Lecture 1

Introduction to Computer Graphics

Copyright@2009, Aous Hammad Al-obaidi, M.Sc
AL-MAARF UNIVERSITY COLLEGE

Sample output to test PDF Combine only

• What is Computer Graphics ?

Computer Graphics: the term has become so widespread now, that we rarely stop to think about what it means. What is Computer Graphics? Simply defined, Computer Graphics (or CG) is the images generated or modified on a computer. These images may be visualizations of real data or imaginary depictions of a fantasy world.

• What is Computer Vision ?

Computer vision is the Analysis, reconstruction, and recognition of 3D object 2D images, using computers. Computer vision and computer graphics are two complementary processes

- Image enhancement
- Feature extraction
- object reconstruction and recognition

• Pixels

The fundamental building block of all computer images is the picture element, or the pixel. A pixel is a dot of light on the computer screen that can be set to different colors. An image displayed on the computer, no matter how complex, is always composed of rows and columns of these pixels, each set to the appropriate color and intensity. The trick is to get the right colors in the right places.

• Computer Display Systems

The computer display, or the monitor, is the most important device on the computer. It provides visual output from the computer to the user. In the Computer Graphics context, the display is everything. Most current personal computers and workstations use Cathode Ray Tube (CRT) technology for their displays.

Copyright@2009, Aous Hammad Al-obaidi, M.Sc
AL-MAARF UNIVERSITY COLLEGE

Sample output to test PDF Combine only

• Applications of Computer Graphics

- Display of information
  1. "An image is worth 1000 words"
  2. Statistical plots, charts
  3. Cartography
  4. Medical imaging
- Design
  1. CAD in Engineering and Architecture
  2. Industrial design
- Simulation
  1. Virtual reality environments
  2. Training and simulation
  3. Scientific visualization
- Entertainment
  1. Computer animations in motion pictures
  2. 3D games
  3. Interactive multimedia

Copyright@2009, Aous Hammad Al-obaidi, M.Sc
AL-MAARF UNIVERSITY COLLEGE

Sample output to test PDF Combine only

Lecture 2

Basic Review: Linear Algebra and Image Files Formats

Copyright@2009, Aous Hammad Al-obaidi, M.Sc
AL-MAARF UNIVERSITY COLLEGE

Sample output to test PDF Combine only

• Vectors and Matrices

Before we jump into the fairly mathematical discussion of transformations, let us first brush up on the basics of vector and matrix math. This math will form the basis for the transformation equations we shall see later. We discuss the math involved as applied to a 2D space. The principles are easily extended to 3D by simply adding a third axes, namely, the z- axis.

• Vectors

A vector is a quantity that has both direction and length. In CG, a vector represents a directed line segment with a start point (its tail) and an end point (the head, shown typically as an arrow pointed along the direction of the vector). The length of the line is the length of the vector.

