Document Information

Analyzed document	Computer Graphics.pdf (D165969245)
Submitted	5/5/2023 10:24:00 AM
Submitted by	Mumtaz B
Submitter email	mumtaz@code.dbuniversity.ac.in
Similarity	16%
Analysis address	mumtaz.dbuni@analysis.urkund.com

Sources included in the report

W	URL: http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf Fetched: 6/20/2021 1:41:52 PM	33
SA	**INF_2046.pdf** Document INF_2046.pdf (D164968115)	9
SA	**CG book _for_publishing_22_12_2018.docx** Document CG book _for_publishing_22_12_2018.docx (D46266915)	8
SA	**Computer Graphics.pdf** Document Computer Graphics.pdf (D165747830)	19
W	URL: https://studio.eecs.umich.edu/confluence/download/attachments/17827062/mathematicsforcomputerg... Fetched: 6/23/2022 3:55:00 PM	13
W	URL: https://docplayer.net/53449703-Computergraphics-dcap504.html Fetched: 8/20/2022 12:05:28 PM	14
W	URL: http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf Fetched: 11/2/2021 8:30:27 AM	17
SA	**Unit-3 (Part-III).docx** Document Unit-3 (Part-III).docx (D77528823)	12
W	URL: https://odl.ptu.ac.in/SLM/msc%20it/3RD/MSIT_301_Computer_Graphics.pdf Fetched: 9/22/2022 2:06:14 PM	4
SA	**Computer Graphics (Final) 190587 .docx** Document Computer Graphics (Final) 190587 .docx (D115191408)	4
SA	**190587 (Jesse Tendai Tswamuno) Computer Graphics Answer sheet.docx** Document 190587 (Jesse Tendai Tswamuno) Computer Graphics Answer sheet.docx (D119943759)	1
SA	**Graphics exam NCSC 300.docx** Document Graphics exam NCSC 300.docx (D104632959)	6
SA	**MSc_Computer Graphics_Course code MCSDSC_1.7.pdf** Document MSc_Computer Graphics_Course code MCSDSC_1.7.pdf (D140276469)	2

W	URL: https://aditipaulsite.files.wordpress.com/2019/10/computer-graphic_final-book.pdf Fetched: 11/7/2021 8:04:03 PM	6
W	URL: https://www.egyankosh.ac.in/bitstream/123456789/63867/1/MMTE-004.pdf Fetched: 11/8/2021 10:23:05 PM	2
SA	**Graphics Examination 190232.pdf** Document Graphics Examination 190232.pdf (D125664974)	1
SA	**CG assaignment 1.rtf** Document CG assaignment 1.rtf (D25685411)	4
W	URL: https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml Fetched: 11/6/2021 8:40:48 PM	21
SA	**ODMCA-404T Palwinder 10th Feb 2020.docx** Document ODMCA-404T Palwinder 10th Feb 2020.docx (D63728007)	5
SA	**Computer_Graphics_Block_2.pdf** Document Computer_Graphics_Block_2.pdf (D149210060)	4
W	URL: https://pdfcoffee.com/mathematicsforcomputergraphicsandgameprogrammingpdf-pdf-free.html Fetched: 11/7/2022 9:41:10 AM	3
SA	**190305 computergraphics.pdf** Document 190305 computergraphics.pdf (D119942496)	1
SA	**190073 computer graphics.pdf** Document 190073 computer graphics.pdf (D119942830)	1
SA	**NCIS300 COMPUTER GRAPHICS 190372.pdf** Document NCIS300 COMPUTER GRAPHICS 190372.pdf (D119943374)	1

Entire Document

3 Unit 1 Overview of Computer Graphics
4 Contents Overview of
Computer Graphics 1.0 Objectives 1.1 Introduction to Computer Graphics: Types of Computer Graphics 1.2 Applications of Computer Graphics 1.3 Techniques of
Object Rendering: Shading, Ray Tracing, Reflection and Transparency 1.4 Graphics Software Packages: Two general classifications for graphics software 1.5 Graphical Input Devices 1.6 Graphical Output Devices 1.7 Polynomial Method for Ellipse and Circle 1.8 DDA Algorithm for Line, Circle and Ellipse 1.9 Bresenham's Algorithm for Line Drawing and Circle 1.10 Midpoint Methods for Line and Circle 1.0 Objectives • To gain insight into Computer Graphics • To enumerate the types of Computer Graphics • To learn applications of Computer Graphics • To understand software packages used in Graphics • To list the input and output devices for Graphics • To acquire knowledge of Scan devices and Scan Conversion Methods • To know the various Algorithms of Graphics 1.1 Introduction to Computer Graphics: Types

83% **MATCHING BLOCK 1/191** W

of Computer Graphics Introduction to Computer Graphics Computer graphics is an art of drawing pictures, lines, charts, etc using computers with the help of programming. Computer graphics is made up of number of pixels. Pixel is the smallest graphical picture or unit represented on the computer screen.

The term
computer graphics includes
almost everything on computers that is not text or
sound.
Today almost every computer can do some graphics, and people have even come to
5 expect to control their computer through icons and pictures rather than just by typing. In

Computer Graphics,
the pictures can be photographs, drawings, movies, or simulations -- pictures of things which do not yet exist and maybe could never exist. Or
they may be pictures from places we cannot see directly, such as medical images from inside our body.
Types of Computer Graphics Basically there are two types of computer graphics namely: •

98%	**MATCHING BLOCK 2/191**	SA	INF_2046.pdf (D164968115)

Interactive Computer Graphics: Interactive Computer Graphics involves a two way communication between computer and user. Here the observer is given some control over the image by providing him with an input device, for example, the video game controller of the ping pong game. This helps him to signal his request to the computer. The computer on receiving signals from the input device can modify the displayed picture appropriately. To the user it appears that the picture is changing instantaneously in response to his commands. He can give a series of commands, each one generating a graphical response from the computer. In this way he maintains a conversation, or dialogue, with the computer. Interactive computer graphics

affects our lives in a number of

94%	**MATCHING BLOCK 3/191**	SA	INF_2046.pdf (D164968115)

indirect ways. For example, it helps to train the pilots of our airplanes. We can create a flight simulator which may help the pilots to get trained not in a real aircraft but on the grounds at the control of the flight simulator. The flight simulator is a mock up of an aircraft flight deck, containing all the usual controls and surrounded by screens on which we have the projected computer generated views of the terrain visible on take off and landing. Flight simulators have many advantages over

the

94%	**MATCHING BLOCK 4/191**	SA	INF_2046.pdf (D164968115)

real aircrafts for training purposes, including fuel savings, safety, and the ability to familiarize the trainee with a large number of the world's airports. •

Non Interactive Computer Graphics: In non interactive computer graphics otherwise known as passive computer graphics. it is the computer graphics in which user does not have any kind of control over the image. Image is merely the product
6 of static stored program and will work according to the instructions given in the program linearly. The image is totally under the control of program instructions not under the user. Example: screen savers. 1.2 Applications of

49%	**MATCHING BLOCK 5/191**	SA	CG book _for_publishing_22_12_2018.docx (D46266915)

Computer Graphics Design processes A major use of computer graphics is in design processes, particularly for engineering and architectural systems, but almost all products are now computer designed. Generally referred to as CAD, computer-aided design methods are now routinely used in the design of buildings, automobiles, aircraft, watercraft, spacecraft, computers, textiles, and

many,
many other products Presentation graphics Another

86%	**MATCHING BLOCK 6/191**	SA	Computer Graphics.pdf (D165747830)

major application area is presentation graphics, used to produce illustrations for reports or to generate 35-mm slides or transparencies

for

74%	**MATCHING BLOCK 7/191**	SA	Computer Graphics.pdf (D165747830)

use with projectors. Presentation graphics is commonly used to summarize financial, statistical, mathematical, scientific, and economic data for research reports, consumer information bulletins, and other types of reports. Workstation devices and service bureaus exist for converting screen displays into 35-mm slides or overhead transparencies for use in presentations. Typical examples of presentation graphics are bar charts, line graphs, surface graphs, pie charts, and other displays

showing relationships between multiple parameters.
Animations

71% **MATCHING BLOCK 8/191** SA Computer Graphics.pdf (D165747830)

Animations are often used in CAD applications. Real-time animations using wise frame displays on a video monitor are useful for testing performance of a vehicle or system.

85% **MATCHING BLOCK 9/191** SA Computer Graphics.pdf (D165747830)

Animations in virtual reality environments are used to determine how vehicle operators are affected by certain motions. When object designs are complete, or nearly complete, realistic lighting models and surface rendering

are

84% **MATCHING BLOCK 10/191** SA Computer Graphics.pdf (D165747830)

applied to produce displays that will show the appearance of the final product. 7

Laying out floor plans Architects use interactive graphics methods to lay out floor plans that

92% **MATCHING BLOCK 11/191** SA Computer Graphics.pdf (D165747830)

show the positioning of rooms, doors, windows, stairs, shelves, counters, and other building features. Working from the display of a building layout on a video monitor, an electrical designer can try out arrangements for wiring, electrical outlets, and fire warning systems. Also, facility-layout packages can be applied to the layout to determine space utilization in an office or on a manufacturing floor. Realistic displays of architectural designs permit both architects and their clients to study the appearance of a single building or a group of buildings, such as a campus or industrial complex.

With virtual-reality systems, designers can even go for a simulated "walk" through the rooms or around the outsides of buildings to better appreciate the overall effect of a particular design.

72% **MATCHING BLOCK 12/191** SA Computer Graphics.pdf (D165747830)

In addition to realistic exterior building displays, architectural CAD packages also provide facilities for experimenting with three-dimensional interior layouts and lighting. Fine art

and commercial art applications Computer graphics methods are widely used in both

75% **MATCHING BLOCK 13/191** SA CG book _for_publishing_22_12_2018.docx (D46266915)

fine art and commercial art applications. Artists use a variety of computer methods, including special-purpose hardware, artist's paintbrush (such as Lumens), other paint packages (

such as Pixelpaint and Superpaint), specially developed software, symbolic mathematics packages (such as

87% **MATCHING BLOCK 14/191** SA CG book _for_publishing_22_12_2018.docx (D46266915)

Mathematics), CAD packages, desktop publishing software, and animation packages that provide facilities for designing object shapes and specifying object motions.

The basic idea behind a paintbrush program is that it

100% **MATCHING BLOCK 15/191** W

allows artists to "paint" pictures on the screen of a video monitor. Actually, the picture is usually painted electronically on a graphics tablet (digitizer) using a stylus, which can simulate different brush strokes, brush widths, and colors.

Motion pictures, music videos, and television shows Computer graphics
methods
are
now commonly

100%	MATCHING BLOCK 16/191	SA	CG book _for_publishing_22_12_2018.docx (D46266915)

used in making motion pictures, music videos, and television shows. Sometimes the graphics scenes are displayed by themselves, and sometimes graphics objects are combined with the actors and live scenes.

100%	MATCHING BLOCK 17/191	SA	Computer Graphics.pdf (D165747830)

Music videos use graphics in several ways. Graphics objects can be combined with

the

97%	MATCHING BLOCK 18/191	SA	Computer Graphics.pdf (D165747830)

live action, or graphics and image processing techniques can be used to produce a transformation of one person or object into another (morphing). 8

Educational aids
and Simulators Computer-generated models of physical, financial, and economic systems are often used as educational aids.
Models of

78%	MATCHING BLOCK 19/191	SA	CG book _for_publishing_22_12_2018.docx (D46266915)

physical systems, physiological systems, population trends, or equipment, such as the color coded diagram, can help trainees to understand the operation of the system. For some training applications, special systems are designed. Examples

of such specialized systems
are the simulators for practice sessions or training of ship captains, aircraft pilots, heavy-equipment operators, and

93%	MATCHING BLOCK 20/191	SA	Computer Graphics.pdf (D165747830)

air traffic control personnel. Some simulators have no video screens; for example, a flight simulator with only a control panel for instrument flying. But most simulators provide graphics screens for visual operation.

68%	MATCHING BLOCK 21/191	SA	Computer Graphics.pdf (D165747830)

Scientists, engineers, medical personnel, business analysts, and others often need to analyze large amounts of information or to study the behavior of certain processes. Numerical simulations carried out on supercomputers frequently produce data files containing thousands and even millions of data values. Similarly, satellite cameras and other sources are amassing large data files faster than they can be interpreted. Scanning these large sets of numbers to determine trends and relationships is a tedious and ineffective process. But if

the data are converted to a visual form,
the trends and patterns are often immediately apparent. Mathematicians, physical

92%	MATCHING BLOCK 23/191	SA	Computer Graphics.pdf (D165747830)

scientists, and others use visual techniques to analyze mathematical functions and processes or simply to produce interesting graphical representations.

97%	MATCHING BLOCK 24/191	SA	Computer Graphics.pdf (D165747830)

Although methods used in computer graphics and Image processing overlap, the two areas concerned with fundamentally different operations.

100%	MATCHING BLOCK 22/191	W

In computer graphics, a computer is used to create a picture. Image processing,

on the other hand

90%	MATCHING BLOCK 25/191	SA Computer Graphics.pdf (D165747830)

applies techniques to modify or interpret existing pictures, such as photographs and TV scans. Two principal applications of image processing are (1) improving picture quality and (2) machine perception of visual information, as used in robotics. To apply image processing methods, we first digitize a photograph or other picture into an image file. Then digital methods can be applied to rearrange picture parts, to enhance color separations, or to improve the quality of shading. These techniques are used

extensively in commercial art applications that involve the

65%	MATCHING BLOCK 26/191	SA Computer Graphics.pdf (D165747830)

retouching and rearranging of sections of photographs and other artwork. Similar methods are used to analyze satellite photos of the earth and photos of galaxies. Medical applications 9 also make extensive use of image processing techniques for picture enhancements, in tomography and in simulations

of operations. 1.3 Techniques of Object Rendering We spend much of our
time improving the way computer pictures can simulate real world scenes. We want
images on computers to not just look more realistic, but also to be more realistic in
their colors, the way objects and rooms are lighted, and the way different materials appear. We call this work "realistic image synthesis", and the following series of pictures will show some of
the
techniques in stages from very simple pictures through very realistic ones.
Object

95%	MATCHING BLOCK 27/191	W

Rendering Rendering is the process of generating an image from a 2

D or 3D model (or models in what collectively could be called a scene file), by means of computer programs. Also, the results of such a model can be called a rendering. Computer graphics uses several simple object rendering techniques to make models appear three-dimensional. Shading Shading techniques extend the realistic appearance of objects and introduce features such as transparency and textures.
10 This vase has been modeled as a symmetrical pattern of vertically- oriented surfaces - tiny flat patches which approximate the round shape of the vase. In this image, each tiny surface is shaded separately with a different gray value based on its orientation to the light source. By introducing a technique called Gouraud shading, we can smooth out the appearance of the vase and hide the individual surfaces from view. The shading is varied on each surface in proportion to values calculated at the edges and from neighboring surfaces. Phong shading improves the apparent realism of the rendering still further by introducing highlights. The way light reflects from real surfaces depends on how shiny the surface is and on the angle you are looking from. Most surfaces are not shiny, but have a more dull or "matte" or "diffuse" appearance.
11 Can you tell where the light is shining from? The surface of this vase is just shiny enough to reflect some light directly to the viewer at certain angles, but around the sides the appearance is much duller. Light hitting the vase at flatter angles is scattered more evenly in all directions. In this last image the shading technique has been extended to let some light pass through the vase - for transparency. Note that the reflection highlights are still there, even if they are less visible. These shading techniques (and more) have all been incorporated into the generalized rendering technique called ray tracing Color Computers don't create color exactly the way we see it. Computers typically display color in three components - red, green, and blue. When combined, these three colors make the full-color image seen in the upper left of this image. By controlling color display, we can simulate on the computer different kinds of color blindness.

98%	MATCHING BLOCK 29/191	W

Ray Tracing In computer graphics, ray tracing is a technique for generating an image by tracing the path of light through pixels in an image plane and simulating the effects of its encounters with virtual objects.

Reflection and Transparency

12 The best way to appreciate how far these simple techniques have been developed is through much more complex graphic images. Radiosity Radiosity is a method of rendering based on an detailed analysis of light reflections off diffuse surfaces. The images that result from a radiosity renderer are characterized by soft gradual shadows. Radiosity is typically used to render images of the interior of buildings, and can achieve extremely photo-realistic results for scenes that are comprised of diffuse reflecting surfaces. Quality of Light The research image sampler shows more current work in radiosity and other techniques. 1.4 Graphics Software Packages: Two general classifications for graphics software There are two general classifications for graphics software: General programming packages and Special-purpose applications packages.
13 • A general graphics

89%	MATCHING BLOCK 28/191	W
programming package provides an extensive set of graphics functions that can be used in		

a high-level programming language, such as C or FORTRAN. An example of a general graphics programming package is the GL (Graphics Library) system on Silicon Graphics equipment. Basic functions in a general package include those for

71%	MATCHING BLOCK 31/191	SA INF_2046.pdf (D164968115)
generating picture components (straight lines, polygons, circles, and other figures), setting color and intensity values, selecting views, and applying		

transformations. • By contrast, application graphics packages are designed for nonprogrammers, so that users can generate displays without worrying about how graphics operations work. The interface to the graphics routines in such packages allows users to communicate with the programs in their own terms. Examples of such applications packages are the artist's painting programs and various business, medical, and CAD systems. Graphics Functions A general-purpose graphics package

95%	MATCHING BLOCK 37/191	SA INF_2046.pdf (D164968115)
provides users with a variety of functions for creating and manipulating		

pictures. These routines

75%	MATCHING BLOCK 30/191	W
can be categorized according to whether they deal with output, input, attributes, transformations, viewing,		

or general control. The basic building blocks for creating and manipulating pictures. These routines

75%	MATCHING BLOCK 32/191	W
can be categorized according to whether they deal with output, input, attributes, transformations, viewing,		

or general control.

78%	MATCHING BLOCK 33/191	W
The basic building blocks for pictures are referred to as output primitives. They include character strings and geometric entities, such as points, straight lines, curved Lines, filled areas (		

polygons, circles, etc.), and shapes defined with arrays of color points. Routines for generating output primitives provide the basic tools for conshucting pictures. Attributes are the properties of the output primitives; that is, an attribute describes how a particular primitive is to be displayed. They include intensity and

100%	MATCHING BLOCK 34/191	W
color specifications, line styles, text styles, and area-filling patterns.		

Functions within this category can be used to set attributes for an individual primitive class or for groups of output primitives. We can

100%	MATCHING BLOCK 35/191	W

change the size, position, or orientation of an object within

a scene using geometric transformations. Similar modeling transformations are used to construct a scene using
14 object descriptions given in modeling coordinates. Given the primitive and attribute definition of a picture in world coordinates, a graphics package projects a selected view of the picture on an output device. Viewing transformations are used to specify the view that is to be presented and the portion of the output display area that is to be used. Pictures can be subdivided into component parts, called structures or segments or objects, depending on the software package in use. Each structure defines one logical unit of the picture. A scene with several objects could reference each individual object in a separate named structure. Routines for processing structures carry out the creation. modification, and transformation of structures. Interactive graphics applications use various kinds of input devices, such as a mouse, a tablet, or a joystick. Input functions are used to control and process the data flow from the interactive devices. Finally, a graphics package contains a number of housekeeping tasks, such as clearing a display screen and initializing parameters, We can lump the functions for carrying out the chores under the heading control operations. 1.5 Graphical Input Devices Various devices are available for data input on graphics workstations. Most systems have a keyboard and one or more additional devices specially designed for interactive input. These include a mouse, trackball, spaceball, joystick, digitizers, dials, and button boxes. Some other input devices used in particular applications are data gloves, touch panels, image scanners, and voice systems. Keyboards-
An alphanumeric keyboard on a graphics system is used primarily as a device for entering text strings. The keyboard is an efficient device for inputting such
nongraphic data as picture labels associated with a graphics display. Keyboards can also be provided with features to facilitate entry of screen coordinates, menu selections, or graphics functions.
Cursor-control keys and function keys are common features on general-purpose keyboards.
Function keys allow users to enter frequently used operations in a single keystroke, and
cursor-control keys can be used to select displayed objects or coordinate
15 positions by positioning the screen cursor.
Other types of cursor-positioning devices, such as a trackball or joystick, are included on some keyboards. Additionally,
a numeric keypad is, often included on the key-board for fast entry of numeric data.
Mouse- A mouse is small hand-held box used to position the screen cursor. Wheels or rollers on the bottom of the mouse can be used to record the amount and direction of movement. Another method for detecting mouse motion is with an optical sensor. For these systems, the mouse is moved over a special mouse pad that has a grid of horizontal and vertical lines. The optical sensor detects movement across the lines in the grid. Since a mouse can be picked up and put down at another position without in the position change in cursor movement, it is used for making relative change in the position of the screen cursor. One, two, or

83%	MATCHING BLOCK 36/191	W

three buttons are usually included on the top of the mouse for signaling the execution of some operation,

such as recording a cursor position or invoking a function. Most general-purpose graphics systems now include a mouse and a keyboard as the major input devices. Additional devices can be included in the basic mouse design to increase the number of allowable input parameters. Trackball and Spaceball- As the name implies, a

78%	MATCHING BLOCK 38/191	W

trackball is a ball that can be rotated with the fingers or palm of the hand to produce screen-cursor movement.

Potentiometers,

83%	MATCHING BLOCK 39/191	W

attached to the ball, measure the amount and direction of rotation. Trackballs are often mounted on keyboards or

other devices such as the Z mouse. While

77%	MATCHING BLOCK 42/191	SA	Unit-3 (Part-III).docx (D77528823)

a trackball is a two- dimensional positioning device, a spaceball provides six degrees of freedom. Unlike the trackball, a spaceball does not actually move.

77%	MATCHING BLOCK 40/191	W

Strain gauges measure the amount of pressure applied to the spaceball to provide input for spatial positioning and orientation as the ball is pushed or pulled in various directions. Spaceballs are used for three-dimensional positioning and selection operations in virtual-reality systems, modeling, animation, CAD, and other applications. Joysticks- A joystick consists of

a small, vertical lever (called the stick)

100%	MATCHING BLOCK 41/191	W

mounted on a base that is used to steer the screen cursor around. Most

joysticks select screen positions with actual stick movement; others respond to pressure on the stick. Some joysticks are 16 mounted on a keyboard; others function as stand-alone units. The distance that the stick is moved in any direction from its center position corresponds to screen-cursor movement in that direction.

84%	MATCHING BLOCK 46/191	SA	Unit-3 (Part-III).docx (D77528823)

Potentiometers mounted at the base of the joystick measure the amount of movement,

and springs return the stick to the center position when it is released. One or more buttons can be programmed to act as input switches to signal certain actions once a screen position has been selected. In another type of movable joystick, the stick is used to activate switches that cause the screen cursor to move at a constant rate in the direction selected. Eight switches, arranged in a circle, are sometimes provided, so that the stick can select any one of eight directions for cursor movement. Pressure-sensitive joy-sticks, also called isometric joysticks, have a nonmovable stick. Pressure on the stick is measured with strain gauges and converted to movement of the cursor in the direction specified. Data Glove- A data glove can be used to grasp a "virtual" object. The glove is constructed with a

58%	MATCHING BLOCK 43/191	W

series of sensors that detect hand and finger motions. Electromagnetic coupling between transmitting antennas and receiving antennas

is used to provide information about the position and orientation of the hand. The transmitting and receiving antennas can each

100%	MATCHING BLOCK 44/191	W

be structured as a set of three mutually perpendicular coils, forming

a three- dimensional

84%	MATCHING BLOCK 45/191	W

Cartesian coordinate system. Input from the glove can be used to position or manipulate objects in a virtual scene.

A two-dimensional projection of the scene can be viewed on a video monitor, or a three-dimensional projection can be viewed with a headset. Digitizers-

100%	MATCHING BLOCK 47/191	SA	Unit-3 (Part-III).docx (D77528823)

A common device for drawing, painting, or interactively selecting coordinate positions on an object is a

digitizer. These

80%	MATCHING BLOCK 48/191	W

devices can be used to input coordinate values in either a two-dimensional or a three-dimensional space.

Typically, a digitizer is used to scan over a drawing or object and to input a set of discrete coordinate positions, which can be joined with straight-line segments to approximate the curve or surface shapes. • One type of digitizer is the graphics tablet (also referred to as a data tablet),

96%	MATCHING BLOCK 54/191	SA	Unit-3 (Part-III).docx (D77528823)

which is used to input two-dimensional coordinates by activating a hand cursor

or stylus at selected positions on a flat surface. Graphics tablets provide a highly accurate
17 method for selecting coordinate positions, with an accuracy that varies from about 0.2 mm on desktop models to about 0.05 mm or less on larger models. • Acoustic (or sonic) tablets use sound waves to detect a stylus position. Either strip microphones or point microphones can be used to detect the sound emitted by an electrical spark from a stylus tip. The position of the stylus is calculated by timing the arrival of the generated sound at the different microphone positions. An advantage of two-dimensional accoustic tablets is that the microphones can be placed on any surface to form the "tablet" work area. This can be convenient for various applications, such as digitizing drawings in a book. • Three-dimensional digitizers use sonic or electromagnetic transmissions to word positions. One electromagnetic transmission method is similar to that used in the data glove: A coupling between the transmitter and receiver is used to compute the location of a stylus as it moves over the surface of an object. Image Scanners- Drawings, graphs, color and

83%	MATCHING BLOCK 49/191	W

black-and-white photos, or text can be stored for computer processing

with an image scanner

100%	MATCHING BLOCK 50/191	W

by passing an optical scanning mechanism over the information to be stored.

The gradations of gray scale or color are then recorded and stored in an array. Once we have the internal representation of a picture, we can apply transformations to rotate, scale, or crop the picture to a particular screen area.

100%	MATCHING BLOCK 51/191	W

We can also apply various image-processing methods to modify the

array representation of the picture. For scanned text input, various editing operations can be performed on the stored documents. Some scanners are able to scan either graphical representations or text, and they come in a variety of sizes and capabilities. Touch Panels- As the
name implies,
touch panels allow displayed objects or
screen positions to be selected with the touch
of a finger. A
typical application of touch panels is for the selection of processing options that are represented with graphical icons. Some systems, such as
the
plasma panels
are designed with touch screens. Other systems can be adapted for touch input by fitting
a transparent device with a touch-sensing mechanism over the video monitor
screen. Touch input can be recorded using
optical, electrical, or acoustical methods.
18 •
Optical touch panels employ a line of infrared light-emitting diodes (LEDs) along one vertical edge and
along one horizontal edge of the frame. The opposite vertical and horizontal edges contain light detectors. These detectors are used to record which beams are interrupted when the panel is touched. The two crossing beams that are interrupted identify the horizontal and vertical coordinates of the screen position selected. Positions can be selected with an accuracy of about 1/4 inch
with closely spaced LEDs, it is possible to b d two horizontal or two vertical beams simultaneously. In this case, an average position between the two interrupted beams is recorded. The LEDs operate at infrared frequencies, so that the light is not visible to a user. • An
electrical touch panel is constructed with two transparent plates separated by

a small distance. One of the plates is coated with a conducting material, and the other plate is coated with a resistive material. When the outer plate is touched, it
comes in
contact with the inner plate. This contact creates a
voltage drop
across the resistive plate that is converted to the coordinate values of the selected screen position. • In acoustical touch panels, high-frequency sound waves are generated
in
the horizontal and vertical directions across a glass plate. Touching the screen causes part of each wave to be reflected from the finger to the emitters. The screen position at the point of contact is calculated from a measurement of the time interval between the transmission of each wave and its reflection to the emitter.
Light Pens- Pencil-shaped devices are

used to select screen positions by detecting the light coming from

points on the CRT screen. They are sensitive to the short burst of light emitted from the phosphorous coating at the instant the electron beam strikes a particular point. Other light sources, such as the background light in the room, are usually not detected by a light pen. An

activated light pen, pointed at a spot on the screen as the electron beam lights up that spot, generates an electrical pulse that causes the coordinate position of the electron beam to be recorded.

As with cursor-positioning devices, recorded Light-pen coordinates can be used to position an object or to select a processing option. Although Light pens are still with us, they are not as popular as they once were since they have several disadvantages compared to other input devices that have been developed. For
19 one, when a light pen is pointed at the screen, part of the main image is obscured by the hand and pen. And prolonged use of the light pen can cause arm fatigue. Also, light pens require special implementations for some applications because they cannot detect positions within black areas. To be able to select positions in any screen area with a light pen, we must have some non-zero intensity assigned to each screen pixel. In addition, light pens sometimes give false readings due to background lighting in a room.
Voice Systems- Speech recognizers are
used in some graphics workstations as input devices to accept voice commands.
The voice-system input can be used to initiate graphics operations or to enter data. These systems operate by matching an input
to
a predefined dictionary of words and phrases. A dictionary is set up for a particular operator by having, the operator speak the command words to be used into the system. Each word is spoken several times, and the system analyzes
the word and establishes a frequency pattern for that word
in the dictionary along with the corresponding function to be performed. Later, when a voice command is given,
the system searches the dictionary for a frequency-pattern match.
Voice input is typically spoken into a microphone mounted on a headset.
The microphone is designed to minimize input of other background sounds.
If a different operator is to use the system, the dictionary must be reestablished with that operator's voice patterns.
Voice systems have some advantage over other input devices, since the attention of the operator does not have to be switched from one device to another to enter a command. 1.6 Graphical Output Devices Typically,
the primary
output device in a graphics system is
a video monitor.
The operation of most video monitors is based on the standard cathode-ray tube (CRT) design,
but several other technologies exist and solid-state monitors may eventually predominate. Refresh Cathode-Ray Tubes-
A beam of electrons (cathode rays),
emitted by an electron gun, passes through focusing and deflection systems
that direct the beam
toward specified positions on the
p h o s p h o m t e d
screen. The phosphor then emits a small spot of light at each position contacted by the electron beam.
Because the light emitted by the phosphor
20 fades very rapidly, some method is needed for maintaining the screen picture.
One way to keep the phosphor glowing
is to redraw

the picture repeatedly by quickly directing the electron beam back over the same points.
This
type of display is called a refresh CRT.
Raster-Scan Displays- The most common type of graphics monitor employing a CRT is the raster-scan display, based on television technology.
In a
raster-
scan system,
the
electron beam is swept across the screen, one row at a time from top to bottom.
As
the electron beam moves across each row,
the beam intensity is turned on and off to create a pattern of illuminated spots.
Picture definition is stored in a memory
area called
the
refresh buffer or frame buffer.
This memory area
holds
the
set of intensity values for all the screen points. Stored intensity values are
then
retrieved from the
refresh buffer and "painted" on the screen one row (
scan line)
at a time.
Each screen point is referred to as a pixel
or
pel (
shortened forms of picture element).
The capability of a raster-scan system to
store intensity information for each screen point
makes it well suited for the realistic display of scenes containing subtle shading and color
patterns.
Home television sets and printers are examples of other systems using raster-scan methods.
Random-Scan Displays- When operated as a random-scan display unit, a CRT has
the electron beam directed only to the parts of the screen where a picture is to be drawn.
Random-scan
monitors draw a picture one line at a time
and for this reason are also referred to as vector displays (or stroke-writing
or calligraphic displays).
The component lines of a picture can be drawn and refreshed
by a random-scan system in any specified order.
A pen plotter operates in a similar way and is an example of a random-scan, hard- copy device.
Random-scan systems are designed for line-drawing applications and cannot display realistic shaded scenes.
Since
picture definition is stored as a set of line drawing instructions
and not
as a set of intensity values
for all screen points,
vector displays generally have higher resolution
than raster
systems.
Also, vector displays produce smooth line drawings because the CRT beam directly follows the line path. A raster system, in contrast, produces jagged lines that are plotted as discreet point sets.
21
Color CRT Monitors-

98%	MATCHING BLOCK 55/191	SA	Computer Graphics (Final) 190587 .docx (D115191408)

A CRT monitor displays color pictures by using a combination of phosphors that emit different-colored light. By combining the emitted light from the different phosphors, a range of colors can be generated. The

two basic techniques for producing color displays with a CRT are: The beam-penetration method and the shadow-mask method. • The beam-penetration method for displaying color pictures has been used with random-scan monitors. Two layers of phosphor, usually red and green, are coated onto the inside of the CRT screen,

100%	MATCHING BLOCK 56/191	SA	Computer Graphics (Final) 190587 .docx (D115191408)

and the displayed color depends on how far the electron beam penetrates

into

60%	MATCHING BLOCK 57/191	SA	190587 (Jesse Tendai Tswamuno) Computer Graphi ... (D119943759)

the phosphor layers. A beam of slow electrons excites only the outer red layer. A beam of very fast electrons penetrates through the red layer and excites the inner green layer. At intermediate beam speeds, combinations of red and green light are emitted to show two additional colors, orange and yellow. The speed of the electrons, and

hence

100%	MATCHING BLOCK 58/191	SA	Computer Graphics (Final) 190587 .docx (D115191408)

the screen color at any point, is controlled by the beam-acceleration voltage.

Beam penetration has been an inexpensive way to produce color in random-scan monitors, but only four colors are possible, and the quality of pictures

83%	MATCHING BLOCK 59/191	SA	CG book _for_publishing_22_12_2018.docx (D46266915)

is not as good as with other methods. • Shadow-mask methods are commonly used in raster-scan systems (

including color TV)

100%	MATCHING BLOCK 60/191	SA	Graphics exam NCSC 300.docx (D104632959)

because they produce a much wider range of colors than the beam- penetration method.

69%	MATCHING BLOCK 62/191	SA	CG book _for_publishing_22_12_2018.docx (D46266915)

A shadow-mask CRT has three phosphor color dots at each pixel position. One phosphor dot emits a red light, another emits a green light, and the third emits a blue light. This type of CRT

has

97%	MATCHING BLOCK 61/191	W	

three electron guns, one for each color dot, and a shadow-mask grid just behind the phosphor-coated screen. The

three

92%	MATCHING BLOCK 63/191	SA	Graphics exam NCSC 300.docx (D104632959)

electron beams are deflected and focused as a group onto the shadow mask, which contains a series of holes aligned with the phosphor-dot patterns. When the three beams pass through a hole in the shadow mask, they activate a dot triangle, which appears as a small color spot on the screen. The phosphor dots in the triangles are arranged so that each electron beam can activate only its corresponding color dot when it passes through the shadow mask. Another configuration for the

three electron guns is an
in-line arrangement in which the
three

71%	MATCHING BLOCK 67/191	SA	Graphics exam NCSC 300.docx (D104632959)

electron guns, and the corresponding red- green-blue color dots on the screen, are aligned along one scan line instead of in a 22 triangular pattern. This in-line arrangement of electron guns is easier to keep in alignment and is commonly used in high-resolution color

CRTs.
Direct-View Storage Tubes- An alternative method for maintaining a screen image is to store the picture information inside the CRT instead of refreshing the screen. A direct-view storage tube (DVST)

94%	MATCHING BLOCK 64/191	W

stores the picture information as a charge distribution just behind the phosphor-coated screen. Two electron guns

are used in a DVST. One, the primary gun, is used to store

76%	MATCHING BLOCK 65/191	W

the picture pattern; the second, the flood gun, maintains the picture display. A

DVST monitor has both disadvantages and advantages compared to the refresh CRT. Because no refreshing is needed,

73%	MATCHING BLOCK 66/191	W

very complex pictures can be displayed at very high resolutions without flicker. Disadvantages of DVST

systems are that they ordinarily do not display color and that selected parts of a picture cannot he erased. To eliminate a picture section, the entire screen must be erased and the modified picture redrawn. The erasing and redrawing process can take several seconds for a complex picture. For these reasons, storage displays have been largely replaced by raster systems.

