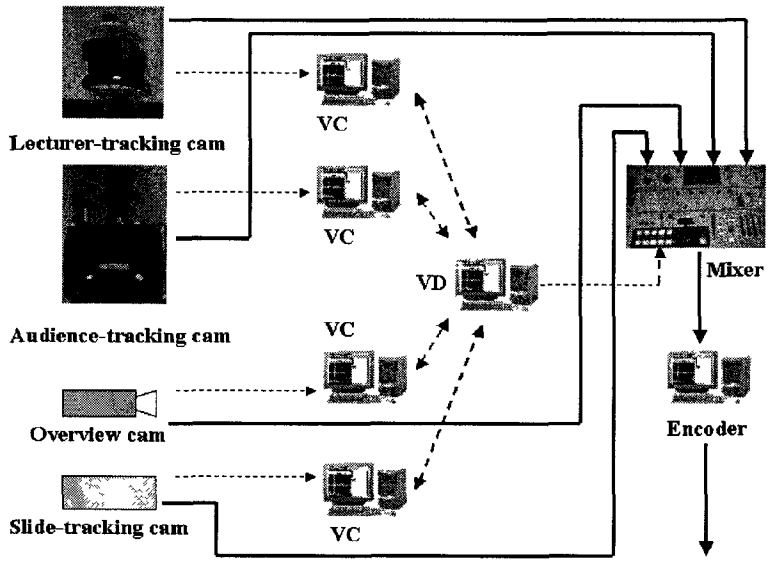

Figure 2.3 iCam system block diagram [22]

Since only the presenter or a talking audience is shown at any point of time in iCam, so this system is considered of type "continuous presence", thus it cannot be used in videoconferencing context where multiple conferees need to be shown at the same time.

2.2.4 Virtual Director for Multi-side Distributed Education System

The virtual director for multi-side distributed education system [23] is an automated display management system used in distributed education where the lecturer and the student can be located in more than one class room. In this system a user can configure his/her preferred streams to be shown and the time and events that trigger the transition between different views. As seen in Figure 2.1, the model is formed of three layers:

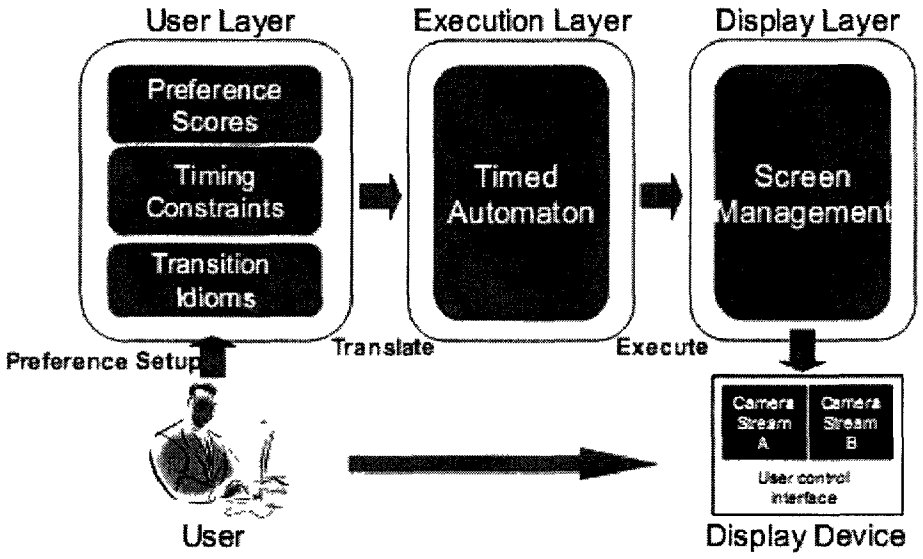

Figure 2.4 The three layer model of the virtual director [23]

- First, the user layer enable the user to set:
  - Preference score for every camera in the system: must show, show in rotation and do not show.
  - Timing constraints: minimum and maximum show time for every camera.
  - Transition idioms: special transitions (example: after x seconds of the beginning of the class show camera C).
- The second layer is the execution layer. It selects the camera streams according to the user preference. Rely on finite state machine: states and transitions (time and event triggered)
- The third layer is the display layer that displays the given streams form the previous layer. It either shows one stream at a time or multiple ones (split screen or picture-in-picture layout).

The advantage of this system is that each site has its own virtual director; this gives the user at each site the ability to enter his/her preference for the display. The disadvantages of this system are that only one view can be shows at the time, and it cannot adapt to different hardware setups.

2.2.5 Automatic Video Production with User Customization

Automatic Video Production with User Customization (AVPUC) system [24] composes multiple camera streams depicting the same event from multiple perspectives into a video program that is customized to audience's preferences. As seen in Figure 2.2, this system is made of four components:

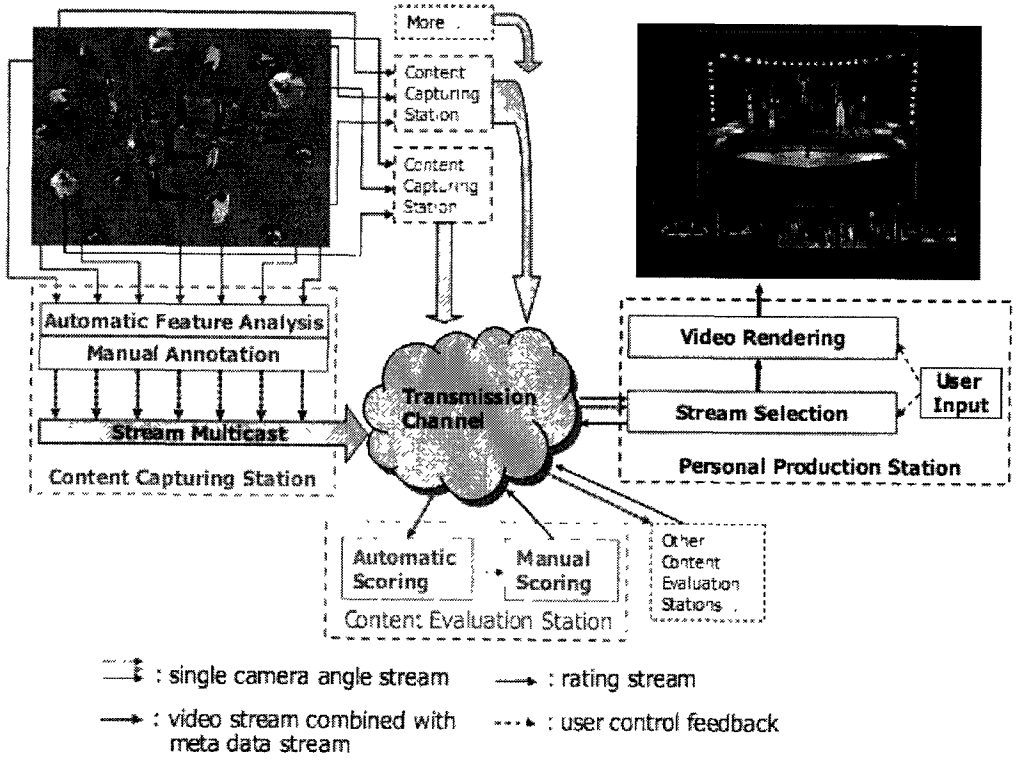

Figure 2.5 The architecture of AVPUC [24]

1- Content Capturing Station: its purpose is to capture video content and provides meta-data that describes the features of the captured content based on computer and human annotation.

2- Transmission Channel: to connect the three other components

3- Content Evaluation Station: evaluates the content streams from the first component and publishes the corresponding rating streams.

4- Personal Production Station: Each audience receives the content and meta-data streams from the Content Capturing Stations and rating streams from Content Evaluation Stations, and performs the task of modifying their rating scores based on audience preference, comparing the streams by their interest scores and composing a final video

program rendered on the user interface. The last step is done by comparing the final score of all video streams, for example if the score of stream 1 is considerably greater than stream 2 score and the latter is greater than the rest of the streams, then the display is picture in picture (stream 1 occupy the larger area of the screen). There are five types of display: single, p-i-p, up-down, 2-p-i-p, and 4 split screen as shown in Figure 2.3.

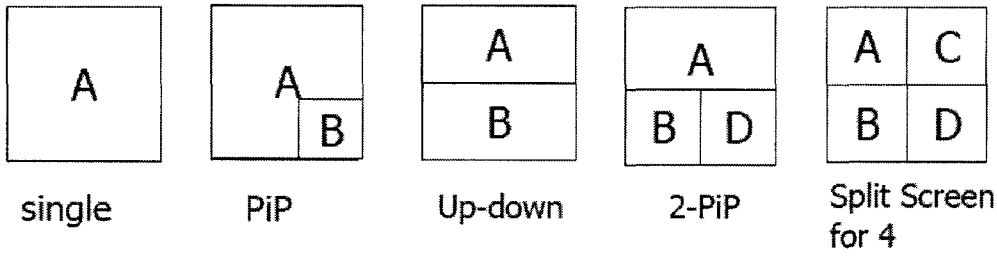

Figure 2.6 Various types of display in AVPUC system [24]

The advantage of AVPUC is that it can show more than one view on the display according to features evaluation results of all existing views, this concept is adopt in the proposed layout manager to make it aware of the context of each site participating in the videoconference. A major limitation of this system is that human is still part of the feature detection and scoring processes which make it impossible to be used as full automated layout manager in videoconferences. Another limitation is that this system does not adapt to different hardware setups.

Chapter 3

System Design

This chapter describes the real estate metaphor used in the system, followed by illustration of the overall system architecture along with detailed description of the design of the two main components: the context awareness component, and the layout manager component, with the novel algorithm discussed in details.

3.1 Real estate metaphor

The real estate metaphor is used as a real-world-scenario to solve the problem of allocating the views received from multiple sites on the display setup. First we introduce

the problem space of the metaphor, followed by an overview of the idea of the metaphor and how it helps to layout views on the display.

3.1.1 Real Estate Problem Space

According to the real estate market theory, the unit that is offered for sale is a piece of land that has a price. A buyer should have money more than the price of the piece of land in order to buy it. When a buyer wants to buy a piece of land he/she searches for the most appropriate one in term of price, location and size. If two buyers are competing on a piece of land, the buyer that is willing to pay more money will get it.

In order to make this problem space similar to our problem space we make the matching between the terms used in the real estate market and the technical terms used in layout systems: a display resembles a "*land*" that is divided into equal size blocks called "*lots*" each having a "*price*" associated with it. A set of lots having a rectangular shape is called "*site*". A view resembles a "*development*"; it is the buyer of a site on the land. From this point on the names reflecting the real-estate metaphor will be used instead of the technical names (view, display, and block). Figure 3.1 shows one screen and the used metaphor terms.

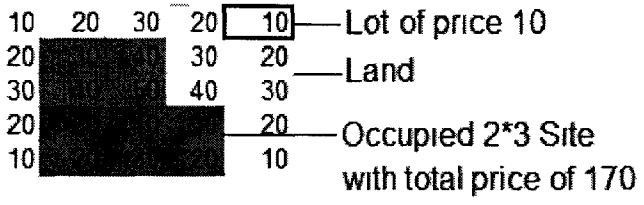

Figure 3.1 A sample land for the real estate metaphor

3.1.2 Using The Metaphor

In this section we are not describing the details of the algorithm of allocating the developments on the land, this is done in section 3.4.2; rather this section mentions the relevance between the real estate market theory and the proposed algorithm for laying out the developments on the display.

In order to apply the real estate metaphor in the layout management problem space we imitate the features of a piece of land and the behavior of the buyer. In the algorithm we consider that a development of size n*m tries to allocate a site with the same size and with a price less than its credit. Among the possible options, the development chooses the most expensive site it can afford. This site can be free so it allocates it, or can be occupied by another development so a competition between the two developments takes place, the winner allocates it and the loser looks for another site to allocate. The competition is represented by raising the price of the site. Therefore similarly to what happen in real estate market the price of a site is subject to changes.

Moreover, to distinguish between the importance of the developments, we assign a credit for each depending on how important is the content. The content can be either a video or a desktop, a video is the view captured by a camera on one location, and a desktop is shared between participants who can manipulate and/or see it. The factors that contribute to give credit to developments are listed in table 3.1. These factors are derived from a brainstorming session with a researcher at Magor Corporation Inc [2], therefore according to videoconferencing experts the following are the most important factors that can be used to measure the importance of a view based on its content.

Table 3.1 List of features used in Context Awareness component

Feature	Possible Value
Participant Role	CEO, Manager, Team leader, Engineer, etc
Participant History	Very active, active, silent.
Participant status	Talking, Silent, Active but currently silent.
Location	Board of directors meeting room, sales department, etc
Video Quality	HD video, medium resolution video, low resolution video.
Network status	Congested, normal flow.

Another aspect relevant to the real estate metaphor is what is referred to as "the first time buyer plan" in which a new development gets extra credit to give it priority over the other developments and thus allow it to allocate a more important site on the land.

