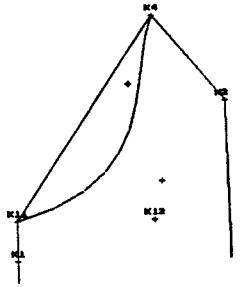

Figure 1: Forming a general arc

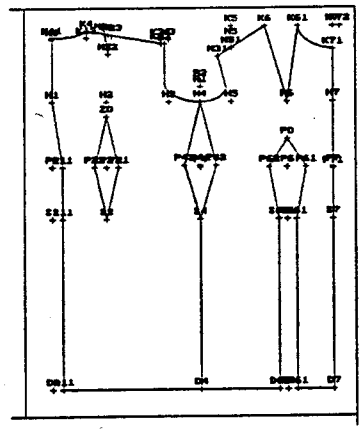

Figure 2: Final form of the style

X Window System

Aleš Limpouch

Department of Computers
Faculty of Electrical Engineering, Czech Technical University
Karlovo nám. 13, 121 35 Prague 2
Czech Republic
e-mail: limpouch@cs.felk.cvut.cz

Abstract

The X Window System has, over the last few years, become increasingly important and it is now accepted as *the* window system for workstations, mainframes and supercomputers. It has become the standard means of providing graphical facilities on UNIX systems. This paper gives a brief overview of fundamental principles, concepts and architecture of the X system and describes in detail essential characteristics of the X server.

Keywords

graphical user interface, window management system, X Window System, X server

1 Fundamental principles

As a visual (graphical) form of information is more natural, illustrative and understandable, intensive change to the graphical way of communication between human and computer has taken place within the past few years. The term **graphical user interface** (GUI) is used for environments, which enable this way of interaction. In addition, it makes it possible to take advantage of multitasking operating systems, such as UNIX, more effectively. The X Window System (X) represents *the* window management system, which has become standard for UNIX workstations.

1.1 X Window System

During the 1980's, the X Window System became the standard means for the development of graphical user interfaces and interactive applications with visual interfaces in UNIX. X began at MIT and its authors had defined ambitious aims, which led to advanced features and model of the system. The important reason for its acceptance was the decision to make the source code of X available free of charge. However, the X system is not public domain and is protected by copyright.

The X Window System does not represent complete GUI. It is a **window management system**, which provides only graphical facilities and window manipulation functions. The main aim during the design of the system was to give the means for the development of graphical user interfaces and applications without restricting people to a particular look and feel. X does not define any particular attribute of the user interface. Authors wanted to provide "mechanism, not policy". High flexibility and various mechanisms for future extensions of the system make it possible to adjust it for user needs, special devices and new hardware.

One fundamental characteristic of the system is its device independence. The X system enables applications to display text or graphics within overlapping windows on any device supporting X protocol. X's network and operating system transparency allow high portability of applications. Sun's NeWS *(Network Extensible Window System)* is the only competitive window management system in the UNIX environment.

1.2 History

The development of X began in 1984 at the Massachusetts Institute of Technology within the Athena and Argus projects. Version 10 was publicly released in late 1985 and was ported during a few months to various platforms. Preparation of the new version was started in May, 1986, taking into account the experience with the previous one. The first technical conference on X was held at MIT in January, 1987. Eleven main computer manufacturers declared full support for the new version during this conference. The X Window System Version 11 Release 1 (X11R1) was then released on September 15, 1987.

The MIT X Consortium was founded in January, 1988 to foster development of X and to control the X standards. The consortium associates more than 100 organizations (e.g. DEC, HP, Sun Microsystems), which are interested in the future development and use of X. The current release is the X Window System 11 Release 5 (X11R5), which has been available since September 5, 1991. This release has introduced support for scalable fonts, font servers, a new device-independent color management system and support for internationalization based on the ANSI standards. A sample PEX *(PHIGS extension to X)* for 3-D graphics was first included in this release. The PEX application interface is fully compatible with the PHIGS and PHIGS PLUS standards. A new release of the X system is expected during the first half of 1994.

Two predominant graphical user interfaces (OPEN LOOK and OSF/Motif) in UNIX are based on X. Motif was developed by the Open Software Foundation (OSF), which is a consortium of manufacturers including DEC, HP and IBM. The look and feel of Motif is similar to that of Microsoft Windows. OPEN LOOK is the standard of the UNIX International (UI) consortium for GUI, which was developed primarily by AT&T and Sun Microsystems. Sun's OpenWindows is user and development environment compliant with the OPEN LOOK standard. This environment takes the X Window and NeWS systems as equivalents.