59%	MATCHING BLOCK 75/191	SA	CG book _for_publishing_22_12_2018.docx (D46266915)

Flat-Panel Displays- Although most graphics monitors are still constructed with CRTs, other technologies are emerging that may soon replace CRT monitors. The term Bat-panel display

96%	MATCHING BLOCK 68/191	W

refers to a class of video devices that have reduced volume, weight, and power requirements compared to a CRT. A significant feature of flat-panel displays is that they are thinner than CRTs, and we can hang them on walls or

wear them on our wrists. Since we can even write on some flat-panel displays, they will soon be available as pocket notepads. Current uses for flat-panel displays include small TV monitors, calculators, pocket video games, laptop computers, armrest viewing of movies on airlines, as advertisement boards in elevators, and as graphics displays in applications requiring rugged, portable monitors. We can separate
flat-panel displays into two categories: Emissive displays and nonemissive

83%	MATCHING BLOCK 69/191	W

displays. • The emissive displays (or emitters) are devices that convert electrical energy into light. Plasma panels, thin-film electroluminescent displays, and Light-emitting diodes

are examples of emissive displays.
Flat CRTs have also been devised, in which electron beams arts accelerated parallel to the screen, then deflected 90' to
23 the screen. But flat CRTs have not proved to be as successful as other emissive devices. A third

100% MATCHING BLOCK 70/191 W

type of emissive device is the light-emitting diode (LED). A matrix of diodes is arranged to form the pixel positions in the display, and picture definition is stored in

a

100% MATCHING BLOCK 71/191 W

refresh buffer. As in scan-line refreshing of a CRT, information is read from the refresh buffer and converted to voltage levels that are applied to the diodes to produce the light patterns

in the display. •
Nonemmissive

95% MATCHING BLOCK 72/191 W

displays (or nonemitters) use optical effects to convert sunlight or light from some other source into graphics patterns. The most important example of a nonemissive flat-panel display is a liquid-crystal device. Plasma panels, also called gas-discharge displays, are constructed by filling the region between two glass plates with a mixture of gases that usually includes neon. A series of vertical conducting ribbons is placed on one glass panel, and a set of horizontal ribbons is built into the other glass panel. Firing voltages applied to a pair of horizontal and vertical conductors cause the gas at the intersection of the two conductors to break down into a glowing plasma of electrons and ions. Picture definition is stored in a refresh buffer, and the firing voltages are applied to refresh the pixel positions (

at the intersections of the conductors) 60

76% MATCHING BLOCK 73/191 W

times per second. Alternating currents methods are used to produce faster application of

the

100% MATCHING BLOCK 74/191 W

firing voltages, and thus brighter displays. Separation between pixels is provided by the electric field of

the conductors. One disadvantage of plasma panels has been that they were strictly monochromatic devices, but systems have been developed that are now capable of displaying color and grayscale. Liquid crystal displays- (LCDS) are commonly used in small systems, such as calculators and portable, laptop computers. These non- emissive

95% MATCHING BLOCK 76/191 W

devices produce a picture by passing polarized light from the surroundings or from an internal light source through a liquid- crystal material that can be aligned to either block or transmit the light. The term liquid crystal refers to the fact that these compounds have a crystalline arrangement of molecules, yet they flow like a liquid. Flat-panel displays commonly use nematic (threadlike) liquid-crystal compounds that tend to keep the long axes of the rod-shaped molecules aligned. A flat-panel display can then be constructed with a nematic liquid 24 crystal. Two glass plates, each containing a light polarizer at right angles to the-other plate, sandwich the liquid-crystal material. Rows of horizontal transparent conductors are built into one glass plate, and columns of vertical conductors are put into the other plate. The intersection of two conductors defines a pixel position. Normally, the molecules are aligned. Polarized light passing through the material is twisted so that it will pass through the opposite polarizer. The light is then reflected back to the viewer. To turn off the pixel, we apply

a

100% MATCHING BLOCK 77/191 W

voltage to the two intersecting conductors to align the molecules so that the light is not .twisted. This type of flat-panel device is referred to as a passive-matrix LCD. Picture definitions are stored in a refresh buffer, and the screen is refreshed at the rate of 60 frames per second, as in the emissive devices.

Three-Dimensional Viewing Devices- Graphics monitors for the display of three- dimensional scenes have been

devised using a technique that reflects a CRT image from a vibrating, flexible mirror.

As the varifocal mirror vibrates, it changes focal length. These vibrations are synchronized with the display of an object on a CRT so that

each point on the object is reflected from the mirror into a spatial position corresponding to

the distance of that point from a specified viewing position. This allows us to walk around an object or scene and view it from different sides. Such systems have been used in medical applications to analyze data from ultrasonography and CAT scan devices, in geological applications to analyze topological and seismic data, in design applications involving solid objects, and in three-dimensional simulations of systems, such as molecules and terrain. Stereoscopic and Virtual-Reality Systems- Another technique for representing three dimensional objects is displaying stereoscopic views. This method does not produce true three-dimensional images, but it does provide a three-dimensional effect by presenting a different view to each eye of an observer so that scenes do appear

to have depth. One way to produce a stereoscopic effect is to display each of the two views with

a

raster system on alternate refresh cycles. The screen is viewed through glasses, with each

lens designed to

act as a rapidly alternating shutter that is synchronized to block out one of the views.

Figure 2-21 shows a pair of stereoscopic glasses constructed with liquid crystal shutters and an infrared emitter that synchronizes the glasses with the views on the screen.
25 Stereoscopic viewing is also a component in virtual-reality systems, where users

can step into a scene and interact with the environment. A headset containing an optical system to generate the stereoscopic views is commonly used in conjuction with interactive input devices to locate and manipulate objects in the scene. A sensing system in the headset keeps track of the viewer's position, so that the front and back of objects can be seen as the viewer "walks through" and interacts with the display. An interactive virtual-reality

environment can also be viewed with stereoscopic glasses and a video monitor, instead of a headset. This provides a means for obtaining a lower-cost virtual-reality system. 1.7 Polynomial Method for Ellipse and Circle Algorithm to draw circles and ellipses This algorithm is based on the parametric form of the circle equation. Recall that this looks like

$x = h + r \cos\theta$ $y = k + r \sin\theta$ where r is the radius of the circle,

and h,k are the coordinates of the center. What these equation do is generate the x,y coordinates of a point on the circle given an angle θ (theta). The algorithm starts with theta at zero, and then loops adding an increment to theta each time round the loop. It draws straight line segments between these successive points on the circle. The circle is thus drawn as a series of straight lines. If the increment is small enough, the result looks like a circle to the eye, even though in strict mathematical terms is not. The algorithm Below is the algorithm in pseudocode showing the basic idea. theta = 0; // angle that will be increased each loop h = 12 // x coordinate of circle center k = 10 // y coordinate of circle center
26 step = 15; // amount to add to theta each time (degrees) repeat until

100%	MATCHING BLOCK 85/191	W

theta &lt;= 360; { x = h + r*cos(theta) y = k + r*sin(theta)

draw a line to x,y add step to theta } The decision about how big to make the step size is a trade off. If it is very small, many lines will be drawn for a smooth circle, but there will be more computer time used to do it. If it is too large the circle will not be smooth and be visually ugly. Example Below is the algorithm written in Javascript using the HTML5 canvas element to draw into. var ctx = canvas.getContext("2d"); var step = 2*Math.PI/20; // see note 1 var h = 150; var k = 150; var r = 50; ctx.beginPath(); //tell canvas to start a set of lines for(var theta=0; theta &gt; 2*Math.PI; theta+=step) { var x = h + r*Math.cos(theta); var y = k - r*Math.sin(theta); //note 2. ctx.lineTo(x,y); } ctx.closePath(); //close the end to the start point
27 ctx.stroke(); //actually draw the accumulated lines Like most graphics systems, the canvas element differs from the usual mathematical coordinate plane: 1. The origin is in the top left corner. The code above compensates by assuming that h, and k are actually relative to the top left. 2. The y axis is inverted. Positive y is down the screen, not up. To correct for this, the k variable (the y coordinate of the center) must be positive to place the center some way down the screen. Also the y calculation has to subtract the sin(x) term instead of add. Marked in the code as Note 1. Note 2. The step size is set to an exact division of 2π to avoid gaps or over-runs in the circle. This code divides the circle into exactly 20 segments. Note too that as in most computer languages, the trig functions operate in radians, not degrees. 360° = 2π radians. Ellipses The circles can be made into ellipses by simply "squashing" them in one direction or the other. For an ellipse that is wider than it is tall, be divide the y coordinate by a number (here 2) to reduce its height. The two inner calculations would then look like: var x = h + r*Math.cos(theta) ; var y = k - 0.5 * r*Math.sin(theta) ; Changing the 0.5 will alter how much the vertical size is squashed. Multiply the y term this way will make an ellipse that is tall and narrow. Representation of a circle: Polynomial method In this method the circle is represented as: $x^2+y^2=r^2$ where, x= x co-ordinate
28 y= y co-ordinate and r= radius of the circle. Coordinates of point A can be calculated by polynomial equations: (x, $\sqrt{r^2 x^2}$) 1.8 DDA Algorithm for Line, Circle and Ellipse In computer graphics, a digital differential analyzer (DDA) is hardware or software

100%	MATCHING BLOCK 87/191	SA	MSc_Computer Graphics_Course code MCSDSC_1.7.pdf (D140276469)

used for linear interpolation of variables over an interval between start and end point. DDAs are used for rasterization of lines, triangles and polygons in

raster graphics devices. In this algorithm, the starting and end position of the line has to be supplied. The intermediary pixel positions will be calculated by the linear interpolation of variables over an interval between the start and end points. The algorithm is as follows: Let the start and end point of the line be (

76%	MATCHING BLOCK 86/191	W

x1, y1) and (x2, y2), respectively. So slope, m= (y2-y1)/(x2-x1).

Depending on the value of m and the Quadrant to which (x, y) belongs, intermediary pixel positions have to be calculated as follows: The positions have to be calculated as follows:
29 Quadrant m&gt;=1 m&lt;1 First

56%	MATCHING BLOCK 88/191	W

x=x+1 y=y + m x=x+1/m y=y+1 Second x=x-1 y=y + m x=x+1/m y=y+1 Third x=x-1 y=y-m x=x-1/m y=y-1 Fourth x=x+1 y=y + m x=x-1/m y=y-1

Here the values of x and y are approximated to the nearest integer value as the pixel position cannot be a floating point number. This was the theoretical approach towards the algorithm where a point can lie in any of the four quadrants virtually. Practically this is not possible as the pixel position cannot be negative in physical display devices. In physical display devices we have only the 1 st quadrant of the Cartesian axis system where both x and y are positive or equal to zero and can assume integer values. At the time of implementing the algorithm for physical display devices, we only have to program for the first quadrant. Another point to be noted here is that the orientation of physical display devices is different from the geometrical Cartesian axis.

30 In the real co-ordinate system for the first quadrant, the x value increases in right direction and y-value increases towards up continuously. In the physical display device coordinate system, the origin is the top left corner pixel. It means that the top left corner pixel will have a coordinate value (0, 0). The x co-ordinate value will increase in the right direction pixel wise and the y co-ordinate value will increase in the downward direction pixel wise. It may be noted that the bottom-right corner pixel will have the highest co-ordinate value in the display device. For a clear visualization, the screen co-ordinate can be viewed as the reflected version of the geometrical first quadrant where the reflection is done by the line y=0. DDA Circle Drawing Algorithm Title:-Write a Program for DDA CIRCLE DRAWING Algorithm. #
include&gt;graphics.h&lt; #
include&gt;stdio.h&lt; #include&gt;conio.h&lt; #include&gt;dos.h&lt; #include&gt;math.h&lt; #
include&gt;iostream.h&lt; #include&gt;process.h&lt;
class Drawing {
31 float redius,x1,y1,
x2,y2;
public: Drawing() { x1=0; y1=0; x2=0; y2=0; } void get_r() { cout&gt;&gt;"\nEnter Value For Redius"; cin&lt;&lt;redius; } void getline() { cout&gt;&gt;"\nEnter Value X1,Y1,X2,Y2(start & end Co_ordinates For Line)"; cin&lt;&lt;x1&lt;&lt;y1&lt;&lt;x2&lt;&lt;y2; } void draw_ddaCir(); void draw_breCir(); void draw_ddaLine(); void draw_breLine(); }; void Drawing:: draw_breCir() { float
d; x1=0;
32 y1=redius; d=3-(2*redius); do {

62%	MATCHING BLOCK 89/191	W

putpixel(x1+200,y1+200,3); putpixel(y1+200,x1+200,3); putpixel(y1+200,200-x1,3); putpixel(x1+200,200-y1,3); putpixel(-x1+200,-y1+200,3); putpixel(-y1+200,-x1+200,3); putpixel(-y1+200,x1+200,3); putpixel(-x1+200,y1+200,3); if(d&gt;0) { d=d+4*x1+6; } else { d=d+4*(x1-y1)+10; y1=y1-1; }

x1=
x1+1; }
while(x1&gt;y1); }
void Drawing:: draw_ddaCir() { float epsilon,n,val; int i=0; float

25%	MATCHING BLOCK 90/191	W

start_x,start_y; 33 start_x=redius; start_y=0; x1=start_x; y1=start_y; { val=pow(2,i); i++; }while(val&gt;redius); epsilon=1/pow(2,i-1); do { x2=x1+(epsilon * y1); y2=y1-(epsilon * x2); delay(10); putpixel(x2+200,y2+200,3); x1=x2; y1=y2; }while((y1-start_y)&gt;epsilon || (start_x-x1)&lt;

epsilon); } void Drawing:: draw_ddaLine() { float dx,dy,len,

62%	MATCHING BLOCK 92/191	W

x,y,x_incr,y_incr; int i=0; dx=x2-x1; dy=y2-y1; 34 x=x1; y=y1; if(abs(dx)&lt;abs(dy)) len=abs(dx); else len=abs(dy);

x_incr=dx/len; y_incr=dy/len;

33%	MATCHING BLOCK 91/191	W

putpixel(x+200,y+200,3); while(i&gt;len) { x=x+x_incr; y=y+y_incr; putpixel(x+200,y+200,3); i++; } flushall(); } void Drawing:: draw_breCir() { } int main() { int gd=DETECT,gm,ch; 35 Drawing draw; clrscr(); initgraph(&gd,&gm,"C:\\TC\\bgi");

do { cout&gt;&gt;"\nEnter Your Choice"; cin&lt;&lt;ch; switch(ch) { 1.9 Bresenham's Algorithm for Line Drawing and Circle Bresenham's line algorithm This is the Bresenham's line algorithm represented by a visual diagram explaining precision of the actual line vs. the pixels representation of the line, and the problem at hand. Take a look at this image. One thing to note here is that it is impossible to draw the true line

36 that we want because of the pixel spacing(in other words there's not enough precision for drawing true lines on a PC monitor especially when dealing with low resolutions). The Bresenham's line-drawing algorithm is based on drawing an approximation of the true line. The true line is indicated in bright color, and its approximation is indicated in black pixels. In this example the starting point of the line is located exactly at 0, 0 and the ending point of the line is located exactly at 9, 6. The way the algorithm works is this. First it decides which axis is the major axis and which is the minor axis. The major axis is longer than the minor axis. On this picture illustrated above the major axis is the X axis. Each iteration progresses the current value of the major axis(starting from the original position), by exactly one pixel. Then it decides which pixel on the minor axis is appropriate for the current pixel of the major axis. How can you approximate the right pixel on the minor axis that matches the pixel on the major axis? - that's what Bresenham's line-drawing algorithm is all about. And it does so by checking which pixel's center is closer to the true line. On the picture above it would be easy to identify these pixels by eye. I added vertical spans to let you grasp the idea better visually. Take a closer look. The center of each pixel is marked with a dot. The algorithm takes the coordinates of that dot and compares it to the true line. If the span from the center of the pixel to the true line is less or equal to 0.5, the pixel is drawn at that location. That span is more generally known as the error term. The whole algorithm is done in straight integer math with no multiplication or division in the main loops (no fixed point math either). How is it possible? Basically, during each iteration through the main drawing loop the error term is tossed around to identify the right pixel as close as possible to the true line. Let's consider these two deltas between the length and height of the line: dx = x1 - x0; dy = y1 – y0; This is a matter of precision and since we're working with integers you will need to scale the deltas by 2 generating two new values:
37 dx2 = dx*2; dy2 = dy*2; These are the values that will be used to change the error term. Why scale? The error term must be first initialized to 0.5 and that cannot be done using an integer. To confuse you even further, finally the scaled values must be substracted by either dx or dy(the original, unscaled delta values) depending on what the major axis is (either x or y). We want to draw a line from (0,0) to (x1,y1), where 0&gt;=y1&gt;=x1, by setting one pixel per column. For example, if x=10, y=7, we get this: Here are several versions, ending with Bresenham's algorithm. The point of this is to use the simplest possible operations. 1. Simple program: m=y1/x1; pixel(0,0); for(x=1;x&gt;=x1;x++) { y=round(m*x); pixel(x,y);} 2. Track the deviation of the current y from the perfect line. When the deviation gets too large, increment y, and decrement the deviation. m=y1/x1;
38 d=0; pixel(0,0); y=0; for(x=1;x&gt;=x1;x++) { d+= m; if (d&lt;= 1/2) { d -= 1; y++; } pixel(x,y); } 3. Scale up m and d by 2x to convert fractions to integers. This is ok since they are used only in the if; and it still branches at the same times as before. m=2*y1; d=0; pixel(0,0); y=0; for(
x=1;

27%	MATCHING BLOCK 94/191	SA	Graphics Examination 190232.pdf (D125664974)

x&gt;=x1;x++) { d+= m; if (d&lt;=x1) { d-= 2*x1; y++;} 39 pixel(x,y); } 4. Shift d by -x1 to remove a subtraction. m=2*y1; d= -x1; pixel(0,0); y=0; for(x=1;x&gt;=x1;x++) { d+= m; if (d&lt;=0) { d-= 2*x1; y++;}

pixel(x,y); } 5.
Assume that the pixels are stored in a frame buffer, and that pixel x y is stored at location fb+x*maxy+y where fb is the address of the start of the frame buffer, and maxy is the max value of y in the framebuffer, i.e., 0&gt;= y&gt;= y1&gt;= maxy. Then we get: m=2*y1; d= -x1; *fb=1; y=0; for(x=1;x&gt;=x1;x++)
40 { d+= m; if (d&lt;=0) { d-= 2*x1; y++;} *(fb+x*maxy+y)=1; } 6. Letting p be the current address, we can optimize. m=2*y1; d= -x1; p=fb; *p=1; for(x=1;x&gt;=x1;x++) { d+= m; p+=maxy; if (d&lt;=0) { d-= 2*x1; p++;} *p=1; } 7. The next problem is that modern computers use a pipelined architecture, where different stages of several instructions are processed in parallel. However, when there is a conditional, the computer can't tell what the next instructions are, and so is forced to flush
41 the pipeline. Therefore we want to eliminate the conditional in the inner loop. The following assumes that the right shift is arithmetic, which it is on the Sun. Mask is all ones when d&lt;=0, all zeroes otherwise. We also pulled 2*x1 out of the loop. m=2*y1; d= -x1; p=fb; *p=1; doublex1= x1&gt;1; for(x=1;x&gt;=x1;x++) { d+= m; p+=maxy; mask = (~(d&lt;&lt;31)); d -= (doublex1&mask); p += (1&mask); *p=1; } 8. We can also offset x by x1 so that the termination condition is a test against 0. Bresenham Circle Algorithm Here are several ways to draw a circle, ending with the Bresenham circle algorithm.
42 Taylor series for sqrt(1-x 2), in the 2nd octant: 0&gt;= x&gt;= y, where the curve doesn't get too vertical. The following is expanded about the origin. The error at x=.71 is 0.0003. y(x) = 1 - (1/2) x 2 - (1/8) x 4 - (1/16) x 6 - (5/128) x 8 - (7/256) x 10 - (21/1024) x 12 + ... This is popular but usually the wrong way. 1. If you must use Taylor, at least expand about the center of the interval: z = x- 0.3536 y(z) = (0.9353967287- 0.3780214203 z- 0.6109173568 z 2 - 0.2468897312 z 3 - 0.2992736736 z 4 - 0.2821915805 z 5 - 0.3420828869 z 6 - 0.4015387039 z 7 - 0.5080487872 z 8 - 0.6481416197 z 9 - 0.8517407025 z 10 - 1.133286086 z 11 - 1.531581071 z 12 - 2.091410948 z 13 +... The error at x=.71 is 0.000 004. This is better, but there are more nonzero coefficients, which means that it is slower. 2. A Chebyshev series, which tries to minimize the maximum error over an interval beats a Taylor series: y(x) = + 1.000000093 - 0.00002511816061 x - 0.4988901592 x 2 - 0.01882886918 x 3 + 0.03498610076 x 4 - 0.7655182976 x 5 + 2.101697099 x 6 - 3.603814683 x 7 + 3.259950635 x 8 - 1.315059748 x 9

43 This has a max error of 1.5*10 -7 , which is much better than above, expecially as there are only 8 coefficients. We could get as good as either Taylor with many fewer terms than they. 3. Rotate the point (R,0) by Dt repeatedly (t is the angle). What should Dt be? 4. Approx the rotation matrix: cos t sin t - sin t cos t approximately = 1 t t 1 with everything multiplied by R. But this circle may grow since the determinant &lt;1. 5. Improve the approx matrix so the determinant =1: 1 t t 1-t 2 6. Binomial theorem on sqrt(1-x 2). Does this do the same as Taylor expanded about 0? 7. Here are two parametric ways to draw circles. 1. x = R cos t y = R sin t What should Dt be? 2. x = R (2t)/(t 2 +1} y = R (t 2 -1)/(t 2 +1) ; -infinity&gt;t&gt;infinity Unfortunately, the speed of the curve wrt t varies with t, so the stepsize must also.

44 3. There is no parametric polynomial, of any degree, that exactly represents a circle. However you can get good approximations. 8. ???? (You suggest) 9. Now let's optimize C code. Do the second octant only. pixel(0,r);

83%	MATCHING BLOCK 96/191	SA	Computer Graphics.pdf (D165747830)

for(x=1;x&gt;r/sqrt(2);x++) { y=sqrt(r 2 -x 2); pixel(x,

y); } 10. Assume that the last point was (x-1,y). Find whether (x,y-½) is above or below the circle, and move SE to (x,y-1) or E to (x,y), respectively. y=r; pixel(0,r); for(x=1;x&gt;r/sqrt(2);x++) { d=x 2 +(y-1/2) 2 -r 2 ; if (d&lt;=0) y--; pixel(x,y); } 11. Optimize: d=x 2 +(y-1/2) 2 -r 2 becomes d=x 2 +y 2 -y+1/4-r 2 . How does the d for a given x differ from d for x-1? If y is unchanged, then dx = dx-1+2x-1. If y decreases, add -2y+2, where that is the old y. Also the only time that 1/4 affects the test d&lt;0 is when d=1/4, so ignore it, but make the test d&lt;=0. y=r;

45 d= -r; pixel(0,r); for(x=1;x&gt;r/sqrt(2);x++) { d+= 2x-1; if (d&lt;=0) { y--; d -= 2y; /* Must do this AFTER y-- */ } pixel(x,y); } This is Bresenham's circle algorithm. Note that you can store 2x-1 and -2y and update them by adding or subtracting 2. 12. Lo-level optimizations can also be done thus: y=r; d= -r; x2m1= -1; pixel(0,r); for(x=1;x&gt;r/sqrt(2);x++) { x2m1 += 2; d+= x2m1; if (d&lt;=0) { y--;

46 d -= (y&gt;&gt;1); /* Must do this AFTER y-- */ } pixel(x,y); } 13. You can also do the same pipelining techniques used for the line algorithm. 14. Finally, on an Intel processor, there are ways to use the available instructions efficiently. 1.10 Midpoint Methods for Line and Circle The Midpoint Line Algorithm Before launching into an explanation of the midpoint line algorithm, it's important to consider the properties that make up a "good" line. First of all, a line, in the theoretical sense, is an infinitely thin, straight connection between two infinitely small points. But a computer screen consists of thousands or millions of pixels with a specific size. Representing a line with a collection of fixed size pixels is an approximation. To achieve the best approximation possible, we should choose the pixels that lie as close to the true line as possible. The collection of pixels should also appear straight, that is, it should be accurate. Consistent brightness is also important. And since the theoretical line is thin, the rendered line should be thin as well - preferably one pixel thick. Of course there are cases where thick lines are desired, but they are out of the scope of this project. To achieve these goals we would like to illuminate one pixel per column in a uniform manner for all lines with slope -1 to 1 and one pixel per row for all other slopes. Finally, any line algorithm should be very fast since it is likely to be a heavily used part of any graphics system. In the field of computer graphics the discussion often arises as to whether the coordinate of a pixel lies at its center or one of its corners. Rather than debate this issue here, it is

47 assumed that the coordinate of the pixel is at its center. This assumption is clearly visible in the Zoom Panel of the What's In A Line? applet. The following description of the midpoint line algorithm applies to all lines with slope between 0 and 1. All other slopes can be handled by modifying the algorithm to reflect the line about the major or minor (y = x, y = -x) axes. In all, there are eight cases to be considered: [0,1], (1, oo], (0, -1], (-1, oo] where P1x &gt;= P2x, and [0,1], (1, oo], (0, -1], (-1, oo] where P1x &lt; P2x. What's In A Line? works for all these cases. However, code is provided for some of them while pseudo-code is provided for the remainder. As an exercise, try deriving the code for the remaining slope cases yourself. The midpoint line algorithm is widely used because it meets all the criteria of a good line algorithm; it has been proven to choose the pixels closest to the line (accuracy, consistency, straightness), it chooses only one pixel per column, and it is very fast. Its speed and compactness are probably why it is so popular. The midpoint line algorithm uses only integer arithmetic and does no multiplication or division in its main loop. The basis for the midpoint line algorithm is simple. Given the previous pixel selection P, there are two candidates for the next pixel closest to the line, E and NE (E stands for East and NE stands for Northeast). M is the midpoint between them. Decide on which side of M the line passes and choose the pixel which is closest to the line. If the M is above the line, choose E. If M is below the line, choose NE. This is obvious from the drawing below (The drawing is from Foley, p. 76. The grid lines are at pixel centers. The empty circles are background pixels. The black circle is a shaded pixel):

48 The midpoint line algorithm visits every column of pixels from the start point to the end point. Since the start point is on the line, it serves as the first "previous" pixel to prime the algorithm. The implicit equation of a line is: F(x, y) = ax + by + c For all points on the line, the solution to F(x, y) is 0. All points above F(x, y) result in a negative number and all points below F(x, y) result in a positive number. This relationship is used to determine the relative position of M. The decision variable, d, is derived from the implicit form of a line. The sign of d is used to make the midpoint determination for all remaining pixels. If d is positive, the midpoint is below the line and NE is chosen. If d is negative or zero, the midpoint is above the line and E is chosen. As the algorithm progresses from pixel to pixel, d is incremented with one of two precalculated values based on the E/NE decision. When using the line algorithm with lines of moderate slope, pay special attention at how the value of d oscillates back and forth between positive and negative. The Midpoint Circle Algorithm A circle has symmetric properties that can be exploited when rendering on a raster display. For circles centered at the origin, the four quadrants are mirror images of each other. However, if a single quadrant of a circle is scan converted in one dimension, pixel continuity goes away as the slope passes +/- 1. If the circle is divided into eights and pixels are mirrored about y = x and y = -x as well as the major axes, pixel continuity is maintained. The midpoint circle algorithm takes advantage of eight way symmetry by calculating just the pixels in the second octant (slope = [1, oo]) and mirroring into all the others. Circles that are not centered at the origin can be re-oriented with a simple translation. The strategy of the midpoint circle algorithm is the same as the midpoint line algorithm; given the last pixel, decide which pixel to visit next based on where the theoretical arc of the circle passes with respect to the midpoint between the two prospective pixels.
49 Scan conversion starts at x = 0 and visits each column in the positive x direction as long as the resulting y values continue to be greater than or equal to the current x value. When y &gt;= x, all the pixels in the quadrant have been rendered. The implicit function of a circle is: F(x, y) = x 2 + y 2 + R F(x, y) is 0 for all (x, y) on the circle.

61%	MATCHING BLOCK 93/191	W

The result is positive for all points outside the circle and negative for all points inside the circle.

This relationship is used to determine the relative location of M with respect the theoretical circle. The decision variable, d, is derived from the implicit form of a circle. At each pixel, d is used to choose between the E (East) and SE (Southeast) pixel. As with the midpoint line algorithm, the sign of d indicates whether the midpoint is inside (below) or outside (above) the circle. If d is positive or zero, the midpoint is outside the circle and SE is chosen. If d is negative, the midpoint is inside the circle and E is chosen. As the algorithm progresses from pixel to pixel, d is incremented based on the E/SE decision. In contrast to the line algorithm where the incremental values are precalculated, the values used to increment d in the circle algorithm are constantly changing. This is because the slope of the circumference is constantly changing. The incremental value of d is based on the new values of x and y. The midpoint circle algorithm uses only integer arithmetic and only works for circles who's
50 center is at the origin and radius is an integer value. Applying a simple translation transformation eliminates the center-at-origin restriction. However, the technique used to define circle primitives in the the Drawing Area can give rise to circles with non-integer radii. What's In A Line? rounds any non-integer radius before applying the circle algorithm. When rounding is significant, some of the pixels rendered in the Zoom Panel appear to be incorrect. To minimize the roundoff error, try to keep the radius of the circle in the Drawing Area completely horizontal or vertical. Roundoff error is introduced when the slope of the radius deviates from 0 or oo. Excercises 1. What are the different areas where Computer Graphics is applied? 2. Explain in brief the various techniques of Object Rendering. 3. Write a program of DDA line drawing algorithm. 4. Write a program to draw an ellipse using MidPoint Ellipse Algorithm. 5. A line has two end points at (10,10) and (20,30). Plot the intermediate points using DDA algorithm. 6. Mention the characteristics of the different types of Digitizers. 7. Differentiate between the beam-penetration method and the shadow-mask method. Recommended Readings: 1.

83%	MATCHING BLOCK 95/191	W

Computer Graphics, C Version, Second Edition by Donald Hearn, M. Pauline Baker 2.

Geometric Tools for Computer Graphics
by Philip Schneider, David H. Eberly 3. Computer Graphics, Multimedia and Animation (Google eBook) Pakhira Malay K.
51 Unit 2 Output Primitives
52 2.0 Objectives 2.1 Scan-Line Polygon Fill Algorithm: Steps taken to turn a set of vertices into a filled polygon 2.2 Methods of Inclusion of a Point in an Arbitrary Polygon: The Crossing Number (cn) method/ the "even-odd" test and the Winding Number (wn) method 2.3

100%	MATCHING BLOCK 98/191	SA	CG assaignment 1.rtf (D25685411)

Polygon Fill Method: Seed Filling: Boundary fill and Flood fill algorithm 2.4

Transformations in the 2D Plane: Translation, Rotation, Scaling 2.5 Reflecting a Point about a Line 2.6 Shearing in 2D 2.7 Homogeneous Coordinates in 2 Dimensions 2.0 Objectives • To learn the skills of filling a polygon • To apply the methods of inclusion of a point in a 2D planar polygon • To know the types of Polygon filling methods • To analyze the various 2D transformations • To list the steps of reflecting an arbitrary point about a line • To adopt the skills of transformation that

distorts the shape of an object along either or both the

axis • To system of coordinates used in projective geometry that have the advantage of representing coordinates of points, including points at infinity, using finite coordinates. 2.1.1 Scan-Line Polygon Fill Algorithm: Steps taken to turn a set of vertices into a filled polygon In order to fill a polygon, we do not want to have to determine the type of polygon that we are filling. The easiest way to avoid this situation is to use an algorithm that works for all
53 three types of polygons. Since both convex and concave polygons are subsets of the complex type, using an algorithm that will work for complex polygon filling should be sufficient for all three types. The

scan-line polygon fill algorithm, which employs the odd/even parity concept previously discussed, works for complex polygon filling.

Reminder:

The basic concept of the scan-line algorithm is to draw points from edges of odd parity to even parity on each scan-line.