3.2 System Architecture

3.2.1 Overview

Figure 3.2 shows the schematic view of the overall system. The followings are explanations for each of its comprising components:

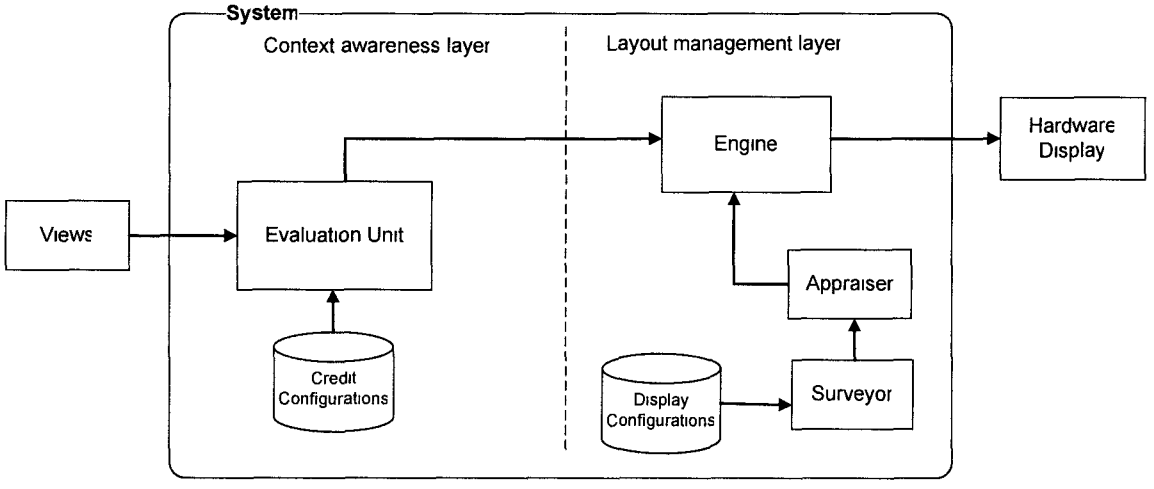

Figure 3.2 Schematic diagram of the proposed system

A **View** can either show the shared desktop between conferees, so it is called desktop view, or it can show what a camera captures on one location so it is called video view. One location can have one or more video views.

Hardware Display is the physical display setup where the views are laid out. It is formed of one or multiple screens, and one or multiple cameras.

The **Context Awareness Layer** receives the views from different sites and evaluates them according to the list of features found in table 3.1. The role of this layer is to assign a credit value for each view by extracting the features from the views and add up the credit worth of each feature. The "**Credit Configuration**" database stores the credit worth of every feature. As an example, a feature is "the participant role", its value is "CEO", and its credit worth is 100. This layer is described in details in section 3.3.

The **Layout Management Layer** receives the list of views along with their evaluated credits from the Context Awareness Layer. The **Engine** module is the core of the system that uses the algorithm described in section 3.4 to allocate the views on the hardware display. The "**Display Configuration"** storage contains the geometric description of the display that are transformed by the Surveyor to 2D matrix of Lots, and then edited by the Appraiser to assign a price for each Lot; this process is detailed in section 3.4.1.

3.2.2 Use Case Model

In this section we describe the high-level user-centric functional requirements of our application using UML-based use case diagram. The use case diagram in Figure 3.3 shows the interactions between external actors and the application. The actors are:

- Display Configurator that describes the geometry of the display i.e. locations and dimensions of screens and locations of cameras.
- Credit Configurator that assigns a credit for every feature value that a view can have i.e. video showing a CEO has credit of 100.

- A Viewer that represents any participant in the videoconference who can see the display and can be captured by any of the cameras.

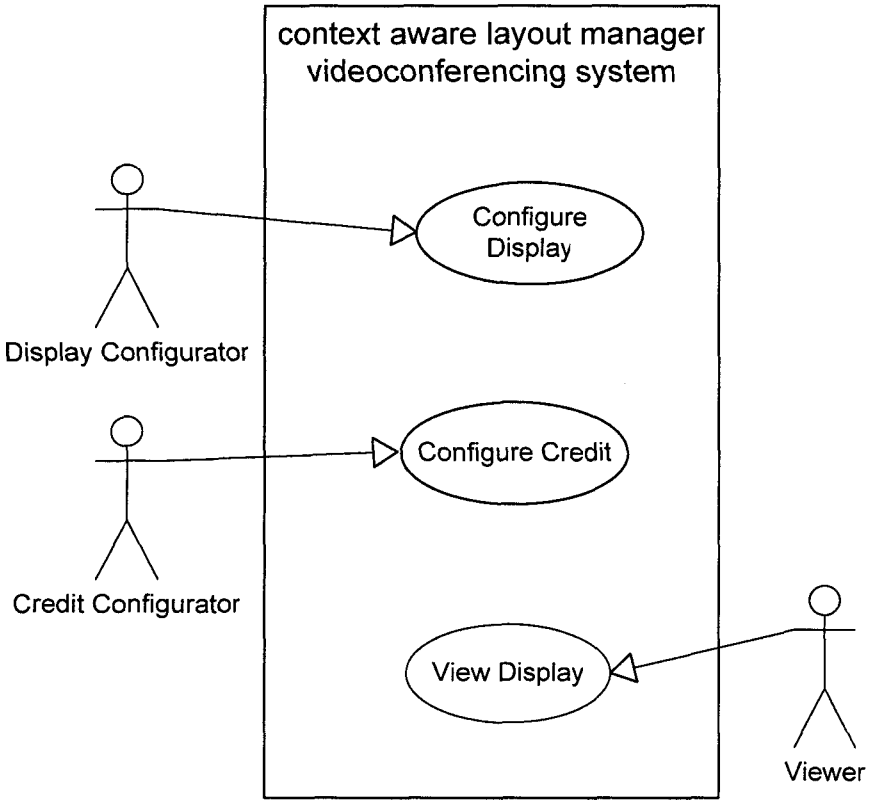

Figure 3.3 Use case diagram of the prototype system.

The use case model consists of several use cases which are described briefly in this section:

- **Configure Display:** this use case is used by the Display Configurator to specify the locations and dimensions of the screens and the locations of the cameras; the reference point is the top left corner of the display setup. The display configuration is specific to one display setup i.e. one participating site.

- **Configure Credits:** this use case is used by the Credit Configurator to assign a credit for every feature value used to evaluate a view. The changes that the user makes are visible by all sites participating in the videoconference.
- **View Display**: this use case is used by the viewer, and actions that the user performs are the same actions in a face to face conference i.e. seeing the display, talking, listening, gestures, etc.

One user can perform all the above three mentioned use cases.

3.2.3 System Features

The prototype of "*context aware layout manager for video conferencing systems*" presents an implementation in accordance to the presented architecture, which is introduced in section 3.1.1 and offers the following highlighted features:

- Controls the layout of any hardware setup and any number of views using the real estate metaphor without the user intervention and specifying any rules for layout.
- Evaluates views according to a list of features in order to give them credit to be able to buy the most appropriate area on the display.
- Gives the user the ability to represent any display setup he/she wishes to use by specifying the dimensions and position of every screen, and the position of every camera.
- Uses a novel algorithm for allocating views to any hardware setup described in section 3.4.2.

These features are described in the following two sections.

3.3 Context Awareness component

The proposed system is considered context aware because, according to the definition of "Context" in [8] the external information is utilized to adapt the data and the behavior of the system. More specifically, the received views from remote participants are analyzed for the purpose of assigning credit value for each of them by extracting the features stored in "Credit Configuration" database as described in section 3.2.1. Thus in the proposed system each receiving site is responsible for the analysis of the remote views in order to assign credit for each of them. The advantage of this technique is that each site can specify its own features that define the context of the videoconference. As an example, one site might want the importance of views to be affected only by the roles of participants in them, regardless of their status (active or silent); other sites can specify new features other than the one mentioned in table 3.1: for example, the lighting condition of the views. However the disadvantage of this technique is the views analysis and the credit computation is done at each site, so possibly the same analysis is done at more than one site. For example it is possible that all sites assign credit for views according to the activity of the participants inside them.

To overcome the latter limitation, another technique can be used. The view(s) produced at a site is (are) evaluated at the same site in order to calculate its (their) credit worth.

Then the credit is sent along with the view(s) to the other participants. The advantage of this technique is that the receiving sites do not need to evaluate the views, and thus reducing the overhead processing time at each site. The disadvantage is that the views are evaluated according to the preference of the sending site not the received site.

The factors listed in table 3.1 are used by the proposed system to evaluate the views and generate credit accordingly. To achieve this, the system needs to measure these factors. For the implementation of the system we omit the context awareness component, instead we show the techniques that can be used to measure some of the factors.

To measure the activity of a participant, his/her audio level can be used, if it exceeds a certain threshold the participant is considered talking and thus active. A more advanced technique is mentioned in [22] where a microphone array is used to estimate the source of the sound; this technique can be used in the proposed system if there are more than one participant at one site. The history of the participant in a view can be measured by calculating the percentage of time during which the participant was active: the higher the percentage the more credit the view gets.

The role of the participant can be identified by the login username for every user; the "Credit Configuration" database should contains the credit value for each user. None identified users can be assigned a default value. This technique can be used to identify the location from which the user logs in to the system. Another, more advanced, well

known technique is face recognition that is feasible for both still images and videos. The techniques mentioned in [1] can be used for this purpose.

The video quality can be measured by measuring the resolution of the remote video in pixels. The resolution is affected by the camera used to capture the video.

3.4 Layout Management component

The **Engine** module is the core of the system, its role is to:

- Transform the list of screens stored in "Display configuration" to 2D matrix of Lots.
- Assign a price to a Lot according to its position on the Land (section 3.4.1).
- Controls the prices of the lots during the allocation of developments.
- Most importantly, allocate a development on a site of the land using the algorithm described in section 3.4.2.

3.4.1 Building the land

As mentioned earlier the system is adaptable to new display setups without needing the user to be involved in the complicated and time consuming process of creating layout rules such as in systems [22] and [23] that are based on finite state machine and time and/or events triggers. All what is required from the user is to specify the geometry of the display i.e. the dimensions and the position of each screen, and the position of each camera, by storing these values in "Display Configuration" database. Afterwards, the Surveyor uses this information to construct a 2D matrix of uniform sized Lots. At last,

the Appraiser uses this 2D matrix and assigns a price for each Lot according to its position on the Land.

Since the middle of the screen is where the user looks at by default, then it is considered more important than the sides of the screen, thus the Lots in the middle have higher price comparing to the Lots close to the sides of the screen. The bezel, which is the black border around the edge of the each screen, is represented by arrays of Lots having infinite price to forbid any development from allocating them (see Figure 3.4).

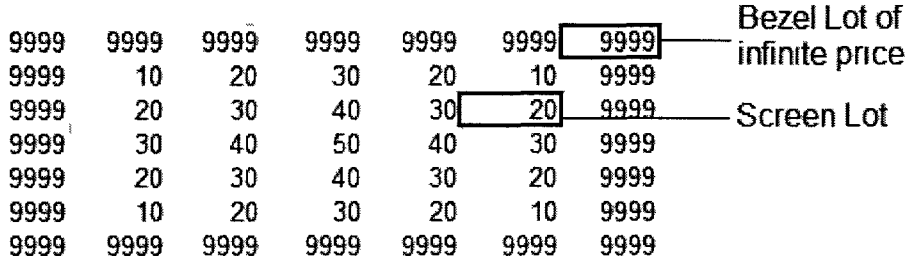

Figure 3.4 A land having high priced lots in its middle

This way of assigning price is not the only one, a user can specify other ways of pricing depending on his/her needs, for example if he/she wants the views to be shown close to the camera(s) to provide eye contact between participants, then he/she can assign the price of a Lot according to the distance between the Lot and the nearest camera, the closer to the camera the higher is the price. Figure 3.5 show a camera on the bezel of the screen, the closest Lot to the camera has the highest price (70) and the furthest Lot has the lowest price (10).

9999	9999	9999	9999	9999	9999	9999
9999	10	20	30	20	10	9999
9999	20	30	40	30	20	9999
9999	30	40	50	40	30	9999
9999	40	50	60	50	40	9999
9999	50	60	70	60	50	9999
9999	9999	9999	Camera	9999	9999	9999

Figure 3.5 A land having high priced lots near the camera

The main advantage of the automated construction of Land is that the system can be used with any display setup and the user's task at this location will be merely to specify the physical geometry of the display. Whereas in rule-based systems, the user has to create new rules for each new display setup and manipulate the system accordingly. This is a clear limitation if these systems are used in videoconferencing because the display setup is not necessarily the same among multiple locations.

3.4.2 Allocation Algorithm

The algorithm imitates the real-estate market theory to allocated developments on the land. Same as the land buyer has many options to buy a piece of land, the algorithm provides a list of options for the sites that a development can allocate, afterwards the best one is chosen and then allocated.

The input of the algorithm is the list of developments. The output is a list of bindings between one development and one site. Following are the details of the different phases for the allocation of one development on the Land.

3.4.2 1 Initialization phase

When a display setup is used for the first time or when it is changed, the Land is built according to the description of section 3.4.1. For example if a screen or a camera is added or removed, the system adapts with the changes dynamically without the need for it to restart or disconnect the current videoconference.

Afterwards the average Lot price is calculated. It is used to give extra credit to the newest development; this is similar to the "first time buyer plan" in real estate. The newest development gets extra credit according to its size, if the size is n*m then the extra credit will be n*m*average Lots price.

We use the Land shown in Figure 3.6 as the illustration example for the rest of the algorithm description. The average Lots price is 26.

9999	9999	9999	9999	9999	9999	9999
9999	10	20	30	20	10	9999
9999	20	30	40	30	20	9999
9999	30	40	50	40	30	9999
9999	20	30	40	30	20	9999
9999	10	20	30	20	10	9999
9999	9999	9999	9999	9999	9999	9999

Figure 3.6 Land used in algorithm description

We assume a development X of size 2*3 and credit of 280 is requesting to be displayed on the Land. Since it is the only and the newest development so it gets extra credit of 2*3*26 = 156. The total credit of the development is 156 + 280 = 436.

After this phase the Land is ready for the allocation phase.