1.3 System architecture

The architecture of the X system is based on the **client–server** model. **X client** is an application program, which sends requests for window manipulations and drawing texts or graphics. X server handles these requests and controls the

display (screen, mouse and keyboard). It transmits user actions to clients in the form of events. The X client and X server are two independent processes, which communicate with each other by the **X protocol**, which is the main standard of the MIT X Consortium. Simplified structure of the client–server model is illustrated in Figure 1.

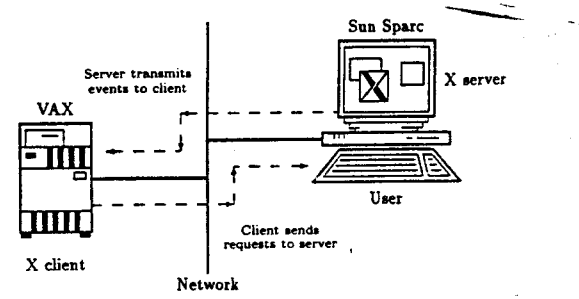

Figure 1: Client–server model

The client and server may run on different machines and communicate through a network. The X system is network transparent and operates over various network protocols including TCP/IP and DECnet. A server can handle more clients and clients can use services of several servers. A client running on one machine can display its results at the same time on any computer for which an X server is available. This fact greatly increases portability of applications. The X server actually provides implementation of the X Window System on a particular device and appears to be a standard graphics terminal for applications. It is the only part of X which is device-dependent. This concept ensures application independence on graphics hardware.

1.4 Application structure

One main part of X is the X client, which sends graphics requests to X server and is responsible for handling user actions as well. It serves a large number of functions, which are not included in the X server – provides look and feel, defines user interface objects, interprets user input and redraws its windows. A special client called a **window manager** gives the user means for window manipulations such as resizing or iconifying and solves problems related to window size or position.

The X system provides several layers for application construction. The basic interface of X represents the X protocol by means of which applications communicate with X server. It is defined in terms of high-level packets (requests and events) and is not practically used for programming. Applications are created by the use of libraries with various levels of abstraction. Some of them are standardized within X; others are part of the MIT X distribution or development environment for particular GUIs.

Basic X client functions are provided by the **Xlib** library. It consist of a large set of C functions for display, window, input, event and resource manipulations. These functions allow direct access to the X protocol requests and events and full control of the system. Application development based on this library is very complex

and tedious. For this reason it is used mainly for the implementation of high-level libraries and is avoided during application development as much as possible. Similar interfaces exist in other programming languages such as LISP, ADA or Fortran.

Programmers prefer **toolkits** for the development of X clients. There are several toolkits for X, available in different languages. They make application implementation, in general, easier and hide some details from programmers. The standard toolkit of the X Window System is the **X Toolkit**. The X Toolkit consists of two parts: the Xt Intrinsics framework and a set of user interface objects called **widgets** in X. The **Xt Intrinsics** is a construction kit for building widgets and provides fundamental mechanisms for the creation and use of widget sets. Its architecture is based on object-oriented principles and techniques, which are very suitable for user interface design. The Xt Intrinsics forms the object-oriented kernel of the X Toolkit in C language.

While the Xt Intrinsics gives only tools for user interface design, the look and feel of applications is defined by the particular widget set, which was used for their creation. The MIT X distribution provides the **Athena Widget Set** as a sample implementation of a widget set based on Xt Intrinsics. It was designed during the development of MIT clients and utilities. The library is written in C language and contains a sort of widgets (buttons, scroll bar, menu) sufficient for a large scale of interactive applications. Figure 2 shows the typical structure of application based on the X Toolkit and Athena Widget Set.

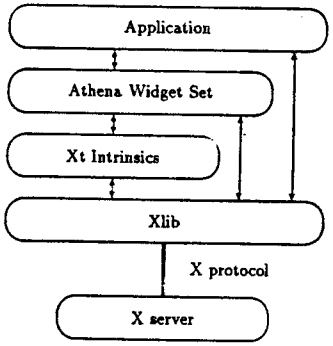

Figure 2: General structure of X clients

While the Xt Intrinsics is the MIT X Consortium standard, there is no standard widget set at present. In addition to Athena, there are a lot of various sets, toolkits and frameworks. The OPEN LOOK Intrinsics Toolkit (OLIT) is an Intrinsics-based toolkit for the design of applications compliant with the OPEN LOOK standard. This toolkit is a standard part of Sun's OpenWindows. The Motif Toolkit is another Intrinsics-based toolkit, which is provided by the OSF for the development of applications within OSF/Motif. There are also a lot of toolkits in C++. InterViews from Stanford University is very popular and widely used. ET++ framework from ETH Zürich constitutes a complete application framework for the development of interactive applications. Clear design, high compactness and portability makes this library very remarkable.