What is a scan-line? A scan-line is a line of constant y value, i.e., y=c, where c lies within our drawing region, e.g., the window on our computer screen. The scan-line algorithm is outlined next. When filling a polygon, you will most likely just have a set of vertices, indicating the x and y Cartesian coordinates of each vertex of the polygon. The following steps should be taken to turn your set of vertices into a filled polygon. 1. Initializing All of the Edges: The first thing that needs to be done is determine how the polygon's vertices are related. The all_edges table will hold this information. Each adjacent set of vertices (the first and second, second and third, ..., last and first) defines an edge. For each edge, the following information needs to be kept in a table: ? The minimum y value of the two vertices. ? The maximum y value of the two vertices. ? The x value associated with the minimum y value. ? The slope of the edge. The slope of the edge can be calculated from the formula for a line: y = mx + b;
54 where m = slope, b = y-intercept, y0 = maximum y value, y1 = minimum y value, x0 = maximum x value, x1 = minimum x value The formula for the slope is as follows: m = (y0 - y1) / (x0 - x1). For example, the edge values may be kept as follows, where N is equal to the total number of edges - 1 and each index into the all edges array contains a pointer to the array of edge values. 2. Initializing the Global Edge Table: The global edge table will be used to keep track of the edges that are still needed to complete the polygon. Since we will fill the edges from bottom to top and left to right. To do this, the global edge table table should be inserted with edges grouped by increasing minimum y values. Edges with the same minimum y values are sorted on minimum x values as follows: ? Place the first edge with a slope that is not equal to zero in the global edge table.
55 ? If the slope of the edge is zero, do not add that edge to the global edge table. ? For every other edge, start at index 0 and increase the index to the global edge table once each time the current edge's y value is greater than that of the edge at the current index in the global edge table. Next, Increase the index to the global edge table once each time the current edge's x value is greater than and the y value is less than or equal to that of the edge at the current index in the global edge table. If the index, at any time, is equal to the number of edges currently in the global edge table, do not increase the index. Place the edge information for minimum y value, maximum y value, x value, and 1/m in the global edge table at the index. The global edge table should now contain all of the edge information necessary to fill the polygon in order of increasing minimum y and x values. 3. Initializing Parity The initial parity is even since no edges have been crossed yet. 4. Initializing the Scan-Line The initial scan-line is equal to the lowest y value for all of the global edges. Since the global edge table is sorted, the scan-line is the minimum y value of the first entry in this table. 5. Initializing the Active Edge Table The active edge table will be used to keep track of the edges that are intersected by the current scan-line. This should also contain ordered edges. This is initially set up as follows: Since the global edge table is ordered on minimum y and x values, search, in order, through the global edge table and, for each edge found having a minimum y value equal to the current scan-line, append the edge information for the maximum y value, x value, and 1/m to the active edge table. Do this until an edge is found with a minimum y value greater than

56 the scan line value. The active edge table will now contain ordered edges of those edges that are being filled as such: 6. Filling the Polygon Filling the polygon involves deciding whether or not to draw pixels, adding to and removing edges from the active edge table, and updating x values for the next scan-line. Starting with the initial scan-line, until the active edge table is empty, do the following: ? Draw all pixels from the x value of odd to the x value of even parity edge pairs. ? Increase the scan-line by 1. ? Remove any edges from the active edge table for which the maximum y value is equal to the scan_line. ? Update the x value for for each edge in the active edge table using the formula x1 = x0 + 1/m. (This is based on the line formula and the fact that the next scan-line equals the old scan-line plus one. ? Remove any edges from the global edge table for which the minimum y value is equal to the scan-line and place them in the active edge table. ? Reorder the edges in the active edge table according to increasing x value. This is done in case edges have crossed.

57 2.2 Methods of Inclusion of a Point in an Arbitrary Polygon: The Crossing Number (cn) method/ the "even-odd" test and the Winding Number (wn) method Determining the inclusion of a point P in a 2D planar polygon is a geometric problem that results in interesting algorithms. Two commonly used methods are: 1. The Crossing Number (cn) method - which counts the number of times a ray starting from the point P crosses the polygon boundary edges. The point is outside when this "crossing number" is even; otherwise, when it is odd, the point is inside. This method is sometimes referred to as the "even-odd" test. The Winding Number (wn) method - which counts the number of times the polygon winds around the point P. The point is outside only when this "winding number" wn = 0; otherwise, the point is inside. If a polygon is simple (i.e., it has no self intersections), then both methods give the same result for all points. But for non-simple polygons, the two methods can give different answers for some points. For example, when a polygon overlaps with itself, then points in the region of overlap are found to be outside using the crossing number, but are inside using the winding number, as shown in the diagrams: Crossing Number Method

58 Winding Number Method Vertices are numbered: 0 1 2 3 4 5 6 7 8 9 In this example, points inside the overlap region have wn = 2, implying that they are inside the polygon twice. Clearly, the winding number gives a better intuitive answer than the crossing number does. Despite this, the crossing number method is more commonly used since cn is erroneously thought to be significantly (up to 20 times!) more efficient to compute than wn [O'Rourke, 1998]. But this is not the case, and wn can be computed with the same efficiency as cn by counting signed crossings. In fact, our implementation for wn_PnPoly() is faster than the code for cn given by [Franklin, 2000], [Haines, 1994] or [O'Rourke, 1998], although the cn "PointInPolygon()" routine of [Moller & Haines, 1999] is comparable to our wn algorithm. But the bottom line is that for both geometric correctness and efficiency reasons, the wn algorithm should always be preferred The Crossing Number for determining the inclusion of a point in a polygon. This method counts the number of times a ray starting from a point P crosses a polygon boundary edge separating it's inside and outside. If this number is even, then the point is outside; otherwise, when the crossing number is odd, the point is inside. This is easy to understand intuitively. Each time the ray crosses a polygon edge, its in-out parity changes (since a boundary always separates inside from outside, right?). Eventually, any ray must

59 end up beyond and outside the bounded polygon. So, if the point is inside, the sequence of crossings "&lt;" must be: in &lt; out &lt; ... &lt; in &lt; out, and there are an odd number of them. Similarly, if the point is outside, there are an even number of crossings in the sequence: out &lt; in &lt; ... &lt; in &lt; out. In implementing an algorithm for the cn method, one must insure that only crossings that change the in-out parity are counted. In particular, special cases where the ray passes through a vertex must be handled properly. These include the following types of ray crossings: Further, one must decide whether a point on the polygon's boundary is inside or outside. A standard convention is to say that a point on a left or bottom edge is inside, and a point on a right or top edge is outside. This way, if two distinct polygons share a common boundary segment, then a point on that segment will be in one polygon or the other, but not both at

60 the same time. This avoids a number of problems that might occur, especially in computer graphics displays. A straightforward "crossing number" algorithm selects a horizontal ray extending to the right of P and parallel to the positive x-axis. Using this specific ray, it is easy to compute the intersection of a polygon edge with it. It is even easier to determine when no such intersection is possible. To count the total crossings, cn, the algorithm simply loops through all edges of the polygon, tests each for a crossing, and increments cn when one occurs. Additionally, the crossing tests must handle the special cases and points on an edge. This is accomplished by the Edge Crossing Rules 1. An upward edge includes its starting endpoint, and excludes its final endpoint; 2. A downward edge excludes its starting endpoint, and includes its final endpoint; 3. Horizontal edges are excluded 4. The edge-ray intersection point must be strictly right of the point P. One can apply these rules to the preceding special cases, and see that they correctly determine valid crossings. Note that Rule #4 results in points on a right-side boundary edge being outside, and ones on a left-side edge being inside. Pseudo-Code: Crossing # Inclusion Code for this algorithm is well-known, and the edge crossing rules are easily expressed. For a polygon represented as an array V[n+1] of vertex points with , popular implementation logic ([Franklin, 2000], [O'Rourke, 1998]) is as follows:

61 typedef struct {int x, y;} Point; cn_PnPoly(Point P, Point V[], int n) { int cn = 0; // the crossing number counter // loop through all edges of the polygon for (each edge E[i]:V[i]V[i+1] of the polygon) { if (E[i] crosses upward ala Rule #1 || E[i] crosses downward ala Rule #2) { if (P.x &gt; x_intersect of E[i] with y=P.y) // Rule #4 ++cn; // a valid crossing to the right of P.x } } return (cn&1); // 0 if even (out), and 1 if odd (in) } Note that the tests for upward and downward crossing satisfying Rules #1 and #2 also exclude horizontal edges (Rule #3). All-in-all, a lot of work is done by just a few tests which make this an elegant algorithm. However, the validity of the crossing number method is based on the "Jordan Curve Theorem" which says that a simple closed curve divides the 2D plane into exactly 2 connected components: a bounded "inside" one and an unbounded "outside" one. The catch is that the curve must be simple (without self intersections), otherwise there can be more than 2 components and then there is no guarantee that crossing a boundary changes the in- out parity. For a closed (and thus bounded) curve, there is exactly one unbounded "outside" component; but bounded components can be either inside or outside. And two of

62 the bounded components that have a shared boundary may both be inside, and crossing over their shared boundary would not change the in-out parity. The Winding Number On the other hand, the winding number accurately determines if a point is inside a nonsimple closed polygon. It does this by computing how many times the polygon winds around the point. A point is outside only when the polygon doesn't wind around the point at all which is when the winding number wn = 0. More generally, one can define the winding number of any closed continuous curve C around a point P in the 2D plane. Let the continuous 2D curve C be defined by the points , for with . And let P be a point not on C. Then, define the vector c(P,u) = C(u) − P from P to C(u), and the unit vector w(P,u) = c(P,u) / |c(P,u)| which gives a continuous function mapping the point C(u) on C to the point w(P,u) on the unit circle . This map can be represented in polar coordinates as where is a positive counter clockwise angle in radians. The winding number is then equal to the integer number of times that W(P) wraps C around S1. This corresponds to a homotopy class of S1, and can be computed by the integral: When the curve C is a polygon with vertices V0,V1,...,Vn = V0, this integral reduces to the sum of the (signed) angles that each edge ViVi+1 subtends with the point P. So, if , we have:
63 . This formula is clearly not very efficient since it uses a computationally expensive arccos() trig function. But, a simple observation lets us replace this formula by a more efficient one. Pick any point Q on S1. Then, as the curve W(P) wraps around S1, it passes Q a certain number of times. If we count (+1) when it passes Q counterclockwise, and (−1) when it passes clockwise, then the accumulated sum is exactly the total number of times that W(P) wraps around S1, and is equal to the winding number . Further, if we take an infinite ray R starting at P and extending in the direction of the vector Q, then intersections where R crosses the curve C correspond to the points where W(P) passes Q. To do the math, we have to distinguish between positive and negative crossings where C crosses R from right-to-left or left-to-right. This can be determined by the sign of the dot product between a normal vector to C and the direction vector q = Q [Foley et al, 1996, p-965], and when the curve C is a polygon, one just needs to make this determination once for each edge. For a horizontal ray R from P, testing whether an edge's endpoints are above and
64 below the ray suffices. If the edge crosses the positive ray from below to above, the crossing is positive (+1); but if it crosses from above to below, the crossing is negative (−1). One then simply adds all crossing values to get . For example: Additionally, one can avoid computing the actual edge-ray intersection point by using the is Left() attribute; however, it needs to be applied differently for ascending and descending edges. If an upward edge crosses the ray to the right of P, then P is on the left side of the edge since the triangle ViVi+1P is oriented counter clockwise. On the other hand, a downward edge crossing the positive ray would have P on the right side since the triangle ViVi+1P would then be oriented clockwise.
65 Pseudo-Code: Winding Number Inclusion This results in the following wn algorithm which is an adaptation of the cn algorithm and uses the same edge crossing rules as before to handle special cases. typedef struct {int x, y;} Point; wn_PnPoly(Point P, Point V[], int n) { int wn = 0; // the winding number counter // loop through all edges of the polygon for (each edge E[i]:V[i]V[i+1] of the polygon) { if (E[i] crosses upward ala Rule #1) { if (P is strictly left of E[i]) // Rule #4 ++wn; // a valid up intersect right of P.x } else if (E[i] crosses downward ala Rule #2) { if (P is strictly right of E[i]) // Rule #4 --wn; // a valid down intersect right of P.x } } return wn; // =0 &gt;=&lt; P is outside the polygon } Clearly, this winding number algorithm has the same efficiency as the analogous crossing number algorithm. Thus, since it is more accurate in general, the winding number
66 algorithm should always be the preferred method to determine inclusion of a point in an arbitrary polygon. The wn algorithm's efficiency can be improved further by rearranging the crossing comparison tests. This is shown in the detailed implementation of wn_PnPoly() given below. In that code, all edges that are totally above or totally below P get rejected after only two (2) inequality tests. However, currently popular implementations of the cn algorithm ([Franklin, 2000], [Haines, 1994], [O'Rourke, 1998]) use at least
2.3

100%	MATCHING BLOCK 101/191	SA	CG assaignment 1.rtf (D25685411)
Polygon Fill Method: Seed Filling: Boundary fill and Flood fill algorithm			

three (3) inequality tests for each rejected edge. Since most of the edges in a large polygon get rejected in practical applications, there is about a 33% (or more) reduction in the number of comparisons done. In runtime tests using very large (1,000,000 edge) random polygons (with edge length &gt; 1/10 the polygon diameter) and 1000 random test points (inside the polygon's bounding box), we measured a 20% increase in efficiency overall.

100%	MATCHING BLOCK 102/191	SA	CG assaignment 1.rtf (D25685411)
Seed Filling: In this method a particular seed point is picked and we start filling upwards and downwards pixels until boundary is reached. Seed fill method is of two types: Boundary fill and Flood fill.			

Boundary fill algorithm: In Boundary filling a seed point is fixed, and then neighboring pixels are checked to match with the boundary color. Then, color filling is done until boundary is reached. A region may be 4 connected or 8 connected:
67

100%	MATCHING BLOCK 103/191	SA	CG assaignment 1.rtf (D25685411)

Procedure for filling a 4- connected region: Color is specified by parameter fill color (f- color) and boundary color is specified by boundary color (b-color). getpixel() function gives the color of specified pixel and putpixel() fills the pixel with particular color. boundary_ fill (x,y, f_color, b_color) { If (getpixel (x,y) != b_color && getpixel (x,y) != f_color) putpixel(x,y,f_color) boundary_fill(x+1, y, f_color, b_color); boundary_fill(x, y+1, f_color, b_color); boundary_fill(x-1, y, f_color, b_color); boundary_fill(x, y-1, f_color, b_color); } } Flood fill algorithm: There are some cases where the boundary color is different than the fill color. For situations like these Flood fill algorithm is used. Here the process is started in a similar way by examining the colors of neighboring pixels. But instead of matching it with a boundary color a specified color is matched. Procedure for filling a 8- connected region: flood_ fill (x,y, old_color, new_color) { putpixel(x,y,new_color) 68 flood_ fill (x+1, y, old_color, new_color) flood_ fill (x-1, y, old_color, new_color) flood_ fill (x, y+1, old_color, new_color) flood_ fill (x, y-1, old_color, new_color) flood_ fill (x+1, y+1, old_color, new_color) flood_ fill (x-1, y-1, old_color, new_color) flood_ fill (x+1, y-1, old_color, new_color) flood_ fill (x-1, y+1, old_color, new_color) } }

Disadvantages of Seed Fill Algorithm: 1) If an inside pixel is in some other color then the fill terminates and the polygon remains unfilled. 2) Seed fill method does not work for large polygons. 2.4 Transformations in the 2D Plane: Translation, Rotation, Scaling, Shearing Many pictures, from 2D drawings to projected views of 3D objects, consist of graphical primitives such as points, lines, circles, and filled polygons. These picture components are often defined in a continuous space at a higher level of abstraction than individual pixel in the discrete image space. For instance, a line is defined by its two endpoints and the line equation, whereas a circle is defined by its radius, center position, and the circle equation. It is the responsibility of the graphics system of the application program to convert each primitive from its geometric definition into a set of pixels that make up the primitive in the image space. This conversion task is generally referred to as scan conversion or
69 rasterization. With the procedures for displaying output primitives and their attributes, we can create variety of pictures and graphs. In many applications, there is also a need for altering or manipulating displays. Design applications and facility layouts are created

96%	MATCHING BLOCK 104/191	SA	INF_2046.pdf (D164968115)

by arranging the orientations and sizes of the component parts of the scene.

And animations are

61%	MATCHING BLOCK 108/191	SA	Computer Graphics.pdf (D165747830)

produced by moving the "camera" or the objects in a scene along animation paths. Changes in orientation, size, and shape are accomplished with geometric transformations that alter the coordinate descriptions of objects. The basic geometric transformations are translation,

rotation, and scaling. Other transformations that are often applied to objects include reflection and shear. We first discuss methods for performing geometric transformations and then consider how transformation functions can be incorporated into graphics packages. Transformations are a fundamental part of computer graphics. Transformations are used to position objects, to shape objects, to change viewing positions, and even to change how something is viewed (e.g. the type of perspective that is used). Translation Translation is nothing more than moving an object or point from one position to another. Let's assume that you have a single point, (x,y), that you want to translate by some amount, (dx,dy). The process of translation is shown in figure below. Translating a single point.
70 Basically, you add the translation factors to the (x,y) coordinate, come up with a new position, and call it (xt,yt). Here's the math: xt = x + dx; yt = y + dy; Here, dx,dy can be either positive or negative. If we're talking about translation with standard display coordinates, where (0, 0) is in the upper-left corner, positive translations in the x-axis move an object to the right. Positive translations in the y-axis move an object down. Negative translations in the x-axis move an object to the left. And finally, negative translations in the y-axis move an object up. To translate an entire object, you simply apply the translation transformation to the center of the object, if the object has an x,y position that all points are relative to (as does the polygon structure). If the object doesn't have a general x,y position, you must apply the formula to every point that makes up the polygon. This brings us to the concept of local and world coordinates. Generally, in 2D/3D computer graphics you want to define all objects to have at least local and world coordinates. The local coordinates of an object are those relative to (0,0) or (0,0,0) in 3D. Then the world coordinates are found by adding the world position (x0,y0) to each of the local coordinates, in essence translating the local coordinates each by (x0,y0) to place the object. This is shown in the next figure.

71 Translation of an object and the resulting vertices. In light of this, in the future you may decide to add more data storage to your polygon so that you can store both local and world coordinates. In fact, you will do this later. Moreover, you'll add storage for camera coordinates. The reasoning for the added storage is that once you transform an object to world coordinates and are ready to draw it, you don't want to have to do it each frame. As long as the object doesn't move or transform, you won't have to. You can just save the last calculated world coordinates. With that all in your little neural net, let's take a look at a general polygon translation function. It's so simple, it should be illegal &gt;BG&lt;. int Translate_Polygon2D(POLYGON2D_PTR poly, int dx, int dy) { // this function translates the center of a polygon // test for valid pointer if (!poly) return(0); // translate poly-&lt;x0+=dx; poly-&lt;y0+=dy;
72 // return success return(1); } // end Translate_Polygon2D Rotation Rotation of bitmaps is rather complex, but rotation of single points in the plane is almost trivial—or at least the actual rotation is. Deriving the rotation is a bit more complex. Referring to the figure below, you can see the point p0 with coordinates (x,y). You want to rotate this point an angle theta about the z-axis (which is running through the paper) to find the rotated coordinates p01 with coordinates (xr,yr). The rotation of a point. Trigonometry Review Obviously, you're going to need to use some trigonometry here. Most trigonometry is based on the analysis of a right triangle, as shown in figure below.
73 The right triangle. Table below shows the differences between radians and degrees. Radians Versus Degrees Degrees Radians (pi) Radians (Numerically) 360 2*pi 6.28 180 1*pi 3.14159 90 pi/2 1.57 57.295 pi/pi 1.0 1 pi/180 0.0175
74 Here are some trigonometric facts: • There are 360 degrees in a complete circle, or 2*pi radians. Hence, there are pi radians in 180 degrees. The computer functions sin() and cos() work in radians, not degrees. • The sum of the interior angles theta1 + theta2 + theta3 = 180 degrees or pi radians. • Referring to the right triangle in Figure 8.20, the side opposite theta1 is called the opposite side, the side below it is called the adjacent side, and the long side is called the hypotenuse. • The sum of the squares of the sides of a right triangle equal the square of the hypotenuse. This is called the Pythagorean theorem. Mathematically, you write it like this: hypotenuse2 = adjacent2 + opposite2 Or sometimes you can use a, b, and c for dummy variables: c2 = a2 + b2 Therefore, if you have two sides of a triangle, you can find the third. • There are three main trigonometric ratios that mathematicians like to use, called the sine, cosine, and tangent. They're defined as adjacent side cos(theta) = ---------------- = ------- hypotenuse r Domain: 0 &gt;= theta &gt;= 2*pi Range: -1 to 1 opposite side y sin(theta) = ---------------- = ------- hypotenuse r Domain: 0 &gt;= theta &gt;= 2*pi Range: -1 to 1
75 sin(theta) opposite/hypotenuse tan(theta) = -------------- = ---------------------- cos(theta) adjacent/hypotenuse opposite y = ----------- = --- = slope = M adjacent x Domain: -pi/2 &gt;= theta &gt;= pi/2 Range: -infinity to +infinity MATH Note the use of the terms domain and range. These simply mean the input and the output, respectively. Figure below shows graphs of all the functions. Notice that all the functions are periodic (repeating) and that sin(theta) and cos(theta) have periodicity of 2*pi, while the tangent has periodicity of pi. Also, notice that tan(theta) goes to +-infinity whenever theta mod pi is pi/2. Graphs of basic trigonometric functions.
76 Now, there are about a bazillion trigonometric identities and tricks, and it would take a math book to prove them all. I'm just going to give you a table of the ones that a game programmer should know. Table below lists some other trigonometric ratios, as well as some neat identities. Useful Trigonometric Identities Cosecant csc(theta) = 1/sin(theta) Secant sec(theta) = 1/cos(theta) Cotangent cot(theta) = 1/tan(theta) Pythagorean theorem in terms of trig functions: sin(theta)2 + cos(theta)2 = 1 Conversion identity: sin(theta1) = cos(theta1 – PI/2) Reflection identities: sin(-theta) = -sin(theta) cos(-theta) = cos(theta) Addition identities: sin(theta1 + theta2) = sin(theta1)*cos(theta2) + cos(theta1)*sin(theta2) cos(theta1 + theta2) = cos(theta1)*cos(theta2) - sin(theta1)*sin(theta2) sin(theta1 - theta2) = sin(theta1)*cos(theta2) - cos(theta1)*sin(theta2) cos(theta1 - theta2) = cos(theta1)*cos(theta2) + sin(theta1)*sin(theta2) Of course, you could derive identities until you turned many shades of blue. In general, identities help you simplify complex trigonometric formulas so you don't have to do the math. Hence, when you come up with an algorithm based on sin, cos, tan, and so on, always take a look in a trigonometry book and see if you can simplify your math so that fewer computations are needed to get the result. Remember, speed, speed, speed!!! Rotating a Point in a 2D Plane
77 Now that you have an idea of what sin, cos, and tan are, let's use them to rotate a point in a 2D plane. Take a look at the figure below, which shows the setup for the rotation formulas. Derivation of the rotation equations. Start off by showing that any point on a circle of radius R is computed a xr = r*cos(theta) yr = r*sin(theta) Hence, if you always wanted to rotate a point that had coordinates (x,0), you could use this equation. However, you need to generalize a little. You want to rotate a point (x,y) about an angle theta, as shown in the figure above. You can think of this in two ways: as rotating the point P, or as rotating the axes themselves. If you think of it as rotating the axes themselves, you have two coordinate systems: one before the rotation, and one after. In the coordinate system before the rotation, you have xr = r*cos(theta) yr = r*sin(theta) But after the rotation, you have Equations 1: xr = r*cos(theta + alpha) yr = r*sin(theta + alpha) Equations 2: x = r*cos(alpha) y = r*sin(alpha)
78 Here, alpha is basically the angle created from the x-axis of the new system and the position vector from the origin to P. If you're confused, let me explain what you're doing here in another way. You always know how to find the point (x,0) rotated about theta, and if you rotate the axes by theta, you can compute P in both the old axes and the new. Then, based on these two formulas, you can come up with the rotation equations. If you take Equations 1 and use the addition identities, you'll get the following results. Equations 3: xr = r*cos(theta)*sin(alpha) – r*sin(theta)*sin(alpha) yr = r*sin(theta)*cos(alpha) + r*sin(theta)*cos(alpha) Wait a minute, let me put some boom in it. You know that x,y are also equal to x = r*cos(alpha) y = r*sin(alpha) Substituting these values into the bolded parts of Equations 3 gives you the results you desire. Equations 4, the rotation formulas: xr = x*cos(theta) – y*sin(theta) yr = x*sin(theta) + y*cos(theta) Q.E.D. MATH For you math-heads, this derivation is very similar to a polar-only rotation with conversion to Cartesian coordinates at the end. That's how I came up with it. Back to reality: You now know that to rotate a point (x,y) by an angle theta, you can use Equations 4. However, there is one detail to remember: The equations rotate a point in the

79 counter clockwise direction for positive theta and in the clockwise direction for negative theta. However, there is one more problem... you did the derivation for quadrant I of the normal Cartesian coordinate system. Thus, the y-axis is inverted on the display screen and the roles of positive and negative are reversed. Later, when you do 3D graphics, you'll transform all screen coordinates so that the x,y-axes are centered in the middle and are both pointing in the positive directions, just like quadrant I of the 2D Cartesian system. But for now, who cares? Rotating a Polygon Taking all of your immense knowledge, let's write a rotation function that rotates a polygon: int Rotate_Polygon2D(POLYGON2D_PTR poly, float theta) { // this function rotates the local coordinates of the polygon // test for valid pointer if (!poly) return(0); // loop and rotate each point, very crude, no lookup!!! for (int curr_vert = 0; curr_vert &gt; poly-&lt;num_verts; curr_vert++) { // perform rotation float xr = poly-&lt;vlist[curr_vert].x*cos(theta) –
80 poly-&lt;vlist[curr_vert].y*sin(theta); float yr = poly-&lt;vlist[curr_vert].x*sin(theta) + poly-&lt;vlist[curr_vert].y*cos(theta); // store result back poly-&lt;vlist[curr_vert].x = xr; poly-&lt;vlist[curr_vert].y = yr; } // end for curr_vert // return success return(1); } // end Rotate_Polygon2D There are a few things you should note. First, the math is performed in floating-point and then the results are stored as integers, so there's loss of precision. Next, the function takes the angle in radians rather than degrees because the function uses the math libraries' sin() and cos() functions, which use radians. The loss of accuracy isn't a huge problem, but the use of trig functions in a real-time program is just about as ugly as it gets. What you need to do is create a lookup table that has the sine and cosine values for, say, 0–360 degrees already precomputed, and then replace the library function calls to sin() and cos() with table lookups. The question is, how do you design the table? This really depends on the situation. Some programmers might want to use a single BYTE to index into the table. Thus, it would have 256 virtual degrees that made up a circle, as shown in the figure below. Breaking a circle into 256 virtual degrees
81 It's up to you and the situation, but usually I like to make the table hold the angles from 0– 359 degrees. Here's how you might create such tables: // storage for our tables float cos_look[360]; float sin_look[360]; // generate the tables for (int ang = 0; ang &gt; 360; ang++) { // convert ang to radians
82 float theta = (float)ang*3.14159/180; // insert next entry into table cos_look[ang] = cos(theta); sin_look[ang] = sin(theta); } // end for ang Then we can rewrite our rotation function to take an angle from 0–359 and use the tables by replacing the sin() and cos() with sin_look[] and cos_look[], respectively: int Rotate_Polygon2D(POLYGON2D_PTR poly, int theta) { // this function rotates the local coordinates of the polygon // test for valid pointer if (!poly) return(0); // loop and rotate each point, very crude, no lookup!!! for (int curr_vert = 0; curr_vert &gt; poly-&lt;num_verts; curr_vert++) { // perform rotation float xr = poly-&lt;vlist[curr_vert].x*cos_look[theta] – poly-&lt;vlist[curr_vert].y*sin_look[theta]; float yr = poly-&lt;vlist[curr_vert].x*sin_look[theta] + poly-&lt;vlist[curr_vert].y*cos_look[theta];
83 // store result back poly-&lt;vlist[curr_vert].x = xr; poly-&lt;vlist[curr_vert].y = yr; } // end for curr_vert // return success return(1); } // end Rotate_Polygon2D To rotate a POLYGON2D object 10 degrees, you would make the call Rotate_Polygon2D(&object, 10); NOTE Note that all this rotation stuff mangles the original polygon's coordinates. Sure, if you rotate 10 degrees, you can then rotate –10 degrees to get back to the original vertices, but you will slowly lose your original vertex coordinates due to integer truncation and rounding. This is the reason for having a second set of coordinates stored in the polygon structure. It can hold transformations, and you always keep the originals too, so you can refresh your data if need be. Scaling After all that, anything should be easy. Scaling is almost as simple as translation. Take a look at the figure below. All you need to do to scale an object is multiply each coordinate by the scaling factor.
84 The math of scaling. Scaling factors that are greater than 1.0 will make the object bigger, while scaling factors less that 1.0 will make the object smaller. A scaling factor of 1.0 will do nothing. The math to scale a point (x,y) by scaling factor s, resulting in (xs,ys), is xs = s*x ys = s*y Also, you can scale each axis non-uniformly—that is, with different scaling factors for the x and y coordinates, like this: xs = sx*x ys = sy*y Most of the time, you want to scale equally. But you might want to make an object grow in one axis, so who knows? Here's a function that scales a polygon. It takes both an x- and a y- axis scaling factor, just in case:
85 int Scale_Polygon2D(POLYGON2D_PTR poly, float sx, float sy) { // this function scales the local coordinates of the polygon // test for valid pointer if (!poly) return(0); // loop and scale each point for (int curr_vert = 0; curr_vert &gt; poly-&lt;num_verts; curr_vert++) { // scale and store result back poly-&lt;vlist[curr_vert].x *= sx; poly-&lt;vlist[curr_vert].y *= sy; } // end for curr_vert // return success return(1); } // end Scale_Polygon2D To scale a polygon to 1/10 its size, you would make the call Scale_Polygon2D(&polygon, 0.1, 0.1);
86 Notice that the x- and y-axis scale factors are both equal to 0.1. Thus, the scaling will be uniform on each axis. 2.1.5 Reflecting a Point about a Line Suppose you have a line drawn on a piece of paper. On this same piece of paper, you have a point. For the moment, suppose that the point does not lie on the line. To 'reflect the point about the line' means, roughly, that you want the point that is the same distance from the line, but on the other side of the line.
This idea is illustrated below: reflecting about an arbitrary line: the distance from P to Q is the same as the distance from P' to Q. (the coordinates of the reflected point are not so easy to find) reflecting about the x- axis (the x-value stays the same; take the opposite of the y-value) reflecting about the y- axis (the y-value stays the same; take the opposite of the x-value) Notice that if you fold the piece of paper along the line of reflection, then the original point and its reflection will land right on top of each other. It's kind of like reflecting in a mirror—except instead of the mirror reflecting 'back at you'; it instead projects to the other
87 side. If a point actually lies on the line that you're reflecting about, then the reflection of the point is itself. Reflection about any line y=mx+c can be accomplished with a combination of translate- rotate-reflect transformations. Steps are: 1. Translate the working coordinate system (WCS)

so that the line passes through the origin. 2. Rotate the

WCS such that one of the coordinate axis lies onto the line. 3. Reflect about the aligned axis. 4. Restore the WCS back by using the inverse rotation and translation transformation. EXAMPLES: Question: Start at the origin. Move to the right 2, and down 4. At what point are you? Solution: (2,−4) Why? (0,0) → right 2 (0+2,0)=(2,0) → down 4 (2,0−4)=(2,−4) Question: Start at the point (1,−3). Move to the left 4, and up 2. At what point are you? Solution: (−3,−1) Why? (1,−3) → left 4 (1−4,−3)=(−3,−3) → up 2 (−3,−3+2)=(−3,−1) Question: Start at the point (−2,5). Stay in the same quadrant, but double your distance from the x-axis,
88 and triple your distance from the y-axis. At what point are you? Solution: (−6,10) Why? The up/down info (the y-value) gives distance from the x-axis. The left/right info (the x-value) gives distance from the y-axis. Thus: (−2,5) original point: currently 5 units from the x-axis (above) and 2 units from the y-axis (to the left) (−2,5·2)=(−2,10) double distance from x-axis; stay above (−2·3,10)=(−6,10) triple distance from y-axis, stay to the left Question: Start at the point (−3,7). Reflect about the x-axis. At what point are you? Solution: (−3,−7) To reflect about the x-axis, the x-value stays the same, and you take the opposite of the y-value. Question: Start at the point (−3,7). Reflect about the y-axis. At what point are you? Solution: (3,7) To reflect about the y-axis, the y-value stays the same, and you take the opposite of the x-value.
89 Question: Start at the point (−3,7). Reflect about the x-axis, and then reflect about the y-axis. At what point are you? Solution: (3,−7) Why? (−3,7) → reflect about x-axis (−3,−7) → reflect about y-axis (3,−7) Question: Start at the point (−3,1). Reflect about the vertical line that passes through the x-axis at 2. At what point are you? Solution: (7,1) Why? On this problem, you need to stop and think. It might be helpful to make a sketch (see below). • Since you're reflecting about a vertical line, the y-value won't change. It was 1, and it will remain 1. • Figure out the distance from the point to the line: 2−(−3)=5. You need to go this far on the other side of the vertical line: 2+5=7. So, the new x-value is 7. Question: Start at the point (−3,1). Reflect about the horizontal line that passes through the y-axis at 4. At what point are you?
90 Solution: (−3,7) Why? Again, you need to stop and think. Again, a sketch may be helpful (see below). • Since you're reflecting about a horizontal line, the x-value won't change. It was −3, and it will remain −3. • Figure out the distance from the point to the line: 4−1=3 You need to go this far on the other side of the horizontal line: 4+3=7 So, the new y-value is 7. 2.6 Shearing in 2 Shear an affine

85%	MATCHING BLOCK 106/191	W

transformation A shear is a transformation that distorts the shape of an object along either or both

of the axis. Like scale and translate, a shear can be done along just one or along both of the coordinate axes. A shear along one axis (say, the

100%	MATCHING BLOCK 107/191	W

x-axis) is performed in terms of the point's coordinate in the other axis (

the
y-axis). Thus a shear of 1 in the x-axis will cause the x- coordinate of the point of distort by 1*(y-coordinate).

92%	MATCHING BLOCK 109/191	W

To shear in the x direction the equation is: x1 = x + ay 91 y1 = y Where b = 0 Where x1 and y1 are the new values, x and y are the original values, and a is the scaling factor in the x direction. The matrix is as follows. Shearing in the y direction is similar except the roles are reversed. x1 = x y1 = y + bx Where a = 0. Where x1 and y1 are the new values, x and y are the original values, and b is the scaling factor in the y direction. The matrix is as follows. Example 92 Original Y-Shear

X-Shear Shear Transformations To complete the set of transformations discussed in the notes one should really add the 3D shear operations. A shear along the x direction in 2D changes a rectangle (with lower right corner at the origin) into a parallelogram as follows: Y ^ Y ^ | | P |____________________ | P' ______________________ | | | / / | | | / / | | |A / / | | | / / |____________________|____ X |/____________________/______ X The transformation has the properties: a) The y coordinate of any point (x,y) is unchanged b) The x coordinate is stretched in a linear way, based on the height of the point above the x axis - i.e. on y. Thus the coordinate change has the form: x' = x + ay y' = y where a is a constant that measures the degree of shearing. If a is negative then the
93 shearing is in the opposite direction. Note that P(0,H) is taken into P'(aH,H). It follows that the shearing angle A (the angle through which the vertical edge was sheared) is given by: tan(A) = aH/H = a. So the parameter a is just the tan of the shearing angle. A suitable multiplicative matrix description of the transform is given by: SHx(a) = | 1 a 0 | | 0 1 0 | | 0 0 1 | We can similarly introduce a shearing along the y axis, which would have the form: x' = x y' = y + bx and in this case a suitable matrix description is given by: SHy(b) = | 1 0 0 | | b 1 0 | | 0 0 1 | Shearing in 3D:

89%	MATCHING BLOCK 111/191	SA	ODMCA-404T Palwinder 10th Feb 2020.docx (D63728007)

In 3D we can shear along the x-axis, y-axis or z-axis.