3.4.1.2 Allocation phase:

Whenever a new development is created, or an existing development is deleted, all the existing developments are allocated even if they are already allocated. The allocation phase is illustrated in Figure 3.7.

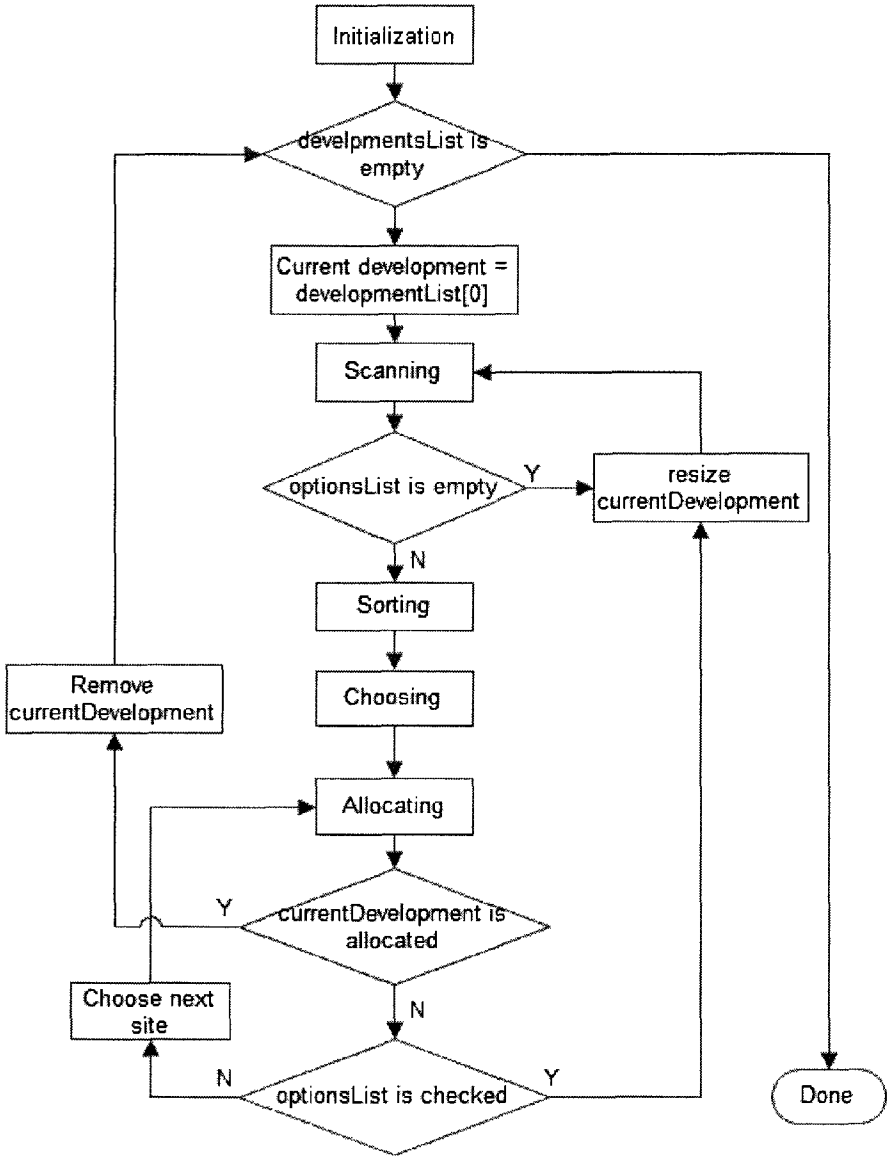

Figure 3.7 Allocation steps of developments list

The major steps to allocate a development are the following:

Step 1: Listing and Sorting all affordable sites

Similarly to the real-estate market where the buyer must afford the price of the land he is buying, a development must afford the price of the site it is trying to allocate. Assuming a development has a size of n*m. In order to find the best site to allocate, it scans the land to search for all n*m sites regardless of whether they are free or occupied by another development. All sites having a price less than the development's credit are stored in options list. The list name indicates that all sites in this list are possible candidates for allocation. Afterwards the list is sorted from the most expensive to the least expensive. If there is more than one site with the same price they will be sorted from the sites with least occupied lots, or free, to the sites with the most occupied lots.

Using the example of the Land in Figure 3.6 we know that the development size is 2*3, so the Land will be scanned to find all possible 2*3 sites. The first site scanned is the top left one which has a price of 10+20+30+20+30+40 = 150. After scanning the Land, the prices of sites in options List are as follows: 150, 170,150, 210, 230, 210, 210, 230, 210, 150, 170, and 150. Then options list is sorted, the order becomes: 230, 230, 210, 210, 210, 210, 170, 170, 150, 150, 150, and 150. That is from the most expensive site to the least expensive.

Step 2: Choosing the best site

In this step, options list is used to choose a candidate site for allocation. This step is divided into two distinct sub-steps.

Sub-step1: Starting from the most expensive site in options list, the first site satisfying the two conditions: *free* (i.e. none of its lots are occupied), and having a *total price more than half of the development credit* (to assure fairness as will be explained later in section 3.4.4.2), is chosen as the candidate site.

Sub-step2: If none of the sites satisfy the two conditions of sub-step 1, *the most expensive* site in options list is chosen as the candidate site for allocation.

For the example used above, the chosen site in options list is the first site because it is free and its price 230 is greater than half the development credit (436/2=218).

Step 3: Allocating the chosen site

In this step the chosen site is allocated. Here there are two possibilities: if the chosen site is free it can be simply allocated by the development, otherwise if at least one of the lots of the chosen site is allocated by another development a competition takes place. Before going into the details of competition, the following two concepts need to be clarified. Each lot has two prices: *actual price* is the price that the occupying development uses when competing with the development trying to allocate this lot that uses the *asking price.* The actual price is always less than the asking price.

If a competition takes place on an occupied site between a requesting development and an occupying development, both the actual and the asking prices of the occupied lots increase by a fixed amount. As a result of this price increase one of three cases take place:

1. *The requesting development looses the competition*: this happens when the occupying development can afford the actual price of the site it is occupying, and the requesting development cannot afford the asking price of the site it is requesting. The result is that both actual and asking price matrices are reset to their original value, and the requesting development choose the next site in options list and repeat step 3 to allocate it.
2. *The requesting development wins the competition*: this happens when the occupying development cannot afford the actual price of the site it is occupying, and the requesting development can afford the asking price of the site it is requesting. The result is that the requesting development is allocated to the requested site, and the occupying development is removed from the land and placed at the first position in the list of developments want to be allocate.
3. Both developments afford their corresponding prices. The result is that step 3 is repeated again until one of both developments wins.

For the illustration example we described before, since the chosen Site is free so it can be allocated the development directly. Therefore the development occupies the Site at position 1, 1 and size 2*3 as seen in Figure 3.8.

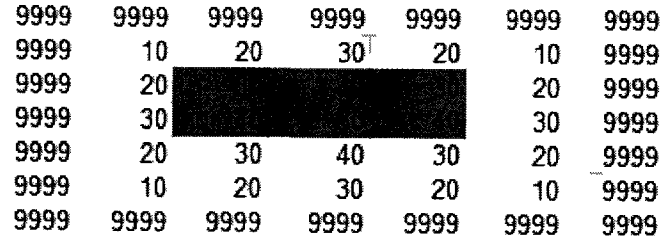

Figure 3.8 Allocation of development X

3.4.3 Walk Through Example

To clarify the algorithm more, and to go through some of the cases that we did not encounter in the simple previous example, we assume there are two developments need to be allocated: the newest development X of size 3*3 of credit 100, and a development Y of size 2*3 of credit 400. Development X gets extra credit of 3*3*26 = 234, its total credit becomes 100 + 234 = 334. The order of prices of 3*3 Sites in the sorted options list is the following: 330, 300, 300, 300, 300, 270, 270, 270, and 270. The site of price 330 is chosen since it is free and has price more than half of X's credit (330 > 334/2). So development X allocates the site at position 2, 2 and size 3*3 as shown in Figure 3.9.

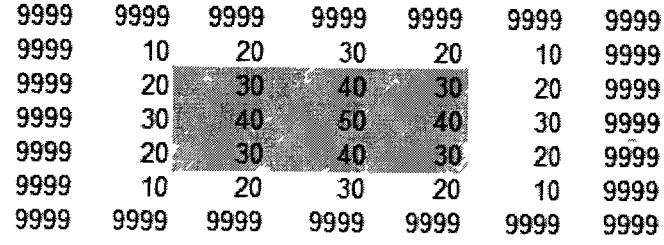

Figure 3.9 Allocation of development X

Now development Y is added. The order of prices of 2*3 Sites in the sorted options list is the following: 230, 230, 210, 210, 210, 210, 170, 170, 150, 150, 150, and 150. None of the sites is free, so the first 2*3 site at position 1, 1 is chosen as shown is Figure 3.10.

9999	9999	9999	9999	9999	9999	9999
9999	10	20	30	20	10	9999
9999	20	30	40	30	20	9999
9999	30	40	50	40	30	9999
9999	20	30	40	30	20	9999
9999	10	20	30	20	10	9999
9999	9999	9999	9999	9999	9999	9999

Figure 3.10 Development X (orange) and the chosen site of development Y (green box)

This Site cannot be allocated directly rather there is competition between developments X and Y. The asking price is the price we were using until now. The actual price is the price that the occupying development uses when competing with a development trying to allocate some of its Lots. For our example the actual price matrix is shown is figure 3.11. The asking price of any Lot is five credits more than its actual price.

9999	9999	9999	9999	9999	9999	9999
9999	5	15	25	15	5	9999
9999	15	25	35	25	15	9999
9999	25	35	45	35	25	9999
9999	15	25	35	25	15	9999
9999	5	15	25	15	5	9999
9999	9999	9999	9999	9999	9999	9999

Figure 3.11 Actual price matrix of the land

Development X uses the actual price in the competition since it is the occupying development. From figure 3.11 we can see that the price of the site it is occupying is 285. As mentioned in the algorithm description, the competition is represented by an increase

of overlapping lots, in this case the lots in the green box in Figure 3.10. The price of each lot increases by five, so the price of both sites increases by 30 (5*6 lots). After two iterations, development X loses the competition because the price of its site become 285 + 2*30 = 345 which is more than 334. This means development Y allocates its requested site as shown in Figure 3.12.

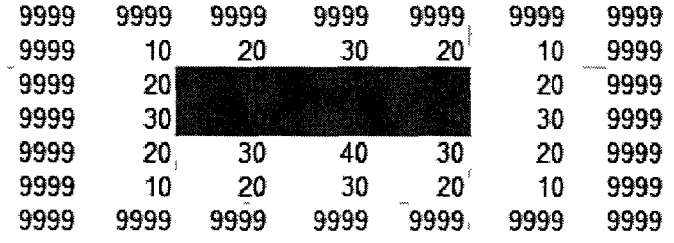

Figure 3.12 Development X loses the competition and Y Allocates the green site.

Now it is the turn of development X again. Since there are no sites of size 3*3 it will be resized, its size is now 2*3. The order of prices of 2*3 Sites in the sorted options list is the following: 230, 230, 210, 210, 210, 210, 170, 170, 150, 150, 150, and 150. The first free site of price more than half of Y's credit is of price 170 (170 > 334/2) at position 3, 1. This site is allocated as in Figure 3.13.

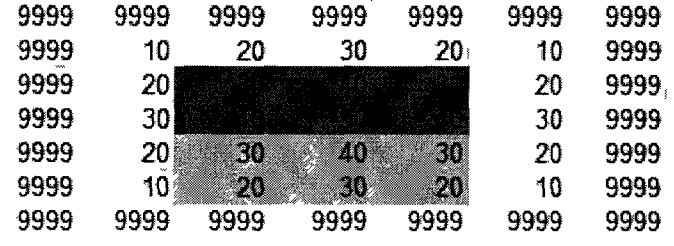

Figure 3.13 Development X us allocated after being resized

3.4.4 Algorithm Discussion

This section discusses the two major concerns in the algorithm, stability and fairness.

3.4.2.1. Stability

The algorithm should be stable when allocating new development, meaning that the land should converge to a stable allocation after executing the algorithm and not looping infinitely trying to find a site for a development. This scenario happens when a development has a small credit comparing to the blocks prices, the development might keep trying infinitely to allocate a site without succeeding. The algorithm assures stability by the following:

a. Resizing: When a development fails to allocate all sites in options list due to insufficient credit, it will be resized. Consequently, with a smaller size and the same credit, the development has more chance of affordance and thus is more capable of allocating a site on the land. This prevents a development to loop infinitely if it cannot afford any site of its size.

b. The algorithm minimizes the competition between a new development and the occupying development. When selecting a site, the new development does not choose the most expensive site it can afford if this site is occupied; rather it chooses the first *free* site in options list having a price more than half of the new development credit. By doing so, the development is not greedy when choosing a site because it accepts to allocate a site with price less than what it can afford.

c. The lots prices are reset to their original value after allocating every developments, this assures that the prices are always stable even if they go up due to competitions between developments, and not increase infinitely.

d. During competition, the algorithm considers two types of site prices: actual price for the occupying development and asking price for the requesting development. The actual price is always less than the asking price to give the occupying development the advantage on the requesting one, thus a requesting development cannot remove another development with the same size unless is has a total credit greater than or equal to the sum of the occupying development credit, plus the total number of lots multiple the difference between the actual and asking price. For example if two developments both of size 3*3 lots, the occupying development have credit of 20, and assuming the difference between the actual and asking price is 2, the requesting window should have at least credit of 38 that is 20+(3*3*2) to be able to remove the occupying development. By doing so the algorithm assures that there is few switching between developments, and that the layout is consistent if there is slight change in developments' credits, which assures stability.