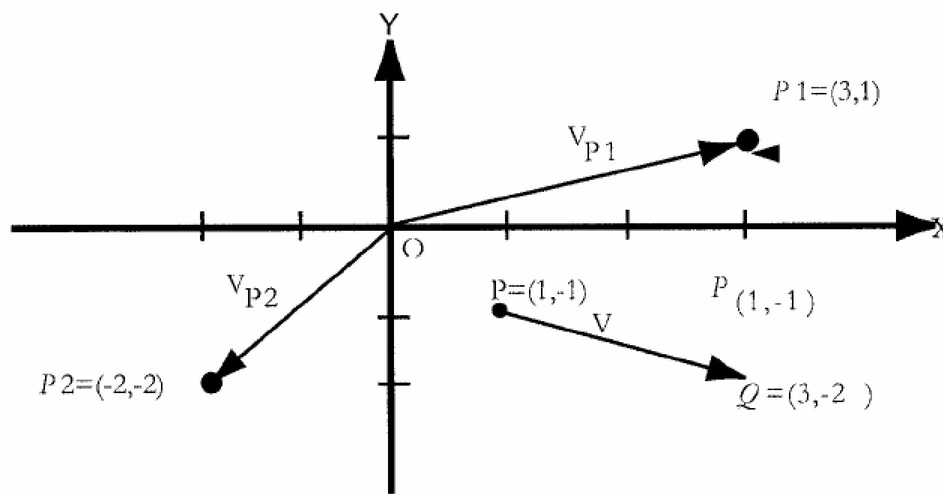


Fig. 2.1: A 2D VectorlPoint: A directional line

A 2D vector that has a length of x units along the x-axis and y units along the y-axis is denoted as $\begin{bmatrix} x \\ y \end{bmatrix}$

It is valuable to think of a vector as a displacement from one point to another. Consider two points *P*(1,-1) and *Q*(3,-2), as shown in Figure 2.1. The displacement from *P* to *Q* is the vector $V = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$, calculated by subtracting the coordinates of the points individually. What this means is that to get from *P* to *Q*, we shift right along the x-axis by two units and down the y-axis by one unit. Interestingly, any point P1 with coordinates (*x*,*y*) corresponds to the vector V_{P1}, with its head at (*x*,*y*) and tail at (0,0) as shown in Figure 2.1. That is, there is a one-to-one correspondence between a vector, with its tail at the origin, and a point on the plane. This means that we can also represent the point *P*(*x*,*y*) by the vector $\begin{bmatrix} x \\ y \end{bmatrix}$.

Copyright@2009, Aous Hammad Al-obaidi, M.Sc
AL-MAARF UNIVERSITY COLLEGE

Sample output to test PDF Combine only

Often, the math of transformation equations uses the vector representation of points in this manner, so do not let this usage confuse you.

- **Operations with Vectors**

Vectors support some fundamental operations: addition, subtraction, and multiplication with a real number. Vectors can be added by performing component wise addition. If V_1 is the vector (x_1,y_1) and V_2 is the vector (x_2,y_2) then V_1+V_2 is

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 + x_2 \\ y_1 + y_2 \end{bmatrix}$$

Conceptually, adding two vectors results in a third vector which is the addition of one displacement with another.

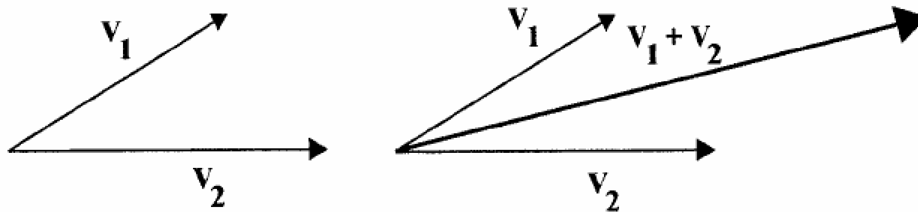


Fig. 2.2: Adding two vectors

Multiplying a vector by a number *s* results in a vector whose length has been scaled by *s*. For this reason, the number s is also referred to as a scalar. If *s* is negative, then this results in a vector whose direction is flipped as well.

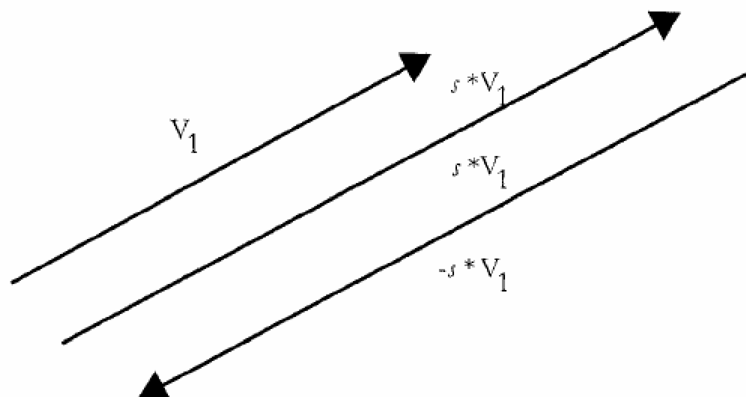


Fig. 2.3: Multiplying a vector with a scalar

Copyright@2009, Aous Hammad Al-obaidi, M.Sc
AL-MAARF UNIVERSITY COLLEGE

Sample output to test PDF Combine only

Mathematically, the scaled vector $sV_1 = \begin{bmatrix} s*x_1 \\ s*y_1 \end{bmatrix}$