AS an example, the shear along the z axis leaves the z coordinate of points the same but changes the x and y coordinates by amounts proportional to their z coordinate: x' = x + az y' = y + bz z' = z and can be represented by the matrix: SHxy(a,b) = | 1 0 a 0 | | 0 1 b 0 | | 0 0 1 0 |
94 | 0 0 0 1 | The shears along the x and y directions have similar forms. Shear Click here for animation Matrix/Vector Representation of Translations shear along x axis = shear along y axis = Combining Transformations We saw that the basic scaling and rotating transformations are always with respect to the origin. To scale or rotate about a particular point (the fixed point) we must first translate the object so that the fixed point is at the origin. We then perform the scaling or rotation and then the inverse of the original translation to move the fixed point back to its original position. For example, if we want to scale the triangle by 2 in each direction about the point fp = (1.5,1), we first translate all the points of the triangle by T = (-1.5,-1), scale by 2 (S) , and then translate back by -T= (1.5,1). Mathematically this looks like
95 q = = (+) + Order Matters! Notice the order in which these transformations are performed. The first (rightmost) transformation is T and the last (leftmost) is -T. If you apply these transformations in a different order then you will get very different results. For example, what happens when you first apply T followed by -T followed by S? Here T and -T cancel each other out and you are simply left with S. Sometimes (but be careful) order does not matter, For example, if you apply multiple 2D rotations, order makes no difference: R1 R2 = R2 R1 But this will not necessarily be true in 3D!! 2.5 Homogeneous Coordinates in 2 Dimensions Scaling and rotations are both handled using matrix multiplication, which can be combined as we will see shortly. The translations cause a difficulty, however, since they use addition instead of multiplication. We want to be able to treat all 3 transformations (translation, scaling, rotation) in the same way - as multiplications. The solution is to give each point a third coordinate (X, Y, W), which will allow translations to be handled as a multiplication also. (Note that we are not really moving into the third dimension yet. The third coordinate is being added to the mathematics solely in order to combine the addition and multiplication of 2-D coordinates.) Two triples (X,Y,W) and (X',Y',W') represent the same point if they are multiples of each other e.g. (1,2,3) and (2,4,6).At least one of the three coordinates must be nonzero. If W is 0 then the point is at infinity. This situation will rarely occur in practice in computer graphics.
96 If W is nonzero we can divide the triple by W to get the cartesian coordinates of X and Y which will be identical for triples representing the same point (X/W, Y/W, 1). This step can be considered as mapping the point from 3-D space onto the plane W=1. Conversely, if the 2-D cartesian coordinates of a point are known as (X, Y), then the homogenous coordinates can be given as (X, Y, 1). So, how does this apply to translation, scaling, and rotation of 2D coordinates? Translation of 2D Homogenous Coordinates

75% MATCHING BLOCK 110/191 W

Point (X,Y) is to be translated by amount Dx and Dy to location (X',Y')

X' = Dx + X Y' = Dy + Y or P' = T * P where _ _ P' = | X' | | Y' | | 1 | - - _ _ T = | 1 0 Dx | = T(Dx,Dy) | 0 1 Dy | | 0 0 1 | - - _ _ P = | X | | Y | | 1 | - - Hey Look! Translation is now a multiplication instead of an addition! Scaling of 2D Homogenous Coordinates
97 P' = S * P where _ _ P' = | X' | | Y' | | 1 | - - _ _ S = | Sx 0 0 | = S(Sx,Sy) | 0 Sy 0 | | 0 0 1 | - - _ _ P = | X | | Y | | 1 | - - Rotation of 2D Homogenous Coordinates P' = R * P where _ _ P' = | X' | | Y' | | 1 | - - _ _ R = | cos(theta) -sin(theta) 0 | = R(theta) | sin(theta) cos(theta) 0 | | 0 0 1 | - - _ _ P = | X |
98 | Y | | 1 | - - Composition of 2D Transformations There are many situations in which the final transformation of a point is a combination of several (often many) individual transformations. For example, the position of the finger of a robot might be a function of the rotation of the robots hand, arm, and torso, as well as the position of the robot on the railroad train and the position of the train in the world, and the rotation of the planet around the sun, and . . . Applying each transformation individually to all points in a model would take a lot of time. Instead of applying several transformations matrices to each point we want to combine the transformations to produce 1 matrix which can be applied to each point. In the simplest case we want to apply the same type of transformation (translation, rotation, scaling) more than once. Translation is additive as expected scaling is multiplicative as expected rotation is additive as expected But what if we want to combine different types of transformations? A very common reason for doing this is to rotate a polygon about an arbitrary point (e.g. the center of the polygon) rather than around the origin. • Translate so that P1 is at the origin T(-Dx,-Dy) Rotate R(theta) • Translate so that the point at the origin is at P1 T(Dx,Dy), note the order of operations here is right to left P' = T(Dx,Dy) * R(theta) * T(-Dx,-Dy) * P i.e. P' = T(Dx,Dy) * { R(theta) * [T(-Dx,-Dy) * P] } i.e. P' = [T(Dx,Dy) * R(theta) * T(-Dx,-Dy)] * P
99 The matrix that results from these 3 steps can then be applied to all of the points in the polygon. Another common reason for doing this is to scale a polygon about an arbitrary point (e.g. the center of the polygon) rather than around the origin. • Translate so that P1 is at the origin Scale • Translate so that the point at the origin is at P1 How do we determine the 'center' of the polygon? • specifically define the center (e.h. the center of mass) • average the location of all the vertices • take the center of the bounding box of the polygon Window to Viewport Generally user's prefer to work in world-coordinates. • 1 unit can be 1 micron 1 unit can be 1 meter 1 unit can be 1 kilometer 1 unit can be 1 mile These coordinates must then be translated to screen coordinates to be displayed in a rectangular region of the screen called the viewport The objects are in world coordinates (with n dimensions) The viewport is in screen coordinates (with n=2) Want one matrix that can be applied to all points: rectangular area of world from (Xmin,Ymin) to (Xmax,Ymax) - world-coordinate window rectangular area of screen from (Umin,Vmin) to (Umax,Vmax) – viewport need to rescale the world-coordinate rectangle to the screen rectangle
100 1. translate world-coordinate window to the origin of the world coordinate system. 2. rescale

56%	MATCHING BLOCK 113/191	SA	ODMCA-404T Palwinder 10th Feb 2020.docx (D63728007)

the window to the size and aspect ratio of the viewport. 3. translate the viewport to its position on the screen in the

screen coordinate system. Pscreen = M * Pworld M = T(Umin,Vmin) * S(deltaU/deltaX, deltaV/deltaY) * T(-Xmin, -Ymin) Exercises 1. Write a procedure for filling the interior of any specified set of "polygon" vertices using the non-zero winding number rule to identify interior regions. 2. What is a Scan-Line? 3. Write a boundary-fill procedure to fill an 8-connected region. 4. Differentiate between the Crossing Number (cn) method and the Winding Number (wn) method. 5. Give an use each of the Global Edge Table and the Active Edge Table. 6. What do we mean by the term 'reflect the point about the line'. Recommended Readings 1. Computer Graphics, Multimedia and Animation by Pakhira Malay K. 2. Computer Graphics C Version Second Edition by Hearn and Baker
101 Unit 3 3D Geometrical Transformations
102 Contents 3D Geometrical Transformations 3. 0 Objectives 3.1 Basic 3D Transformation Matrices for Translation, Scaling and Rotation 3.2 3D Reflections and Shears 3.3 Viewing 3D Transformations, Computing the 3D Viewing Transformation, The Eye Coordinate System, Constructing the Viewing Matrix 3.4 Projection- Parallel and Perspective; Orthographic Parallel Projections, Oblique Parallel Projection 3.5 Vertex Processing: Coordinates Transformation, OpenGL's Model-View Matrix and Projection Matrix, World Space to Camera Space 3. 0 Objectives • To understand the three dimensional matrices for Geometrical transformations in Computer Graphics • To list the various three dimensional transformations • To differentiate between parallel and perspective projection • To analyze the image formation inside a camera 3.1 Basic 3D Transformation Matrices for Translation, Scaling and Rotation Methods for

92%	MATCHING BLOCK 112/191	W

geometric transformations and object modelling in three dimensions are extended from two-dimensional methods by including considerations for the z coordinate.

We now translate an object by specifying a three-dimensional translation vector, which determines how much the object is to be moved in each of the three coordinate directions. Similarly, we scale an object with three coordinate scaling factors. The extension for three-
103 dimensional rotation is less straight forward. When we discussed

69%	MATCHING BLOCK 114/191	SA	Computer Graphics.pdf (D165747830)

two-dimensional rotations in the xy plane, we needed to consider only rotations about axes that were perpendicular to the xy plane. In three-dimensional space,

we can now select any spatial orientation for the rotation axis. Most graphics packages handle three-dimensional rotation

87%	MATCHING BLOCK 115/191	SA	Computer Graphics.pdf (D165747830)

as a composite of three rotations, one for each of the three Cartesian axes. Alternatively, an user can easily set up a general rotation matrix, given the orientation of the axis and the required rotation angle.

As in the two-dimensional case, we express geometric transformations in matrix form. Any sequence of transformations is then represented as, a single matrix, formed by concatenating the matrices for the individual transformations in the sequence.

100%	MATCHING BLOCK 117/191	SA	ODMCA-404T Palwinder 10th Feb 2020.docx (D63728007)

In 3D graphics, transformation is often used to operate on vertices and vectors. It is also used to convert them in one space to another. Transformation is performed via multiplication with a matrix.

There are typically three types of primitive transformation that can be performed on vertices: translation (where it lies in space relative to the origin), rotation (its direction in relation to the x, y, z frame), and scaling (its distance from origin). In addition to those, projection transformation is used to go from view space to projection space. The D3DX library contains APIs that can conveniently construct a matrix for many purposes such as translation, rotation, scaling, world-to-view transformation, view-to- projection transformation, etc. An application can then use these matrices to transform vertices in its scene. A basic understanding of matrix transformations is required. We will briefly look at some examples below. Translation Translation refers to moving or displacing for a certain distance in space. In 3D, the matrix used for translation has the form 1 0 0 0 0 1 0 0 0 0 1 0 a b c 1

104 where (a, b, c) is the vector that defines the direction and distance to move. For example, to move a vertex -5 unit along the X axis (negative X direction), we can multiply it with this matrix: 1 0 0 0 0 1 0 0 0 0 1 0 -5 0 0 1 If we apply this to a cube object centered at origin, the result is that the box is moved 5 units towards the negative X axis, as figure 5 shows, after translation is applied. The effect of translation
105 In 3D, a space is typically defined by an origin and three unique axes from the origin: X, Y and Z. There are several spaces commonly used in computer graphics: object space, world space, view space, projection space, and screen space. Rotation Rotation refers to rotating vertices about an axis going through the origin. Three such axes are the X, Y, and Z axes in the space. An example in 2D would be rotating the vector [1 0] 90 degrees counter-clockwise. The result from the rotation is the vector [0 1]. The matrix used for rotating ? Degrees clockwise about the Y axis looks like this: cos? 0 -sin? 0 0 1 0 0 sin? 0 cos? 0 0 0 0 1 Figure shows the effect of rotating a cube centered at origin for 45 degrees about the Y axis.
106 Scaling Scaling refers to enlarging or shrinking the size of vector components along axis directions. For example, a vector can be scaled up along all directions or scaled down along the X axis only. To scale, we usually apply the scaling matrix below:
107 p 0 0 0 0 q 0 0 0 0 r 0 0 0 0 1 Where p, q, and r are the scaling factor along the X, Y, and Z direction, respectively. The figure below shows the effect of scaling by 2 along the X axis and scaling by 0.5 along the Y axis.
108 3.2 3D Reflections and Shears

86% **MATCHING BLOCK 116/191** **W**

Reflections A three-dimensional reflection can be performed relative to a selected reflection axis or with respect to a selected reflection plane.

In general, three-dimensional reflection matrices are set up similarly to those for two dimensions. Reflections relative to a given axis are equivalent to 180 o rotations about that axis. Reflections with respect to a plane are equivalent to 180 o rotations in four-dimensional

89% **MATCHING BLOCK 118/191** **W**

space. When the reflection plane is a coordinate plane (either xy, xz, or yz),

we can think of

80% **MATCHING BLOCK 119/191** **W**

the transformation as a conversion between Left-handed and right-handed systems. An example of a reflection that converts coordinate specifications from a right handed system to a left-handed system (or vice versa) is shown in Fig. 4-3. This transformation changes the sign of

the

93% **MATCHING BLOCK 120/191** **W**

z coordinates, Leaving the x and y-coordinate values unchanged. The matrix representation for this reflection of points relative to the xy plane is

given below. Transformation matrices for inverting x and y values are defined similarly, as reflections

73% **MATCHING BLOCK 121/191** **W**

relative to the yz plane and xz plane, respectively Reflections about other planes can be obtained as a combination of rotations and coordinate-plane reflections. 109 Shears Shearing transformations

85% **MATCHING BLOCK 122/191** **W**

can be used to modify object shapes. They are also useful in three-dimensional viewing for obtaining general projection transformations.

In two dimensions, we discussed transformation relative to the x or y axes to produce distortions in the shapes of objects. In three dimensions, we can also generate shears relative to the z axis. As an example of three-dimensional shearing

86% MATCHING BLOCK 123/191 W

the following transformation produces a z- axis shear: Parameters a and b can be assigned any real values.

The effect of

86% MATCHING BLOCK 124/191 W

this transformation matrix is to alter x- and y-coordinate values by an amount that is proportional to the 2 value, while leaving the z coordinate unchanged. Boundaries of planes that are perpendicular to the z axis are

thus shifted by an amount proportional to z. An example of the effect of this shearing matrix on a unit cube is shown in Fig. 4-4, for shearing values a = b = 1. Shearing matrices for the x axis and y axis are defined similarly planes that are perpendicular to the z axis are thus shifted by an amount proportional to z. An example of the effect of this shearing matrix on a unit cube is shown in Fig. 4-4, for shearing values a = b = 1. Shearing matrices for the x axis and y axis are defined similarly.

86% MATCHING BLOCK 125/191 W

Reflections A three-dimensional reflection can be performed relative to a selected reflection axis or with respect to a selected reflection plane.

In general, three-dimensional reflection matrices are set u p similarly to those for two dimensions. Reflections

87% MATCHING BLOCK 126/191 W

relative to a given axis are equivalent to 180°" rotations about that axis.

Reflections with respect to a plane are equivalent to 180°" rotations in four-dimensional

89% MATCHING BLOCK 127/191 W

space. When the reflection plane is a coordinate plane (either xy, xz, or yz),

we can think of

88% MATCHING BLOCK 128/191 W

the transformation as a conversion between Left-handed and right-handed systems. An example of a reflection

is one

76% MATCHING BLOCK 129/191 W

that converts coordinate specifications from a right-handed system to a left-handed system (or vice versa). This transformation changes the sign of

the z coordinates, leaving the x- and y- coordinate values unchanged.
110
Transformation matrices for inverting x and y values are defined similarly, as reflections

67% MATCHING BLOCK 130/191 W

relative to the yz plane and xz plane, respectively Reflections about other planes can be obtained as a combination of rotations and coordinate-plane reflections. Shearing Shearing transformations

85% MATCHING BLOCK 131/191 W

can be used to modify object shapes. They are also useful in three dimensional viewing for obtaining general projection transformations.

In two dimensions, we discussed transformations relative to the x or y axes to produce distortions in the shapes of objects. In three dimensions, we can also generate shears relative to the z axis. Transformations in 3D Type Column Vector Matrix Row Vector Matrix Properties Translation translate (tx ty tz) 1 0 0 tx 0 1 0 ty 0 0 1 tz 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 tx ty tz 1 Rigid Body Orthogonal Submatrix Scale scale (sx sy sy) sx 0 0 0 0 sy 0 0 0 0 sz 0 0 0 0 1 sx 0 0 0 0 sy 0 0 0 0 sz 0 0 0 0 1 Non-rigid Body
111 Rotation around X rotate (1 0 0) (a) 1 0 0 0 0 cos(a) -sin(a) 0 0 sin(a) cos(a) 0 0 0 0 1 1 0 0 0 0 cos(a) sin(a) 0 0 -sin(a) cos(a) 0 0 0 0 1 Rigid Body Orthogonal Rotation around Y rotate (0 1 0) (a) cos(a) 0 sin(a) 0 0 1 0 0 -sin(a) 0 cos(a) 0 0 0 0 1 cos(a) 0 -sin(a) 0 0 1 0 0 sin(a) 0 cos(a) 0 0 0 0 1 Rigid Body Orthogonal Rotation around Z rotate (0 0 1) (a) cos(a) -sin(a) 0 0 sin(a) cos(a) 0 0 0 0 1 0 0 0 0 1 cos(a) sin(a) 0 0 -sin(a) cos(a) 0 0 0 0 1 0 0 0 0 1 Rigid Body Orthogonal Rotation around an Unit Axis rotate (x y z) (a) w = [x, y, z, 0] W = CrossProduct(w) R = I + W sin(a) + W2 (1 - cos(a)) w = [x, y, z, 0] W = CrossProduct(w) R = I + W sin(a) + Rigid Body Orthogonal
112 W2 (1 - cos(a)) Create Cross Product Matrix w is a unit vector W = CrossProduct(w) b = w * a = W a 0 -wz wy 0 wz 0 -wx 0 -wy wx 0 0 0 0 0 0 0 wz -wy 0 -wz 0 wx 0 wy -wx 0 0 0 0 0 0 Skew Symmetric W3 = -W Shear XY [x, y, z, 1] =&lt; [x + shx*z, y + shy*z, z, 1] 1 0 shx 0 0 1 shy 0 0 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 shx shy 1 0 0 0 0 1 Non-rigid Body Shear XZ [x, y, z, 1] =&lt; [x + shx*y, y, z + shz*y, 1] 1 shx 0 0 0 1 0 0 0 shz 1 0 1 0 0 0 shx 1 shz 0 0 0 1 0 0 0 0 1 Non-rigid Body
113 0 0 0 1 Shear YZ [x, y, z, 1] =&lt; [x, y + shy*x, z + shz*x, 1] 1 0 0 0 shy 1 0 0 shz 0 1 0 0 0 0 1 1 shy shz 0 0 1 0 0 0 0 1 0 0 0 0 1 Non-rigid Body Change of Basis Compute GA&gt;-B or GB-&lt;A Given BOA, BXA, BYA, and BZA (the origin and orthonormal basis vectors of the B coordinate system expressed with respect to A's coordinate system) BXAx BYAx BZAx BOAx BXAy BYAy BZAy BOAy BXAz BYAz BZAz BOAz 0 0 0 1 BXAx BXAy BXAz 0 BYAx BYAy BYAz 0 BZAx BZAy BZAz 0 BOAx BOAy BOAz 1 Rigid Body 3.3 Viewing 3D Transformations, Computing the 3D Viewing Transformation, The Eye Coordinate System, Constructing the Viewing Matrix Computing the 3D Viewing Transformation
114 A method of computing the 3D viewing transformation which transforms the right-hand world coordinate system to left-hand eye coordinate system is presented. The initial viewing parameters are chosen so as to be able to give an unrestricted view of the scene. In practice, however, some simplifications are most often used as default viewing parameters. The projection plane, shown in Figure 1, has the view plane defined by a point on the plane (VRP) and the view plane normal (VPN). Figure 1: Initial Viewing Parameters The VPN gives the orientation of the view plane and is often (but not required to be) parallel to the view direction. The VPN is used to define a left-handed coordinate system screen coordinate system. The VUP vector defines a direction which is not parallel to VPN and is taken to be the viewer's concept of up. VUP need not (but often is taken to) be perpendicular to VPN. The projection of VUP on the view plane defines the V axis of the screen coordinate system. The U axis of the screen coordinate system is chosen to be perpendicular to both (orthogonal to) V and VPN. These vectors are chosen so as to form a
115 left-handed V, U, VPN 3D coordinate system. The VRP is the origin of the 2D screen coordinate system. However, VRP is not the origin of the left-handed 3D coordinate system we wish to define. Its origin is the location of the eye (COP). The coordinates of COP are defined relative to the VRP using world coordinates. The Eye Coordinate System We now have all of the parameters necessary to describe the 3D viewing transformation which maps the world coordinate system into the eye coordinate system. A rectangular region, Figure 2, (viewport) describes the clipping region of the screen coordinate system which is visible to the viewer. This 2 dimensional viewport has sides which are parallel to the V U axis and the location and size of the viewport are given in units of the screen coordinate system using VRP as the origin. The viewport forms a viewing pyramid which gives the visible portion of world coordinate space from COP. All objects outside this pyramid are clipped from the scene. Actually, two additional clipping planes (Near and Far; see Figure 3) which are parallel to the view plane define portions of the seen which are either too close or too far from COP to be seen. The line from the COP through the center of the viewport defines the viewing direction and will be the positive Z axis of the eye coordinate system.
116 Figure 2: Viewport Some Linear Algebra A few concepts from elementary linear algebra are useful at this point. Let and be two 3D vectors. The dot product or inner product of and is defined as: Notice that the inner product of two vectors is a real number. The cosine of the acute angle between two vectors and is defined as: where is the length of the vector and is the length of the vector . Hence we may write:
117 Figure 3: Viewing Pyramid Notice that if and are two perpendicular vectors, then the angle between each other is and . Hence, the inner product of two perpendicular vectors is . The length of a vector is defined as . The inner product of with itself yields: or If , then . Given two vectors and , then the sum of and is the vector which is shown in Figure 4.
118 Figure 4: Sum of Vectors v1 and v2 Inner product (dot product) distributes over vector addition. That is, given vectors , , , and a real number , then Suppose is a vector whose length is 1 and is a vector not parallel to . We wish to project to a vector which lies on a plane perpendicular to (see Figure 5). Figure 5: Projecting on To solve this problem define the vector by the equation: Then,
119 But since is of length one. Hence, and from this it follows that and are perpendicular. Therefore, is a vector that lies on a plane perpendicular to The cross product of two vectors and is the vector: The cross product of two non-parallel vectors is a vector which is perpendicular to both vectors. Hence, the inner product of either or with is zero. The direction of the vector is such that if the fingers of the right hand are curled around in the direction of , then is pointing in the direction of the thumb. Cross product is not commutative. which means that the direction of is the opposite of the direction of . Constructing the Viewing Matrix The 3-D viewing pipeline is shown in the figure below. The second step of the pipeline involves transforming the vertices of model objects which are given in world coordinates to the eye coordinate system. This process starts from the initial parameters of , , , and . These vectors are first used to compute the coordinate system as: The Graphics Pipeline

120 The next step is to transform the left-handed eye coordinate system defined by , and into the right-handed world coordinate system. This is accomplished by three steps: 1. The origin of the eye coordinate system, is translated to the origin of the world coordinate system. 2. Rotate so that ? The axis in the direction is parallel to the world coordinate x-axis. ? The axis in the direction is parallel to the world coordinate y-axis. ? The is parallel to the negative z-axis, that is, going into the display screen. 3. Make the negative z-axis the positive direction so that the positive z direction goes into the screen, i.e., make the eye coordinate system left-handed. The matrices required to accomplish this transformation are given next. Since the center of projection, is defined relative to the , the translation matrix is: The rotation matrix is The matrix to change the direction of the z-axis is
121 We can combine the matrices and by computing the matrix product and rename it producing the matrix The final matrix to produce the transformation from world coordinates to eye coordinates is the product of the two matrices 3. 4 Projection- Parallel and Perspective; Orthographic Parallel Projections, Oblique Parallel Projection A Projection is defined as a process which transforms

92%	MATCHING BLOCK 132/191	SA	INF_2046.pdf (D164968115)

points in a coordinate system of dimension n into points in a coordinate system of dimension less than n.

The mapping of 3D objects onto the 2

89%	MATCHING BLOCK 133/191	SA	Computer_Graphics_Block_2.pdf (D149210060)

D screen is done by straight projection rays (called projectors) emanating from a center of projection, passing through each point of the object, and intersecting a projection plane to form the projection.

Projections can be divided into two basic classes: • Perspective or Vanishing Point Method (VPM) • Parallel Projection

64%	MATCHING BLOCK 137/191	SA	INF_2046.pdf (D164968115)

The distinction is in the relation of center of projection and projection plane. If the distance between the center of projection and projection plane is finite then the projection is perspective and if the distance is infinite, the projection is parallel. 122

Perspective Projections Perspective projections are planar geometric projections where all the projectors intersect in one point, center of projection. A property of perspective projections is the existence of one or more vanishing points. A vanishing point is located on an axis of the object's coordinate system. Lines that are parallel to the axis intersect in that point. There can be only one vanishing point for each axis, and the axis needs to intersect the projection plane. Depending on the number of vanishing points a perspective projection is also called one- point, two-point, or three-point perspective. Parallel Projections: Parallel projections are planar geometric projections where all the projectors are parallel to each other. The direction of projection or the angle between the projectors and the projection plane leads to further subclassing. All parallel projections have the following properties in common: angles between lines that are located on a plane that is parallel to the projection plane remain in the projected image; the projection of any pair of parallel lines is also parallel; parallel lines not parallel to the projection plane are equally foreshortened; as long as no scaling is applied to the projection, the distance between two points on a plane parallel to the projection plane is the same as the distance between the projection of those points. The orthographic projection is a parallel projection with the projectors perpendicular to the projection plane. The most common types of orthographic projections are: • Multiple View Projection,
123 • Axonometric Projection Multiple View Projection: It is further classified into a) Front Elevation Projection b) Top Elevation or Plan-elevation Projection c) Side Elevation Projection
Axonometric orthogonal projections -Axonometric orthogonal projections use

77%	MATCHING BLOCK 134/191	W

projection planes that are not normal to a principal axis and therefore show several faces of an object at once.

They resemble the

88%	MATCHING BLOCK 135/191	W

perspective projection in this way, but differ in that the foreshortening is uniform rather than being related to the distance from the center of projection.

It is further classified into i) Isometric Projection ii) Dimetric Projection iii) Trimetric Projection Oblique Parallel Projection In this projection plane normal and the direction of projection differ. It combines properties of the front, top and side orthogonal projections with those of the

66%	MATCHING BLOCK 136/191	W

axonometric projections, the projection plane is normal to a principal axis, so the projections of the face of the object parallel to this plane allows measurement of angles and distances. 124

Two frequently used oblique projections are: • Cavalier Projection –Direction of projection makes a 45 o angle with the projection plane. • Cabinet Projection - Direction of projection that makes an angle of 63.4 o with the projection plane.
125 Perspective projection is a

83%	MATCHING BLOCK 138/191	W

projection in which center of projection is at a finite distance from

a projection plane. The technique of perspective projection is used in preparing perspective drawings of three dimensional objects and scenes. Perspective Projection

90%	MATCHING BLOCK 139/191	W

is characterized by perspective foreshorting and vanishing points. Perspective foreshortening is the illusion that objects and length appear smaller as their distance from the center of projection increases. The illusion that certain sets of parallel lines appear to meet at a point is another feature of perspective drawings. These points are Vanishing Points. 126

Let P1P2 be an object and we take a

73%	MATCHING BLOCK 141/191	SA	Computer_Graphics_Block_2.pdf (D149210060)

projection of this object. Straight Projectors called projectors emanating from a center of projection, passing through each point of object and intersecting a projection plane form

a perspective image P1'P2'of the object P1P2. P2' is the perspective image of P2 and P1' is the perspective image of P1'. Let the distance between the center of projection and plane of projection is d (focal length). In Perspective projection, image size is not true size; it depends upon center of projection, plane of projection and size of the object. The size of the image will be bigger when distance between center of projection and plane of projection decreases (i.e. d decreases). Projection Transformations After the viewing transformation we have everything oriented as we would like them to appear in the final image. All that remains is to project out the depth, or z-dimension, so that the the three-dimensional view-space primitives are reduced to two-dimensional screen-space primitives. There are many different types of projection. The simplest, is to simply ignore the z- dimension. This form of projection is called orthographic or parallel. It is the common form of projection used by drafts people for top, bottom, and side views. The advantage of parallel projection is that the you can make accurate measurements of image features in the two dimensions that remain. The disadvantage is that the images don't appear natural (i.e. they lack perspective foreshorting).
127 Here is an example of a parallel projection of our scene. The projection matrix for orthographic projection is very simple There are some problems with this simple form, however. To begin with the units of the transformed points are still the same as the model. This is great for drafting, but in our case we'd like for the units to be in pixels. We can incorporate this change of units, and perform the flip of the y-axis required for raster coordinates into our projection matrix as follows. For realistic three-dimensional viewing we'd like for our objects to be displayed with proper perspective. Perspective causes objects nearer to the viewing position to appear larger that the same object would when place farther from the viewpoint.
128 Artists (Donatello, Brunelleschi, and Da Vinci) during the renaissance discovered the importance of perspective for making images appear realistic. This outdates mathematicians by more than 300 years. The real reason for introducing homogenous, or projective coordinates to computer graphics was to accomplish perspective projections using linear operators. Not, as mentioned earlier, for combining rotations and translations in the same framework. Lets start by looking at a perspective projection. Note how lines known to be parallel in image space appear to converge to intersect when viewed in perspective. This is an important attribute of lines in projective spaces; they always intersect at a point. The projection matrix for perspective projection matrix is:
129 Notice how similar this transform is to the original parallel projection. It also has all of the disadvantages of the parallel form; its units are not screen space units. The following transformation accomplishes the projection and the conversion to pixels in a single transform. For the purpose of clipping we can modify this transformation so that it is mapped into a canonical space. Following is a canonical space mapping for orthographic projections. Next is a canonical space mapping for perspective projections.

130 3. 5 Vertex Processing: Coordinates Transformation, OpenGL's Model-View Matrix and Projection Matrix, World Space to Camera Space Coordinates Transformation The process used to produce a 3D scene on the display in Computer Graphics is like taking a photograph with a camera. It involves four transformations: • Arrange the objects (or models, or avatar) in the world (Model Transformation or World transformation). • Position and orientation the camera (View transformation). • Select a camera lens (wide angle, normal or telescopic), adjust the focus length and zoom factor to set the camera's field of view (Projection transformation). • Print the photo on a selected area of the paper (Viewport transformation) - in rasterization stage. A transform converts a vertex V from one space (or coordinate system) to another space V'. In computer graphics, transform is carried by multiplying the vector with a transformation matrix, i.e., V' = M V.
131 Model Transform (or Local Transform, or World Transform) Each object (or model or avatar) in a 3D scene is typically drawn in its own coordinate system, known as its model space (or local space, or object space). As we assemble the objects, we need to transform the vertices from their local spaces to the world space, which is common to all the objects. This is known as the world transform. The world transform consists of a series of scaling (scale the object to match the dimensions of the world), rotation (align the axes), and translation (move the origin). Rotation and scaling belong to a class of transformation called linear transformation (by definition, a linear transformation preserves vector addition and scalar multiplication). Linear transform and translation form the so-called affine transformation. Under an affine transformation, a straight line remains a straight line and ratios of distances between points are preserved.

100%	MATCHING BLOCK 140/191	W

In OpenGL, a vertex V at (x, y, z) is represented as

a 3x1 column vector: Other systems, such as Direct3D, use a row vector to represent a vertex.
132 Scaling 3D scaling can be represented in a 3x3 matrix: where αx, αy

76%	MATCHING BLOCK 142/191	W

and αz represent the scaling factors in x, y and z direction, respectively.

If all the factors are the same, it is called uniform scaling. We can obtain the transformed result V' of vertex V via matrix multiplication, as follows:

88%	MATCHING BLOCK 143/191	W

Rotation 3D rotation operates about an axis of rotation (2D rotation operates about a center of rotation). 3D Rotations about the x, y and z axes for an angle θ (measured in counter- clockwise manner) can be represented

in the following 3x3 matrices:

85%	MATCHING BLOCK 146/191	W

The rotational angles about x, y and z axes, denoted as θx, θy and θz, are known as Euler angles, which can be used to specify any arbitrary orientation of an object.

The combined transform is called Euler transform.

83%	MATCHING BLOCK 144/191	W

Translation Translation does not belong to linear transform, but can be modelled via a vector addition, as follows:

Fortunately, we can represent translation using a 4x4 matrices and obtain the transformed result via matrix multiplication, if the vertices are represented in the so-called 4- component homogeneous coordinates (x, y, z, 1), with an additional forth w-component of 1. We shall describe the significance of the w-component later in projection transform. In
133 general, if the w-component is not equal to 1, then (x, y, z, w) corresponds to Cartesian coordinates of (x/w, y/w, z/w). If w=0, it represents a vector, instead of a point (or vertex). Using the 4-component homogeneous coordinates, translation can be represented in a 4x4 matrix, as follows: The transformed vertex V' can again be computed via matrix multiplication: Summary of Affine Transformations We rewrite the scaling and rotation into 4x4 matrices using the homogenous coordinates.