3.4.4.2. Fairness

Similarly to what happens in the real estate market, a buyer with a certain credit buys a piece of land which has a price less than or close to his credit. As and example, a buyer

with total credit of 100 would not accept to buy a piece of land that has a price of 30, simply because he can afford more expensive piece, so why buying a cheap one! The algorithm assures fairness by:

a. In step 2, a new development chooses the first free site in options list with the condition that this site has a price *more than half* of the development credit. This means that a development chooses only the sites that do not have low credit relative to its credit.

b. When a new development is introduced, the algorithm gives it a bonus credit proportional to its size. This is similar to the "first home plan" in real estate market where a buyer gets a discount if he is buying his first home. The algorithm considers that only one development has this privilege at a time. This assures that a new development is displayed even if its credit is relatively small.

Chapter 4

Implementation

This chapter presents the implementation details of the real estate layout manager. We start with the technology and the deployment of the system, followed by a brief description of the layout viewer tool used to simulate the layout manager, then a description of the interface between the tool and the manager, and finally we describe the software architecture of our layout manager system.

4.1 Technology and Deployment

The technology used in the implementation is Java2 Platform, Standard Edition, v1.6 (J2SE). The layout manager that uses the real estate market metaphor was first

implemented as a standalone Java application. A simple GUI similar to the one shown in the example presented in section 3.4.3 was first implemented to proof the correctness of the layout management algorithm. All components were implemented according to the design except the sub-components of the context-awareness components that are responsible for detecting and extracting the factors that generate credits for views. However the implementation of the context-awareness component is able to add up the hardcoded credit worth of each factor and return the total credit of each view to the layout manager.

Then the real estate layout manager was integrated with a layout viewer application developed at Magor communication Inc [2]. Similar to the first implementation the credit factors were hardcoded to generate credit for each view. A brief description of the layout viewer application before integration the real estate layout manger is given below in section4.2.

4.2 Layout Viewer Application

The layout viewer application was developed at Magor communication Inc. to simulate real display setup with screens and cameras. The layout manager used to control the display is a rule-based or constraint-based manager similar to the technique described in [20] and mentioned in section 2.2.1. The application specifies the layout rules for six predefined setups as seen in the menu of Figure 4.1, this screen shot shows how one

desktop view and two video views are allocated on a setup formed of 2-panels and 2-cameras.

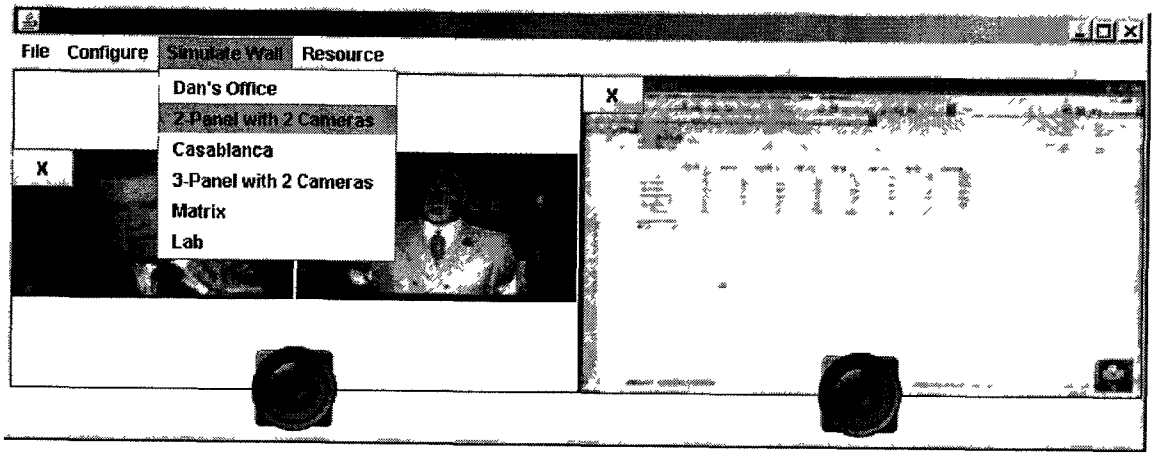

Figure 4.1 Screenshot of layout viewer application with predefined setups in the menu

The user can also specify a customized display setup if the desired setup is none of the predefined setups. As seen in Figure 4.2, the user can specify the number of screens horizontally and vertically, each screen resolution, and bezel width.

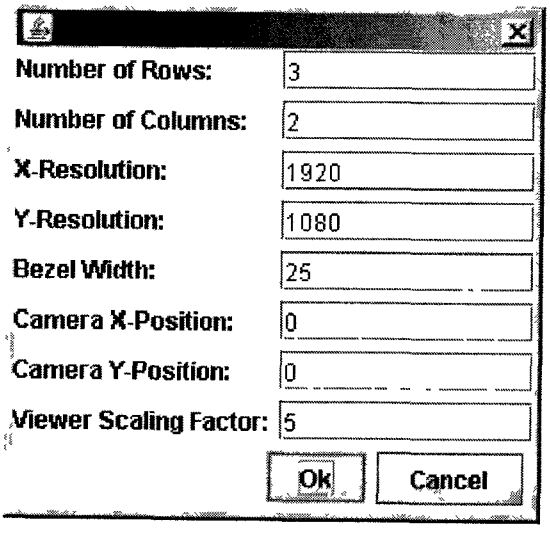

Figure 4.2 Customized display setup configuration

4.3 The Layout Manager-Viewer Interface

An interface between the layout viewer application and the layout manager is needed because each application has its specific problem-space i.e. its own data types and techniques to allocate views on the display. For example a view in layout viewer application (called resource) has no size and credit, where as in real estate layout manager the views (called developments) should have a size and a credit and the display should be a matrix of Lots.

Figure 4.3 shows the UML class diagram of the interface. The most important class is `RealEstateLayoutAdapter`. The role of the adapter is to make the translation between the layout viewer and the real estate layout controller, by:

- Transforming the list of screens to a Land i.e. a 2D matrix of Lots as described in section 3.4.1. The class `Surveyor` is used to create the 2D matrix of Lots and the class `Appraiser` is used to assign prices for Lots in this matrix.
- Transforming the list of Resources to a list of Developments.
- Using the `RealEstateLayoutController` to get the list of bindings between developments and sites namely `DevelopmentSiteBinding` list.
- Transform `DevelopmentSiteBinding` list to `ResourceLayoutBinding` list used by the `LayoutController`.

Implementation

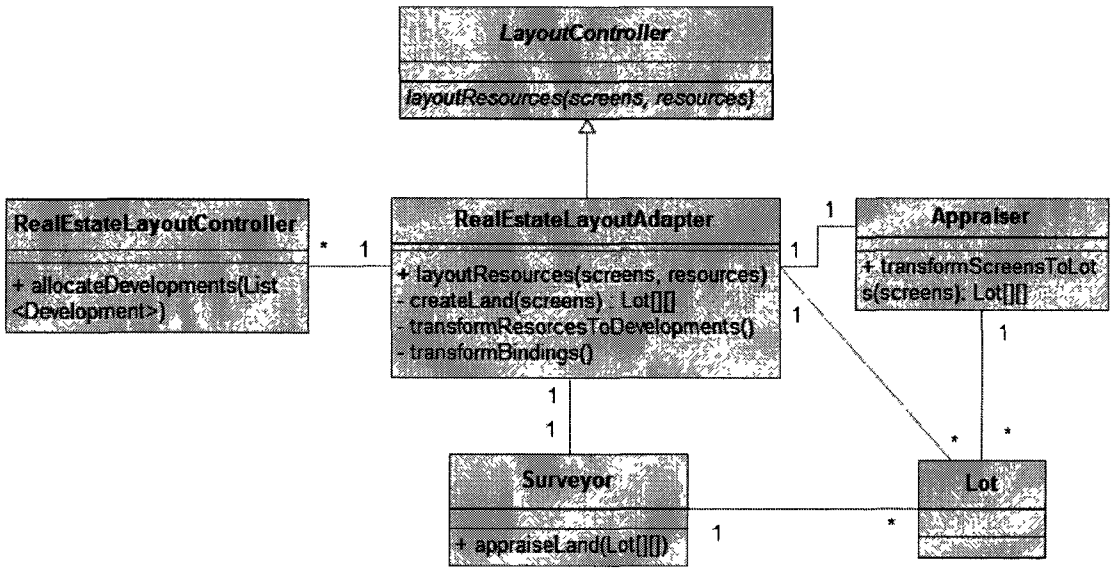

Figure 4.3 UML class diagram of the interface between layout manager and layout viewer

As shown in Figure 4.3, the `RealEstateLayoutAdapter` class implements the `LayoutController` abstract class by providing a definition for method `layoutResources()`. The `RealEstateLayoutAdapter` class uses `Surveyor` and `Appraiser` classes to create a 2D matrix of lots that represents the land, this happens when the display setup is changes i.e. screens positions or dimension, or cameras position is changed. In such case a new `RealEstateLayoutController` is created, and then the method `allocateDevelopments()` is called to get a list of bindings between developments and sites. Afterwards this list is transformed to `ResourceLayoutBinding` list that is returned to the parent class `LayoutController`.

Implementation

The UML sequence diagram in figure 4.4 shows how the class `RealEstateLayoutAdapter` uses the classes Surveyor and `Appraiser` to create the 2D matrix of lots. First the `Surveyor` uses the list of screens to create a 2D matrix of lots, and then this matrix is used by the `Appraiser` to assign a price for each lot according to the method described in section 3.4.1.

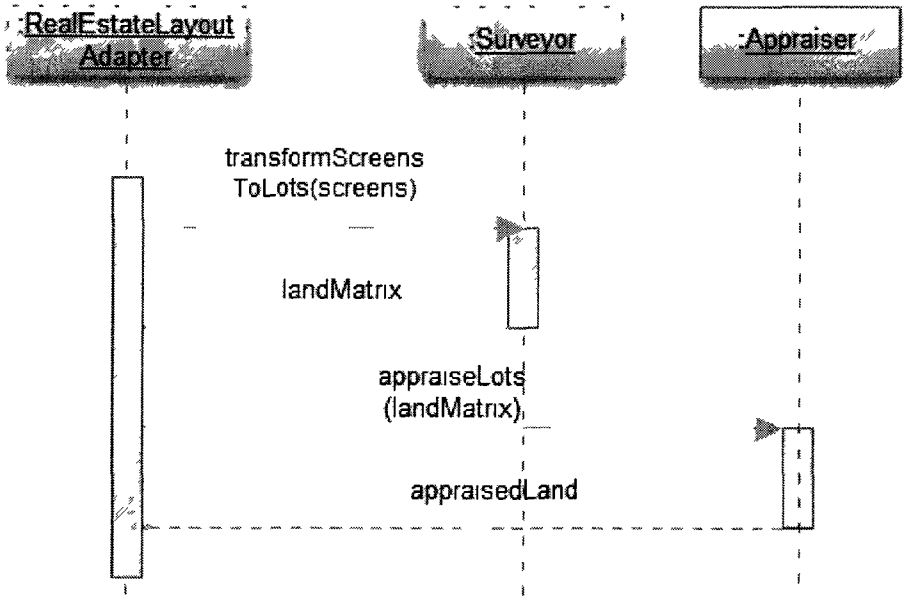

Figure 4.4 UML class diagram of land creation

4.4 Layout Manager Implementation

The real estate layout manager has been implemented using the design concepts described in the chapter 3. We start with the explanation of the software architecture of the system, and then we present selected pseudo-code sections of the implementation of

the algorithm describe in section 3.4, and at the end we present a UML sequence diagram showing the allocation of a Site by Development.

4.4.1 Software Architecture

The UML class diagram in Figure 4.5 shows the software architecture of the layout manager. The main class that implements the real estate layout management algorithm is the `RealEstaeLayoutController`. This class has a 2D matrix of lots i.e. the land, and has a list of developments that need to be allocated on the land.