2 X server

The X server plays the key role in X and actually represents its implementation for a particular graphics device, workstation or operating system. In this section, we describe its essential characteristics and in more detail some features important from the point of view of computer graphics.

2.1 Server concepts

The main role of the X server is handling connections to multiple simultaneous remote and local applications. Server-client communication is based on the X protocol and is transmitted asynchronously. The communications link can be available in any form of inter-process communication (shared memory, pipe) or network protocol. Clients send **requests** to the server, where they are stored in queue. Although the server processes requests from each particular application in the order in which they arrive, they are not necessarily processed immediately. Clients do not usually wait for processing, but they can explicitly request synchronous handling, which usually results in considerable slowing down of the system.

Graphics output processing is one of the main functions of the X server. It handles clients' requests for window manipulations and drawing texts or graphics. At the same time, the X server maintains all **resources** used by X, including windows, pixmaps, cursors, colormaps, fonts and graphics contexts. These resources are created and controlled by clients, but they are stored and maintained by the X server. This feature decreases the amount of transmitted data between client and server considerably. Clients can identify resources by unique **resource identifiers**, which are assigned by the X server. Resources can be shared among different applications.

The X server informs applications about user activities in the form of **events**, which are generated either as direct or indirect results of user actions (key press, mouse move, mouse button press). The X server also informs clients about changes of window state, e.g. sends a client an `Expose` event when a window's contents need to be restored. X supports a variety of input devices. Depending on implementation, a server can support tablets, trackballs or other data input and pointing devices. However, the most common input devices are the keyboard, used for textual input, and the mouse, which serves as both a pointing and selection device.

The MIT X Consortium produces sample implementations of the X system for a wide range of UNIX systems. Hardware manufacturers and third-party vendors have developed servers for other workstations and operating systems. An X server is usually software, but some graphics device manufacturers provides X-terminals, where hardware and firmware provide an implementation of the server. X-terminals represent, like ordinary terminals, input and output devices for X clients, which are running remotely on central hosts. Some vendors have ported the X server in a similar way to personal computers such as DOS PC or Atari, which are not suitable for running X clients.

In X, many functions, which in other systems would usually be handled by the base window system, are not performed by the X server but are left to the client. The server does not automatically redraw the contents of windows. There are no special facilities for resizing pictures in a window. The server does not provide any user interface components such as menus, buttons, or scroll bars and does not interpret mouse or keyboard input. The server does not give users any means for window manipulations such as resizing or iconifying and does not solve problems of window

size or position. These tasks are left to a special client – a window manager. These features fully correspond with authors' wishes to avoid any particular definition of look and feel of the system and must be handled by clients.

2.2 Window management

The most important resource of X is the window. The **X window** represents a bordered rectangular area of the screen filled with a background color or pattern. In contrast to other systems, the X window has no title bar, scroll bar or other decoration. The X server creates windows in response to requests of clients. The server stores and maintains them and clients refer to the window using its identifier. Each client, which knows the identifier, can manipulate the window. This feature is used by window managers to control the position of windows on the screen.

X organizes windows on the screen hierarchically into the **window tree**. The **root window** is created automatically by the X server and occupies an entire screen. Every window except the root window has a **parent** window *(ancestor)* and can also have **child** windows *(subwindows)*. As child windows may also have subwindows, an arbitrarily deep window tree structure can be created. Figure 3 illustrates this hierarchical model and these relationships. Only the area of a child window, which lies within the bounds of its parent, is visible. X automatically maintains the clipping of subwindows.

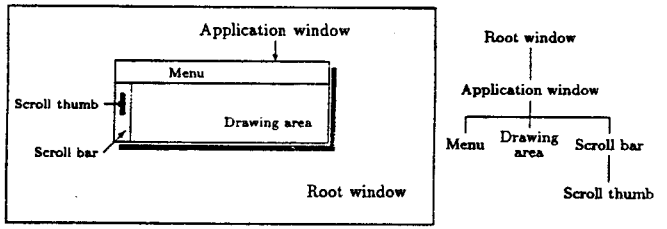

Figure 3: Window tree hierarchy

X supports overlapping windows in a way that resembles papers on a desk. Windows may occupy the same region on the screen and one window can completely or partially obscure another one. The X system maintains automatically correct overlapping of windows based on the stack metaphor. Although each X window is associated with a rectangular area on the screen, all windows are not necessarily visible. A window is not displayed automatically after its creation. A client must issue a **map** request to display that window. But it still might not be visible for various reasons. It might be clipped by its parent or completely obscured by another window. Another reason might be that any ancestor of the window is not mapped. A window that is mapped, but has an ancestor that is not mapped, is called **unviewable**. Such a window automatically becomes **viewable** when all ancestors are mapped.