134 Successive Transforms A series of successive affine transforms (T1, T2, T3, ...) operating on a vertex V can be computed via concatenated matrix multiplications V' = ...T3T2T1V. The matrices can be combined before applying to the vertex because matrix multiplication is associative, i.e., T3 (T2 (T1 V)) = (T3T2T1) V. Transformation of Vertex-Normal Recall that a vector has a vertex-normal, in addition to (x, y, z) position and color. Suppose that M is a transform matrix, it can be applied to vertex-normal only if the transforms does not include non-uniform scaling. Otherwise, the transformed normal will not be orthogonal to the surface. For non-uniform scaling, we could use (M-1)T as the transform matrix, which ensure that the transformed normal remains orthogonal. View Transform After the world transform, all the objects are assembled into the world space. We shall now place the camera to capture the view. Positioning the Camera In 3D graphics, we position the camera onto the world space by specifying three view parameters: EYE, AT and UP, in world space.
135 1. The point EYE (ex, ey, ez) defines the location of the camera. 2. The vector AT (ax, ay, az) denotes the direction where the camera is aiming at, usually at the center of the world or an object. 3. The vector UP (ux, uy, uz) denotes the upward orientation of the camera roughly. UP is typically coincided with the y-axis of the world space. UP is roughly orthogonal to AT, but not necessary. As UP and AT define a plane, we can construct an orthogonal vector to AT in the camera space. Notice that the 9 values actually produce 6 degrees of freedom to position and orientate the camera, i.e., 3 of them are not independent. OpenGL In OpenGL, we can use the GLU function gluLookAt() to position the camera: void gluLookAt(GLdouble xEye, GLdouble yEye, GLdouble zEye, GLdouble xAt, GLdouble yAt, GLdouble zAt, GLdouble xUp, GLdouble yUp, GLdouble zUp) The default settings of gluLookAt() is: gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, -100.0, 0.0, 1.0, 0.0) That is, the camera is positioned at the origin (0, 0, 0), aimed into the screen (negative z- axis), and faced upwards (positive y-axis). To use the default settings, you have to place the objects at negative z-values Computing the Camera Coordinates From EYE, AT and UP, we first form the coordinate (xc, yc, zc) for the camera, relative to the world space. We fix zc to be the opposite of AT, i.e., AT is pointing at the -zc. We can obtain the direction of xc by taking the cross-product of AT and UP. Finally, we get the direction of yc by taking the cross-product of xc and zc. Take note that UP is roughly, but not necessarily, orthogonal to AT.
136 Transforming from World Space to Camera Space Now, the world space is represented by standard orthonormal bases (e1, e2, e3), where e1=(1, 0, 0), e2=(0, 1, 0) and e3=(0, 0, 1), with origin at O=(0, 0, 0). The camera space has orthonormal bases (xc, yc, zc) with origin at EYE=(ex, ey, ez). It is much more convenience to express all the coordinates in the camera space. This is done via view transform. The view transform consists of two operations: a translation (for moving EYE to the origin), followed by a rotation (to axis the axes): The View Matrix We can combine the two operations into one single View Matrix: Model-View Transform In Computer Graphics, moving the objects relative to a fixed camera (Model transform), and moving the camera relative to a fixed object (View transform) produce the same image, and
137 therefore are equivalent. OpenGL, therefore, manages the Model transform and View transform in the same manner on a so-called Model-View matrix. Projection transformation (in the next section) is managed via a Projection matrix. Projection Transform - Perspective Projection Once the camera is positioned and oriented, we need to decide what it can see (analogous to choosing the camera's field of view by adjusting the focus length and zoom factor), and how the objects are projected onto the screen. This is done by selecting a projection mode (perspective or orthographic) and specifying a viewing volume or clipping volume. Objects outside the clipping volume are clipped out of the scene and cannot be seen. View Frustum in Perspective View The camera has a limited field of view, which exhibits a view frustum (truncated pyramid), and is specified by four parameters: fovy, aspect, zNear and zFar. 1. Fovy: specify the total vertical angle of view in degrees. 2. Aspect: the ratio of width vs. height. For a particular z, we can get the height from the fovy, and then get the width from the aspect. 3. zNear; the near plane. 4. zFar: the far plane. The camera space (xc, yc, zc) is renamed to the familiar (x, y, z) for convenience.
138 The projection with view frustum is known as perspective projection, where objects nearer to the COP (Center of Projection) appear larger than objects further to the COP of the same size. An object outside the view frustum is not visible to the camera. It does not contribute to the final image and shall be discarded to improve the performance. This is known as view- frustum culling. If an object partially overlaps with the view frustum, it will be clipped in the later stage. OpenGL In OpenGL, there are two functions for choosing the perspective projection and setting its clipping volume: 1. More commonly-used GLU function gluPerspective(): void gluPerspective(GLdouble fovy, GLdouble aspectRatio, GLdouble zNear, GLdouble zFar) // fovy is the angle between the bottom and top of the projectors; // aspectRatio is the ratio of width and height of the front (and also back) clipping plane; // zNear and zFar specify the front and back clipping planes. 2. Core GL function glFrustum(): void glFrustum(GLdouble xLeft, GLdouble xRight, GLdouble yBottom, GLdouble yTop, GLdouble zNear, GLdouble zFar) // xLeft, xRight, yBottom and yTop specifies the front clipping plane. // zNear and zFar specify the positions of the front and back clipping planes. Clipping-Volume Cuboid Next, we shall apply a so-called projection matrix to transform the view-frustum into a axis- aligned cuboid clipping-volume of 2x2x1 centered on the near plane, as illustrated. The near plane has z=0, whereas the far plane has z=-1. The planes have dimension of 2x2, with range from -1 to +1.
139 The Perspective Projection Matrix The projection matrix is given by: Take note that the last row of the matrix is no longer [0 0 0 1]. With input vertex of (x, y, z, 1), the resultant w-component would not be 1. We need to normalize the resultant homogeneous coordinates (x, y, z, w) to (x/w, y/w, z/w, 1) to obtain position in 3D space. (It is amazing that homogeneous coordinates can be used for translation, as well as the perspective projection.) The final step is to flip the z-axis, so that the near plane is still located at z=0, but the far plane is flipped and located at z=1 (instead of z=-1). In other words, the larger the z, the further is the object. To perform flipping, we can simply negate the third row of the projection matrix.

140 After the flip, the coordinate system is no longer a Right-Hand System (RHS), but becomes a Left-hand System (LHS). OpenGL's Model-View Matrix and Projection Matrix OpenGL manages the transforms via two matrices: a model-view matrix (GL_MODELVIEW for handling model and view transforms) and a projection matrix (GL_PROJECTION for handling projection transform). These two matrices can be manipulated independently. We need to first select the matrix for manipulation via: void glMatrixMode(GLenum matrix) // Select matrix for manipulating, e.g., GL_PROJECTION, GL_MODELVIEW.
141 We can reset the currently selected matrix via: void glLoadIdentity() We can save the value of the currently selected matrix onto the stack and restore it back via: void glPushMatrix() void glPopMatrix() Push and pop use a stack and operate in a last-in-first-out manner, and can be nested. Projection Transform - Orthographic Projection Besides the commonly-used perspective projection, there is another so-called orthographic projection (or parallel projection), which is a special case where the camera is placed very far away from the world (analogous to using telescopic lens). The view volume for orthographic projection is a parallelepiped (instead of a frustum in perspective projection). OpenGL In OpenGL, we can use glOrtho() function to choose the orthographic projection mode and specify its clipping volume:
142 void glOrtho(GLdouble xLeft, GLdouble xRight, GLdouble yBottom, GLdouble yTop, GLdouble zNear, GLdouble zFar) For 2D graphics, we can use gluOrtho2D() (GLU function instead of GL) to choose 2D orthographic projection and set its clipping area: void gluOrtho2D(GLdouble xLeft, GLdouble xRight, GLdouble yBottom, GLdouble yTop) The default 3D projection in OpenGL is the orthographic (instead of perspective) with parameters (-1.0, 1.0, -1.0, 1.0, -1.0, 1.0), i.e., a cube with sides of 2.0, centered at origin. Outputs of the Vertex Processing Stage Each vertex is transformed and positioned in the clipping-volume cuboid space, together with their vertex-normal. The x and y coordinates (in the range of -1 to +1) represent its position on the screen, and the z value (in the range of 0 to 1) represents its depth, i.e., how far away from the near plane. The vertex processing stage transforms individual vertices. The relationships between vertices (i.e., primitives) are not considered in this stage. Exercises 1. What is orthographic or parallel projection? Mention an advantage and disadvantage of this type of projection. 2. What is meant by scaling? What is the scaling matrix? 3. Which are the three types of primitive transformation that can be performed on vertices? Give examples of each transformation. 4. What is the function of shearing? 5. Write a routine to implement scaling as a raster transformation of a pixel block. 6. Show that two successive reflections about either of the co-ordinate axes is equivalent to a single rotation about the coordinate origin. 7. What is a projection? What are the two basic classes of projection?
143 8. Using the 4-component homogeneous coordinates, represent translation in a 4x4 matrix. 9. How do we transform from world space to camera space? 10. “The process used to produce a 3D scene on the display in Computer Graphics is like taking a photograph with a camera. It involves four transformations”. List the four transformations.
144 Unit 4 2D Viewing and Clipping
145 4. 0 Objectives 2D Viewing and Clipping 4. 1 Windows and Viewports- Window-To-Viewport Coordinate Transformation, OpenGL Window-To-Viewport Mapping, Development of the Window-To-Viewport Mapping 4.2 Clipping of Lines in 2D- Cohen Sutherland Clipping, Liang-Barsky Line Clipping 4. 3 Cohen-Sutherland Clipping Algorithm- Inside-Outside Window Codes 4. 4 Midpoint Subdivision Method- Midpoint Subdivision Algorithm 4. 5 Polygon Clipping- Sutherland-Hodgeman Polygon Clipping, Weiler-Atherton Polygon Clipping 4. 6 Curve design techniques- Parametric Equation of a Circle, 2-D Curve Generation, LeGrange interpolated Curve, B-Spline Curves, Bezier Curves, 3-D Curves 4.0 Objectives • To understand the process of mapping or transforming a two-dimensional, world- coordinate scene to device coordinates • To be able to clip lines in Two dimensional graphics • To analyse the Cohen-Sutherland Clipping Algorithm • To learn the skills for polygon clipping • To list the various Classic and modern Curve designing techniques 4. 1 Windows and Viewports- Window-To-Viewport Coordinate Transformation, Development of the Window-To-Viewport Mapping

91%	MATCHING BLOCK 145/191	W	

The Viewing Pipeline A world-coordinate area selected for display is called a window. An area on a display device to which a window is mapped is called a viewport. The window defines what is to be viewed; the viewport defines where it is to be displayed.

Often, windows and viewports are rectangles in standard position, with the rectangle edges parallel to the coordinate axes. Other window or viewport geometries, such as general polygon shapes and circles, are
146 used in some applications, but these shapes take longer to process.
In general,

100%	MATCHING BLOCK 148/191	SA	MSc_Computer Graphics_Course code MCSDSC_1.7.pdf (D140276469)

the mapping of a part of a world-coordinate scene to device coordinates is referred to as a viewing transformation.

Sometimes the two-dimensional viewing transformation is simply referred to as the window-to-viewport transformation or the windowing transformation. But, in general, viewing involves more than just the transformation from the window to the viewport. Some graphics packages that provide window and viewport operations allow only standard rectangles, but a more general approach is to allow the rectangular window to have any orientation. In this case, we carry out

93%	MATCHING BLOCK 147/191	W

the viewing transformation in several steps. First, we construct the scene in world coordinates using the output primitives and attributes. Next, to obtain a particular orientation for the window, we can set up a two-dimensional viewing-coordinate system in the world-coordinate plane, and define a window In the viewing-coordinate system. The viewing- coordinate reference frame is used to provide a method for setting up arbitrary orientations for rectangular windows. Once the viewing reference frame is established, we can transform descriptions in world coordinates to viewing coordinates. We then define a viewport in normalized coordinates (in the range from O to I) and map the viewing-coordinate description of the scene to normalized co- ordinates. At the final step, parts of the picture that are outside the viewport are clipped, and the contents of the viewport are transferred to device coordinates. By changing the position of the viewport, we can view objects at different positions on the display area of an output device.

Also, by varying the size of
the viewports, we can change the size and proportions of displayed objects.
We achieve zooming effects by successively mapping different-sized windows on a fixed-size viewport. As the windows are made smaller, we zoom in on some part of a scene to view details that are not shown with larger windows.
Similarly, more overview is obtained by zooming out from a section of a scene with successively larger windows. Panning effects are produced by moving a fixed-size window across the various objects in a scene. Viewports are typically defined within the unit square (normalized coordinates). This provides a means for separating the viewing and other transformations from specific output-device requirements, so that the graphics package is largely device-independent.
147 Once the scene has been transferred to normalized coordinates, the unit square is simply mapped to the display area for the particular output device in use at that time. Different output devices can be used by providing the appropriate device drivers. Clipping procedures are of fundamental importance in computer graphics. They are used not only in viewing transformations, but also in window-manager systems and drawing packages to eliminate parts of a picture inside or outside of a designated screen area, and in many other applications. Window-To-Viewport Coordinate Transformation • World Coordinate System - This is object space or

84%	MATCHING BLOCK 149/191	SA	Unit-3 (Part-III).docx (D77528823)

the space in which the application model is defined. Definitions • Screen Coordinate System -

The space in which the image is displayed. • World Window (or clipping) - This is the rectangle in the world defining the region that is to be displayed. •

100%	MATCHING BLOCK 150/191	SA	ODMCA-404T Palwinder 10th Feb 2020.docx (D63728007)

Interface Window - The window opened on the raster graphics screen in which the image will be displayed. •

Viewport -

90%	MATCHING BLOCK 155/191	SA	Unit-3 (Part-III).docx (D77528823)

The rectangular portion of the interface window that defines where the image will actually appear (

usually the entire interface window but in some cases modified to be a portion of the interface window). • Viewing Transformation - The process of mapping a world window in World Coordinates to the Viewport.
148 Window-to-Viewport mapping is the process of mapping or transforming a two- dimensional, world-coordinate scene to device coordinates. In particular, objects inside

100%	MATCHING BLOCK 151/191	W

the world or clipping window are mapped to the viewport. The

viewport is displayed in

33%	MATCHING BLOCK 152/191	W

the interface window on the screen. In other words, the clipping window is used to select the part of the scene that is to be displayed. The viewport then positions the scene on the output device.

Example Development of the Window-To-Viewport Mapping This mapping or transformation involves developing formulas that start with

72%	MATCHING BLOCK 153/191	W

a point in the world window, say (xw, yw). The formula is used to produce a corresponding point in viewport coordinates, say (xv, yv).

We would like for this mapping to be "proportional" in the sense that

84%	MATCHING BLOCK 154/191	W

if xw is 30% of the way from the left edge of the world window, then xv is 30% of the way from the left edge of the viewport. Similarly, if yw is 30% of the way from the bottom edge of the world window, then yv is 30% of the way from the bottom edge of the viewport. The

picture below shows this proportionality.
149 Using this proportionality, the following ratios must be equal. By solving these equations for the unknown viewport position (xv, yv), the following becomes true: The scale factors (Sx, Sy) would be:
150 And the translation factors (Tx, Ty) would be:

96%	MATCHING BLOCK 156/191	W

The position of the viewport can be changed allowing objects to be viewed at different positions

on the Interface Window.

100%	MATCHING BLOCK 157/191	W

Multiple viewports can also be used to display different sections of a scene at different screen positions.

Also, by changing the dimensions of the viewport, the size and proportions of the objects being displayed can be manipulated. Thus, a zooming affect can be achieved by successively

100%	MATCHING BLOCK 158/191	W

mapping different dimensioned clipping windows on a fixed sized viewport.

58%	MATCHING BLOCK 159/191	W

If the aspect ratio of the world window and the viewport are different, then the image may look distorted. OpenGL Window-To-Viewport

Mapping In OpenGL, the function void gluOrtho2D(left, right, bottom, top); is used to set up the world window. For example, if we wanted a world window with x varying from -1.0 to 1.0 and y varying from 3.0 to 5.0, we would use the following code to accomplish this: glMatrixMode(GL_PROJECTION); glLoadIdentity(); gluOrtho2D(-1.0, 1.0, 3.0, 5.0); To set up the viewport, we would use glViewport(left, bottom, width, height); For
151 example, if we wished a viewport to start at the lower left corner of the interface window, have a width of 200 and a height of 300, we would use the following OpenGL statement to accomplish this: glViewport(0, 0, 200, 300); Tiling using the Window-To-Viewport transformation If we draw a

97%	MATCHING BLOCK 160/191	W

number of copies of the same image in rows and columns across the interface window so that they cover the entire window,

it is called tiling. The picture that is drawn many times is called a motif. To achieve tiling in computer graphics, the window remains static and the viewport is changed many times and the picture is redrawn each time the viewport is changed. The following picture shows the same image from the world drawn 4 times in 4 different viewports. This can be achieved using the following code: for (int i = 0; i &gt; 2; i++) for (int j = 0; j &gt; 2; j++) { glViewport (i * screenWidth/2.0, j * screenHeight/2.0, screenWidth/2, screenHeight/2);

152 drawSaturn(); } Zooming or Panning Using Window-to-Viewport transformation Zooming is a technique in which users can change the size of the area to be viewed in order to see more detail or less detail. Panning or roaming means sliding

the camera across the scene, taking in different parts of it at different times.

Both zooming and panning can be achieved in computer graphics by leaving the viewport static and changing the world window. The picture below shows these concepts. Once object descriptions have been transferred to the viewing reference frame, we choose the window extents in viewing coordinates and select the viewport limits in normalized coordinates. Object descriptions are then transferred to normalized device coordinates. We do this using a transformation that maintains the same relative placement of objects in normalized space as they had in viewing coordinates. If a coordinate position is at the center of the viewing window, for instance, it will be displayed at the center of the viewport. 4.2 Clipping of Lines in 2D- Cohen Sutherland Clipping, Liang-Barsky Line

Clipping 153 Generally, any procedure that identifies those portions of a picture that are either inside or outside of a specified region of space is referred to as a clipping algorithm, or simply clipping. The region against which an object is to be clipped is called a clip window.

In computer graphics, '

line clipping' is the process of removing lines or portions of lines outside of an area of interest. Typically, any line or part thereof which is outside of the viewing area is removed.

There are two common algorithms for line clipping: Cohen–Sutherland and Liang– Barsky. Applications

of clipping include extracting part of a detained scene for viewing; identifying visible surfaces in three-dimensional views;

antialiasing line segments or

object boundaries; creating objects using solid-modelling procedures; displaying a multi window environment; and drawing and painting operations

that allow parts of a picture to be selected for copying, moving, erasing, or duplicating.

Depending on the application, the clip window can be a general polygon or it can even have curved boundaries. We

first consider clipping methods using rectangular clip regions, then we discuss methods for other clip-region shapes.

For the viewing transformation, we want to display only those picture parts that are within the window area (assuming that the clipping flags have not been set to no clip). Everything outside the window is discarded. Clipping algorithms can be applied in world coordinates, so that only the contents of the window interior are mapped to device coordinates. Alternatively, the complete world-coordinate picture can be mapped first to device coordinates, or normalized device coordinates, then clipped against the viewport boundaries. World-coordinate clipping removes those primitives outside the window from further consideration, thus eliminating the processing necessary to transform those primitives to device space. Viewport clipping, on the other hand, can reduced calculations by allowing concatenation of viewing and geometric transformation matrices. But viewport clipping does require that the 154 transformation to device coordinates be performed for all objects, including those outside the window area. On raster systems, clipping algorithms are often combined with scan conversion. In the following sections, we consider algorithms for clipping the following

primitive types ? Point Clipping ? Line Clipping (straight-line segments) ? Area Clipping (polygons) ? Curve Clipping ? Text Clipping Line and polygon clipping routines are standard components of graphics packages,

but many packages accommodate curved objects, particularly spine curves and conics, such as circles and ellipses. Another way to handle curved objects is to approximate them with straight-line segments and apply the line- or polygon- clipping procedure. Since we have a separation between the models and the image created from those models, there can be parts of the model that do not appear in the current view when they are rendered. Pixels outside the clip rectangle are clipped, and are not displayed. Can clip analytically - Knowing where the clip rectangle is clipping can be done before scan- line converting a graphics primitive (point, line, polygon) by altering the graphics primitive so the new version lies entirely within the clip rectangle. Can clip by brute force (scissoring) - scan convert the entire primitive but only display those pixels within the clip rectangle by checking each pixel to see if it is visible. • clipping a point against a rectangle -&lt; nothing or single point • clipping a line against a rectangle -&lt; nothing or single line segment • clipping a rectangle against a rectangle -&lt; nothing or single rectangle • (Assuming the rectangle is aligned. Otherwise treat as convex polygon.) • clipping a convex polygon against a rectangle -&lt; nothing or single single convex
155 polygon • clipping a concave polygon against a rectangle -&lt; nothing or 1 or more concave polygons
as with scan conversion, this must be done as quickly as possible as it is a very common operation.
Point Clipping point (X,Y) clipping rectangle with corners (Xmin,Ymin) (Xmax,Ymax) point is within the clip rectangle if: Xmin &gt;= X&gt;= Xmax Ymin &gt;= Y&gt;= Ymax Cohen Sutherland
Clipping When drawing a 2D line on screen, it might happen that one or both of the endpoints are outside the screen while a part of the line should still be visible. In that case, an efficient algorithm is needed to find two new endpoints that are on the edges on the screen, so that the part of the line that's visible can now be drawn. This way, all those points of the line outside the screen are clipped away and you don't need to waste any execution time on them. A good clipping algorithm is the Cohen-Sutherland algorithm. The function containing this algorithm is already included in QuickCG in the file QuickCG.cpp and is called clipLine. You pass the coordinates of the old line, and the coordinates of the new line by reference so that the function can return the coordinates of the new line by changing those parameters. Clipping is a very important aspect of 3D graphics, and so in the 3D Lines tutorial, this 2D Clipping function is used often. Liang-Barsky Line Clipping The ideas for clipping line of Liang-Barsky and Cyrus-Beck are the same. The only difference is Liang-Barsky algorithm has been optimized for an upright rectangular clip window. So we will study only the idea of Liang-Barsky.
156 Liang and Barsky have created an algorithm that uses floating-point arithmetic but finds the appropriate end points with at most four computations. This algorithm uses the parametric equations for a line and solves

four inequalities to find the range of the parameter for which the line is in the viewport.

Let be the line which we want to study. The parametric equation of the line segment from gives x-values and y-values for every point in terms of a parameter that ranges from 0 to 1. The equations are and We can see that when t = 0, the point computed is P(x1,y1); and when t = 1, the point computed is Q(x2,y2). Algorithm 1. Set and 2. Calculate the values of tL, tR, tT, and tB (tvalues). • if or ignore it and go to the next edge • otherwise classify the tvalue as entering or exiting value (using inner product
157 to classify) • if t is entering value set ; if t is exiting value set 3. If then draw a line from (x1 + dx*tmin, y1 + dy*tmin) to (x1 + dx*tmax, y1 + dy*tmax) 4. If the line crosses over the window, you will see (x1 + dx*tmin, y1 + dy*tmin) and (x1 + dx*tmax, y1 + dy*tmax) are intersection between line and edge. The next step we consider if t value is entering or exiting by using inner product. (Q-P) = (15+5,9-3) = (20,6) At left edge (Q-P)nL = (20,6)(-10,0) = -200 &gt; 0 entering so we set tmin = ¼ At right edge (Q-P)nR = (20,6)(10,0) = 200 &lt; 0 exiting so we set tmax = ¾ Because then we draw a line from (-5+(20)*(1/4), 3+(6)*(1/4)) to (- 5+(20)*(3/4), 3+(6)*(3/4)) Example 2 - Line Not Passing Through Window
158 The next step we consider if tvalue is entering or exiting by using inner product. (Q-P) = (2+8,14-2) = (10,12) At top edge (Q-P)nT = (10,12)(0,10) = 120 &lt; 0 exiting so we set tmax = 8/12 At left edge (Q-P)nL = (10,12)(-10,0) = -100 &gt; 0 entering so we set tmin = 8/10 Because tmin &lt; tmax then we don't draw a line. 4.3 Cohen-Sutherland Clipping Algorithm When drawing a 2D line, if one endpoint of the line is outside the screen, and the other inside, you have to clip the line so that only the part of it that's inside the screen remains. Even if both endpoints are outside the screen, it's still possible that a part of the line should be visible. The clipping algorithm needs to find new endpoints of the lines, that are inside or on the edges of the screen. Here are a few cases, where the black rectangle represents the screen, in red are the old endpoints, and in blue the ones after clipping: • Case A: both endpoints are inside the screen, no clipping needed. • Case B: one endpoint outside the screen, that one had to be clipped
159 • Case C: both endpoint are outside the screen, and no part of the line is visible, don't draw it at all. • Case D: both endpoint are outside the screen, and a part of the line is visible, clip both endpoints and draw it. There are tons of different cases, each endpoint can be inside the screen, left of it, right of it, above, below, etc... The Cohen Sutherland Clipping Algorithm can recognize these cases quite efficiently and do the clipping. The algorithm divides the 2D space in 9 regions: The center region is the screen, and the other 8 regions are on different sides outside the screen. Each region is given a binary number, called an "outcode". The codes are chosen as follows: •

38%	MATCHING BLOCK 169/191	W

If the region is above the screen, the first bit is 1 • If the region is below the screen, the second bit is 1 • If the region is to the right of the screen, the third bit is 1 • If the region is to the left of the

screen, the fourth bit is 1 Obviously an area can't be to the left and the right at the same time, or above and below it at the same time, so the third and fourth bit can't be 1 together, and the first and second bit can't be 1 together. The screen itself has all 4 bits set to 0. Both endpoints of the line can lie in any of these 9 regions, and there are a few trivial cases:
160 • If both endpoints are inside or on the edges of the screen, the line is inside the screen or clipped, and can be drawn. This case is the trivial accept. • If both endpoints are on the same side of the screen (e.g., both endpoints are above the screen), certainly no part of the line can be visible on the screen. This case is the trivial reject, and the line doesn't have to be drawn. These two cases can easily be detected thanks to the outcodes of the regions: • Trivial Accept: both endpoints have to be in the region with code 0000, so the trivial accept case only happens if code1 | code2 == 0, where code1 and code2 are the codes of both endpoints of the line, and '|' is the binary OR operator, which can only return 0 if both codes are 0. • Trivial Reject: because of the way the codes of the regions were chosen, only if both endpoints of the line are on the same side of the region, both codes will have two corresponding bits that are both 1. For example, only if both endpoints are on the left of the screen, the fourth bit of both codes is 1. So, the trivial reject case is detected if code1 & code2 != 0, where & is the binary AND operation. The binary AND operation only returns a non zero result if two corresponding bits are 1. All other cases (i.e. no trivial reject and no trivial accept) have to be turned into a trivial case by doing one clip operation. The Cohen Sutherland algorithm is a loop, that does only one clipping operation at the time. It can clip one endpoint of the line, and only clip it to a vertical or horizontal region border. In many cases, it has to clip multiple times before it can finally detect if the line is to be accepted, or rejected. It never has to be clipped more than about 4 times though, so it's quite fast. The function that uses the Cohen Sutherland Clipping Algorithm is the clipLine function from QuickCG, which is in the QuickCG.cpp file. It uses an auxiliary function, findRegion, that returns the binary code of the region a given endpoint is in. For example to set the second bit to 1, you have to 'OR' the code with 4 (the first bit represents 8, the second 4, the third 2, and the firth (the primary bit) represents 1). int findRegion(int

74%	MATCHING BLOCK 170/191	W

x, int y) 161 { int code=0; if(y &lt;= h) code |= 1; //top else if(y &gt; 0) code |= 2; //bottom if(x &lt;= w) code |= 4; //right else if (x &gt; 0) code |= 8; //left return(code); }

Cohen-Sutherland Line Clipping The Cohen-Sutherland line clipping algorithm quickly detects and dispenses with two common and trivial cases. To clip a line, we need to consider only its endpoints. If both endpoints of a line lie inside the window, the entire line lies inside the window. It is trivially accepted and needs no clipping. On the other hand, if both endpoints of a line lie entirely to one side of the window, the line must lie entirely outside of the window. It is trivially rejected and needs to be neither clipped nor displayed. Inside-Outside Window Codes To determine whether endpoints are inside or outside a window, the algorithm sets up a half-space code for each endpoint. Each edge of the window defines an infinite line that divides the whole space into two half-spaces, the inside half-space and the outside half- space, as shown below.
162 As you proceed around the window, extending each edge and defining an inside half-space and an outside half-space, nine regions are created - the eight "outside" regions and the one "inside"region. Each of the nine regions associated with the window is assigned a 4-bit code to identify the region. Each bit in the code is set to either a 1(true) or a 0(false).

26%	MATCHING BLOCK 171/191	W

If the region is to the left of the window, the first bit of the code is set to 1. If the region is to the top of the window, the second bit of the code is set to 1. If to the right, the third bit is set, and if to the bottom, the fourth bit is set. The 4

bits in the code then identify each of the nine regions as shown below. For any endpoint (x , y) of a line, the code can be determined that identifies which region the endpoint lies. The code's bits are set according to the following conditions:
163

100%	MATCHING BLOCK 173/191	SA	Unit-3 (Part-III).docx (D77528823)

The sequence for reading the codes' bits is LRBT (Left, Right, Bottom, Top).

Once the codes for each endpoint of a line are determined, the logical AND operation of the codes determines if the line is completely outside of

43%	MATCHING BLOCK 174/191	SA	Unit-3 (Part-III).docx (D77528823)

the window. If the logical AND of the endpoint codes is not zero, the line can be trivally rejected. For example, if an endpoint

had a code of 1001 while the other endpoint had a code of 1010, the logical AND would be 1000 which indicates the line segment lies outside of the window. On the other hand, if the endpoints had codes of 1001 and 0110, the logical AND would be 0000, and the line could not be trivally rejected.

96%	MATCHING BLOCK 175/191	SA	Unit-3 (Part-III).docx (D77528823)

The logical OR of the endpoint codes determines if the line is completely inside the window. If the logical OR is zero, the line can be trivially accepted. For example, if the endpoint codes are 0000 and 0000, the logical OR is 0000 - the line can be trivially accepted. If the

73%	MATCHING BLOCK 176/191	SA	Unit-3 (Part-III).docx (D77528823)

endpoint codes are 0000 and 0110, the logical OR is 0110 and the line cannot be trivially accepted. Algorithm The

Cohen-Sutherland algorithm uses a divide-and-conquer strategy. The line segment's endpoints are tested to see if the line can be trivially accepted or rejected. If the line cannot be trivially accepted or rejected, an intersection of the line with a window edge is determined and the trivial reject/accept test is repeated. This process is continued until the line is accepted.

100%	MATCHING BLOCK 177/191	SA	Unit-3 (Part-III).docx (D77528823)

To perform the trivial acceptance and rejection tests, we extend the edges of the window to divide the plane of the window into the nine regions.

Each end point of the line segment is then assigned the code of the region in which it lies.
164 1. Given a line segment with endpoint and 2. Compute the 4-bit codes for each endpoint.

96%	MATCHING BLOCK 178/191	SA	Unit-3 (Part-III).docx (D77528823)

If both codes are 0000,(bitwise OR of the codes yields 0000) line lies completely inside the window:

pass the endpoints to the draw routine. If both codes have a 1 in the same bit position (bitwise AND of the codes is not 0000), the line lies outside the window. It can be trivially rejected. 3. If a line cannot be trivially accepted or rejected, at least one of the two endpoints must lie outside the window and the line segment crosses a window edge. This line must be clipped at the window edge before being passed to the drawing routine. 4. Examine one of the endpoints, say . Read 's 4-bit code in order: Left-to-Right, Bottom-to-Top. 5. When a set bit (1) is found, compute the intersectionI of the corresponding window edge with the line from to . Replace with I and repeat the algorithm. Before Clipping 1. Consider the line segment AD. Point A has an outcode of 0000 and point D has an outcode of 1001. The logical AND of these outcodes is zero; therefore, the line cannot be trivally rejected. Also, the logical OR of the outcodes is not zero; therefore, the line cannot be trivally accepted. The algorithm then chooses D as the outside point (its outcode contains 165 1's). By our testing order, we first use the top edge to clip AD at B. The algorithm then recomputes B's outcode as 0000. With the next iteration of the algorithm, AB is tested and is trivially accepted and displayed. 2. Consider the line segment EI Point E has an outcode of 0100, while point I's outcode is 1010. The results of the trivial tests show that the line can neither be trivally rejected or accepted.

45%	MATCHING BLOCK 179/191	SA	NCIS300 COMPUTER GRAPHICS 190372.pdf (D119943374)

Point E is determined to be an outside point, so the algorithm clips the line against the bottom edge of the window.