Subtraction follows easily as the addition of a vector that has been flipped: that is V_1-V_2 = V_1+ (-V_2).

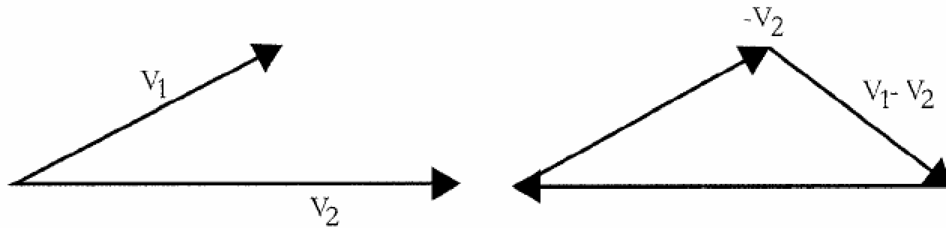


Fig. 2.4: Subtracting two vectors

- **The Magnitude of a Vector**

The length of a vector $V = \begin{bmatrix} x \\ y \end{bmatrix}$ is also referred to as its magnitude. The magnitude of a vector is the distance from the tail to the head of the vector. It is represented as $|v|$ and is equal to $\sqrt{x^2 + y^2}$

It is very useful to scale a vector so that the resultant vector has a length of 1, with the same direction as the original vector. This process is called *normalizing* a vector. The resultant vector is called a *unit vector*. To normalize a vector V, we simply scale it by the value $1/|v|$. The resultant unit vector is represented as $V=V/|v|$.

Interestingly, a unit vector along the x-axis is quite simply the vector $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$

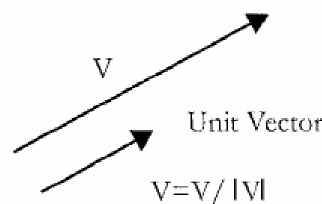


Fig. 2.5: A Unit Vector

and a unit vector along the y axis is $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Copyright@2009, Aous Hammad Al-obaidi, M.Sc
AL-MAARF UNIVERSITY COLLEGE

Sample output to test PDF Combine only

- **The Dot Product**

The dot product of two vectors is a scalar quantity. The dot product is used to solve a number of important geometric problems in graphics. The dot product is written as $V_1 .V_2$ and is calculated as

$$v_1 \bullet v_2 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \bullet \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = x_1 * y_1 + x_2 * y_2$$

The dot product is also related to the angle between the two vectors, but it doesn't tell us the angle

$$a.b = \|a\|\|b\|\cos(\phi)$$

- **Matrices**

A matrix is an array of numbers. The number of rows (m) and columns (n) in the array defines the cardinality of the matrix ($m * n$). In reality, a vector is simply a matrix with one column (or a one-dimensional matrix). We saw how displays are just a matrix of pixels. A frame buffer is a matrix of pixel values for each pixel on the display.

Matrices can be added component wise, provided they have the same cardinality:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix}$$

Multiplication of a matrix by a scalar is simply the multiplication of its components by this scalar:

$$s * \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} s * a_{11} & s * a_{12} \\ s * a_{21} & s * a_{22} \end{bmatrix}$$

Two matrices can be multiplied if and only if the number of columns of the first matrix ($m * n$) is equal to the number of rows of the second ($n * p$). The result is calculated by applying the dot product of each row of the first matrix with each column of the second. The resultant matrix has a cardinality of ($m * p$).