Every window system with overlapping windows must in some way solve the problem of preserving window's contents when that window is covered by another window to be able to restore its contents later. At present, there are three common methods of solving this problem. Some systems preserve covered windows and all their changes are performed off-screen in memory, so they can be restored at any time.

This method is called **backing-store**. Some systems preserves areas, which are obscured by another window in the frozen state and any change of this area is lost. This method is called **save-unders**. Other systems do not preserve any window's contents and pass this responsibility to applications. X allows the combination of all three methods, but the first two are not guaranteed by the system and have only supplementary character. Its availability depends on the particular X server implementation and the required amount of memory. The primary principle of X is based on the last method. Every client must be able to redraw the contents of its windows on request at any time.

The previous paragraphs show fundamental principles and concepts of window management. It should be noted, that the X server is not in fact responsible for window management. For the reason of high flexibility, window management is handled by a client with special rights – a **window manager**. Entire output to the screen is limited to main application windows, which are direct children of the root window. Problems related to size, position, stacking order, overlapping and iconifying of main application windows are solved by the window manager, which also adds their title bars, controls and other decorations. The window manager also mediates inter-client communication and handles their inter-operation, co-existence and competition for system resources. The ICCCM *(Inter-Client Communication Conventions Manual)* standard regulates inter-client communication and co-existence. X provides standard functions and mechanisms, which make it possible to create window managers and to carry out their tasks. The MIT X distribution includes Tab Window Manager (twm). Two predominant GUIs have their own window managers – OPEN LOOK Window Manager (olwm) and Motif Window Manager (mwm).

2.3 Graphics

The X server enables applications to send requests for drawing graphics or text within windows. In relation to drawing graphics, there are problems of graphical attributes, color specification and text fonts. X supports two-dimensional graphics. A picture is represented by a matrix of **pixels**. Graphics operations can be performed within windows as well as **pixmaps**, which are stored off-screen in memory. The common term **drawable** is used in X for both windows and pixmaps.

Each drawable has its own integer coordinate system with the base point (0,0) in the upper left corner. The x coordinate increases toward the right and the y coordinate increases toward the bottom. All dimensions are expressed in pixels. The X server has no special facilities for resizing pictures in an X window. The third dimension of drawables is the number of bits representing pixel, which is called **depth**. Drawable can be divided in this dimension into **planes**, whose depth is 1. The X server provides requests for drawing graphics primitives and their sets, including point, line and polyline, rectangle, arc and filled rectangle, polygon and arc. There are also functions for clearing area and copying area between drawables available.

X represents pictures internally in the server as pixmaps. **Bitmap** (1-bit depth pixmap) can be considered as a special case used in X for the definition of filling patterns, icons and cursors. Bitmap is also used to specify a clipping region for graphics operations. This technique provides an easy way to display non-rectangular windows on the screen. **Image** is a structure defined in X for easier off-screen manipulation with raster data, which are independent of particular X server and portable across different platforms and graphics devices.

Each result of graphics operations is dependent on a large number of graphical attributes, which are provided in X. These attributes are stored within the X server in the **graphics context** resource. The X system supports many usual and some unusual attributes: foreground and background colors, line width and style, cap and join styles, dash lengths and offset, fill style, tile and stipple pixmaps and offset, polygon filling rule, arc filling mode and font. The clipping mask, plane mask and logical display function define exactly destination pixels of graphics operations.

2.4 Colors

Window management systems should solve problems of color display, matching and management. X supports a wide range of displays from simple monochrome to modern true color displays. The most commonly used displays are based on colormap architecture. These devices are usually able to display at one time fewer colors than can be chosen from the palette. **Colormap** is a color lookup table, which stores actual colors, and pixel value in some way represents an index to this table. The X system uses color representation based on colormaps.