Now line EI has been clipped to be line FI. Line FI is tested and cannot be trivially accepted or rejected. Point F has an outcode of 0000, so the algorithm chooses point I as an outside point since its outcode is1010. The line FI is clipped against the window's top edge, yielding a new line FH. Line FH cannot be trivally accepted or rejected. Since H's outcode is 0010, the next iteration of the algorthm clips against the window's right edge, yielding line FG. The next iteration of the algorithm tests FG, and it is trivially accepted and display. After Clipping After clipping the segments AD and EI, the result is that only the line segment AB and FG can be seen in the window. 4.5 Midpoint Subdivision Method Midpoint Subdivision Algorithm Midpoint subdivision algorithm is an extension of the Cyrus Beck algorithm. This algorithm is mainly used to compute visible areas of lines that are present in the view port are of the sector or the image. It follows the principle of the bisection method and works similarly to

166 the Cyrus Beck algorithm by bisecting the line in to equal halves. But unlike the Cyrus Beck algorithm, which only bisects the line once, Midpoint Subdivision Algorithm bisects the line numerous times. Working
Midpoint Subdivision Algorithm also
follows the basic principle of coordinate geometry. It basically marks the gives line with two coordinates, say P 0 (x1 and x2)P 1 (y1 and y2). Now a mid point is made on the line, say M with coordinates (x1 + x2)/2 and (y1 + y2)/2. Now the line or the abscissa is divided into two equal parts. And each of the two lines is now tested for its position in the view port. Now first of all, the end points of the line is tested to see where it falls, if both the ends are inside the display box or the view port, it means that both the abscissa lie inside the box. and the whole line will be displayed, this is the case of acceptance. Similarly other lines are tested one by one. Now again if both the abscissa of the line is outside the view port, then the
whole is omitted to be displayed, and whole line is clipped, this is the case of rejecting the line.
Now if any one of the abscissa of any line lies outside the view port, the part outside is clipped, this called the case of partial acceptance,
and this line is now bisected
into two halves, and the line is again checked for its position in the view port. If both the abscissa of the line
are inside the view port, algorithm moves on to check the other line, and if any of the lines' abscissa in not inside the view port, again that line is bisected into two halves and the same operation is carried out, until the partial case of occurrence is eliminated. That is the algorithm follows the principle of either inside the view port or outside the view port, it does not clip the partial line that is out but directly eliminates the half of it.
Example or Sample Working Supposing there is a line in a 2D image, sap L. Now the image is divided into 9 sectors, and the middle is going to be considered. Now the line is marked as L1 and L2. It is partially appearing in the view port, hence following the Midpoint Subdivision Algorithm, it is bisected into LINE 1(L1, LM) and LINE 2 (LM, L2). And a check is done to find out there visibility in the view port, and the one which is not visible is eliminated, the one with partial visibility is selected for Midpoint Subdivision Algorithm again. Suppose the LINE 1 is
167 partially visible, so bisecting we get LINE 1(L1, LM1) and LINE 2 (LM1, LM). This procedure goes on till the case of partial visibility is eliminated. The strength of this algorithm over the Cohen-Sutherland algorithm is that it requires no floating point arithmetic to find the point of intersection with the line and the clip boundary. The midpoint subdivision algorithm clips a line by finding the endpoints of the visible portion of the line segment. Each endpoint can be found by an identical process and given appropriate hardware; this can be done in parallel for both endpoints. We still use the region codes do find line segments that are trivially accepted or rejected. When a line crosses the boundary, the midpoint is computed as follows: Xmid = (X1+X2)/2 Ymid = (Y1+Y2)/2 Essentially we have two scenarios that can exist and they look like the following: This algorithm wants to find the visible point of the line segment (P0P1) that is farthest from P0. Midpoint subdivision algorithm: Step#1: Is P1 visible? If it is, then P1 is the farthest point from P0 and the process is complete; otherwise, it is invisible, continue. Step#2: Is segment(P0P1) entirely invisible? If it is, then clip entire line; otherwise, continue. Step#3: We will take a guess at the farthest point from P0 which is Pm (midpoint of the segment (P0P1)). If segment(PmP1) can be trivially rejected, then go to Step#2 using segment(P0Pm); otherwise, go to Step#2 using segment(PmP1). What are the terminating conditions of the midpoint subdivision algorithm? Remember that while the process is going on for segment (P0P1) and a similar process is going on for segment(P1P0).
168 Problem: Given set window(10,50,10,50) and segment(P0P1) such that P0=(20,15) and P1=(20,8), show how the midpoint subdivision algorithm would proceed and clip the line segment(P0,P1) to the window. How would the midpoint subdivision algorithm handle the following case? 4.6Polygon Clipping Polygon clipping is one of those humble tasks computers do all the time. It's a basic operation in creating graphic output of all kinds.
There are several well-known polygon clipping algorithms, each having its strengths and weaknesses. The oldest one (from 1974) is called the Sutherland-Hodgman algorithm.
In its basic form, it
is relatively simple. It is also very efficient in two important cases, one being when the polygon is completely inside the boundaries, and the other when it's completely outside. The Laing-Barsky algorithm (1983) is a good deal more complicated, but in certain cases fewer intersections need to be calculated than for Sutherland-Hodgman. Therefore, it may be somewhat faster when many polygon lines intersect with the clipping boundaries. However, it is more sensitive to 'nasty' polygons. The Weiler algorithm (1977) is even more complicated, but it is the one you'll need if you want to clip a polygon against a non-rectangular window. Even more ways to clip a polygon exist. None of them is totally perfect. Often, it is possible to feed a weird polygon to an algorithm and retrieve an incorrect result. One of the vertices will disappear, or a ghost vertex will be created. Therefore, the hunt for the perfect clipping algorithm is still open.
169
A polygon is generally stored as a collection of vertices. Any clipping algorithm takes one collection, and outputs a new collection. A clipped polygon, after all, is also a polygon. Notice that the clipped polygon often will have more vertices than the unclipped one, but it can also have the same number, or less. If the unclipped polygon lies completely outside the clipping boundary,
the clipped polygon even has zero vertices.
Sutherland-Hodgman After some investigation, I opted for the old and trusted Sutherland-Hodgman method to attack the problem. For the project I was working on, the non-rectangular capabilities of Weiler would be overkill. With the current CPUs and their fast floating point operations, the slightly higher efficiency of Laing-Barsky hardly makes sense. In fact, after a rather crude experiment, I suspect it to be even somewhat slower.
Sutherland-Hodgman uses a
divide-and-conquer strategy to attack the problem. First, it clips the polygon against

the right clipping boundary.
The resulting, partly clipped polygon is then clipped against the top boundary,
and then the process is repeated for the two remaining boundaries. (
Of course, it also works in another order.) In a way, it is the most natural thing to do. If you had to clip a paper polygon with a pair of scissors, you would probably proceed the same way. Sutherland-Hodgman's divide-and-conquer strategy. To clip a polygon against a rectangular boundary window, successively clip against each boundary.
170 To clip against one boundary, the algorithm loops through all polygon vertices. At each step, it considers two of them, called 'previous' and 'current.' First, it determines whether these vertices are inside or outside the clipping boundary. This, of course, is simply a matter of comparing the horizontal or vertical position to the boundary's position. Then, it applies the following simple rules:
If 'previous' and 'current' are both inside: Output 'current.' If 'previous' is inside, and 'current' is outside: Output the intersection point of the corresponding edge and the clipping boundary. If 'previous' and 'current' are both outside: Do nothing. If 'previous' is outside, and 'current' is inside: Output the intersection
point, and then output 'current.'

86%	MATCHING BLOCK 181/191	SA	INF_2046.pdf (D164968115)

This way, you get a new polygon, clipped against one boundary, and ready to be clipped against the next boundary. The method works, but has one disadvantage: To clip against all four boundaries, you

have to store the intermediate, partly clipped polygons. It's evident that this is a costly operation.
171 Luckily, there are ways to avoid the intermediate storage. Instead of storing an output point, you can send it to the next stage immediately, where it can be clipped against the next boundary. The classical way to do this is to make one general boundary-clipping function, which calls itself recursively to put a vertex to the next stage. An extra parameter tells the function which boundary to use for clipping. Inside the function, a few switch statements account for the differences. A somewhat faster option is to create separate functions for the four boundaries, thus eliminating the nasty switch statements. However, this means developing and debugging four very similar functions, which is an open invitation for all kinds of trouble.

100%	MATCHING BLOCK 180/191	W

To clip polygons, we need to modify the line-clipping procedures

discussed in the previous section.

98%	MATCHING BLOCK 182/191	W

A polygon boundary processed with a line clipper may be displayed as a series of unconnected line segments, depending on the orientation of the polygon to the clipping window.

What we really want to display is a bounded area after
clipping.

94%	MATCHING BLOCK 183/191	W

For polygon clipping, we require an algorithm that will generate one or more closed areas that are then scan converted for the appropriate area fill. The output of a polygon clipper should be a sequence of vertices that defines the clipped polygon boundaries. Sutherland-Hodgeman Polygon Clipping

We can
correctly clip

94%	MATCHING BLOCK 184/191	W

a polygon by processing the polygon boundary as a whole against each window edge. This could be accomplished by processing all polygon vertices against each clip rectangle boundary

in turn. Beginning with the
initial set of polygon vertices, we could first clip the polygon against the left rectangle boundary to produce a new sequence of vertices. The new set of vertices could then successively passed to a

right boundary clipper, a bottom boundary clipper, and a top boundary clipper.

At each step, a new sequence of output vertices is generated and passed to the next window

boundary clipper. There are four possible cases when processing vertices in sequence around the perimeter of a polygon. As each pair of adjacent polvgon vertices is passed to a window boundary clipper,

we
make the following tests:
172 (1)

If the first vertex is outside the window boundary and the second vertex is inside, both the intersection point of the polygon edge with the window boundary and the second vertex are added to the output vertex list. (2) If both input vertices are inside the window boundary, only the second vertex is added to the output vertex list. (3) If the boundary and the second vertex is outside, only first vertex is inside the window, the edge intersection with the window boundary is added to the output vertex list. (4) If both input vertices are outside the window boundary, nothing is added to the output list.

Once all vertices have been processed for one clip window boundary, the output list of vertices is clipped against the next window boundary.

Convex polygons are correctly clipped by the Sutherland-Hodgeman algorithm, but concave polygons may be displayed with extraneous lines.

This occurs when the clipped polygon should have two or more separate sections. But since there is only one output vertex list, the last vertex in the list is always joined to the first vertex. There are several things we could d o to correctly display concave polygons. For one, we could split the concave polygon into two or more convex polygons and process each convex polygon separately. Another possibility is to modify the Sutherland- Hodgeman approach to check the final vertex list for multiple vertex points along any clip window boundary and correctly join pairs of vertices. Finally, we could use a more general polygon clipper, such as either the Weiler-Atherton algorithm or the Weiler algorithm. The Sutherland - Hodgman algorithm performs a clipping of a polygon against each window edge in turn. It accepts an ordered sequence of vertices v1, v2, v3, ..., vn and puts out a set of vertices defining the clipped polygon. This figure represents a polygon (the large, solid, upward pointing arrow) before clipping has occurred.
173 The following figures show how this algorithm works at each edge, clipping the polygon. a. Clipping against the left side of the clip window. b. Clipping against the top side of the clip window. c. Clipping against the right side of the clip window. d. Clipping against the bottom side of the clip window. Four Types of Edges As the algorithm goes around the edges of the window, clipping the polygon, it encounters four types of edges. All four edge types are illustrated by the polygon in the following figure. For each edge type, zero, one, or two vertices are added to the output list of vertices that define the clipped polygon. The four types of edges are: 1. Edges that are totally inside the clip window. - add the second inside vertex point 2. Edges that are leaving the clip window. - add the intersection point as a vertex 3. Edges that are entirely outside the clip window. - add nothing to the vertex output
174 list 4. Edges that are entering the clip window. - save the intersection and inside points as vertices How to Calculate Intersections Assume that we're clipping a polgon's edge with vertices at (x1,y1) and (x2,y2) against a clip window with vertices at (xmin, ymin) and (xmax,ymax). The location (IX, IY) of the intersection of the edge with the left side of the window is: i. IX = xmin ii. IY = slope*(xmin-x1) + y1, where the slope = (y2-y1)/(x2-x1) The location of the intersection of the edge with the right side of the window is: i. IX = xmax ii. IY = slope*(xmax-x1) + y1, where the slope = (y2-y1)/(x2-x1) The intersection of the polygon's edge with the top side of the window is: i. IX = x1 + (ymax - y1) / slope ii. IY = ymax Finally, the intersection of the edge with the bottom side of the window is: i. IX = x1 + (ymin - y1) / slope ii. IY = ymin Some Problems With This Algorithm 1. This algorithm does not work if the clip window is not convex. 2. If the polygon is not also convex, there may be some dangling edges. Weiler-Atherton Polygon Clipping Here, the vertex-processing procedures for window boundaries are modified so that concave polygons are displayed correctly.

This clipping procedure was developed as a method for identifying visible surfaces, and so it can be applied with arbitrary polygon-clipping regions. 175 The basic idea in this algorithm is that instead of always proceeding around the polygon edges as vertices are processed, we sometimes want to follow the window boundaries. Which path we follow depends on the polygon-processing direction (clockwise or counterclockwise) and whether tile pair of polygon vertices currently being processed represents an outside-to-inside pair or an inside-to-outside pair. For clockwise processing of polygon vertices, we use the following rules: • For an outside-to-inside pair of vertices, follow the polygon boundary. • For an inside-to-outside pair of vertices,. follow the window boundary in a clockwise direction.

An improvement on the Weiler-Atherton algorithm is the Weiler algorithm, which applies constructive solid geometry ideas to clip an arbitrary polygon against any polygon dipping region.

Sutherland-Hodgman Clipping Algorithm Clipping a polygon can produce a polygon with fewer vertices or a polygon with more vertices. Give an example of each. Remember that a polygon consists of an ordered set of vertices or an ordered set of edges. The polygon is drawn by connecting the vertices in order or more specifically by connecting successive pairs of consecutive vertices with straight lines which produces the edges. So how does the clipping algorithm work? We can call the clipping algorithm once for each vertex of the original polygon. For each vertex, the algorithm will either return: 1) no vertex at all 2) the original vertex 3) one or more new vertices Essentially, we use a divide and conquer algorithm such that we solve a series of smaller, simpler problems which combine to solve the larger problem. What is a simpler problem than clipping a polygon to four regions? Clipping a polygon to one region. We can note that successive clips against four clip boundaries, where each boundary is defined by the window, will clip the polygon. The following is a good example.
176 1) Before hitting the specifics of the Sutherland-Hodgman algorithM, we will discuss some generalities: 2) The algorithm accepts a series of vertices: V1,V2,...Vn. 3) In general, the polygon edges are from Vi to Vi+1 and from Vn to V1. The algorithm will clip against an edge and then output another series of vertices that define the clipped polygon to that edge. Remember, we must clip againt all four boundaries. 4.7 Curve design techniques- Parametric Equation of a Circle, 2-D Curve Generation, LeGrange interpolated Curve, B-Spline Curves, Bezier Curves, 3-D Curves Computer graphics instructors frequently skip the curves and surfaces chapters due to the problem of "too much mathematics". Fortunately, with our tools DesignMentor, we can take an intuitive approach, avoid the use of the involved mathematics, and get to the core of curve and surface design quickly. These tools have been classroom tested by our junior and senior students and the results have been very successful. In this workshop all four types of parametric curves and surfaces (e.g., Bézier, rational Bézier, B-spline and NURBS) will be covered. Our discussion starts with the hierarchy of these curves and surfaces: For each curve type, we shall discuss the meaning and the merit of its definition and
177 important and fundamental properties: • The definition of each curve. • Important properties of each curve, which include the convex hull property, partition of unity, affine and/or projective invariance, local modification scheme and curve tracing. The following shows the computation steps of de Casteljau's and de Boor's algorithms for computing a point on a Bézier and a B-spline curve. The yellowish polygons are the convex hulls at a particular point on the curve and the line segments are the de Casteljau's and de Boor's nets. • Basic shape modification techniques. These include changing the position of a control point, the value of the weight of a control point, and the value of a knot. The local modification scheme, which states that modifying the above mentioned quantities will only affect the curve locally, will also be discussed. The following figures show the effect of changing the position of a selected control point. It is clear that the left part of the curve does not change even though the position of the selected control point is changed drastically. The following shows the effect of changing the weight of a selected control point. The
178 weight of the selected control point is 1. The left figure shows the result of increasing the weight to 5 (the curve is pulled toward the selected control point), and the right figure shows the result of decreasing the weight to 0.2 (the curve is pushed away from the selected control point). • Fundamental algorithms. We shall discuss three extremely important fundamental algorithms, namely: knot insertion - inserting a new knot into the knot vector of a B- spline or a NURBS curve, degree elevation - increasing the degree of a curve by one, and curve subdivision - dividing a curve into two segments. These algorithms perform some modification to the defining parameters of a curve without changing its shape. The following shows the effect of inserting a new knot 0.25 into the knot vector of the left curve. The right figure is the result. Please note that the shape of the curve does not change. The only change is the three marked control points being replaced by four new control points.
179 The following shows the effect of subdividing the left curve at 0.45 into two curve segments, each of which has its own control points and knot vector. Thus, one can modify each component without affecting the other. For the surface part, we shall start with the meaning of tensor product surfaces and focus on B-spline and NURBS surfaces. Almost all properties and algorithms for curves have natural and easy extensions to surfaces. For example, increasing (resp., decreasing) the weight of a selected control point will pull (resp., push) the surface toward (resp., away) from that control point. The following figures illustrate the effect of increasing the weight of the selected control point from 1 to 10. Both figures show the control points and control nets of the surfaces. The most interesting application of the theory is the cross section technique. This technique reduces the design of a surface to the design of a few curves, and hence simplifies the whole

180 design process. The cross section design technique requires one or more profile curves and a trajectory curve. The surface is constructed by sliding the profile curves, possibly with transformations (i.e., rotation and scaling), along the trajectory curve. Many classical surfaces can be constructed this way. This includes ruled surfaces, surfaces of revolution, swung surfaces and skinned surfaces. All of them will be discussed in some detail. In our system, the input curves are NURBS ones and the output is a NURBS surface. The following shows the generation of a hyperboloid of one sheet as a ruled surface from two circles: The following shows the construction of a vase as a surface of revolution from its profile curve: The following is a swung surface. It is constructed by swinging an up-side-down omega symbol about a squarish curve. As a result, the up-side-down omega curve provides the profile in the vertical direction, while the squarish curve gives the shape of the horizontal direction.
181 In general, a curve may not have a simple mathematical definition. In the past, curves and shapes were defined using different geographical notations which were not very easy to manipulate. Different techniques were used to design different types of curves and surfaces. These methods include the following among others: 1. Polygon meshes: This is possibly the most common boundary representation technique. Here a 3-D graphical object is modelled as a set of polygonal surfaces that enclose the object interior. The surfaces may be plain or curved in nature. Standard graphical objects can be represented by plane polygonal surfaces. But for complex object boundaries, we need spline surfaces or other schemes. Objects defined with polygonal surfaces look like a polygon mesh. 2. Parametric instancing: Generally the equation of a curve is used to determine the co- ordinate points on the curve. Parametric Equation of a Circle A circle can be defined as the locus of all points that satisfy the equations x = r cos(t) y = r sin(t) where x,y

62%	MATCHING BLOCK 191/191	SA	ODMCA-404T Palwinder 10th Feb 2020.docx (D63728007)

are the coordinates of any point on the circle, r is the radius of the circle

and t is the parameter - the angle subtended by the point at the circle's center.
182 Coordinates of a point on a circle Looking at the figure above, point P is on the circle at a fixed distance r (the radius) from the center. The point P subtends an angle t to the positive x-axis. Click 'reset' and note this angle initially has a measure of 40°. Using trigonometry, we can find the coordinates of P from the right triangle shown. In this triangle the radius r is the hypotenuse. The x coordinate is therefore r cos(t) and the y coordinate is r sin(t) To see why this is, recall that in a right triangle, the sine of an angle is the opposite side divided by the hypotenuse. In the figure on the right In the applet above, the side opposite t has a length of y, the y coordinate of P. The hypotenuse is the radius r. Therefore Multiply both sides by r By similar means we find that The parametric equation of a circle From the above we can find the coordinates of any point on the circle if we know the radius and the subtended angle. So in general we can say that a circle centered at the origin, with radius r, is the locus of all points that satisfy the equations x = r cos(t) y = r sin(t) for all values of t It also follows that any point not on the circle does not satisfy this pair of equations. Example
183 If we have a circle of radius 20 with its center at the origin, the circle can be described by the pair of equations x = 20 cos(t) y = 20 sin(t) What if the circle center is not at the origin? Then we just add or subtract fixed amounts to the x and y coordinates. If we let h and k be the coordinates of the center of the circle, we simply add them to the x and y coordinates in the equations, which then become: x = h + r cos(t) y = k + r sin(t) This is really just translating ("moving") the circle from the origin to its proper location. In the figure above, drag the center point C to see this. What does 'parametric' mean? In the above equations, the angle t (theta) is called a 'parameter'. This is a variable that appears in a system of equations that can take on any value (unless limited explicitly) but has the same value everywhere it appears. A parameter values are not plotted on an axis. Algorithm for drawing circles This form of defining a circle is very useful in computer algorithms that draw circles and ellipses. In fact, all the circles and ellipses in the applets on this site are drawn using this equation form. Other forms of the equation Using the Pythagorean Theorem to solve the triangle in the figure above we get the more common form of the equation of a circle
184 To demonstrate that these forms are equivalent, consider the figure on the right. In the right triangle, we can see that Recall the trig identity Substitute x/r and y/r into the identity: Remove the parentheses: Multiply through by r 2 2-D Curve Generation Curves are one of the most essential objects to create high-resolution graphics. While using many small polylines allows creating graphics that appear smooth at fixed resolutions, they do not preserve smoothness when scaled and also require a tremendous amount of storage for any high-resolution image. Curves can be stored much easier, can be scaled to any resolution without losing smoothness, and most importantly provide a much easier way to specify real-world objects. All of the popular curves used in graphics are specified by parametric equations. Instead of specifying a function of the form y = f(x), the equations y = fy(u) and x = fx(u) are used. Using parametric equations allows curves that can double back and cross themselves, which are impossible to specify in a single equation in the y = f(x) case. Parametric equations are also easier to evaluate: changing u results in moving a fixed distance along the curve, while in the traditional equation form much work is needed to determine

185 whether to step through x or y, and determining how large a step to take based on the slope. The remainder of this document describes LeGrange interpolated curves, cubic B-splines, and Bezier curves. While LeGrange interpolated curves are not often used in computer graphics, they help give a simple foundation for understanding Bezier curves and B-splines, the most commonly used graphical curves. LeGrange interpolated Curve The goal of LeGrange interpolation is to create a smooth curve that passes through an ordered group of points. When used in this fashion, these points are called the control points. The general idea, as with all curve algorithms, is to use the control points to generate parametric equations. To draw the curve, all that is then needed is to step through u at some small amount, drawing straight lines between the calculated points. The smaller the step size the more smooth the curve will appear. The step size can be calculated based on the amount of pixels available in the output image. To do LeGrange interpolation, we wish to specify a curve that will pass through any number of control points. The curve's function can be constructed as a sum of terms, one for each control point. fx(u) = sum from i = 1 to n of x[i] B[i](u) fy(u) = sum from i = 1 to n of y[i] B[i](u) Each function B[i](u) specifies how much the ith control point effects the position of the curve. This can be thought of as a function specifying how much the ith control point draws the curve towards it. The name for these functions are Blending Functions. If for some control point i the value of this function is 1 and for every other control point the value of this function is 0, the curve will go through i at that value. To achieve the goal of LeGrange interpolation, getting a curve that passes through all the specified control points, the blending functions must be defined so that the curve will go through each control point in the proper order. This can be done by creating blending functions where the first control point is passed through (has total control) at u = -1, the
186 second control point is passed through at u = 0, the third at u = 1, and so on. A mathematical expression that does so is: u (u - 1) (u - 2) . . . [u - (n - 2)] At u = -1 this expression is: -1 (-2) (-3) . . . (1 - n) So dividing the first by the second gives us 1 when u = -1, and gives us 0 when u is a whole number greater then -1 and less then or equal to n. Using this method for the other blending functions gives us the same type of behavior for the other control points. This function defines the LeGrange interpolation blending functions.
(u + 1) (u) (u - 1) ... [u - (i - 3)] [u - (i - 1)] ... [u - (
i - 2)]
B[i](u) = -- (i - 1) (i - 2) (i - 3) ... (1) (-1) ... (i - n) To draw a curve using this, it is necessary to decide how many control points we will use for the basis of the curve, and then calculate the blending functions B[1](u) to B[n](u), where n is the number of control points. The final functions for x, y, and z
are then:
x = x[1] B[1](u) + x[2] B[2](u) + x[3] B[3](u) + . . . + x[n] B[n](u)
y = y[1] B[1](u) + y[2] B[2](u) + y[3] B[3](u) + . . . + y[
n] B[n](u) Where x[i] is the x coordinate of the ith control point, and y[i] is the y coordinate of the ith control point. The number of control points used is most commonly 4. In this case, the calculated curve is particularly good between the second and third control points (0 &gt;= u &gt;= 1). Thus to draw a full
LeGrange
interpolated curve using many control points, it is common to work with sets of four points. First start with the first through fourth control points, and step through u from 0 to 1 in small steps, drawing straight lines between each. When u = 1 is reached, switch to using the second through fifth control points. Now step through u from 0 to 1 in small steps again. Repeat this, each time shifting down along the control points in the curve
187 by one point until the end of the curve is reached. At this point all of the curve will have been drawn, except for the first and last segments (the curve between the first and second control points and the curve between the second to last and last control points). These can be dealt with as special cases, calculating from -1 &gt;= u &gt;= 0 using the first four control points and 1 &gt;= u &gt;= 2 using the last four control points. Some sample LeGrange interpolated curves appear below. The last two of the above images shows why LeGrange interpolated curves are not usually used in graphics. Although the control points form a straight line, the curve wiggles between the control points. This results because the blending functions sum to 1 at the control points, but do not necessarily do so at fractional values. To eliminate the wiggle we would need to dividing the value of each blending function by the sum of the blending function values before using them. However, there is also another problem. While each section of the curve connects to the next section at a control point, the slopes for the connected curves at the control point may be different. This results in the ability to have corners at control points. This results because while at control points only that control point has control, at any other part of the curve all points have some control. The third problem is that the
LeGrange
curve goes outside the convex hull of the control points. This means that clipping LeGrange curves must be done with every single line segment drawn,
188 instead of clipping the control points and then just drawing the curve. This adds much computation.
B-Spline Curves B-splines correct the deficiencies of LeGrange interpolated curves by defining blending functions that vary each control point's control from 0 far away from the point to its maximum near the point. This can be done by making it so the curve does not pass through each control point, but instead just passes near them. The sum of the B-Spline blending functions also always sum to 1, and the slopes between curve segments are continuous. Finally, the final
curve always lies within the convex hull of the control points.
While B-splines can be of higher order, usually cubic B-splines are used. Cubic functions allow for enough curve flexibility for most tasks, and are easier to work with then higher order functions since they behave more predictably (higher order functions may get unexpected wiggles and kinks). The cubic B-Spline blending functions work over four control points. The blending functions appear below. Note that special blending functions are needed for the first and last two sections of the B-Spline so that it passes through the first and last points. First section:
B[1](

u) = (1 - u)^3 B[2](u) = 21 u^3 / 12 - 9 u^2 / 2 + 3 u B[3](u) = -11 u^3 / 12 + 3 u^2 / 2 B[4](u) = u^3 / 6
Second section:
B[1](u) = (1 - u)^3 / 4 B[2](
u) = 7
u^3 / 12 - 5
u^2 / 4 + u / 4 + 7 / 12 B[3](u) = - u^3 / 2 + u^2 / 2 + u / 2 + 1 / 6
189 B[4](u) = u^3 / 6
Middle sections:
B[1](u) = (1 - u)^3 / 6 B[2](u) = u^3 / 2 - u^2 + 2 / 3 B[3](
u) = - u^3 / 2 + u^2 / 2 +
u / 2 + 1 / 6 B[4](u) = u^3 / 6
Second to last section: B[i](u) = F[5 - i](1 - u) where F[i](u) is the ith blending function for the second section. Last section: B[i](u) = F[5 - i](1 - u) where F[i](u) is the ith blending function for the first section. Note that B-Spines are designed to eliminate sharp corners, as seen above. A sharp corner can be obtained, however, by using duplicate sequential control points. Using three identical control points causes the curve to pass through the point, as illustrated below.
Bezier Curves
190 Bezier curves are another very popular curve type. Bezier curves, unlike B-splines, must have precisely n control points, where n + 1 is the degree of the Bezier polynomial. To create a longer curve, it is necessary to connect multiple Bezier curves. This is usually done by making the last control point of one curve
the same as the first control point of the next curve. Bezier curves pass through the first and last control points
of each curve segment, however, which makes them quite easy to work with and popular for use in interactive design programs. Bezier curves, like B-Spline curves, always lie within the convex hull of the control points, and always have the sum of the basis functions add to 1. The idea behind Bezier curves is quite simple. To create a smooth curve, one intuitive mechanism is to first connect the four control points with lines. Then draw new lines connecting the midpoints of those lines, and more new lines to connect the midpoints of the new lines, and so forth until the resulting curve is sufficiently smooth. This is illustrated below. While it is possible to define a curve type based on connecting the last third of one segment to the first third of the next, or the last fourth of one segment to the first fourth of the next, and so on, using midpoints allows for a very efficient subdivision algorithm to be derived. The basis functions for cubic Bezier curves, which have the same results as the above subdivision process, appear below. These are based on Bernstein polynomials.
B[1](u) = (1 - u)^3 B[2](u) = 3 u (1 - u)^2 B[3](u) = 3 u^2 (1 - u) B[4](
u) = u^3
191 The DeCasteljau representation of Bezier
curves is usually used, however. This is a recursive definition of the above polynomials that is based on the subdivision illustrated earlier. r r-1 r-1 p[i](u) = (1 - u) p[i](u) + u p[i + 1](u) where p[1], p[2], . . ., p[n] are the control points r = 1, 2, . . ., n i = 0, 1, . . ., n - r p[i](u) = p[i] A point on the curve is given by p[0]^n (u). Using this derivation, it is possible to see a simple recursive subdivision process for generating cubic Bezier curves. This works by taking the original Bezier curve and breaking it into two smaller Bezier curves, as illustrated below. These Bezier curves can then be broken into smaller Bezier curves, and so on, until the control points of the Bezier curve are very close to a straight line. When this happens, a straight line is drawn between the first and last control points, and the recursion pops back up. At each stage of subdivision, the control points of the sub-curves are calculated by:
192
q[0] = p[0] q[1] = (p[0] + p[1]) / 2 q[2] = (q[1] / 2) + (p[1] + p[2]) / 4 q[3] = (q[2] + r[1]) / 2 r[0] = q[3] r[1] = (p[1] + p[2]) / 4 + (
r[2] / 2) r[2] = (p[2] + p[3]) / 2 r[3] = p[3] where p[1..4] are the control points of the original Bezier, and q[1..4] and r[1..4] are the control points of the sub-curves. This derivation was first done by Lane. The linearity test for the above method is quite simple: all that needs to be done is calculate the sum of the distance from points p[1] and p[2] to the line connecting p[0] and p[4], and see if this is greater then some tolerance. 3-
D Curves All of the above can be applied to 3-D curves by including a third equation to specify z: fz(u) = sum from i = 1 to n of z[i] B[i](u) and using the same blending functions. The optimized version of the Bezier curve formulation, however, would need to be changed. Exercises 1. What is a viewport? 2. An algorithm goes around the edges of the window, clipping the polygon, it encounters four types of edges. List the four types. 3. What do we mean by line clipping? 4. How are concave and convex polygons clipped? 5. What is the advantage of the Cyrus Beck algorithm Midpoint over the Subdivision
193 Algorithm? 6. Differentiate between B-Spline curves and Bezier Curves. 7. What is the goal of LeGrange interpolation? 8. Write the basis functions for cubic Bezier curves, which are based on Bernstein polynomials.

Hit and source - focused comparison, Side by Side

Submitted text	As student entered the text in the submitted document.
Matching text	As the text appears in the source.

1/191	SUBMITTED TEXT 50 WORDS	83% MATCHING TEXT 50 WORDS
	of Computer Graphics Introduction to Computer Graphics Computer graphics is an art of drawing pictures, lines, charts, etc using computers with the help of programming. Computer graphics is made up of number of pixels. Pixel is the smallest graphical picture or unit represented on the computer screen.	of Computer Computer Graphics What is computer Graphics? Computer graphics is an art of drawing pictures, lines, charts, etc. using computers with the help of programming. Computer graphics image is made up of number of pixels. Pixel is the smallest graphical unit represented on the computer screen.

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

2/191	SUBMITTED TEXT 118 WORDS	98% MATCHING TEXT 118 WORDS
	Interactive Computer Graphics: Interactive Computer Graphics involves a two way communication between computer and user. Here the observer is given some control over the image by providing him with an input device, for example, the video game controller of the ping pong game. This helps him to signal his request to the computer. The computer on receiving signals from the input device can modify the displayed picture appropriately. To the user it appears that the picture is changing instantaneously in response to his commands. He can give a series of commands, each one generating a graphical response from the computer. In this way he maintains a conversation, or dialogue, with the computer. Interactive computer graphics	

SA INF_2046.pdf (D164968115)

3/191	SUBMITTED TEXT 90 WORDS	94% MATCHING TEXT 90 WORDS
	indirect ways. For example, it helps to train the pilots of our airplanes. We can create a flight simulator which may help the pilots to get trained not in a real aircraft but on the grounds at the control of the flight simulator. The flight simulator is a mock up of an aircraft flight deck, containing all the usual controls and surrounded by screens on which we have the projected computer generated views of the terrain visible on take off and landing. Flight simulators have many advantages over	

SA INF_2046.pdf (D164968115)

4/191	SUBMITTED TEXT 27 WORDS	94% MATCHING TEXT 27 WORDS
	real aircrafts for training purposes, including fuel savings, safety, and the ability to familiarize the trainee with a large number of the world's airports. •	

SA INF_2046.pdf (D164968115)

5/191	SUBMITTED TEXT	53 WORDS	49%	MATCHING TEXT	53 WORDS

Computer Graphics Design processes A major use of computer graphics is in design processes, particularly for engineering and architectural systems, but almost all products are now computer designed. Generally referred to as CAD, computer-aided design methods are now routinely used in the design of buildings, automobiles, aircraft, watercraft, spacecraft, computers, textiles, and

SA CG book _for_publishing_22_12_2018.docx (D46266915)

6/191	SUBMITTED TEXT	20 WORDS	86%	MATCHING TEXT	20 WORDS

major application area is presentation graphics, used to produce illustrations for reports or to generate 35-mm slides or transparencies

SA Computer Graphics.pdf (D165747830)

7/191	SUBMITTED TEXT	66 WORDS	74%	MATCHING TEXT	66 WORDS

use with projectors. Presentation graphics is commonly used to summarize financial, statistical, mathematical, scientific, and economic data for research reports, consumer information bulletins, and other types of reports. Workstation devices and service bureaus exist for converting screen displays into 35-mm slides or overhead transparencies for use in presentations. Typical examples of presentation graphics are bar charts, line graphs, surface graphs, pie charts, and other displays

SA Computer Graphics.pdf (D165747830)

8/191	SUBMITTED TEXT	28 WORDS	71%	MATCHING TEXT	28 WORDS

Animations are often used in CAD applications. Real-time animations using wise frame displays on a video monitor are useful for testing performance of a vehicle or system.

SA Computer Graphics.pdf (D165747830)

9/191	SUBMITTED TEXT	32 WORDS	85%	MATCHING TEXT	32 WORDS

Animations in virtual reality environments are used to determine how vehicle operators are affected by certain motions. When object designs are complete, or nearly complete, realistic lighting models and surface rendering

SA Computer Graphics.pdf (D165747830)

10/191	SUBMITTED TEXT	23 WORDS	84%	MATCHING TEXT	23 WORDS

applied to produce displays that will show the appearance of the final product. 7

SA Computer Graphics.pdf (D165747830)

11/191	SUBMITTED TEXT 94 WORDS	92% MATCHING TEXT 94 WORDS
	show the positioning of rooms, doors, windows, stairs, shelves, counters, and other building features. Working from the display of a building layout on a video monitor, an electrical designer can try out arrangements for wiring, electrical outlets, and fire warning systems. Also, facility-layout packages can be applied to the layout to determine space utilization in an office or on a manufacturing floor. Realistic displays of architectural designs permit both architects and their clients to study the appearance of a single building or a group of buildings, such as a campus or industrial complex.	

SA Computer Graphics.pdf (D165747830)

12/191	SUBMITTED TEXT 24 WORDS	72% MATCHING TEXT 24 WORDS
	In addition to realistic exterior building displays, architectural CAD packages also provide facilities for experimenting with three-dimensional interior layouts and lighting. Fine art	

SA Computer Graphics.pdf (D165747830)

13/191	SUBMITTED TEXT 25 WORDS	75% MATCHING TEXT 25 WORDS
	fine art and commercial art applications. Artists use a variety of computer methods, including special-purpose hardware, artist's paintbrush (such as Lumens), other paint packages (	

SA CG book _for_publishing_22_12_2018.docx (D46266915)

14/191	SUBMITTED TEXT 21 WORDS	87% MATCHING TEXT 21 WORDS
	Mathematics), CAD packages, desktop publishing software, and animation packages that provide facilities for designing object shapes and specifying object motions.	

SA CG book _for_publishing_22_12_2018.docx (D46266915)

15/191	SUBMITTED TEXT 38 WORDS	100% MATCHING TEXT 38 WORDS
	allows artists to "paint" pictures on the screen of a video monitor. Actually, the picture is usually painted electronically on a graphics tablet (digitizer) using a stylus, which can simulate different brush strokes, brush widths, and colors.	allows artists to paint pictures on the screen of a video monitor. Actually, the picture is usually painted electronically on a graphics tablet (digitizer) using a stylus, which can simulate different brush strokes, brush widths, and colors.