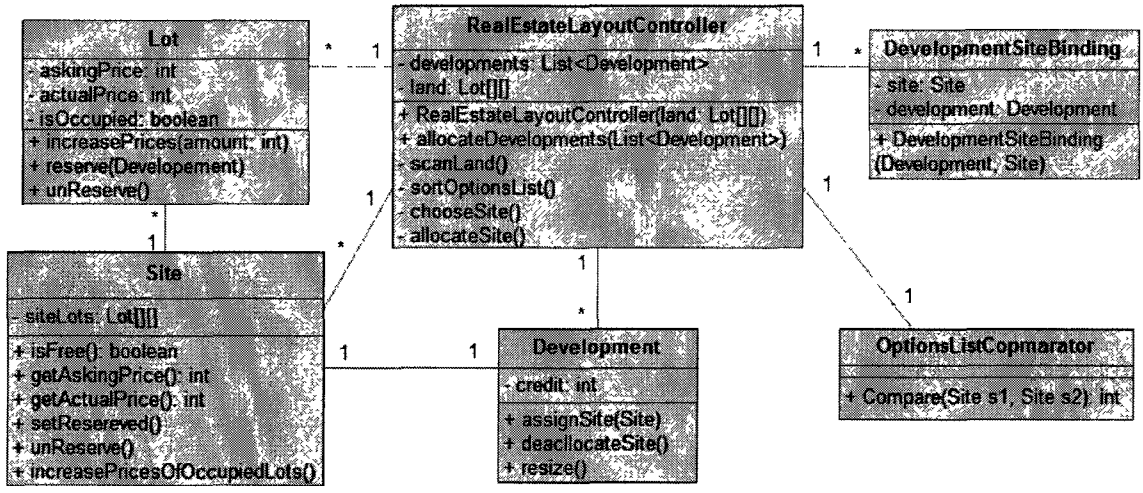

Figure 4.5 UML class diagram of the software architecture of the layout manager

The method `allocateDevelopments()` is used to allocate the of developments on the land. The following is the pseudo-code in this method:

```
while (the list of developments is not empty) {
      current development is the first in the list;
      scan the Land and store Sites in options list;

      if (options list is not empty) {
            sort options list;
            choose a site from options list;
            allocate the chosen site;
```

```
        }
}
```

This method uses the following four private methods:

- `scanLand()` to create options list that contains all sites affordable by a development. The following is its pseudo-code:

```
for (all vertical lots in Land:  i) {
      for (all horizontal lots in land: j) {
            create new  site that starts at position i,j of same
            size as current development;
            if (price if site < credit of current development) {
                  store it in options list;
            }
      }
}
```

- `sortOptionsList()` to sort options list using the comparator specified in `OptionsListComparator` class. The following is the pseudo-code of the comparator:

```
compare (site s1, site s2) {
      if (price of s1<s2) {
            position of s1 greater than position of s2;
      }
      else if (price of s1>s2) {
            position of s1 lower than position of s2;
      }
      else if (both prices are equal) {
            if  (number  of  occupied  Lots  is  s1  <  number  of
            occupied Lots in s2) {
                  position of s1 lower than position of s2;
            }
            else {
                  position of s1 lower than position of s2;
            }
      }
}
```

- `chooseSite()` to choose a site from options list to allocate. The following is its pseudo-code:

```
current site = the first element in options list;
while (a site is not chosen yet  AND the end of  options list is
not reached) {
      if  (the  current  site  is  free  and  has  price  >  half  of
      current development)
      {
            choose this site;
            place it at the first position is options list;
      }
      else {
            current site =  next element in options list
      }
}
if (no site is chosen) {
      choose the first Site in options list;
}
```

- `allocateSite()` to allocate the chosen site. The following is its pseudo-code:

```
current site = the first element in options list;
while (end of options list is not reached AND current development
still not allocated) {
      if (current site is free) {
            current development allocates current site;
            create a binding between both;
            if (current development  is not already in bindings
            list) {
                  add the binding to the binding list;
            }
      }
      else { // there is competition between occupying and
            // requesting developments
            increase the asking and actual prices of the occupied
            Lots in current site;
            if (current development can afford  the asking price
            of current site) {
                  for (all occupying developments: d) {
                        if (development d cannot afford the
                        actual price of its site) {
                              deallocate d from its site;
                              place it at position 1 of
                              developments list;
                        }
                  }
            }
            else {
                  reset the prices of all Lots;
                  current site =  next element in options list;
            }
      }
} // end of while loop
```

Implementation

```
if (current development is not allocated) {
      resize current development;
}
```

4.4.2 Site Allocation Sequence Diagram

The sequence diagram in Figure 4.6 shows the interaction between different objects to assign a site to a development. We start by assuming that the site that a development wants to allocate is free, this means none of its Lots is occupied. First the sole object of class `RealEstateLayoutController` checks if the object of the chosen site is free, the value is then returned. The box titled "opt" represents an option that the interaction sequence may go through. We assume that the chosen site is free so the sequence continues inside this box. The method `assignSite()` of the object of class `Development` is called, then this object calls `setReserved()` method of the chosen site object. The latter, in its turn, calls the method `reserve()` on all the `Lot` objects of the chosen site, here lote_n_m is the object of the last Lot in the chosen site. At last, a binding between the development and the chosen site is formed by creating an object of type `DevelopmentSiteBinding`.

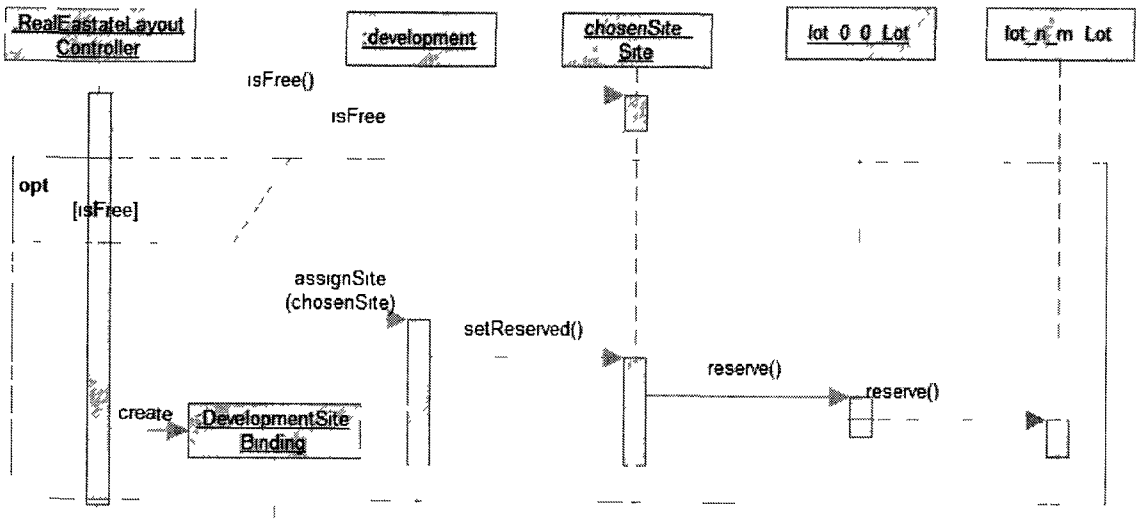

Figure 4.6 UML sequence diagram for site allocation

Chapter 5

Evaluation and Results

In this chapter we evaluate the real estate layout manager system to justify the suitability of our proposed approach. Using the layout viewer application described in chapter 4, we compare the layout results of both: the rule-based layout manager and our proposed real estate layout manager. Then we show the performance of both managers, followed by the results of the usability study that we performed to confirm the effectiveness of different aspects of our system. Finally, we provide a discussion as per the obtained results.

5.1 Experimental Setup and Procedures

The following experiment is performed using the layout viewer application on both layout managers: the rule-based layout manager and our proposed real estate layout manager. The experiment is performed on a Desktop PC (Intel Pentium 4, 2.2 GHz, 1 GB RAM) running Eclipse SDK v3.4 on Windows XP Home edition.

A resource is a window that contains a video or a desktop view. The following test cases are performed:

1. Using the rule-based layout manager and the "2 panel with 2 cameras" predefine display setup, we use up to seven resources.
2. Using the rule-based layout manager and the "Matrix" predefine display setup, we use up to seven resources.
3. Using the rule-based layout manager and a customized display setup made of 3*2 screens, we use up to eleven resources.
4. Using the real estate layout manager and the "2 panel with 2 cameras" predefine display setup using up to seven resources.
5. Using the real estate layout manager and the "Matrix" predefine display setup, we use up to eleven resources.
6. Using the real estate layout manager and a customized display setup made of 3*2 screens, we use up to eleven resources.

5.2 Results

The results of the six test cases are presented here. First we show the results of our proposed system followed by the results of the rule-based system, and then a usability study, and at the end we discuss the obtained results.

5.2.1 Rule-based Layout Manager Results

In this section we show results of test cases 1, 2, and 3 where the rule-based layout manager is used.

5.2.1.1 Test Case 1

In test case 1 a "2 panel 2 cameras" display setup is used. Figure 5.1 shows the layout of seven resources. They are all aligned horizontally and the sizes of the resources on the left screen do not allow viewers to see the content of the video clearly.

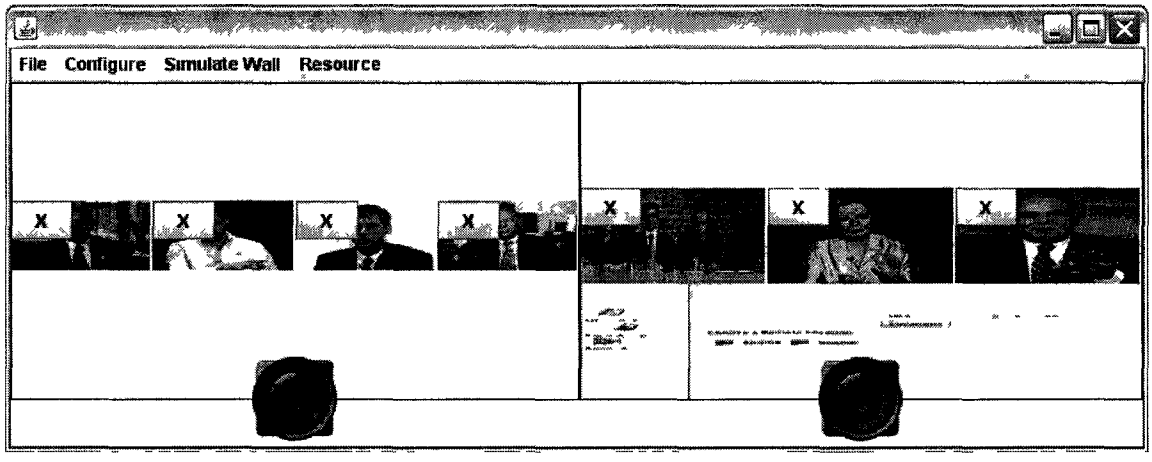

Figure 5.1 Rule-based manager with seven resources on "2 panel 2 cameras" setup

5.2.1.2 Test Case 2

In test case 2 a "Matrix" display setup is used. Figure 5.2 shows the layout of seven video resources and one desktop resource. One of the screens is left empty, the desktop resource allocates one screen, and the video resources allocate the lower two screens similar to test case 1.

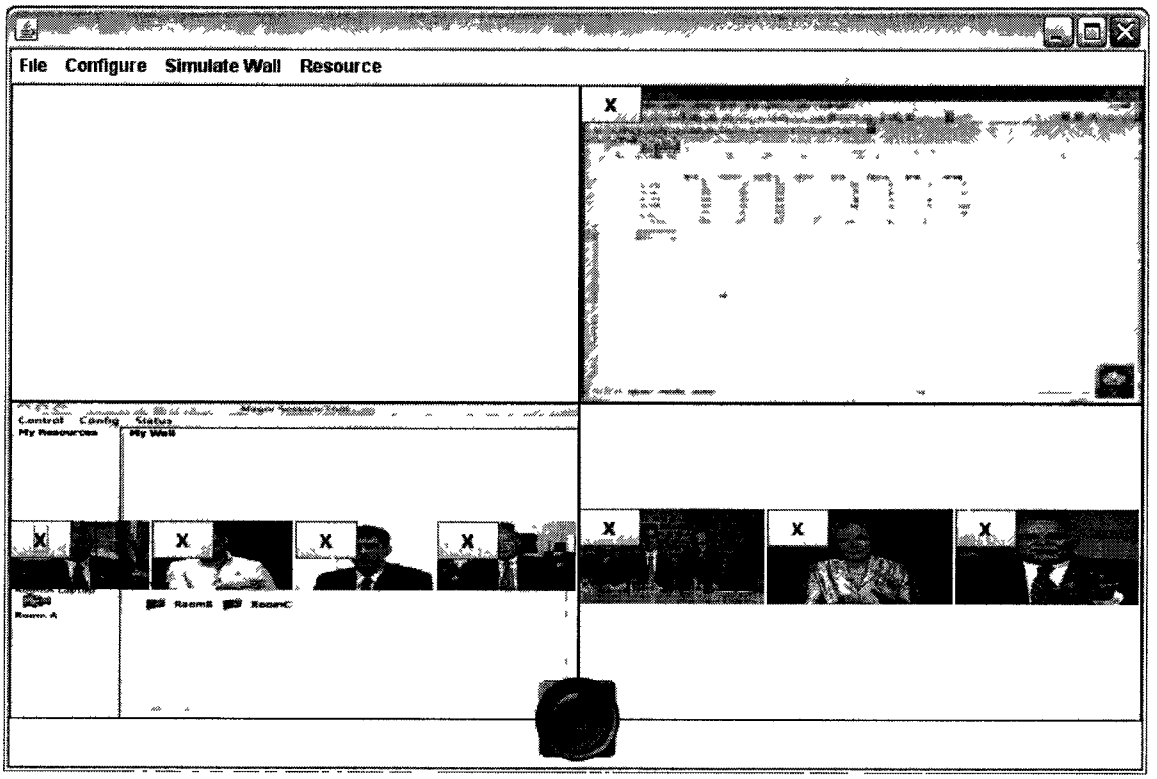

Figure 5.2 Rule-based manager with one desktop and seven video resources on "Matrix" setup

5.2.1.3 Test Case 3

In test case 3 a customized display setup, made of 3*2 screens, is used. A customized display setup means that there are no rules used to manage the layout; rather they are allocated in a pseudo-random way. Figure 5.3 shows the layout result of seven resources.

We made the width of the bezel larger to show that resources can be allocated on two or more screens as in the case of the video in lower right corner, and that resources can overlap, as in the video in the middle left, or even completely cover each other.

Figure 5.3 Rule-based manager with seven resources on 3*2 screens setup

5.2.2 Real Estate Layout Manager Results

In this section we show the results of test cases 4, 5, and 6 where the real estate layout manager is used.

5.2.2.1 Test Case 4

In test case 4 a "2 panel 2 cameras" display setup is used. Figure 5.4 shows the layout of two resources, each occupying the total space of a screen.