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} * \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} a_{11} * b_{11} + a_{12} * b_{21} + a_{13} * b_{31} & a_{11} * b_{12} + a_{12} * b_{22} + a_{13} * b_{32} \\ a_{21} * b_{11} + a_{22} * b_{21} + a_{23} * b_{31} & a_{12} * b_{12} + a_{22} * b_{22} + a_{23} * b_{32} \end{bmatrix}$$

An *identity matrix* is a square matrix (an equal number of rows and columns) with all zeroes, except for 1s in its diagonal.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Copyright@2009, Aous Hammad Al-obaidi, M.Sc
AL-MAARF UNIVERSITY COLLEGE

Sample output to test PDF Combine only

Multiplication of a vector/matrix by an identity matrix has no effect. Prove this by multiplying a vector with the appropriate identity matrix

- ## Determinant of a Matrix

Determinant is very important to find inversion, If det(A) = 0, then A has no inverse. For two dimensional matrixes like $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$:

$$Det(A) = ad - bc$$

Using Laplace expansion around any row or column says the first row for

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

$$Det(A) = (-1)^{1+1}(a)\begin{vmatrix} e & f \\ h & i \end{vmatrix} + (-1)^{1+2}(b)\begin{vmatrix} d & f \\ g & i \end{vmatrix} + (-1)^{1+3}(c)\begin{vmatrix} d & e \\ g & h \end{vmatrix} + \ldots\ldots + (-1)^{3+3}(i)\begin{vmatrix} a & b \\ d & e \end{vmatrix}$$

- ## Inverse of a Matrix

It can be found using cofactors, factorial and pivots. If det(A) = 0, then A has no inverse. The inverse for matrix of any dimension can be found as fellow:

$$A^{-1} = \frac{(cof(a))^T}{|A|}$$

Where A is n*n matrix.

- # Image file format

1. Need to store an image on disk
   - Real photos
   - Synthetic renderings
   - Composed images (Multiple sources)
2. Desirable Features
   - High quality (Lossy vs Lossless formats ,Channel depth – bit per pixel)
3. Small file size
   - Quality of compression
   - Small overhead – small headers

Copyright@2009, Aous Hammad Al-obaidi, M.Sc
AL-MAARF UNIVERSITY COLLEGE

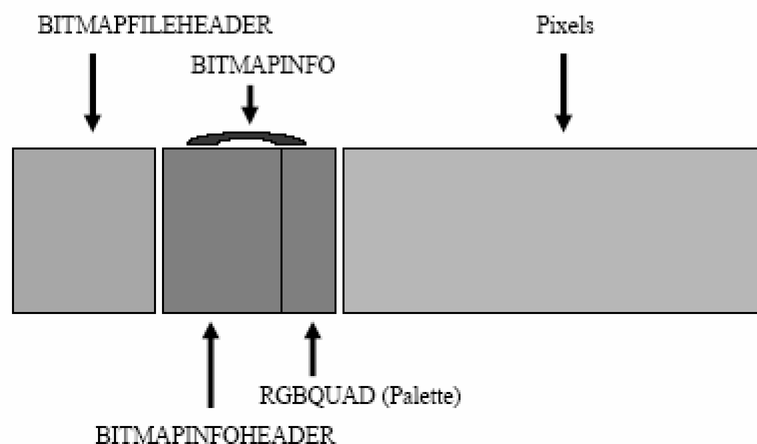
Sample output to test PDF Combine only

4. Application data (Save application specific data of the image processing tool)

- **Bitmap**

Its One of the most common image formats available,very simple to implement and inefficient in storage. it has wide support on Windows but still exist on Unix too . Bitmap is a raster Data In contrast to vector data like postscript.it consist of amatrix of pixels

Like most common image formats, a bitmap image consists of *Header* which contains descriptive information about the image, such as width, height, etc. *Body* which contains the actual (raster scanned) colors of the image pixels.