W https://studio.eecs.umich.edu/confluence/download/attachments/17827062/mathematicsforcomputergrap ...

16/191	SUBMITTED TEXT 31 WORDS	100% MATCHING TEXT 31 WORDS
	used in making motion pictures, music videos, and television shows. Sometimes the graphics scenes are displayed by themselves, and sometimes graphics objects are combined with the actors and live scenes.	

SA CG book _for_publishing_22_12_2018.docx (D46266915)

17/191	SUBMITTED TEXT	14 WORDS	100%	MATCHING TEXT	14 WORDS

Music videos use graphics in several ways. Graphics objects can be combined with

SA Computer Graphics.pdf (D165747830)

18/191	SUBMITTED TEXT	30 WORDS	97%	MATCHING TEXT	30 WORDS

live action, or graphics and image processing techniques can be used to produce a transformation of one person or object into another (morphing). 8

SA Computer Graphics.pdf (D165747830)

19/191	SUBMITTED TEXT	34 WORDS	78%	MATCHING TEXT	34 WORDS

physical systems, physiological systems, population trends, or equipment, such as the color coded diagram, can help trainees to understand the operation of the system. For some training applications, special systems are designed. Examples

SA CG book _for_publishing_22_12_2018.docx (D46266915)

20/191	SUBMITTED TEXT	33 WORDS	93%	MATCHING TEXT	33 WORDS

air traffic control personnel. Some simulators have no video screens; for example, a flight simulator with only a control panel for instrument flying. But most simulators provide graphics screens for visual operation.

SA Computer Graphics.pdf (D165747830)

21/191	SUBMITTED TEXT	79 WORDS	68%	MATCHING TEXT	79 WORDS

Scientists, engineers, medical personnel, business analysts, and others often need to analyze large amounts of information or to study the behavior of certain processes. Numerical simulations carried out on supercomputers frequently produce data files containing thousands and even millions of data values. Similarly, satellite cameras and other sources are amassing large data files faster than they can be interpreted. Scanning these large sets of numbers to determine trends and relationships is a tedious and ineffective process. But if

SA Computer Graphics.pdf (D165747830)

22/191	SUBMITTED TEXT	14 WORDS	100%	MATCHING TEXT	14 WORDS
In computer graphics, a computer is used to create a picture. Image processing,				In computer graphics, a computer is used to create a picture. Image processing	

W https://studio.eecs.umich.edu/confluence/download/attachments/17827062/mathematicsforcomputergrap ...

23/191	SUBMITTED TEXT	21 WORDS	92%	MATCHING TEXT	21 WORDS

scientists, and others use visual techniques to analyze mathematical functions and processes or simply to produce interesting graphical representations.

SA Computer Graphics.pdf (D165747830)

24/191	SUBMITTED TEXT	19 WORDS	97%	MATCHING TEXT	19 WORDS

Although methods used in computer graphics and Image processing overlap, the two areas concerned with fundamentally different operations.

SA Computer Graphics.pdf (D165747830)

25/191	SUBMITTED TEXT	80 WORDS	90%	MATCHING TEXT	80 WORDS

applies techniques to modify or interpret existing pictures, such as photographs and TV scans. Two principal applications of image processing are (1) improving picture quality and (2) machine perception of visual information, as used in robotics. To apply image processing methods, we first digitize a photograph or other picture into an image file. Then digital methods can be applied to rearrange picture parts, to enhance color separations, or to improve the quality of shading. These techniques are used

SA Computer Graphics.pdf (D165747830)

26/191	SUBMITTED TEXT	50 WORDS	65%	MATCHING TEXT	50 WORDS

retouching and rearranging of sections of photographs and other artwork. Similar methods are used to analyze satellite photos of the earth and photos of galaxies. Medical applications 9 also make extensive use of image processing techniques for picture enhancements, in tomography and in simulations

SA Computer Graphics.pdf (D165747830)

27/191	SUBMITTED TEXT	12 WORDS	95%	MATCHING TEXT	12 WORDS
Rendering Rendering is the process of generating an image from a 2				Rendering Time: Rendering is the process of generating an image from a	

W https://docplayer.net/53449703-Computergraphics-dcap504.html

28/191	SUBMITTED TEXT	15 WORDS	89%	MATCHING TEXT	15 WORDS
programming package provides an extensive set of graphics functions that can be used in				programming package provides an extensive set of graphics function that can be used in	

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

29/191	SUBMITTED TEXT 37 WORDS	98% MATCHING TEXT 37 WORDS
	Ray Tracing In computer graphics, ray tracing is a technique for generating an image by tracing the path of light through pixels in an image plane and simulating the effects of its encounters with virtual objects.	Ray tracing method In computer graphics, ray tracing is a technique for generating an image by tracing the path of light through pixels in an image plane and simulating the effects of its encounters with virtual objects.

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

30/191	SUBMITTED TEXT 15 WORDS	75% MATCHING TEXT 15 WORDS
	can be categorized according to whether they deal with output, input, attributes, transformations, viewing,	can be classified according to whether they deal with graphics output, input, attributes, transformation viewing,

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

31/191	SUBMITTED TEXT 20 WORDS	71% MATCHING TEXT 20 WORDS
	generating picture components (straight lines, polygons, circles, and other figures), setting color and intensity values, selecting views, and applying	

SA INF_2046.pdf (D164968115)

32/191	SUBMITTED TEXT 15 WORDS	75% MATCHING TEXT 15 WORDS
	can be categorized according to whether they deal with output, input, attributes, transformations, viewing,	can be classified according to whether they deal with graphics output, input, attributes, transformation viewing,

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

33/191	SUBMITTED TEXT 29 WORDS	78% MATCHING TEXT 29 WORDS
	The basic building blocks for pictures are referred to as output primitives. They include character strings and geometric entities, such as points, straight lines, curved Lines, filled areas (	The basic building blocks for pictures are referred to as output primitives. They includes character, string, and geometry entities such as point, straight lines, curved lines, filled areas

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

34/191	SUBMITTED TEXT 10 WORDS	100% MATCHING TEXT 10 WORDS
	color specifications, line styles, text styles, and area-filling patterns.	color specifications, line styles, text styles and area filling patterns.

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

35/191	SUBMITTED TEXT 11 WORDS	100% MATCHING TEXT 11 WORDS
	change the size, position, or orientation of an object within	change the size, position or orientation of an object within

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

36/191	SUBMITTED TEXT 19 WORDS	83% MATCHING TEXT 19 WORDS
	three buttons are usually included on the top of the mouse for signaling the execution of some operation,	Three buttons are placed on the top of the mouse for signaling the execution of some operation.? ?

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

37/191	SUBMITTED TEXT 12 WORDS	95% MATCHING TEXT 12 WORDS
	provides users with a variety of functions for creating and manipulating	

SA INF_2046.pdf (D164968115)

38/191	SUBMITTED TEXT 21 WORDS	78% MATCHING TEXT 21 WORDS
	trackball is a ball that can be rotated with the fingers or palm of the hand to produce screen-cursor movement.	Trackball is ball that can be rotated with the finger or palm of the hand to produce cursor movement.?

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

39/191	SUBMITTED TEXT 19 WORDS	83% MATCHING TEXT 19 WORDS
	attached to the ball, measure the amount and direction of rotation. Trackballs are often mounted on keyboards or	attached to the ball, measure the amount and direction of rotation.? ? They are often mounted on keyboard or

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

40/191	SUBMITTED TEXT 54 WORDS	77% MATCHING TEXT 54 WORDS
	Strain gauges measure the amount of pressure applied to the spaceball to provide input for spatial positioning and orientation as the ball is pushed or pulled in various directions. Spaceballs are used for three-dimensional positioning and selection operations in virtual-reality systems, modeling, animation, CAD, and other applications. Joysticks- A joystick consists of	strain gauges measure the amount of pressure applied to the space ball to provide input for spatial positioning and orientation as the ball is pushed or pulled in various directions.? ? Space balls are used in 3D positioning and selection operations in virtual reality system, modeling, animation, CAD and other application.? Joysticks ? A joy stick consists of

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

41/191	SUBMITTED TEXT 15 WORDS	100% MATCHING TEXT 15 WORDS
	mounted on a base that is used to steer the screen cursor around. Most	mounted on a base that is used to steer the screen cursor around.? ? Most

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

42/191	SUBMITTED TEXT 24 WORDS	77% MATCHING TEXT 24 WORDS
	a trackball is a two- dimensional positioning device, a spaceball provides six degrees of freedom. Unlike the trackball, a spaceball does not actually move.	

SA Unit-3 (Part-III).docx (D77528823)

43/191	SUBMITTED TEXT 18 WORDS	58% MATCHING TEXT 18 WORDS
	series of sensors that detect hand and finger motions. Electromagnetic coupling between transmitting antennas and receiving antennas	series of sensors that detect hand and figure motions.? ? Electromagnetic coupling is used between transmitter and receiver antennas

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

44/191	SUBMITTED TEXT 12 WORDS	100% MATCHING TEXT 12 WORDS
	be structured as a set of three mutually perpendicular coils, forming	be structured as a set of three mutually perpendicular coils forming 3

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

45/191	SUBMITTED TEXT 20 WORDS	84% MATCHING TEXT 20 WORDS
	Cartesian coordinate system. Input from the glove can be used to position or manipulate objects in a virtual scene.	Cartesian coordinates system.? ? Input from the glove can be used to position or manipulate object in a virtual scene.?

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

46/191	SUBMITTED TEXT 14 WORDS	84% MATCHING TEXT 14 WORDS
	Potentiometers mounted at the base of the joystick measure the amount of movement,	

SA Unit-3 (Part-III).docx (D77528823)

47/191	SUBMITTED TEXT 17 WORDS	100% MATCHING TEXT 17 WORDS
	A common device for drawing, painting, or interactively selecting coordinate positions on an object is a	

SA Unit-3 (Part-III).docx (D77528823)

48/191	SUBMITTED TEXT 17 WORDS	80% MATCHING TEXT 17 WORDS
	devices can be used to input coordinate values in either a two-dimensional or a three-dimensional space.	devices could be used to input co- ordinate values in either a two-dimensional or a three dimensional space.

W https://odl.ptu.ac.in/SLM/msc%20it/3RD/MSIT_301_Computer_Graphics.pdf

49/191	SUBMITTED TEXT 11 WORDS	83% MATCHING TEXT 11 WORDS
	black-and-white photos, or text can be stored for computer processing	black and white photos or text and can stored for computer processing

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

50/191	SUBMITTED TEXT 13 WORDS	100% MATCHING TEXT 13 WORDS
	by passing an optical scanning mechanism over the information to be stored.	by passing an optical scanning mechanism over the information to be stored.? ?

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

51/191	SUBMITTED TEXT 11 WORDS	100% MATCHING TEXT 11 WORDS
	We can also apply various image-processing methods to modify the	We can also apply various image processing methods to modify the

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

52/191	SUBMITTED TEXT 12 WORDS	100% MATCHING TEXT 12 WORDS
	used to select screen positions by detecting the light coming from	used to select screen positions by detecting the light coming from

W https://docplayer.net/53449703-Computergraphics-dcap504.html

53/191	SUBMITTED TEXT 35 WORDS	82% MATCHING TEXT 35 WORDS
	activated light pen, pointed at a spot on the screen as the electron beam lights up that spot, generates an electrical pulse that causes the coordinate position of the electron beam to be recorded.	Activated light pens pointed at a spot on the screen as the electron beam lights up that spot and generate electronic pulse that causes the coordinate position of the electron beam to be recorded.?

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

54/191	SUBMITTED TEXT 13 WORDS	96% MATCHING TEXT 13 WORDS
	which is used to input two-dimensional coordinates by activating a hand cursor	

SA Unit-3 (Part-III).docx (D77528823)

55/191	SUBMITTED TEXT 34 WORDS	98% MATCHING TEXT 34 WORDS
	A CRT monitor displays color pictures by using a combination of phosphors that emit different-colored light. By combining the emitted light from the different phosphors, a range of colors can be generated. The	

SA Computer Graphics (Final) 190587 .docx (D115191408)

56/191	SUBMITTED TEXT 13 WORDS	100% MATCHING TEXT 13 WORDS
	and the displayed color depends on how far the electron beam penetrates	

SA Computer Graphics (Final) 190587 .docx (D115191408)

57/191	SUBMITTED TEXT	58 WORDS	60%	MATCHING TEXT	58 WORDS

the phosphor layers. A beam of slow electrons excites only the outer red layer. A beam of very fast electrons penetrates through the red layer and excites the inner green layer. At intermediate beam speeds, combinations of red and green light are emitted to show two additional colors, orange and yellow. The speed of the electrons, and

SA 190587 (Jesse Tendai Tswamuno) Computer Graphics Answer sheet.docx (D119943759)

58/191	SUBMITTED TEXT	13 WORDS	100%	MATCHING TEXT	13 WORDS

the screen color at any point, is controlled by the beam-acceleration voltage.

SA Computer Graphics (Final) 190587 .docx (D115191408)

59/191	SUBMITTED TEXT	18 WORDS	83%	MATCHING TEXT	18 WORDS

is not as good as with other methods. • Shadow-mask methods are commonly used in raster-scan systems (

SA CG book _for_publishing_22_12_2018.docx (D46266915)

60/191	SUBMITTED TEXT	15 WORDS	100%	MATCHING TEXT	15 WORDS

because they produce a much wider range of colors than the beam- penetration method.

SA Graphics exam NCSC 300.docx (D104632959)

61/191	SUBMITTED TEXT	19 WORDS	97%	MATCHING TEXT	19 WORDS
three electron guns, one for each color dot, and a shadow-mask grid just behind the phosphor-coated screen. The				three electron guns one for each color dot. And a shadow mask grid just behind the phosphor coated screen.? ? The	

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

62/191	SUBMITTED TEXT	36 WORDS	69%	MATCHING TEXT	36 WORDS

A shadow-mask CRT has three phosphor color dots at each pixel position. One phosphor dot emits a red light, another emits a green light, and the third emits a blue light. This type of CRT

SA CG book _for_publishing_22_12_2018.docx (D46266915)

63/191	SUBMITTED TEXT	83 WORDS	92% MATCHING TEXT	83 WORDS

electron beams are deflected and focused as a group onto the shadow mask, which contains a series of holes aligned with the phosphor-dot patterns. When the three beams pass through a hole in the shadow mask, they activate a dot triangle, which appears as a small color spot on the screen. The phosphor dots in the triangles are arranged so that each electron beam can activate only its corresponding color dot when it passes through the shadow mask. Another configuration for the

SA Graphics exam NCSC 300.docx (D104632959)

64/191	SUBMITTED TEXT	17 WORDS	94% MATCHING TEXT	17 WORDS

Submitted	Matching
stores the picture information as a charge distribution just behind the phosphor-coated screen. Two electron guns	stores the picture information as a charge distribution just behind the phosphor coated screen.? ? DVST two electron guns

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

65/191	SUBMITTED TEXT	14 WORDS	76% MATCHING TEXT	14 WORDS

Submitted	Matching
the picture pattern; the second, the flood gun, maintains the picture display. A	the picture pattern and the flood gun maintains the picture display.? ? A

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

66/191	SUBMITTED TEXT	16 WORDS	73% MATCHING TEXT	16 WORDS

Submitted	Matching
very complex pictures can be displayed at very high resolutions without flicker. Disadvantages of DVST	Very complex pictures can be displayed at very high resolution without flicker.? ? Flat screen.? Disadvantage of DVST ?

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

67/191	SUBMITTED TEXT	49 WORDS	71% MATCHING TEXT	49 WORDS

electron guns, and the corresponding red- green-blue color dots on the screen, are aligned along one scan line instead of in a 22 triangular pattern. This in-line arrangement of electron guns is easier to keep in alignment and is commonly used in high-resolution color

SA Graphics exam NCSC 300.docx (D104632959)

68/191	SUBMITTED TEXT	41 WORDS	96% MATCHING TEXT	41 WORDS

Submitted	Matching
refers to a class of video devices that have reduced volume, weight, and power requirements compared to a CRT. A significant feature of flat-panel displays is that they are thinner than CRTs, and we can hang them on walls or	refers to a class of video devices that have reduced volume, weight, and power requirements compared to a CRT. A significant feature of flat-panel displayed is that they are thinner than CRTs, and we can hang them on walls or

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

69/191	SUBMITTED TEXT 25 WORDS	83% MATCHING TEXT 25 WORDS
	displays. • The emissive displays (or emitters) are devices that convert electrical energy into light. Plasma panels, thin-film electroluminescent displays, and Light-emitting diodes	displays: - the emissive display or emitters are devices that convert electrical energy into light. For Plasma panel, thin film electroluminescent displays and light emitting diodes. 2.

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

70/191	SUBMITTED TEXT 30 WORDS	100% MATCHING TEXT 30 WORDS
	type of emissive device is the light-emitting diode (LED). A matrix of diodes is arranged to form the pixel positions in the display, and picture definition is stored in	type of emissive device is the light-emitting diode (LED). A matrix of diodes is arranged to form the pixel positions in the display, and picture definition is stored in

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

71/191	SUBMITTED TEXT 33 WORDS	100% MATCHING TEXT 33 WORDS
	refresh buffer. As in scan-line refreshing of a CRT, information is read from the refresh buffer and converted to voltage levels that are applied to the diodes to produce the light patterns	refresh buffer. As in scan- line refreshing of a CRT, information is read from the refresh buffer and converted to voltage levels that are applied to the diodes to produce the light patterns.

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

72/191	SUBMITTED TEXT 133 WORDS	95% MATCHING TEXT 133 WORDS
	displays (or nonemitters) use optical effects to convert sunlight or light from some other source into graphics patterns. The most important example of a nonemissive flat-panel display is a liquid-crystal device. Plasma panels, also called gas-discharge displays, are constructed by filling the region between two glass plates with a mixture of gases that usually includes neon. A series of vertical conducting ribbons is placed on one glass panel, and a set of horizontal ribbons is built into the other glass panel. Firing voltages applied to a pair of horizontal and vertical conductors cause the gas at the intersection of the two conductors to break down into a glowing plasma of electrons and ions. Picture definition is stored in a refresh buffer, and the firing voltages are applied to refresh the pixel positions (	displays (or non use optical effects to convert sunlight or light from some other source into graphics patterns. The most important example of a flat- panel display is a liquid- crystal device. Plasma panels, also called gas discharge displays, are constructed by filling the region between two glass plates with a mixture of gases that usually include neon. A series of vertical conducting ribbons is placed on one glass panel, and a set of horizontal ribbons is built into the other glass panel. Firing voltages applied to a pair of horizontal and vertical conductors cause the gas at the intersection of the two conductors to break down into a glowing plasma of electrons and ions. Picture definition is stored in a refresh buffer, and the firing voltages are applied to refresh the pixel positions 60

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

73/191	SUBMITTED TEXT 14 WORDS	76% MATCHING TEXT 14 WORDS
	times per second. Alternating currents methods are used to produce faster application of	times per second.? ? Alternating current methods are used to provide faster application of

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

74/191	SUBMITTED TEXT 17 WORDS	100% MATCHING TEXT 17 WORDS
	firing voltages, and thus brighter displays. Separation between pixels is provided by the electric field of	firing voltages and thus brighter displays.? ? Separation between pixels is provided by the electric field of

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

75/191	SUBMITTED TEXT 26 WORDS	59% MATCHING TEXT 26 WORDS
	Flat-Panel Displays- Although most graphics monitors are still constructed with CRTs, other technologies are emerging that may soon replace CRT monitors. The term Bat-panel display	

SA CG book _for_publishing_22_12_2018.docx (D46266915)

76/191	SUBMITTED TEXT 181 WORDS	95% MATCHING TEXT 181 WORDS
	devices produce a picture by passing polarized light from the surroundings or from an internal light source through a liquid-crystal material that can be aligned to either block or transmit the light. The term liquid crystal refers to the fact that these compounds have a crystalline arrangement of molecules, yet they flow like a liquid. Flat-panel displays commonly use nematic (threadlike) liquid-crystal compounds that tend to keep the long axes of the rod-shaped molecules aligned. A flat-panel display can then be constructed with a nematic liquid 24 crystal. Two glass plates, each containing a light polarizer at right angles to the-other plate, sandwich the liquid-crystal material. Rows of horizontal transparent conductors are built into one glass plate, and columns of vertical conductors are put into the other plate. The intersection of two conductors defines a pixel position. Normally, the molecules are aligned. Polarized light passing through the material is twisted so that it will pass through the opposite polarizer. The light is then reflected back to the viewer. To turn off the pixel, we apply	devices produce a picture by passing polarized light from the surrounding or from an internal light source through a liquid-crystal material that can be aligned to either block or transmit the light. The term liquid crystal refers to the fact that these compounds have a crystalline arrangement of molecules, yet they flow like a liquid. Flat-panel displays commonly use nematic (threadlike) liquid- crystal compounds that tend to keep the long axes of the rod-shaped molecules aligned. A flat-panel display can then be constructed with a nematic liquid crystal, as in fig. Two glass plates, each containing a light polarizer at right angles to the other sandwich the liquid-crystal material. Rows of horizontal transparent conductors are built into one glass plate, and columns of vertical conductors are put into the other plate. The intersection of two conductors defines a pixel position. Normally, the molecules are aligned as shown in the "on Polarized light passing through the material is twisted so that it will pass through the opposite polarizer. The light is reflected back to the viewer. To turn off the pixel, we apply

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

77/191	SUBMITTED TEXT 57 WORDS	100% MATCHING TEXT 57 WORDS
	voltage to the two intersecting conductors to align the molecules so that the light is not .twisted. This type of flat-panel device is referred to as a passive-matrix LCD. Picture definitions are stored in a refresh buffer, and the screen is refreshed at the rate of 60 frames per second, as in the emissive devices.	voltage to the two intersecting conductors to align the molecules so that the light is not twisted. This type of flat-panel device is referred to as a passive matrix LCD. Picture definitions are stored in a refresh buffer, and the screen is refreshed at the rate of 60 frames per second, as in the emissive devices.

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

78/191	SUBMITTED TEXT 15 WORDS	100% MATCHING TEXT 15 WORDS
	devised using a technique that reflects a CRT image from a vibrating, flexible mirror.	devised using a technique that reflects a CRT image from a vibrating flexible mirror.?

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

79/191	SUBMITTED TEXT 17 WORDS	90% MATCHING TEXT 17 WORDS
	each point on the object is reflected from the mirror into a spatial position corresponding to	each point on the object is reflected from the mirror into spatial position corresponding to

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

80/191	SUBMITTED TEXT 20 WORDS	92% MATCHING TEXT 20 WORDS
	to have depth. One way to produce a stereoscopic effect is to display each of the two views with	to have depth.? ? One way to produce stereoscopic effect is to display each of the two views with

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

81/191	SUBMITTED TEXT 15 WORDS	100% MATCHING TEXT 15 WORDS
	raster system on alternate refresh cycles. The screen is viewed through glasses, with each	raster system on alternate refresh cycles.? ? The screen is viewed through glasses with each

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

82/191	SUBMITTED TEXT 17 WORDS	100% MATCHING TEXT 17 WORDS
	act as a rapidly alternating shutter that is synchronized to block out one of the views.	act as a rapidly alternating shutter that is synchronized to block out one of the views.?

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

83/191	SUBMITTED TEXT 76 WORDS	75% MATCHING TEXT 76 WORDS
	can step into a scene and interact with the environment. A headset containing an optical system to generate the stereoscopic views is commonly used in conjuction with interactive input devices to locate and manipulate objects in the scene. A sensing system in the headset keeps track of the viewer's position, so that the front and back of objects can be seen as the viewer "walks through" and interacts with the display. An interactive virtual-reality	can step into a scene and interact with the environment.? Unit-1 – Basics of Computer Graphics 14 Computer Graphics ? A head set containing an optical system to generate the stereoscopic views is commonly used in conjunction with interactive input devices to locate and manipulate objects in the scene.? ? Sensor in the head set keeps track of the viewer's position so that the front and back of objects can be seen as the viewer "walks through" and interacts with the display.? ? Virtual reality

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

84/191	SUBMITTED TEXT 26 WORDS	78% MATCHING TEXT 26 WORDS
	x = h + r cosθ y = k + r sinθ where r is the radius of the circle,	x = h + r cosθ y = k + r sinθ where (h, k) are the center coordinates and r is the radius of the circle. '

W https://odl.ptu.ac.in/SLM/msc%20it/3RD/MSIT_301_Computer_Graphics.pdf

85/191	SUBMITTED TEXT 15 WORDS	100% MATCHING TEXT 15 WORDS
	theta &lt;= 360; { x = h + r*cos(theta) y = k + r*sin(theta)	theta &gt; =) x = h + r cos(theta) y = k + r sin(theta)

W https://odl.ptu.ac.in/SLM/msc%20it/3RD/MSIT_301_Computer_Graphics.pdf

86/191	SUBMITTED TEXT 11 WORDS	76% MATCHING TEXT 11 WORDS
	x1, y1) and (x2, y2), respectively. So slope, m= (y2-y1)/(x2-x1).	x1, y1) and (x2, y2) respectively 2. Set m = (y2-y1)/ (x2- x1)

W https://docplayer.net/53449703-Computergraphics-dcap504.html

87/191	SUBMITTED TEXT	26 WORDS	100% MATCHING TEXT	26 WORDS

used for linear interpolation of variables over an interval between start and end point. DDAs are used for rasterization of lines, triangles and polygons in

SA MSc_Computer Graphics_Course code MCSDSC_1.7.pdf (D140276469)

88/191	SUBMITTED TEXT	27 WORDS	56% MATCHING TEXT	27 WORDS

x=x+1 y=y + m x=x+1/m y=y+1 Second x=x-1 y=y + m x=x+1/m y=y+1 Third x=x-1 y=y-m x=x-1/m y=y-1 Fourth x=x+1 y=y + m x=x-1/m y=y-1

X X Y start &gt; end ⇒ m is negative X start &lt; end Y start &lt; Y end ⇒ m is positive |m|&gt;=1 X i+1 = X i + 1 Y i+1 = Y i + m X i+1 = X i – 1 Y i+1 = Y m X i+1 = X i – 1 Y i+1 = Y m |m|&lt;=1 Y i+1 = Y i – 1 X i+1 = 1 – X m i+1 = Y X i+1 = 1 – X m Y Y

W https://aditipaulsite.files.wordpress.com/2019/10/computer-graphic_final-book.pdf

89/191	SUBMITTED TEXT	155 WORDS	62% MATCHING TEXT	155 WORDS

putpixel(x1+200,y1+200,3); putpixel(y1+200,x1+200,3); putpixel(y1+200,200-x1,3); putpixel(x1+200,200-y1,3); putpixel(-x1+200,-y1+200,3); putpixel(-y1+200,-x1+200,3); putpixel(-y1+200,x1+200,3); putpixel(-x1+200,y1+200,3); if(d&gt;0) { d=d+4*x1+6; } else { d=d+4*(x1-y1)+10; y1=y1-1; }

putpixel(-x,-y,putpixel(y,x,YELLOW); y,-x,YELLOW); putpixel(-y,x,putpixel(y,-x,YELLOW); putpixel(x,y,YELLOW); putpixel(x,-y,YELLOW); if(d=d+4*x=x+1; } Appendix: Programming Using C A.5 else { d=d+4*(x-y)+10; x=x+1; y=y-1; } }

W https://aditipaulsite.files.wordpress.com/2019/10/computer-graphic_final-book.pdf

90/191	SUBMITTED TEXT	181 WORDS	25% MATCHING TEXT	181 WORDS

start_x,start_y; 33 start_x=redius; start_y=0; x1=start_x; y1=start_y; { val=pow(2,i); i++; }while(val&gt;redius); epsilon=1/pow(2,i-1); do { x2=x1+(epsilon * y1); y2=y1-(epsilon * x2); delay(10); putpixel(x2+200,y2+200,3); x1=x2; y1=y2; }while((y1-start_y)&gt;epsilon || (start_x-x1)&lt;

start &gt; X end Y start &lt; Y end ⇒ m negative start &lt; X end Y start &gt; Y end ⇒ m is negative X start &lt; X end Y start &lt; Y end ⇒ m is positive |m|&gt;=1 X i+1 = X i + 1 Y i+1 = Y i + X i+1 = X i – 1 Y i+1 = Y i – m X i+1 = X i – 1 Y i+1 = Y i – m |m|&lt;=1 Y i+1 = Y i – 1 X i+1 = 1 – i X

W https://aditipaulsite.files.wordpress.com/2019/10/computer-graphic_final-book.pdf

91/191	SUBMITTED TEXT	73 WORDS	33% MATCHING TEXT	73 WORDS

putpixel(x+200,y+200,3); while(i&gt;len) { x=x+x_incr; y=y+y_incr; putpixel(x+200,y+200,3); i++; } flushall(); } void Drawing:: draw_breCir() { } int main() { int gd=DETECT,gm,ch; 35 Drawing draw; clrscr(); initgraph(&gd,&gm,"C:\\TC\\bgi");

putpixel(x,y,RED); fill_left(--x,y); x = x + 1 ; fill_left(y-1); left(x,y+1); } delay(1); } void main() { int x,y,n,i; int gd=DETECT,gm; clrscr(); initgraph(&gd,&gm,"c:\\tc\\bgi"); /*-

W https://aditipaulsite.files.wordpress.com/2019/10/computer-graphic_final-book.pdf

92/191	SUBMITTED TEXT	61 WORDS	62% MATCHING TEXT	61 WORDS

x,y,x_incr,y_incr; int i=0; dx=x2-x1; dy=y2-y1; 34 x=x1; y=y1; if(abs(dx)&lt;abs(dy)) len=abs(dx); else len=abs(dy);

x1,int y1,int x2,int y2) { int dy,i; float x=x1,y=y1; x2-dy=y2-y1; if(abs(dx)&lt;=abs(dy)) delta=abs(dx); else delta=abs(dy);

W https://www.egyankosh.ac.in/bitstream/123456789/63867/1/MMTE-004.pdf

93/191	SUBMITTED TEXT 19 WORDS	61% MATCHING TEXT 19 WORDS
	The result is positive for all points outside the circle and negative for all points inside the circle.	the function is positive for the points outside the circle (or above the arc) and is negative for the points inside the circle (

W https://docplayer.net/53449703-Computergraphics-dcap504.html

94/191	SUBMITTED TEXT 78 WORDS	27% MATCHING TEXT 78 WORDS
	x&gt;=x1;x++) { d+= m; if (d&lt;=x1) { d-= 2*x1; y++;} 39 pixel(x,y); } 4. Shift d by -x1 to remove a subtraction. m=2*y1; d= -x1; pixel(0,0); y=0; for(x=1;x&gt;=x1;x++) { d+= m; if (d&lt;=0) { d-= 2*x1; y++;}	

SA Graphics Examination 190232.pdf (D125664974)

95/191	SUBMITTED TEXT 14 WORDS	83% MATCHING TEXT 14 WORDS
	Computer Graphics, C Version, Second Edition by Donald Hearn, M. Pauline Baker 2.	Computer Graphics" C-Version, 2 nd Edition by Donald Hearn and M. Pauline Baker

W https://www.egyankosh.ac.in/bitstream/123456789/63867/1/MMTE-004.pdf

96/191	SUBMITTED TEXT 12 WORDS	83% MATCHING TEXT 12 WORDS
	for(x=1;x&gt;r/sqrt(2);x++) { y=sqrt(r 2 -x 2); pixel(x,	

SA Computer Graphics.pdf (D165747830)

97/191	SUBMITTED TEXT 12 WORDS	100% MATCHING TEXT 12 WORDS
	distorts the shape of an object along either or both the	distorts the shape of an object along either or both the

W https://studio.eecs.umich.edu/confluence/download/attachments/17827062/mathematicsforcomputergrap ...

98/191	SUBMITTED TEXT 13 WORDS	100% MATCHING TEXT 13 WORDS
	Polygon Fill Method: Seed Filling: Boundary fill and Flood fill algorithm 2.4	

SA CG assaignment 1.rtf (D25685411)

99/191	SUBMITTED TEXT 18 WORDS	68% MATCHING TEXT 18 WORDS
	scan-line polygon fill algorithm, which employs the odd/even parity concept previously discussed, works for complex polygon filling.	SCAN LINE POLIGON FILL ALGORITHM It employs the odd/even parity concept and works for complex polygon filling.

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

100/191	SUBMITTED TEXT	23 WORDS	97% MATCHING TEXT	23 WORDS

The basic concept of the scan-line algorithm is to draw points from edges of odd parity to even parity on each scan-line.

The basic concept of the scan line algorithm is to draw points from edges of the odd parity to even parity on each scan line. 2.7.1

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

101/191	SUBMITTED TEXT	13 WORDS	100% MATCHING TEXT	13 WORDS

Polygon Fill Method: Seed Filling: Boundary fill and Flood fill algorithm

SA CG assaignment 1.rtf (D25685411)

102/191	SUBMITTED TEXT	36 WORDS	100% MATCHING TEXT	36 WORDS

Seed Filling: In this method a particular seed point is picked and we start filling upwards and downwards pixels until boundary is reached. Seed fill method is of two types: Boundary fill and Flood fill.

SA CG assaignment 1.rtf (D25685411)

103/191	SUBMITTED TEXT	209 WORDS	100% MATCHING TEXT	209 WORDS

Procedure for filling a 4- connected region: Color is specified by parameter fill color (f- color) and boundary color is specified by boundary color (b-color). getpixel() function gives the color of specified pixel and putpixel() fills the pixel with particular color. boundary_ fill (x,y, f_color, b_color) { If (getpixel (x,y) != b_color && getpixel (x,y) != f_color) putpixel(x,y,f_color) boundary_fill(x+1, y, f_color, b_color); boundary_fill(x, y+1, f_color, b_color); boundary_fill(x-1, y, f_color, b_color); boundary_fill(x, y-1, f_color, b_color); } } Flood fill algorithm: There are some cases where the boundary color is different than the fill color. For situations like these Flood fill algorithm is used. Here the process is started in a similar way by examining the colors of neighboring pixels. But instead of matching it with a boundary color a specified color is matched. Procedure for filling a 8- connected region: flood_ fill (x,y, old_color, new_color) { putpixel(x,y,new_color) 68 flood_ fill (x+1, y, old_color, new_color) flood_ fill (x-1, y, old_color, new_color) flood_ fill (x, y+1, old_color, new_color) flood_ fill (x, y-1, old_color, new_color) flood_ fill (x+1, y+1, old_color, new_color) flood_ fill (x-1, y-1, old_color, new_color) flood_ fill (x+1, y-1, old_color, new_color) flood_ fill (x-1, y+1, old_color, new_color) } }

SA CG assaignment 1.rtf (D25685411)

104/191	SUBMITTED TEXT	14 WORDS	96% MATCHING TEXT	14 WORDS

by arranging the orientations and sizes of the component parts of the scene.