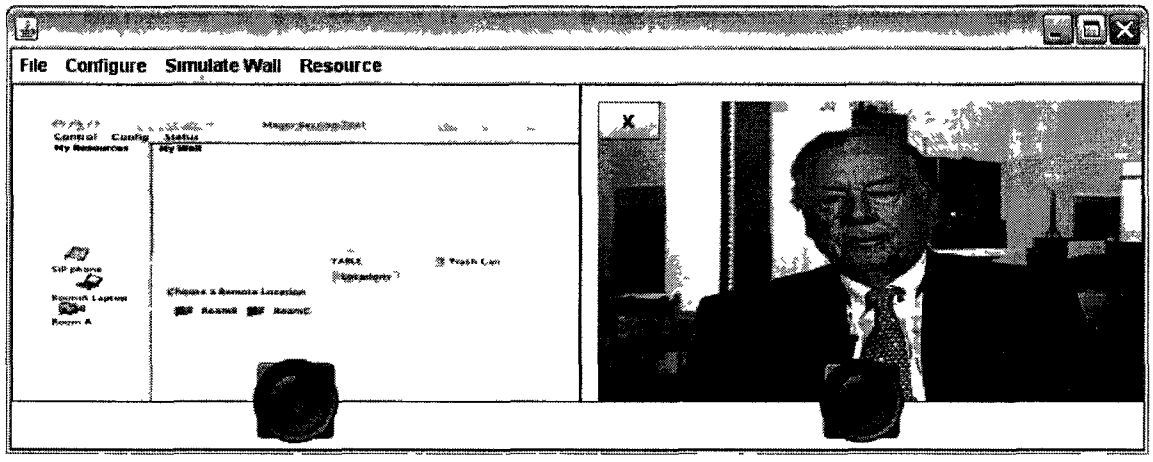

Figure 5.4 Layout with two resources on 2-panel setup

Figure 5.5 shows the layout of three resources, the new resource occupies one screen and the two older resources are allocated on the other screen.

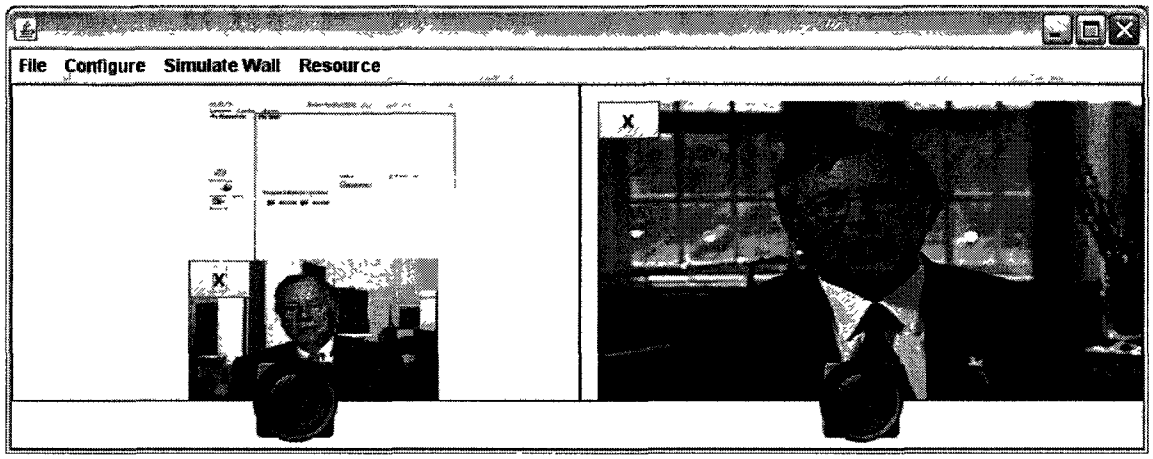

Figure 5.5 Layout with three resources on 2-panel setup

Figure 5.6 shows the layout of more than five resources. Similar to Figure 5.5, the new resource occupies one screen and the four other resources occupy the other screen, the remaining resources are not displayed because they are resized to a point that their size becomes less than the minimum acceptable size.

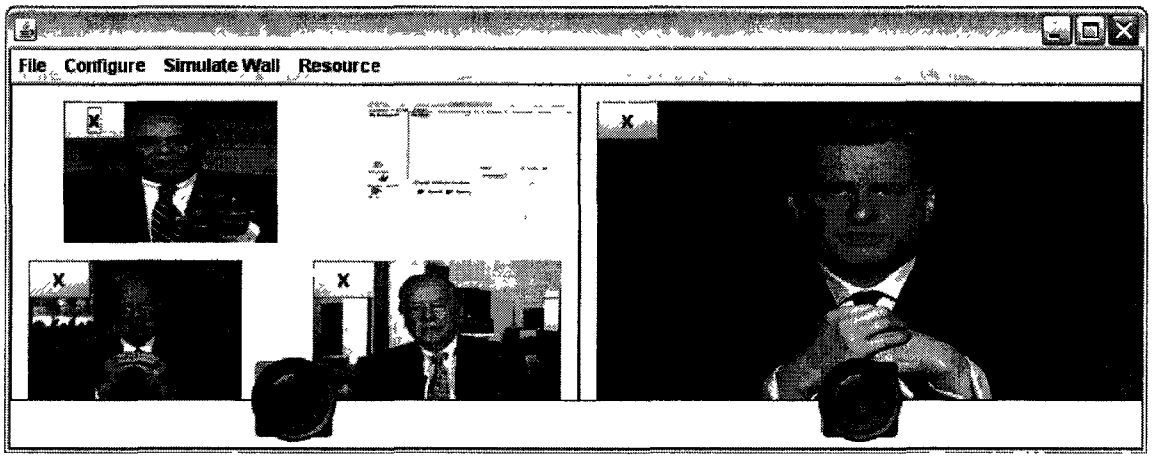

Figure 5.6 Layout with more than five resources on 2-panel setup

5.2 2.2 Test Case 5

In test case 5 the "Matrix" display setup, made of 2*2 screens, is used. Figure 5.7 shows the layout of seven resources, the newest resource is always the one that gets the largest space on the display.

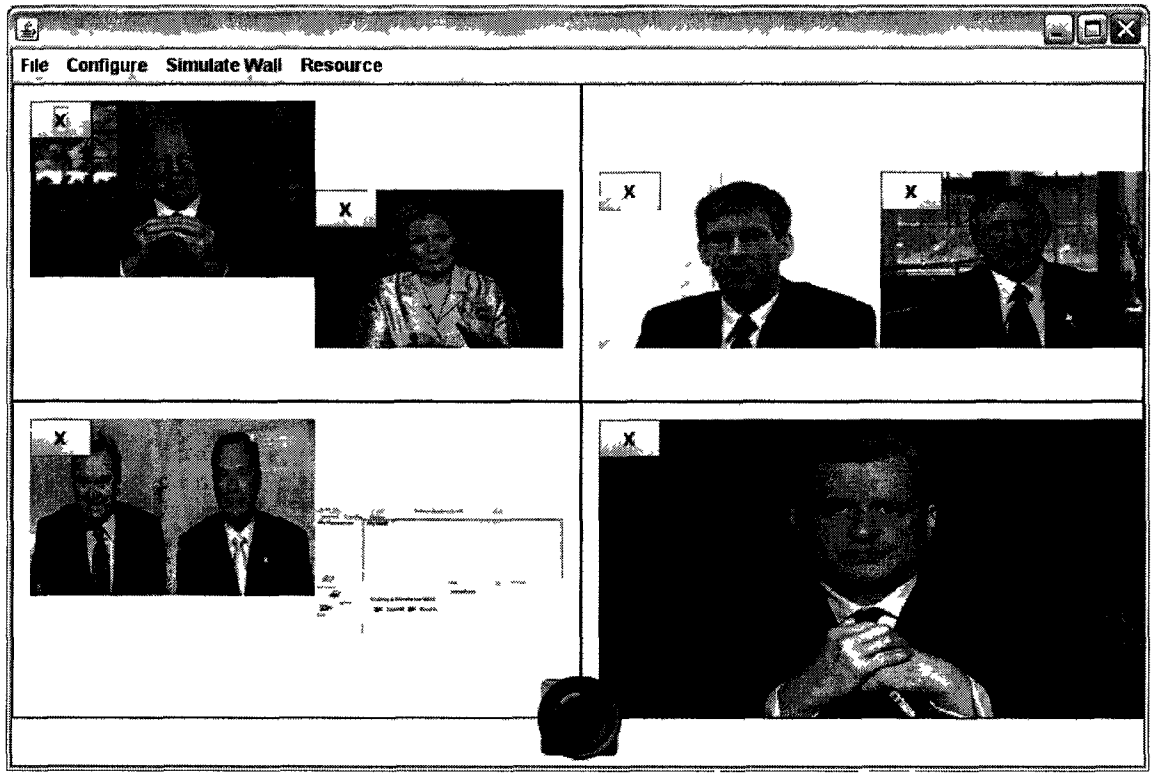

Figure 5.7 Layout with seven resources on Matrix setup

Figure 5.8 shows the layout of eleven resources. The new resource is allocated on one separate screen; three resources are allocated on one screen; three other resources on another screen; and the remaining four screens are allocated on the fourth screen.

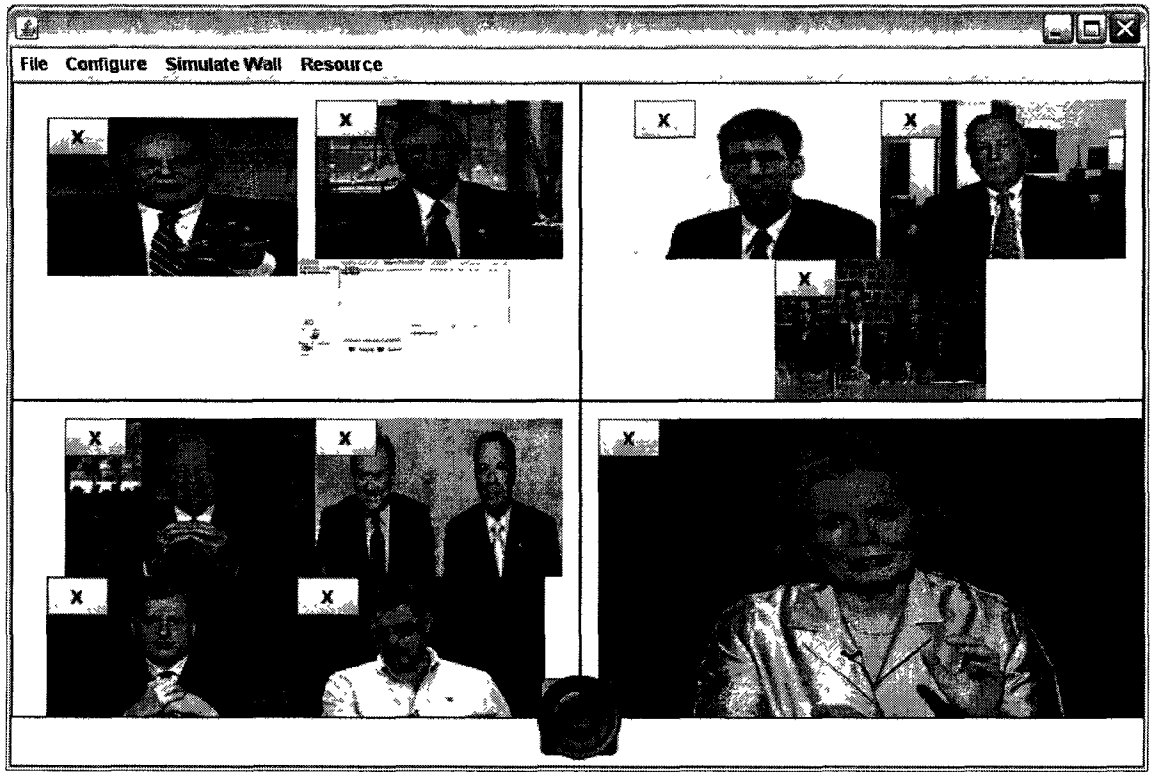

Figure 5.8 Layout with eleven resources on Matrix setup

5.2.2.3 Test Case 6

In test case 6 a customized display setup, made of 3*2 screens, is used. Figure 5.9 shows the layout result of eleven resources. The new resource is allocated on one screen, and on the remaining five screens a pair of resources is allocated on each.