Considering the big differences of graphics devices, X divides them according to visual type into 6 **visual classes** *(visuals)*. A list of supported visuals with detailed descriptions is then available for each device. Server representation of colors is internally based on a triplet of 16-bit red, green and blue (RGB) values. The server also maintains the database of color names and their RGB representation. This database enables clients to indicate colors by names such as "green", "orange" or "pink". Colors can also be specified by their RGB values and the server automatically chooses a matching color, which the device is able to display.

A much bigger problem seems to be the allocation of colormap to applications and sharing colormap among them. X provides multiple software colormaps and each window can have its own colormap associated with it. One of these is then installed into hardware colormap. The window manager is responsible for the installation of active colormap and uses the colormap of the current window in compliance with convention. This means that other applications and windows may be displayed in incorrect colors, because their colors are changed. X allows the sharing of particular colors in colormaps across windows and applications and then helps to eliminate this problem in some cases. But at the same time the provided mechanism grants the right to allocate and change the whole colormap to applications.

The RGB scheme for specifying colors is closely dependent on a particular graphics device. Then other color spaces were developed, which are independent of graphics hardware. Release X11R5 has introduced **Xcms** *(X Color Management System)*, which provides the CIE XYZ, CIE xyY, CIE L*u*v, CIE L*a*b and TekHVC color spaces for the specification of colors. This extension includes functions for conversion between color spaces and also mechanisms for the definition and use of new individual color spaces. Specification of color and illumination characteristics or device calibration is required for the use of this extension. Xcms is not part of X server, which still uses the RGB scheme, but is provided as an extension of the standard Xlib library.

2.5 Texts and fonts

In the X system, there are several requests for drawing and handling texts and manipulating fonts. Besides drawing texts, font characteristics and text dimensions

can be queried. Texts are drawn horizontally and cannot be rotated. Multiple fonts and character sets can be used simultaneously. Characters in fonts can be encoded as one or two bytes. Up to release X11R4, raster fonts, which are stored as bitmaps on the server, were used exclusively. Release X11R5 has introduced support for scalable and outline fonts. A scaler for bitmap fonts has been included and the Bitstream Speedo technology for scalable outline fonts has been incorporated.

The **XLFD** *(X Logical Font Description)* standard defines the font name scheme and the method of font description. This standard provides conventions for uniform font names and characteristics, which are independent of a particular X server, operating and file system. The full XLFD font name also includes a character set indication. X presently supports ISO 8859 character encodings (Latin 1–5, Arabic, Cyrillic, Greek, Hebrew) as well as some oriental standards (JIS, KS, GB). **CTEXT** *(Compound Text)* is a format for multiple character set data, such as multi-lingual texts. It is intended as an external representation and can be used for inter-client communication and interchange among various environments and platforms.

The distribution of raster fonts across platforms is enabled by the standardized portable, plain-text, format called **BDF** *(Bitmap Distribution Format)*. X servers use the effective internal format **PCF** *(Portable Compiled Format)* for text handling in release X11R5, or **SNF** *(Server Natural Format)* in earlier releases. Despite these standards, administration of fonts is a complex problem. The installation of new fonts and special technologies is especially difficult in a large network with many different platforms. Also the server memory and disk space consumption is remarkable for the use of many fonts.

To overcome these problems, font servers were introduced in release X11R5. X font server is a special server, which sends the X server requested font information and character bitmaps. The X server, instead of reading font files directly, can ask the font server for particular information. Font servers enable central storage of font files and easy installation. The X server can be connected to more than one font server, which can be chained together. The introduction of new font technology or format could be easily accommodated through the installation of a new font server supplied by its font vendor.

2.6 Server extensions

Certain functionality was for various reasons not incorporated into the base X system and would have to be added later on. X has been built as an open system and provides various methods for the extension and addition of new functions. It makes it possible to adjust X in a clean way for novel hardware and special graphics devices. This feature can be used for solving special user needs or demands found during the use and administration of X.

One way to extend X is by means of server extension. The X system provides a well-defined standard mechanism for such an extension, which actually represents an X protocol extension and defines new primitive services supported by the X server. Therefore a special extension library usually makes it possible to invoke new functionality and to issue extra requests for new services. Queries for available extensions and their versions can be made. So applications may be written independently of server extensions. An X client, which needs a particular extension, unfortunately depends on a certain X server, but may still be device-independent.

SHAPE *(X11 Nonrectangular Window Shape Extension)* provides non-rectangular windows in X. It is the most popular extension, which has become the X standard.