SA INF_2046.pdf (D164968115)

105/191	SUBMITTED TEXT 12 WORDS	100% MATCHING TEXT 12 WORDS
	so that the line passes through the origin. 2. Rotate the	so that the line passes through the origin. 2. Rotate the

W https://aditipaulsite.files.wordpress.com/2019/10/computer-graphic_final-book.pdf

106/191	SUBMITTED TEXT 18 WORDS	85% MATCHING TEXT 18 WORDS
	transformation A shear is a transformation that distorts the shape of an object along either or both	Transformation A shear is a transformation that, similar to scale and translate, distorts the shape of an object along either or both

W https://studio.eecs.umich.edu/confluence/download/attachments/17827062/mathematicsforcomputergrap ...

107/191	SUBMITTED TEXT 14 WORDS	100% MATCHING TEXT 14 WORDS
	x-axis) is performed in terms of the point's coordinate in the other axis (	x-axis) is performed in terms of the point's coordinate in the other axis (

W https://studio.eecs.umich.edu/confluence/download/attachments/17827062/mathematicsforcomputergrap ...

108/191	SUBMITTED TEXT 39 WORDS	61% MATCHING TEXT 39 WORDS
	produced by moving the "camera" or the objects in a scene along animation paths. Changes in orientation, size, and shape are accomplished with geometric transformations that alter the coordinate descriptions of objects. The basic geometric transformations are translation,	

SA Computer Graphics.pdf (D165747830)

109/191	SUBMITTED TEXT 125 WORDS	92% MATCHING TEXT 125 WORDS
	To shear in the x direction the equation is: x1 = x + ay 91 y1 = y Where b = 0 Where x1 and y1 are the new values, x and y are the original values, and a is the scaling factor in the x direction. The matrix is as follows. Shearing in the y direction is similar except the roles are reversed. x1 = x y1 = y + bx Where a = 0. Where x1 and y1 are the new values, x and y are the original values, and b is the scaling factor in the y direction. The matrix is as follows. Example 92 Original Y-Shear	To shear in the x direction the equation is: Where x1 = x + ay y1 = y b=0 Where x1 and y1 are the new values, x and y are the original values, and a is the scaling factor in the x direction. The matrix is as follows. 1a0 010 001 Shearing in the y direction is similar except the roles are reversed. Where x1 = x y1 = y + bx a = 0. Where x1 and y1 are the new values, x and y are the original values, and b is the scaling factor in the y direction. The matrix is as follows. 3.5.2.3 Y Shear

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

110/191	SUBMITTED TEXT 23 WORDS	75% MATCHING TEXT 23 WORDS
	Point (X,Y) is to be translated by amount Dx and Dy to location (X',Y')	point (X, Y, Z) is to be translated by amount tx, ty and tz to location (X', Y',

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

111/191	SUBMITTED TEXT 13 WORDS	89% MATCHING TEXT 13 WORDS
	In 3D we can shear along the x-axis, y-axis or z-axis.	

SA ODMCA-404T Palwinder 10th Feb 2020.docx (D63728007)

112/191	SUBMITTED TEXT 21 WORDS	92% MATCHING TEXT 21 WORDS
	geometric transformations and object modelling in three dimensions are extended from two-dimensional methods by including considerations for the z coordinate.	Geometric transformations and object modeling in three dimensions are extended from two-dimensional methods by including considerations for the z-coordinate.

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

113/191	SUBMITTED TEXT 24 WORDS	56% MATCHING TEXT 24 WORDS
	the window to the size and aspect ratio of the viewport. 3. translate the viewport to its position on the screen in the	

SA ODMCA-404T Palwinder 10th Feb 2020.docx (D63728007)

114/191	SUBMITTED TEXT 25 WORDS	69% MATCHING TEXT 25 WORDS
	two-dimensional rotations in the xy plane, we needed to consider only rotations about axes that were perpendicular to the xy plane. In three-dimensional space,	

SA Computer Graphics.pdf (D165747830)

115/191	SUBMITTED TEXT 37 WORDS	87% MATCHING TEXT 37 WORDS
	as a composite of three rotations, one for each of the three Cartesian axes. Alternatively, an user can easily set up a general rotation matrix, given the orientation of the axis and the required rotation angle.	

SA Computer Graphics.pdf (D165747830)

116/191	SUBMITTED TEXT 22 WORDS	86% MATCHING TEXT 22 WORDS
	Reflections A three-dimensional reflection can be performed relative to a selected reflection axis or with respect to a selected reflection plane.	Reflections reflection can be performed relative to a selected reflection axis or with respect to a selected reflection plane.

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

117/191	SUBMITTED TEXT 34 WORDS	100% MATCHING TEXT 34 WORDS
	In 3D graphics, transformation is often used to operate on vertices and vectors. It is also used to convert them in one space to another. Transformation is performed via multiplication with a matrix.	

SA ODMCA-404T Palwinder 10th Feb 2020.docx (D63728007)

118/191	SUBMITTED TEXT 15 WORDS	89% MATCHING TEXT 15 WORDS
	space. When the reflection plane is a coordinate plane (either xy, xz, or yz),	space. When the reflection plane in a coordinate plane (either xy, xz or yz)

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

119/191	SUBMITTED TEXT 43 WORDS	80% MATCHING TEXT 43 WORDS
	the transformation as a conversion between Left-handed and right-handed systems. An example of a reflection that converts coordinate specifications from a right handed system to a left-handed system (or vice versa) is shown in Fig. 4-3. This transformation changes the sign of	the transformation can be a conversion between left-handed and right-handed systems. ? An example of a reflection that converts coordinate specifications from a right handed system to a left-handed system is shown in the This transformation changes the sign of

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

120/191	SUBMITTED TEXT 24 WORDS	93% MATCHING TEXT 24 WORDS
	z coordinates, Leaving the x and y-coordinate values unchanged. The matrix representation for this reflection of points relative to the xy plane is	z coordinates, leaves the x and y coordinate values unchanged. ? The matrix representation for this reflection of points relative to the xy plane is 1 0 0 0

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

121/191	SUBMITTED TEXT 34 WORDS	73% MATCHING TEXT 34 WORDS
	relative to the yz plane and xz plane, respectively Reflections about other planes can be obtained as a combination of rotations and coordinate-plane reflections. 109 Shears Shearing transformations	relative to the xy plane is 1 000 RFz = 0 100 0 0 –1 0 0 001 Reflections about other planes can be obtained as a combination of rotations and coordinate plane reflections. 5.10.2 Shears Shearing transformations

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

122/191	SUBMITTED TEXT 20 WORDS	85% MATCHING TEXT 20 WORDS
	can be used to modify object shapes. They are also useful in three-dimensional viewing for obtaining general projection transformations.	can be used to modify object shapes.? ? They are also useful in 3 D viewing for obtaining general projection transformations.? ?

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

123/191	SUBMITTED TEXT 18 WORDS	86% MATCHING TEXT 18 WORDS
	the following transformation produces a z- axis shear: Parameters a and b can be assigned any real values.	The following transformation produces a z-axis shear. 100 1 0 a 0 SHz = 0 1 b 0 Observation/Note 0 01 0 0 00 1 Parameters a and b can be assigned any real values.

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

124/191	SUBMITTED TEXT 38 WORDS	86% MATCHING TEXT 38 WORDS
	this transformation matrix is to alter x- and y-coordinate values by an amount that is proportional to the 2 value, while leaving the z coordinate unchanged. Boundaries of planes that are perpendicular to the z axis are	This transformation matrix is used to alter x and y coordinate values by an amount that is proportional to the z value, and the z coordinate will be unchanged.Boundaries of planes that are perpendicular to the z axis are

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

125/191	SUBMITTED TEXT 22 WORDS	86% MATCHING TEXT 22 WORDS
	Reflections A three-dimensional reflection can be performed relative to a selected reflection axis or with respect to a selected reflection plane.	Reflections reflection can be performed relative to a selected reflection axis or with respect to a selected reflection plane.

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

126/191	SUBMITTED TEXT 14 WORDS	87% MATCHING TEXT 14 WORDS
	relative to a given axis are equivalent to 180°" rotations about that axis.	relative to a given axis are equivalent to 1800 rotations about the axis.

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

127/191	SUBMITTED TEXT 15 WORDS	89% MATCHING TEXT 15 WORDS
	space. When the reflection plane is a coordinate plane (either xy, xz, or yz),	space. When the reflection plane in a coordinate plane (either xy, xz or yz)

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

128/191	SUBMITTED TEXT 16 WORDS	88% MATCHING TEXT 16 WORDS
	the transformation as a conversion between Left-handed and right-handed systems. An example of a reflection	the transformation can be a conversion between left-handed and right-handed systems. An example of a reflection

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

129/191	SUBMITTED TEXT 22 WORDS	76% MATCHING TEXT 22 WORDS
	that converts coordinate specifications from a right-handed system to a left-handed system (or vice versa). This transformation changes the sign of	that converts coordinate specifications from a right handed system to a left-handed system is shown in the This transformation changes the sign of

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

130/191	SUBMITTED TEXT 28 WORDS	67% MATCHING TEXT 28 WORDS
	relative to the yz plane and xz plane, respectively Reflections about other planes can be obtained as a combination of rotations and coordinate-plane reflections. Shearing Shearing transformations	relative to the xy plane is Reflections about other planes can be obtained as a combination of rotations and coordinate plane reflections. 5.10.2 Shears Shearing transformations

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

131/191	SUBMITTED TEXT 21 WORDS	85% MATCHING TEXT 21 WORDS
	can be used to modify object shapes. They are also useful in three dimensional viewing for obtaining general projection transformations.	can be used to modify object shapes.? ? They are also useful in 3 D viewing for obtaining general projection transformations.? ?

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

132/191	SUBMITTED TEXT 20 WORDS	92% MATCHING TEXT 20 WORDS
	points in a coordinate system of dimension n into points in a coordinate system of dimension less than n.	

SA INF_2046.pdf (D164968115)

133/191	SUBMITTED TEXT 33 WORDS	89% MATCHING TEXT 33 WORDS
	D screen is done by straight projection rays (called projectors) emanating from a center of projection, passing through each point of the object, and intersecting a projection plane to form the projection.	

SA Computer_Graphics_Block_2.pdf (D149210060)

134/191	SUBMITTED TEXT 21 WORDS	77% MATCHING TEXT 21 WORDS
	projection planes that are not normal to a principal axis and therefore show several faces of an object at once.	Projection planes are not perpendicular to a principal axis and therefore show multiple faces of an object at once.

W https://studio.eecs.umich.edu/confluence/download/attachments/17827062/mathematicsforcomputergrap ...

135/191	SUBMITTED TEXT 26 WORDS	88% MATCHING TEXT 26 WORDS
	perspective projection in this way, but differ in that the foreshortening is uniform rather than being related to the distance from the center of projection.	perspective projection in this way, but differs in that the shortening is uniform, rather than being related to the distance from the center of projection.

W https://studio.eecs.umich.edu/confluence/download/attachments/17827062/mathematicsforcomputergrap ...

136/191	SUBMITTED TEXT 39 WORDS	66% MATCHING TEXT 39 WORDS
	axonometric projections, the projection plane is normal to a principal axis, so the projections of the face of the object parallel to this plane allows measurement of angles and distances. 124	axonometric projection. The projection plane is normal to the principal axes, so the projection of the face of the object must be parallel to the projection plane. This allows the measurement of angles and distances.

W https://studio.eecs.umich.edu/confluence/download/attachments/17827062/mathematicsforcomputergrap ...

137/191	SUBMITTED TEXT	53 WORDS	64% MATCHING TEXT	53 WORDS

The distinction is in the relation of center of projection and projection plane. If the distance between the center of projection and projection plane is finite then the projection is perspective and if the distance is infinite, the projection is parallel. 122

SA INF_2046.pdf (D164968115)

138/191	SUBMITTED TEXT	13 WORDS	83% MATCHING TEXT	13 WORDS

Submitted	Matching
projection in which center of projection is at a finite distance from	projection. In general, the center of projection is at a finite distance from

W https://studio.eecs.umich.edu/confluence/download/attachments/17827062/mathematicsforcomputergrap ...

139/191	SUBMITTED TEXT	62 WORDS	90% MATCHING TEXT	62 WORDS

Submitted	Matching
is characterized by perspective foreshorting and vanishing points. Perspective foreshortening is the illusion that objects and length appear smaller as their distance from the center of projection increases. The illusion that certain sets of parallel lines appear to meet at a point is another feature of perspective drawings. These points are Vanishing Points. 126	is characterized by perspective foreshortening and vanishing points. Perspective foreshortening is the illusion that objects and lengths appear smaller as their distance from the center of projection increases. The illusion that certain sets of parallel lines appear to meet at a point is another feature of perspective drawing. These points are called vanishing points.

W https://aditipaulsite.files.wordpress.com/2019/10/computer-graphic_final-book.pdf

140/191	SUBMITTED TEXT	13 WORDS	100% MATCHING TEXT	13 WORDS

Submitted	Matching
In OpenGL, a vertex V at (x, y, z) is represented as	in OpenGL, a vertex V at (x, y, z) is represented as

W https://studio.eecs.umich.edu/confluence/download/attachments/17827062/mathematicsforcomputergrap ...

141/191	SUBMITTED TEXT	27 WORDS	73% MATCHING TEXT	27 WORDS

projection of this object. Straight Projectors called projectors emanating from a center of projection, passing through each point of object and intersecting a projection plane form

SA Computer_Graphics_Block_2.pdf (D149210060)

142/191	SUBMITTED TEXT	14 WORDS	76% MATCHING TEXT	14 WORDS

Submitted	Matching
and αz represent the scaling factors in x, y and z direction, respectively.	and az represent the scaling factors in x, y, and z directions, respectively.

W https://pdfcoffee.com/mathematicsforcomputergraphicsandgameprogrammingpdf-pdf-free.html

143/191	SUBMITTED TEXT 38 WORDS	88% MATCHING TEXT 38 WORDS
	Rotation 3D rotation operates about an axis of rotation (2D rotation operates about a center of rotation). 3D Rotations about the x, y and z axes for an angle θ (measured in counter-clockwise manner) can be represented	Rotation 3D rotation operates about an axis of rotation (2D rotation operates about a center of rotation). 3D rotations about the x, y, and z axes for an angle q (measured in counterclockwise manner) can be represented

W https://pdfcoffee.com/mathematicsforcomputergraphicsandgameprogrammingpdf-pdf-free.html

144/191	SUBMITTED TEXT 20 WORDS	83% MATCHING TEXT 20 WORDS
	Translation Translation does not belong to linear transform, but can be modelled via a vector addition, as follows:	Translation Translation does not belong to linear transformation, but can be modeled via a vector addition as follows:

W https://studio.eecs.umich.edu/confluence/download/attachments/17827062/mathematicsforcomputergrap ...

145/191	SUBMITTED TEXT 47 WORDS	91% MATCHING TEXT 47 WORDS
	The Viewing Pipeline A world-coordinate area selected for display is called a window. An area on a display device to which a window is mapped is called a viewport. The window defines what is to be viewed; the viewport defines where it is to be displayed.	The Viewing PipelineA world coordinate area selected for display is called a window. An area on a displaydevice to which a window is mapped is called a view port. The window defines what is viewed the view port defines where it is to be displayed.

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

146/191	SUBMITTED TEXT 33 WORDS	85% MATCHING TEXT 33 WORDS
	The rotational angles about x, y and z axes, denoted as θx, θy and θz, are known as Euler angles, which can be used to specify any arbitrary orientation of an object.	The rotational angles about x, y, and z axes, denoted as qx, qy, and qz, are known as Euler angles, which can be used to specify any arbitrary orientation of an object.

W https://pdfcoffee.com/mathematicsforcomputergraphicsandgameprogrammingpdf-pdf-free.html

147/191	SUBMITTED TEXT 161 WORDS	93% MATCHING TEXT 161 WORDS
	the viewing transformation in several steps. First, we construct the scene in world coordinates using the output primitives and attributes. Next, to obtain a particular orientation for the window, we can set up a two-dimensional viewing-coordinate system in the world-coordinate plane, and define a window In the viewing-coordinate system. The viewing- coordinate reference frame is used to provide a method for setting up arbitrary orientations for rectangular windows. Once the viewing reference frame is established, we can transform descriptions in world coordinates to viewing coordinates. We then define a viewport in normalized coordinates (in the range from O to I) and map the viewing-coordinate description of the scene to normalized co- ordinates. At the final step, parts of the picture that are outside the viewport are clipped, and the contents of the viewport are transferred to device coordinates. By changing the position of the viewport, we can view objects at different positions on the display area of an output device.	The viewing transformation in several steps, as indicated in Fig. 4.1, First, we construct the scene in world coordinates using the output primitives. Next to obtain a particular for the window, we can set up a two-dimensional viewing-coordinate system in the world coordinate plane, and define a window in the viewing-coordinate system. The viewing- coordinate reference frame is used to provide a method for setting up arbitrary orientations for rectangular windows. Once the viewing reference frame is established, we can transform descriptions in world coordinates to viewing coordinates. We then define a viewport in normalized coordinates (in the range from 0 to 1) and map the viewing-coordinate description of the scene to normalized coordinates. the final step all parts of the picture that lie outside the viewport are clipped, and the the viewport are transferred to device coordinates. By changing the position of the viewport, we can view objects at different positions on the display area of an output device.

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

148/191	SUBMITTED TEXT	20 WORDS	100% MATCHING TEXT	20 WORDS

the mapping of a part of a world-coordinate scene to device coordinates is referred to as a viewing transformation.

SA MSc_Computer Graphics_Course code MCSDSC_1.7.pdf (D140276469)

149/191	SUBMITTED TEXT	16 WORDS	84% MATCHING TEXT	16 WORDS

the space in which the application model is defined. Definitions
• Screen Coordinate System -

SA Unit-3 (Part-III).docx (D77528823)

150/191	SUBMITTED TEXT	21 WORDS	100% MATCHING TEXT	21 WORDS

Interface Window - The window opened on the raster graphics screen in which the image will be displayed. •

SA ODMCA-404T Palwinder 10th Feb 2020.docx (D63728007)

151/191	SUBMITTED TEXT	12 WORDS	100% MATCHING TEXT	12 WORDS

Submitted	Matching
the world or clipping window are mapped to the viewport. The	the world or clipping window are mapped to the viewport. The

W https://docplayer.net/53449703-Computergraphics-dcap504.html

152/191	SUBMITTED TEXT	37 WORDS	33% MATCHING TEXT	37 WORDS

Submitted	Matching
the interface window on the screen. In other words, the clipping window is used to select the part of the scene that is to be displayed. The viewport then positions the scene on the output device.	The interface window on the screen displays the viewport. To display a part of the scene, the clipping window is selected and then the viewport positions the scene on the output device.

W https://docplayer.net/53449703-Computergraphics-dcap504.html

153/191	SUBMITTED TEXT	25 WORDS	72% MATCHING TEXT	25 WORDS

Submitted	Matching
a point in the world window, say (xw, yw). The formula is used to produce a corresponding point in viewport coordinates, say (xv, yv).	a point in the world window like (XW, YW). The formula helps to produce a corresponding point in the viewport coordinates, (XV, YV).

W https://docplayer.net/53449703-Computergraphics-dcap504.html

154/191	SUBMITTED TEXT	61 WORDS	84% MATCHING TEXT	61 WORDS

Submitted	Matching
if xw is 30% of the way from the left edge of the world window, then xv is 30% of the way from the left edge of the viewport. Similarly, if yw is 30% of the way from the bottom edge of the world window, then yv is 30% of the way from the bottom edge of the viewport. The	If XW is 4 per cent of the way from the right edge of the world window, then XV is 3% of the way from the right edge of the viewport. Similarly, if YW is 4per cent of the way from the top edge of the world window, then YV is 4 percent of the way from the top edge of the viewport. The

W https://docplayer.net/53449703-Computergraphics-dcap504.html

155/191	SUBMITTED TEXT 16 WORDS	90% MATCHING TEXT 16 WORDS
	The rectangular portion of the interface window that defines where the image will actually appear (	
SA	Unit-3 (Part-III).docx (D77528823)	

156/191	SUBMITTED TEXT 17 WORDS	96% MATCHING TEXT 17 WORDS
	The position of the viewport can be changed allowing objects to be viewed at different positions	the position of the viewport can be changed by allowing objects to be viewed at different positions.
W	https://docplayer.net/53449703-Computergraphics-dcap504.html	

157/191	SUBMITTED TEXT 18 WORDS	100% MATCHING TEXT 18 WORDS
	Multiple viewports can also be used to display different sections of a scene at different screen positions.	Multiple viewports can also be used to display different sections of a scene at different screen positions.
W	https://docplayer.net/53449703-Computergraphics-dcap504.html	

158/191	SUBMITTED TEXT 11 WORDS	100% MATCHING TEXT 11 WORDS
	mapping different dimensioned clipping windows on a fixed sized viewport.	mapping different dimensioned clipping windows on a fixed sized viewport,
W	https://docplayer.net/53449703-Computergraphics-dcap504.html	

159/191	SUBMITTED TEXT 22 WORDS	58% MATCHING TEXT 22 WORDS
	If the aspect ratio of the world window and the viewport are different, then the image may look distorted. OpenGL Window-To-Viewport	if the aspect ratio of the world window and the viewport are different. Tiling Using the Window-to-Viewport
W	https://docplayer.net/53449703-Computergraphics-dcap504.html	

160/191	SUBMITTED TEXT 23 WORDS	97% MATCHING TEXT 23 WORDS
	number of copies of the same image in rows and columns across the interface window so that they cover the entire window,	number of copies drawn of the same image in rows and columns across the interface window so that they cover the entire window.
W	https://docplayer.net/53449703-Computergraphics-dcap504.html	

161/191	SUBMITTED TEXT 15 WORDS	100% MATCHING TEXT 15 WORDS
	the camera across the scene, taking in different parts of it at different times.	the camera across the scene, taking in different parts of it at different times.
W	https://docplayer.net/53449703-Computergraphics-dcap504.html	

162/191	SUBMITTED TEXT	58 WORDS	96%	MATCHING TEXT	58 WORDS

Clipping 153 Generally, any procedure that identifies those portions of a picture that are either inside or outside of a specified region of space is referred to as a clipping algorithm, or simply clipping. The region against which an object is to be clipped is called a clip window.

SA Graphics exam NCSC 300.docx (D104632959)

163/191	SUBMITTED TEXT	34 WORDS	84%	MATCHING TEXT	34 WORDS

line clipping' is the process of removing lines or portions of lines outside of an area of interest. Typically, any line or part thereof which is outside of the viewing area is removed.

SA Computer Graphics (Final) 190587 .docx (D115191408)

164/191	SUBMITTED TEXT	18 WORDS	75%	MATCHING TEXT	18 WORDS

of clipping include extracting part of a detained scene for viewing; identifying visible surfaces in three-dimensional views;

SA Graphics exam NCSC 300.docx (D104632959)

165/191	SUBMITTED TEXT	18 WORDS	91%	MATCHING TEXT	18 WORDS

object boundaries; creating objects using solid-modelling procedures; displaying a multi window environment; and drawing and painting operations

SA Graphics exam NCSC 300.docx (D104632959)

166/191	SUBMITTED TEXT	21 WORDS	70%	MATCHING TEXT	21 WORDS

Depending on the application, the clip window can be a general polygon or it can even have curved boundaries. We

SA Computer_Graphics_Block_2.pdf (D149210060)

167/191	SUBMITTED TEXT	36 WORDS	72%	MATCHING TEXT	36 WORDS

Submitted	Matching
primitive types ? Point Clipping ? Line Clipping (straight-line segments) ? Area Clipping (polygons) ? Curve Clipping ? Text Clipping Line and polygon clipping routines are standard components of graphics packages,	primitive types: Point clipping Line clipping (Straight-line segment) Area clipping Curve clipping Text clipping Line and polygon clipping routines are standard components of graphics packages. 2.6.1

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

168/191	SUBMITTED TEXT 164 WORDS	98% MATCHING TEXT 164 WORDS
	For the viewing transformation, we want to display only those picture parts that are within the window area (assuming that the clipping flags have not been set to no clip). Everything outside the window is discarded. Clipping algorithms can be applied in world coordinates, so that only the contents of the window interior are mapped to device coordinates. Alternatively, the complete world-coordinate picture can be mapped first to device coordinates, or normalized device coordinates, then clipped against the viewport boundaries. World-coordinate clipping removes those primitives outside the window from further consideration, thus eliminating the processing necessary to transform those primitives to device space. Viewport clipping, on the other hand, can reduced calculations by allowing concatenation of viewing and geometric transformation matrices. But viewport clipping does require that the 154 transformation to device coordinates be performed for all objects, including those outside the window area. On raster systems, clipping algorithms are often combined with scan conversion. In the	

SA 190305 computergraphics.pdf (D119942496)

169/191	SUBMITTED TEXT 55 WORDS	38% MATCHING TEXT 55 WORDS
	If the region is above the screen, the first bit is 1 • If the region is below the screen, the second bit is 1 • If the region is to the right of the screen, the third bit is 1 • If the region is to the left of the	if the endpoint is above the window. Bit 2- set if the endpoint is below the window. Bit 3- set to 1 if the endpoint is to the right of the window. Bit 4- set to 1 if the endpoint is to the left of the

W https://odl.ptu.ac.in/SLM/msc%20it/3RD/MSIT_301_Computer_Graphics.pdf

170/191	SUBMITTED TEXT 48 WORDS	74% MATCHING TEXT 48 WORDS
	x, int y) 161 { int code=0; if(y &lt;= h) code |= 1; //top else if(y &gt; 0) code |= 2; //bottom if(x &lt;= w) code |= 4; //right else if (x &gt; 0) code |= 8; //left return(code); }	x, float y) { outcode code = 0; if(y &lt; ymax) code| = TOP; else if(y &gt; ymin) code| = BOTTOM; if(x &lt; xmax) code| = RIGHT; else if(x &gt; xmin) code| = LEFT; return code; }

W https://studio.eecs.umich.edu/confluence/download/attachments/17827062/mathematicsforcomputergrap ...

171/191	SUBMITTED TEXT 62 WORDS	26% MATCHING TEXT 62 WORDS
	If the region is to the left of the window, the first bit of the code is set to 1. If the region is to the top of the window, the second bit of the code is set to 1. If to the right, the third bit is set, and if to the bottom, the fourth bit is set. The 4	if the endpoint is to the left of the window, second-bit set— if the endpoint is to the right of the window, third-bit set—if the endpoint is below the window, fourth-bit set— if the endpoint is above the window, fifth-bit set—if the

W https://studio.eecs.umich.edu/confluence/download/attachments/17827062/mathematicsforcomputergrap ...

172/191	SUBMITTED TEXT 19 WORDS	94% MATCHING TEXT 19 WORDS
	four inequalities to find the range of the parameter for which the line is in the viewport.	

SA 190073 computer graphics.pdf (D119942830)

173/191	SUBMITTED TEXT	14 WORDS	100%	MATCHING TEXT	14 WORDS

The sequence for reading the codes' bits is LRBT (Left, Right, Bottom, Top).

SA Unit-3 (Part-III).docx (D77528823)

174/191	SUBMITTED TEXT	25 WORDS	43%	MATCHING TEXT	25 WORDS

the window. If the logical AND of the endpoint codes is not zero, the line can be trivally rejected. For example, if an endpoint

SA Unit-3 (Part-III).docx (D77528823)

175/191	SUBMITTED TEXT	53 WORDS	96%	MATCHING TEXT	53 WORDS

The logical OR of the endpoint codes determines if the line is completely inside the window. If the logical OR is zero, the line can be trivially accepted. For example, if the endpoint codes are 0000 and 0000, the logical OR is 0000 - the line can be trivially accepted. If the

SA Unit-3 (Part-III).docx (D77528823)

176/191	SUBMITTED TEXT	21 WORDS	73%	MATCHING TEXT	21 WORDS

endpoint codes are 0000 and 0110, the logical OR is 0110 and the line cannot be trivially accepted. Algorithm The

SA Unit-3 (Part-III).docx (D77528823)

177/191	SUBMITTED TEXT	27 WORDS	100%	MATCHING TEXT	27 WORDS

To perform the trivial acceptance and rejection tests, we extend the edges of the window to divide the plane of the window into the nine regions.

SA Unit-3 (Part-III).docx (D77528823)

178/191	SUBMITTED TEXT	19 WORDS	96%	MATCHING TEXT	19 WORDS

If both codes are 0000,(bitwise OR of the codes yields 0000) line lies completely inside the window:

SA Unit-3 (Part-III).docx (D77528823)

179/191	SUBMITTED TEXT	23 WORDS	45%	MATCHING TEXT	23 WORDS

Point E is determined to be an outside point, so the algorithm clips the line against the bottom edge of the window.

SA NCIS300 COMPUTER GRAPHICS 190372.pdf (D119943374)

180/191	SUBMITTED TEXT 11 WORDS	100% MATCHING TEXT 11 WORDS
	To clip polygons, we need to modify the line-clipping procedures	To clip polygons, we need to modify the line-clipping procedures.

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

181/191	SUBMITTED TEXT 35 WORDS	86% MATCHING TEXT 35 WORDS
	This way, you get a new polygon, clipped against one boundary, and ready to be clipped against the next boundary. The method works, but has one disadvantage: To clip against all four boundaries, you	

SA INF_2046.pdf (D164968115)

182/191	SUBMITTED TEXT 30 WORDS	98% MATCHING TEXT 30 WORDS
	A polygon boundary processed with a line clipper may be displayed as a series of unconnected line segments, depending on the orientation of the polygon to the clipping window.	A polygon boundary processed with a line clipper may be displayed as a series of unconnected line segments (Fig. 4.10.), depending on the orientation of the polygon to the clipping window.

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

183/191	SUBMITTED TEXT 47 WORDS	94% MATCHING TEXT 47 WORDS
	For polygon clipping, we require an algorithm that will generate one or more closed areas that are then scan converted for the appropriate area fill. The output of a polygon clipper should be a sequence of vertices that defines the clipped polygon boundaries. Sutherland-Hodgeman Polygon Clipping	For polygon clipping, we require an algorithm that will generate one or more closed areas that are then scan converted for the appropriate area fill. The output of a polygon clipper should be a sequence of vertices that defines the clipped polygon boundaries. Fig. 4.11 Showing a Rectangle Clipping 4.4.1 Sutherland – Hodgeman Polygon Clipping:

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

184/191	SUBMITTED TEXT 29 WORDS	94% MATCHING TEXT 29 WORDS
	a polygon by processing the polygon boundary as a whole against each window edge. This could be accomplished by processing all polygon vertices against each clip rectangle boundary	A polygon can be clipped by processing the polygon boundary as a whole against each could be accomplished by processing all polygon vertices against each clip rectangle boundary.

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

185/191	SUBMITTED TEXT 13 WORDS	100% MATCHING TEXT 13 WORDS
	right boundary clipper, a bottom boundary clipper, and a top boundary clipper.	right boundary clipper, a bottom boundary clipper and a top boundary clipper,

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

186/191	SUBMITTED TEXT 33 WORDS	85% MATCHING TEXT 33 WORDS
	boundary clipper. There are four possible cases when processing vertices in sequence around the perimeter of a polygon. As each pair of adjacent polvgon vertices is passed to a window boundary clipper,	boundary. There are four possible cases when processing vertices in sequence around the perimeter As each point of adjacent polygon vertices is passed to a window boundary clipper,

W https://notes.nmeict.online/mca/COMPUTER%20GRAPHICS/files/search/bookText.xml

187/191	SUBMITTED TEXT 19 WORDS	100% MATCHING TEXT 19 WORDS
	Convex polygons are correctly clipped by the Sutherland-Hodgeman algorithm, but concave polygons may be displayed with extraneous lines.	Convex polygons are correctly clipped by the Sutherland-Hodgeman algorithm but concave polygons may be displayed with extraneous lines.? ?

W http://oms.bdu.ac.in/ec/admin/contents/175_P16MCA11_2020052205522139.pdf

188/191	SUBMITTED TEXT 128 WORDS	98% MATCHING TEXT 128 WORDS
	This clipping procedure was developed as a method for identifying visible surfaces, and so it can be applied with arbitrary polygon- clipping regions. 175 The basic idea in this algorithm is that instead of always proceeding around the polygon edges as vertices are processed, we sometimes want to follow the window boundaries. Which path we follow depends on the polygon-processing direction (clockwise or counterclockwise) and whether tile pair of polygon vertices currently being processed represents an outside-to-inside pair or an inside-to-outside pair. For clockwise processing of polygon vertices, we use the following rules: • For an outside-to-inside pair of vertices, follow the polygon boundary. • For an inside-to-outside pair of vertices,. follow the window boundary in a clockwise direction.	This clipping procedure was developed as a method for identifying visible surfaces, and so it can be applied with arbitrary polygon-clipping regions. The basic idea in this algorithm is that instead of always proceeding around the polygon edges as vertices are processed, we sometimes want to follow the window boundaries. Which path we follow depends on the polygon-processing direction (clockwise or counterclockwise) and whether the pair of polygon vertices currently being processed represents an outside-to-inside pair or an inside- to-outside pair. For clockwise processing of polygon vertices, we use the following rules: ? For an outside-to-inside pair of vertices, follow the polygon boundary. ? For an inside-to-outside pair of vertices,. follow the window boundary in a clockwise direction.

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

189/191	SUBMITTED TEXT 29 WORDS	94% MATCHING TEXT 29 WORDS
	An improvement on the Weiler-Atherton algorithm is the Weiler algorithm, which applies constructive solid geometry ideas to clip an arbitrary polygon against any polygon dipping region.	An improvement on the Weiler-Atherton algorithm is the Weiler algorithm, which applies constructive solid geometry ideas to clip an arbitrary polygon against any polygon clipping region.

W http://www.miet.edu/course/wp-content/uploads/2019/07/3.5-CG-cse_Optimized.pdf

Ouriginal by Turnitin

190/191	SUBMITTED TEXT	111 WORDS	72%	MATCHING TEXT	111 WORDS

If the first vertex is outside the window boundary and the second vertex is inside, both the intersection point of the polygon edge with the window boundary and the second vertex are added to the output vertex list. (2) If both input vertices are inside the window boundary, only the second vertex is added to the output vertex list. (3) If the boundary and the second vertex is outside, only first vertex is inside the window, the edge intersection with the window boundary is added to the output vertex list. (4) If both input vertices are outside the window boundary, nothing is added to the output list.

SA Computer_Graphics_Block_2.pdf (D149210060)

191/191	SUBMITTED TEXT	17 WORDS	62%	MATCHING TEXT	17 WORDS

are the coordinates of any point on the circle, r is the radius of the circle

SA ODMCA-404T Palwinder 10th Feb 2020.docx (D63728007)