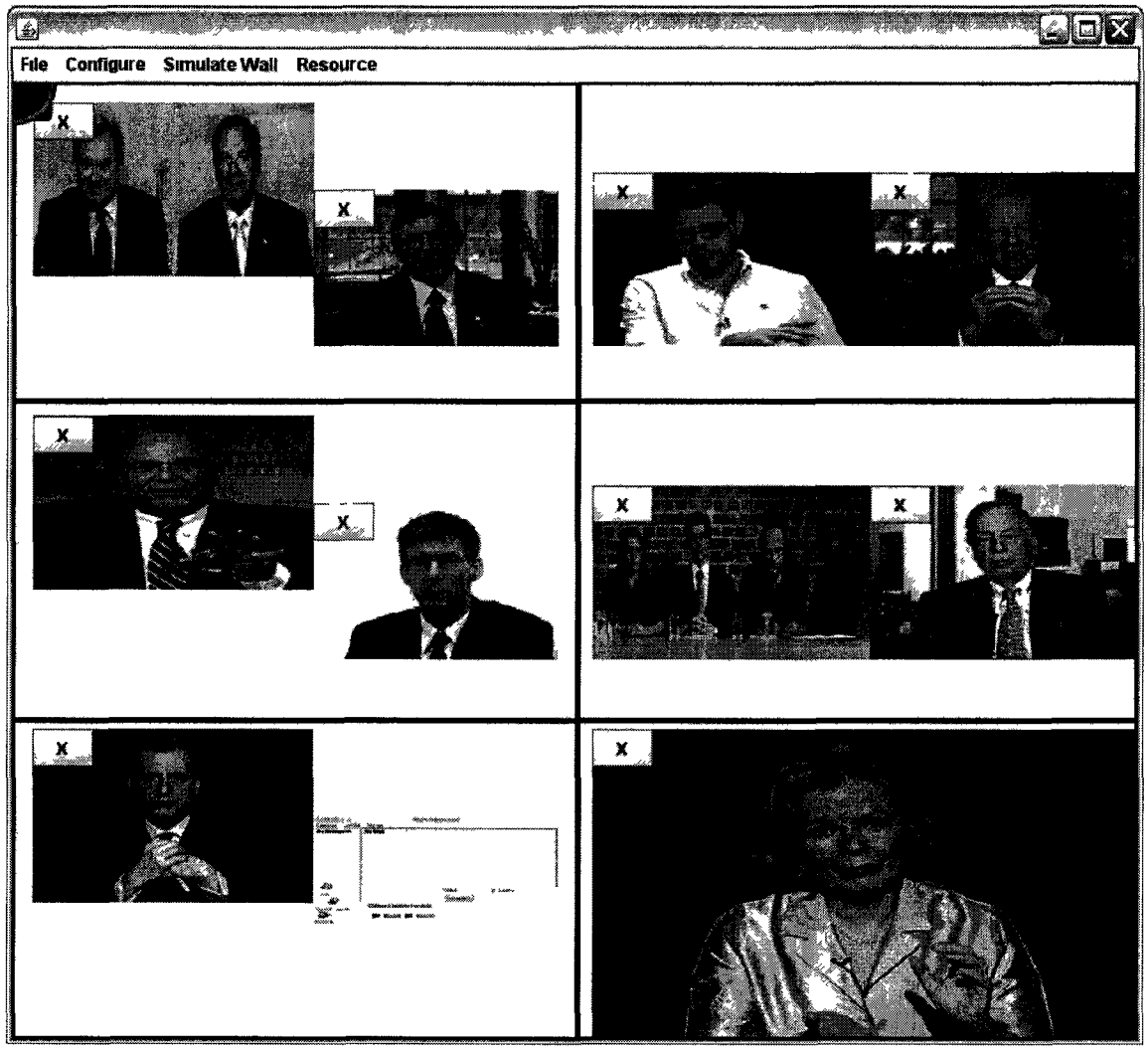

Figure 5.9 Layout with eleven resources on 3*2 screens setup

5.2.3 Performance Evaluation

The only quantitative measurement of the layout manager is the response time. This factor is measured in the layout viewer application using the "Matrix" display setup. Figure 5.10 shows the response time of the rule-based layout manager for a number of views between one and eleven views. The response time of the system as function of number of views is always less than three milliseconds. This high performance is due to

the limited processing in the rule-based approach where the rules are simply retrieved and applied on the existing views.

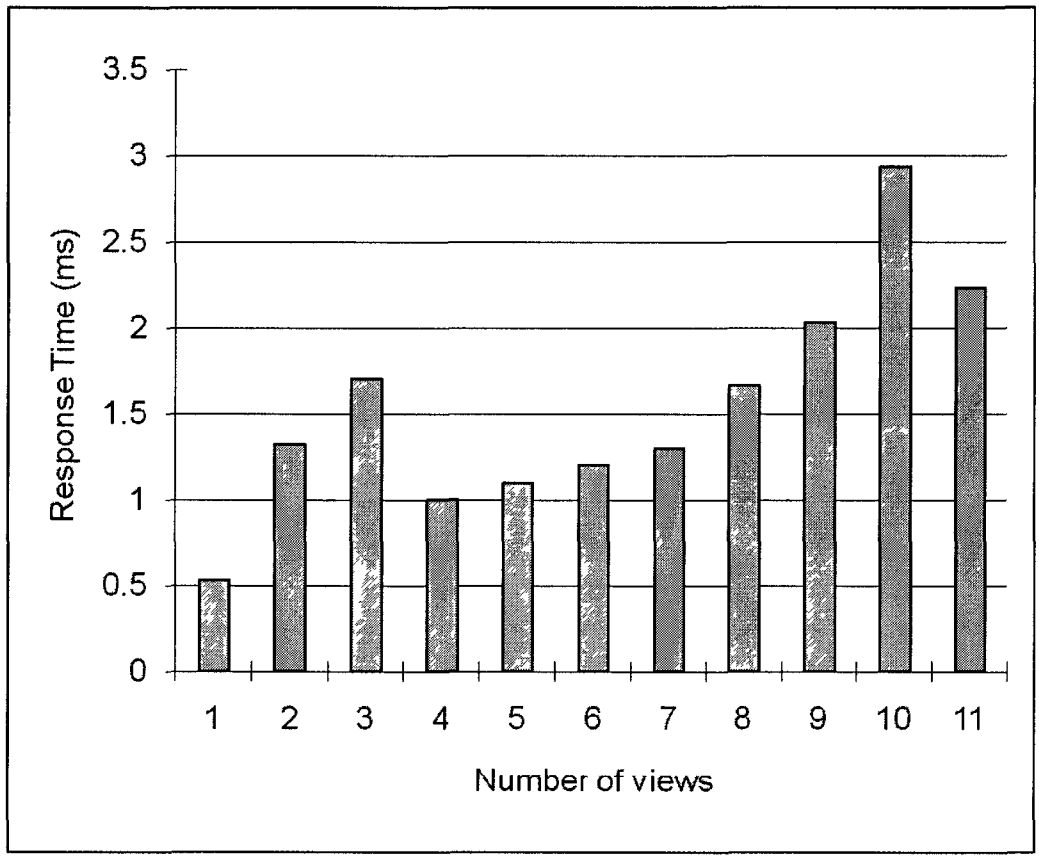

Figure 5.10 Response time for rule-based layout manager

Figure 5.11 shows the response time of the real estate layout manager for the same number of views in Figure 5.10. The response time for the first four views is less than 100 ms; afterwards it increases significantly. This is because the "Matrix" display setup can layout four views, each on one screen. After adding the fifth view, competitions occur between the views and some of them are resized which cause the system to perform more processing and eventually experience higher response times.

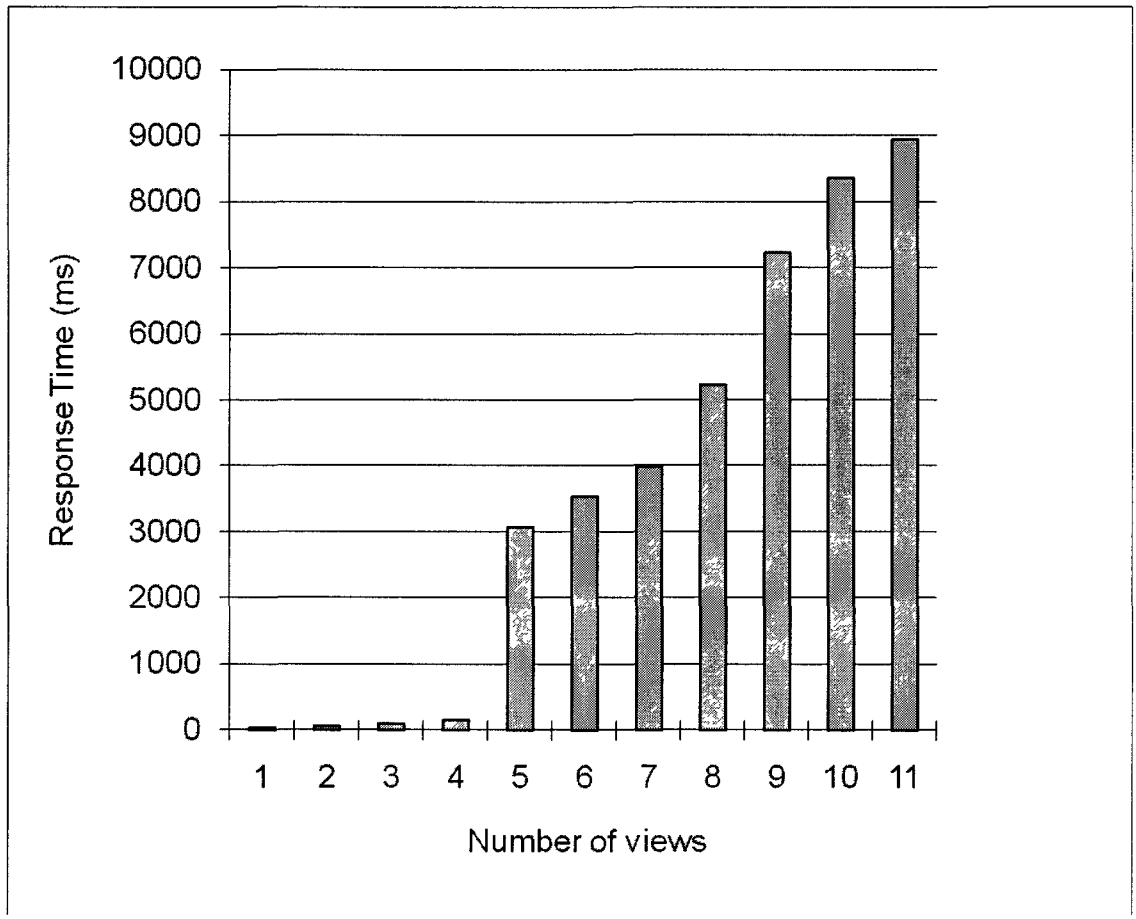

Figure 5.11 Response time for real estate layout manager

5.2.4 Usability Test

We have performed a usability test to qualitatively evaluate our proposed system. The usability test consists of twelve volunteers of age group between 22 and 35, and of experience with videoconferencing systems. The users were requested to perform the six test cases listed in section 5.1 and to fill up a questionnaire to evaluate the proposed system using Likert scale [6] in which the answers are in the range of 0-4 (the higher the rating, the greater the satisfaction).

Figure 5.12 shows the average values and the standard deviation for responses of users on four questions on both the rule based and real estate layout managers. Following are the questions 1 to 4:

- Question 1: The sizes of views are visually pleasant.
- Question 2: The positions of views are visually pleasant.
- Question 3: As the number of conferees increases, the layout of the videoconference becomes unacceptable.
- Question 4: The response time is acceptable.

Figure 5.12 shows that the users are more satisfied with real estate layout manager than the rule-based layout manager as per the size and position of the views on the display, which means that the real estate layout manager makes better utilization of the display area and the layout results are visually pleasant to the viewer. The standard deviation of responses about the sizes of views in rule-based layout manager is around 1.3; this means users have different opinions about this question, because some of them consider the situation when there are many views and the sizes are very small to be seen, and other users do not consider this factor.

The results of question 3 showed that users are much more satisfied with the real estate layout manager as the number of conferees increases; the average satisfaction of the latter is 3.5, whereas this value is 0.8 for the rule-based layout manager. The reason of this satisfaction is that only the important views are displayed in a size greater than the

minimum acceptable size in the real estate layout manager, whereas in the rule-based layout manager all views are displayed on a horizontal line in the middle of the screen regardless of their sizes.

The results of question 4 show that users' satisfaction of the response time of the rule based layout manager is 2.8, and it is 2 for the real estate layout manager, which is the middle of their satisfaction in Likert scale. This means that even though the response time of the layout manager is in the order of few seconds as shown is Figure 5.11, but this is still acceptable and usable for the users.

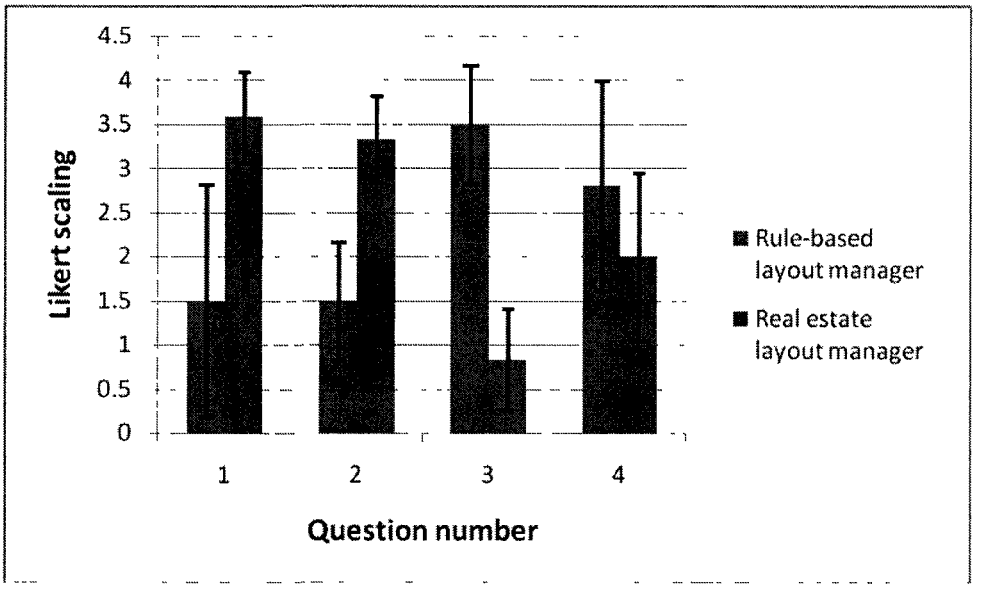

Figure 5.12 Average values and standard deviations for responses of users on questions comparing rule-based and real estate layout managers

Figure 5.13 shows the user response percentage of the following three questions:

- Question 1: The newest view has always occupied the largest space on the screen compared to other views.
- Question 2: Real estate layout manger has clear advantages over rule-based layout manager.
- Question 3: The consistency issue has distracted me in the videoconferencing session.