- **BITMAPFILEHEADER**

```
typedef struct tagBITMAPFILEHEADER
{
        WORD bfType;
        DWORD bfSize;
        WORD bfReserved1;
        WORD bfReserved2;
        DWORD bfOffBits;
} BITMAPFILEHEADER;
```

The BITMAPFILEHEADER structure contains information about the type, size, and layout of a file that contains a device-independent bitmap (DIB).

- bfType - Specifies the file type. It must be BM.
- bfSize - Specifies the size, in bytes, of the bitmap file.

Copyright@2009, Aous Hammad Al-obaidi, M.Sc
AL-MAARF UNIVERSITY COLLEGE

Sample output to test PDF Combine only

- bfOffBits - Specifies the offset, in bytes, from the BITMAPFILEHEADER structure to the bitmap data.

A BITMAPINFO structure immediately follows the BITMAPFILEHEADER structure in the DIB file

- **BITMAPINFO**

```
typedef struct tagBITMAPINFO
{
        BITMAPINFOHEADER        bmiHeader;
        RGBQUAD         bmiColors[1];
} BITMAPINFO;
```

The BITMAPINFO structure combines the BITMAPINFOHEADER structure and a color table to provide a complete definition of the dimensions and colors of a DIB.

- **BITMAPINFOHEADER**

```
typedef struct tagBITMAPINFOHEADER
{
        DWORD biSize;
        LONG biWidth;
        LONG biHeight;
        WORD biPlanes;
        WORD biBitCount;
        DWORD biCompression;
        DWORD biSizeImage;
        LONG biXPelsPerMeter;
        LONG biYPelsPerMeter;
        DWORD biClrUsed;
        DWORD biClrImportant;
} BITMAPINFOHEADER;
```

- biSize – size of the struct in bytes
- biWidth – width in pixels
- biHeight – height in pixels
- biPlanes – layers in bitmap – must be 1
- biBitCount – bit per pixel 1,2,4,8,16,24,32 we can use 24 bits for 3 channels with no palette images.
- More fields – look at Visual .Net manual for the rest of the fields. The above are the most important.

Copyright@2009, Aous Hammad Al-obaidi, M.Sc
AL-MAARF UNIVERSITY COLLEGE

Sample output to test PDF Combine only

Lecture 3

Line and Circle Drawing (Part One)

Copyright@2009, Aous Hammad Al-obaidi, M.Sc
AL-MAARF UNIVERSITY COLLEGE

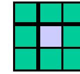
Sample output to test PDF Combine only

• Raster Display

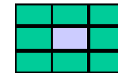
- Discrete grid of elements (frame buffers pixels)
- Shapes drawn by setting the "right" elements
- Frame buffer is scanned, one line at a time, to refresh the image (as opposed to vector display)
- Properties :
  1. Difficult to draw smooth lines
  2. Displays only a discrete approximation of any shape
  3. Refresh of entire frame buffer

• Terminology

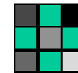
- Pixel: Picture element
  1. Smallest accessible element in picture
  2. Usually rectangular or circular



- Aspect Ratio: Ratio between physical dimensions of pixel
  1. (not necessarily 1 !!)

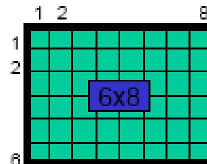


- Dynamic Range: Ratio between minimal (not zero!) and maximal light intensity emitted by displayed pixel

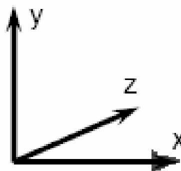


- Resolution: number of distinguishable rows and columns on device Measured in
  1. Absolute values (1K x 1K)
  2. Relative values (300 dots per inch)

- Screen Space: discrete 2D Cartesian coordinate system of screen pixels



- Object Space: 3D Cartesian coordinate system of the universe where the objects (to be displayed) are embedded