MIT-SHM *(MIT Shared Memory Extension)* facilitates memory-sharing for certain data on one machine between the server and client. **Input extension** *(X11 Input Extension Protocol Specification* and *X11 Input Extension Library Specification)* is the standard mechanism for the definition and use of additional input devices. Support for 3-D graphics introduced in the current release X11R5 is implemented as an extension of the X server as well. The **X3D-PEX** *(PHIGS extension to X)* extension is defined in the draft X standard.

Some vendors provide their own extensions of the X server. The MIT X Consortium controls the registration and standardization of these extensions. Currently, there are more than 40 different extensions of the X server registered. Some of them are limited to one platform; others are widely available. A well-known frequent extension is support for the **Display PostScript** language *(Adobe-DPS-Extension)*. Special hardware and software of Silicon Graphics workstations can be used by means of the **SGI-XGL** extension.

References

[1] Asente, P. – Swick, R.R.: *X Window System Toolkit. The Complete Programmer's Guide and Specification.* X Version 11, Release 4. Digital Press, Bedford, MA, 1990.

[2] Barkakati, N.: *UNIX Desktop Guide to OPEN LOOK.* SAMS, Carmel, IN, 1992.

[3] Limpouch, A.: *X Window System - programování aplikací.* Grada, Praha, 1993.

[4] McCormack, J. – Asente, P.: An Overview of the X Toolkit. *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software* (Banff, Oct. 17–19, 1988), ACM Press, 1988, pp. 46–55.

[5] Mikes, S.: *X Window System Program Design and Development.* Addison-Wesley, Reading, MA, 1992.

[6] Scheifler, R.W. – Gettys, J.: *X Window System. The Complete Reference to Xlib, X Protocol, ICCM, XLFD.* X Version 11, Release 4. Digital Press, Bedford, MA, 1990.

[7] Young, D.A.: *X Window Systems: Programming and Applications with Xt.* Prentice Hall, Englewood Cliffs, NJ, 1989.

[8] Peterson, Ch.D.: *Athena Widget Set - C Language Interface.* X Window System. X Version 11, Release 5. MIT, Cambridge, MA, 1991.

[9] Rosenthal, D.: *Inter-Client Communication Conventions Manual.* Version 1.1. MIT X Consortium Standard. X Version 11, Release 5. MIT, Cambridge, MA, 1991.

[10] Flowers, J.: *X Logical Font Description Conventions.* Version 1.4. MIT X Consortium Standard. X Version 11, Release 5. MIT, Cambridge, MA, 1991.

[11] Gettys, J. – Scheifler, R.W.: *Xlib - C Language Interface.* MIT X Consortium Standard. X Version 11, Release 5. First Revision. MIT, Cambridge, MA, 1991.

[12] McCormack, J. – Asente, P. – Swick, R.R.: *X Toolkit Intrinsics - C Language Interface.* X Window System. X Version 11, Release 5. First Revision. MIT, Cambridge, MA, 1991.

An Algorithm for Fast Voxel Scene Traversal

Miloš Šrámek*
Institute of Measurement Science
Slovak Academy of Sciences,
Dúbravská cesta 9, 842 19 Bratislava, ČSFR

Abstract

Together with improvement of tomographic methods of scanning 3D data, residing in an increase of distinguishing ability of the scanners, the quality of final 3D reconstruction of the data comes into foreground. An inevitable condition for visualization of small details, comparable with the voxel size is application of such algorithms, which enables to define a surface of the scanned object on subvoxel level. Coming out from paper published at previous year of *Winter School on Computer Graphics and CAD Systems* we present an algorithm for fast traversal of the voxel scene, which enables to find a nearest intersection of a ray with the object surface defined in such a way. High speed of the algorithm utilizes object coherency of the scene, which means that voxels belonging to the object are usually grouped in connected regions.

The algorithm consists from two steps: preprocessing and scene traversal. In the preprocessing phase we assign to each background voxel of the segmented scene a value equal to its chessboard distance from the nearest object voxel, which defines a cubic macro region with no object voxels inside. While generating a discrete ray, we can jump over this region, which results in up to 6-fold speed up of the traversal. The algorithm has no additional demands on memory, since the distance is stored in originally "empty" background voxels.

1 Introduction

Together with improvement of tomographic methods of scanning 3D data, residing in an increase of distinguishing ability of the scanners, the quality of final 3D reconstruction of the data comes into foreground. An inevitable condition for visualization of small details, comparable with the voxel size is application of such algorithm, which enables to define a surface of the scanned object on subvoxel level. Traditionally, triangulation methods[LC87] are used, which approximate object surface defined within a neighborhood of 8 voxels(cell)