The answers of question 1 show that eight out of twelve users strongly agree that the newest view has always occupied the largest space on the screen when the real estate layout manager is used. This means that the implementation of algorithm performed according to its design that has specified that the newest view gets the "first time buyer bonus" and thus allocated a larger area on the display.

The answers of question 2 show that all users believe that the real estate layout manager has clear advantages over rule-based layout manager, simply because the position and size of the former are more satisfying than the latter, and because the former can adapt to any display setup which is not the case in the latter.

Question 3 is included in the questionnaire to evaluate the effect of one of the limitations of the system, namely inconsistency in the layout of views, in which the positions of views change between screens after adding or removing one view. Six of the twelve users agree that the inconsistency distracted them during the test, one strongly agrees, three did not agree, and two are neutral. Thus for most of the users, inconsistency is an

important issue that should be considered in the real estate layout manager despite their overall satisfaction.

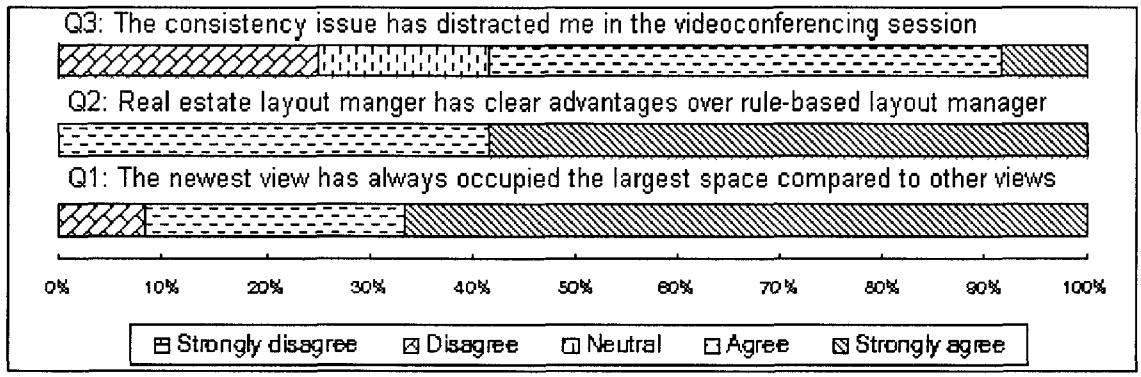

Figure 5.13 Users response percentage on the evaluation of the real estate layout manager

5.2.5 Results Discussion

After examining the allocation results of the real estate layout manager we can easily notice that it takes advantage of the total space of the display as demonstrated by the results of test cases 4, 5 and 6, and does not leave any screen empty as in the results of test case 1 of the rule-based manager. Also it assures that the size of a resource is not less than the minimum acceptable size for a viewer, in contrast with the results of the rule-based manager that reduces the sizes of resources when a new resource is introduced regardless of whether they are visible by the viewer or not, as shown in the results of test cases 1 and 2. These advantages have been confirmed by the results of the usability test where the satisfaction of users scores higher values for question about positions and sizes of views of the real estate layout manager compared to their satisfaction with the rule-based layout manager.

The most important advantage of the real estate layout manager is that it adapts with any display configuration, which is not the case in the rule-based layout manager. Comparing test cases 3 and 6 where a customized display is used, the real estate layout manager was able to allocate eleven resources on positions that look visually organized and symmetric, and their sizes is acceptable with respect to the screens sizes. Whereas the rule-based layout manage was allocating the resources in a pseudo-random fashion, and many resources are completely or partially covered by other resources, and half of one of the screens is empty. Thus we can claim that our manager is dynamic in the sense it can adapt to any display setup and generally produces better layout results than the rule-based layout manager. This causes all users in the usability test to confirm that the former has clear advantages over the latter.

Finally concerning the two limitations of the system, high response time and inconsistency, the usability test shows that the users are slightly affected by the high response time, but it does not diminish their satisfaction with the proposed system as long as it produces desirable results within reasonable response time. On the other hand, they notice the inconsistency in the positions of the views, which is not desirable for them when they focus on one of the views and it moves from one screen to another. This issue can be addressed in the future by allowing a view to change its position in the same screen and not among screens.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we presented a novel approach for managing the layout of views of sites in videoconferences using the real estate metaphor. This approach is considered novel because it manages the layout of views on any display setup dynamically without needing the user to specify any layout rules; his/her task is limited to providing the display setup geometry to the system.

Moreover, the context of the videoconference and the content of remote views affect their layout on the display. The proposed approach is able to evaluate the views according to a list of features specified by the user; therefore the system is context-aware.

The important improvement is that the user is freed from the burden of controlling the videoconferencing technology which makes the interface between them more transparent.

There are some limitations in the proposed system. The response time of the system is higher than its competitors, specifically the rule based layout manager; this is because there are more memory usage and computation involved. Using high performance computers can make the response time of the system more acceptable. Another limitation is the consistency issue; during the videoconferencing session participants join and leave causing a change in the layout. Viewers can be distracted when the positions of views change. We want to address this issue in the future releases of the system.

6.2 Future Work

The most important work need to be done in the future is to make the system able to measure the context parameters and automatically compute the credits for views. Some techniques are suggested in section 3.3. Also more context parameters can be investigated and added to the system to meet users' expectations.

One possible extension of the system is to give the viewer the freedom to make/change the layout manually. This work is part of the layout viewer not the layout manager, but adding this feature to the system increases the user satisfaction and allow him/her to control the layout in case he/she has a unique preference or in case of a system crash.

Bibliography

[1] W. Zhao, R. Chellappa, P.J. Phillips, "Face Recognition: A Literature Survey," *ACM Computing Surveys 35(4), 399-458,* 2003.

[2] Magor Corportation web, site http://www.magorcorp.com, accessed May 9, 2009.

[3] Y. Rui, A. Gupta, J. Gurdin, "Videography for telepresentations," *ACM conference on Computer Human Interface*, 2003.

[4] C. Zhang, J. Crawford, Y. Rui, L. He, "An Automated End-to-End Lecture Capturing and Broadcasting System", *Proceedings of ACM Multimedia 2005, Singapore, 2005, 808-809.*

[5] R. Heck, M. Wallick, M. Gleicher, "Virtual Videography," *ACM Transaction on Multimedia Computing, Communications and Applications (TOMCAAP), Volume 3, Issue 1, February 2007.*

[6] Likert scale, http://en.wikipedia.org/wiki/Likert_scale, accessed May 9, 2009.

[7] M. Wallick, Y. Rui, L. He, "A portable solution for automatic lecture room camera management", *In Proceedings of the IEEE International Conference on Multimedia and Expo 2004.*

[8] L. Barkhuus, and A. Dey, "Is Context-Aware Computing Taking Control away from the User? Three Levels of Interactivity Examined." *In Proceedings of the Ubi-Comp2003 conference (2003), LNCS 2864, pp. 149 - 156*

[9] W.C. Chen et al.: "Toward a Compelling Sensation of Telepresence: Demonstrating a portal to a distant (static) office", *Proc. of IEEE Visualization 2000, Salt Lake City,*

UT, USA, Oct. 2000.

[10] T. Aoki et al, "MONJUnoCHIE System: Videoconference System with Eye Contact for Decision Making," *International Workshop on Advanced Image Technology (IWAIT), 1999.*

[11] S.J. Gibbs et al, "TELEPORT-Towards Immersive Copresence", *Multimedia Systems, No.7, pp.214-221,Springer-Verlag, 1999.*

[12] Hewett, Baecker, Card, Carey, Gasen, Mantei, Perlman, Strong and Verplank, *"ACM SIGCHI Curricula for Human-Computer Interaction".*

[13] K. Vredenburg, J.-Y. Mao, P. W. Smith, and T. Carey, "A Survey of User-Centered Design Practice". *In L. Terveen, D. Wixon, E. Comstock, & A. Sasse (Eds.), Proceedings of ACM Conference on Human Factors in Computing Systems (2002).* (pp. 472-478). New York NY: ACM Press

[14] N. Danino, "Human-Computer Interaction and Your Site", November 14th 2001, http://www.sitepoint.com/article/computer-interaction-site, accessed May 9, 2009.

[15] B. Schilit, M.Theimer, "Disseminating active map information to mobile hosts." *IEEE Network* 1994; 8: 22–32

[16] B. Schilit, N. Adams, and R. Want. "Context-aware computing applications." *In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pages 85-90, Santa Cruz, California, December 1994. IEEE Computer Society Press.*

[17] A. Dey and G. Abowd. "Towards a Better Understanding of context and context-awareness." *Technical Report GIT-GVU-99-22, Georgia Institute of Technology, College of Computing, June 1999*

[18] G. Chen and D. Kotz. "A survey of context-aware mobile computing research." *Technical Report TR2000-381, Dartmouth College, Computer Science, Hanover, NH,* November 2000.

[19] J. Pascoe. "Adding generic contextual capabilities to wearable computers." *In Proceedings of the Second International Symposium on Wearable Computers, Pittsburgh, Pennsylvania, October 1998.* IEEE Computer Society Press

[20] S. Lok and S. Feiner. "A Survey of Automated Layout Techniques for Information Presentations". *In Proceedings of SmartGraphics 2001, March 2001*

[21] A. Kroener, "The DesignComposer: Context-Based Automated Layout for the Internet." *Proceedings of the AAAI Fall Symposium Series: Using Layout for the Generation, Understanding, or Retrieval of Documents, 1999.*

[22] Y. Rui, A. Gupta, J. Grudin and L.W. He, "Automating lecture capture and broadcast: technology and videography", *in ACM Multimedia Systems Journal (Springer), 10:3-15 2004*

[23] B. Yu, C. Zhang, Y. Rui, K. Nahrstedt, "A Three-Layer Virtual Director Model for Supporting Automated Multi-Site Distributed Education," *ICME, IEEE International Conference on Multimedia and Expo, 2006,* pp. 637-640, 2006

[24] B. Yu and K. Nahrstedt, "AVPUC: Automatic Video Production with User Customization," *in Twelfth Annual Multimedia Computing and Networking Conference (MMCN '05)*

[25] US Patent 7266189 Issued on September 4, 2007 – "Who said that? teleconference speaker identification apparatus and method".

[26] D. Ingalls, "Design Principles Behind Smalltalk.", *BYTE Magazine, August 1981.*

[27] K. Bonhringer, F. Paulisch, "Using Constarints to Achieve Stability in Automatic Graph Layout Algorithms", *Proc. of ACM SIGCHI Conf. on Human Factors in Computing Systems, Seattle, WA, April 1990.*

[28] W. Graf, "The Constraint-Based Layout Framework LayLab and Its Applications", *Electronic Proceedings of the ACM Workshop on Effective Abstractions in Multimedia, November 4, 1995 San Francisco, California.*

[29] J. Vanderdonckt, X. Gillo, "Visual Techniques for Traditional and Multimedia Layouts", *Proceedings of the workshop on Advanced visual interfaces, p.95-104, June 01-04, 1994, Bari, Italy.*

[30] L. Weitzman and K. Wittenburg. "Automatic presentation of multimedia documents using relational grammars", *In Proceedings of the Second ACM International Conference on Multimedia (MULTIMEDIA '94)*, pages 443–452, New York, Oct. 1994. ACM Press.

[31] E. André, W. Finkler, W. Graf, T. Rist, A. Schauder and W. Wahlster, "WIP: the automatic synthesis of multimodal presentations", M. Maybury, ed., *Intelligent Multimedia Interfaces (AAAI Press, Cambridge, MA, 1993).*

[32] W. Graf, "Constraint-Based Graphical Layout of Miltimodal Presentations", *In T. Catarci, M. F. Costabile, and S. Levialdi, editors, Advanced Visual Interfaces, AVI, volume 36 of Series in Computer Science*, pages 365–385. World Scientific, 27–29 May 1992.

[33] B. V. Zanden and B. A. Myers. "Automatic, look-and-feel independent dialog creation for graphical user interfaces", *In Proceedings of ACM CHI'90 Conference on Human Factors in Computing Systems,* Constraint Based UI Tools, pages 27–34, 1990.

[34] A. Borning and R. Duisberg. "Constraint-based tools for building user interfaces" *ACM Transactions on Graphics*, 5(4):345–374, Oct. 1986.

[35] S. Hudson and S. Mohamed. "Interactive specification of flexible user interface displays", *ACM Transactions on Information Systems*, 8(3):269–288, July 1990.

[36] S. E. Hudson and I. Smith. "Ultra-lightweight constraints", *In Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 147–155, 1996.

[37] S. Kochhar, J. Marks, and M. Friedell. "Interaction paradigms for human-computer cooperation in graphical-object modeling", *In Proceedings of Graphics Interface '91*, pages 180–191, June 1991.

[38] B. Myers et al. "The garnet user interface development environment", *In Proceedings of the CHI '94 conference companion on Human factors in computing systems*, pages 457–458, 1994.

[39] B. Myers, R. G. McDaniel, and D. S. Kosbie. "Marquise: Creating complete user interfaces by demonstration", *In INTERCHI '93, Human Factors in Computing Systems, Apr. 1993.*

[40] M. Zhou and S. Ma. "Toward applying machine learning for to design rule acquisition for automated graphics generation" *Technical report, IBM Watson Research Center, 1999.*

[41] http://soundbites.typepad.com/photos/uncategorized/2007/04/24/hollywood_squares.jpg, accessed September 7th, 2009.

[42] Q. Liu, Y. Rui, A. Gupta and J. Cadiz, "Automating Camera Management for Lecture Room Environments", *Technical Report MSR-TR-2000-90, September 21, 2000.*