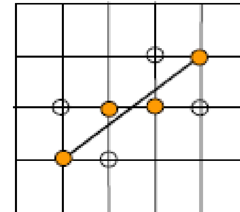
Copyright@2009, Aous Hammad Al-obaidi, M.Sc
AL-MAARF UNIVERSITY COLLEGE

Sample output to test PDF Combine only

• Basic Line Drawing

Assume x1< x2 and line slope absolute value is ≤ 1

```
Line (x1,y1, x2,y2)
begin
      float dx,dy,x,y,slope;
      dx=x2-x1;
      dy=y2-y1;
      slope=dx/dy;
      y=y1;
      for x from x1 to x2 do
      begin
            PlotPixel(x,Round(y));
            y=y+slope;
      end;
end;
```



Questions:

1. Can this algorithm use integer arithmetic?
2. Does it accumulate error?
3. Is the error significant?

• Recursive Line Drawing

Simple, recursive, integer, line drawing:

```
Line (x1,y1, x2,y2)
begin
      int x,y;
      x=(x2+x1)/2;
      d=(y2+y1)/2;
      if((x= x1 and  y=y1) or (x= x2 and  y=y2))
            return;
      else begin
            PlotPixel(x, y);
            Line(x1,y1,x,y);
            Line(x,y,x2,y2);
      end;
end;
```

Copyright@2009, Aous Hammad Al-obaidi, M.Sc
AL-MAARF UNIVERSITY COLLEGE

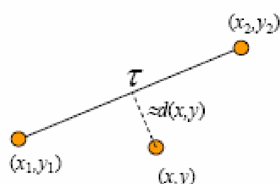
Sample output to test PDF Combine only

- **Potential Problems:**
  1. Line is not drawn sequentially.
  2. Function call for each pixel.
  3. Slow and non accurate algorithm.

• Midpoint (Bresenham) Algorithm

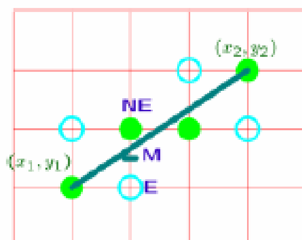
Assumptions:

$$x_2 > x_1, y_2 > y_2 \quad and \quad \frac{dy}{dx} = \frac{y_2 - y_1}{x_2 - x_1} < 1$$



Idea:
1. Define error function
2. Proceed along the line incrementally
3. Select direction that minimizes accumulated error

Algorithm
1. Starting point satisfies d(x_1, y_1)= 0 .
2. Each step moves right (east) or upper right (northeast).
3. Sign of $d(x_1+1, y_1+½)$ indicates whether to move east or northeast .
4. At (x_1,y_1):
   d_{start}= d(x_1+1,y_1+½)=2dy-dx .
5. Increment in d (after each step)
   Move east: $\Delta_e = d(x+2, y+½) - d(x+1, y+½) = 2dy$
   Move northeast: $\Delta_{ne} = d(x+2, y+{}^3/_2) - d(x+1, y+½) = 2(dy-dx)$



Sample output to test PDF Combine only

```
Line (x1,y1, x2,y2)
begin
        int x,y,dx,dy,De,Dne,d;
        x=x1;
        y=y1;
        dx=x2-x1;
        dy=y2-y1;
        d=2*dy-dx;
        De=2*dy;
        Dne=2*(dy-dx);
        PlotPixel(x, y);
        While(x<x2) do
                If ( d<0) then
                begin
                        d=d +De;
                        x=x+1;
                end;
                else begin
                        d=d +Dne;
                        x=x+1;
                        y=y+1;
                end;
                PlotPixel(x, y);
        end;
end;
```

Copyright@2009, Aous Hammad Al-obaidi, M.Sc
AL-MAARF UNIVERSITY COLLEGE

Sample output to test PDF Combine only