

ICS370 Computer Graphics

Instructor: Toby Daniel

Lecture 3

Lab 1

Implications of the graph?

Homework #1

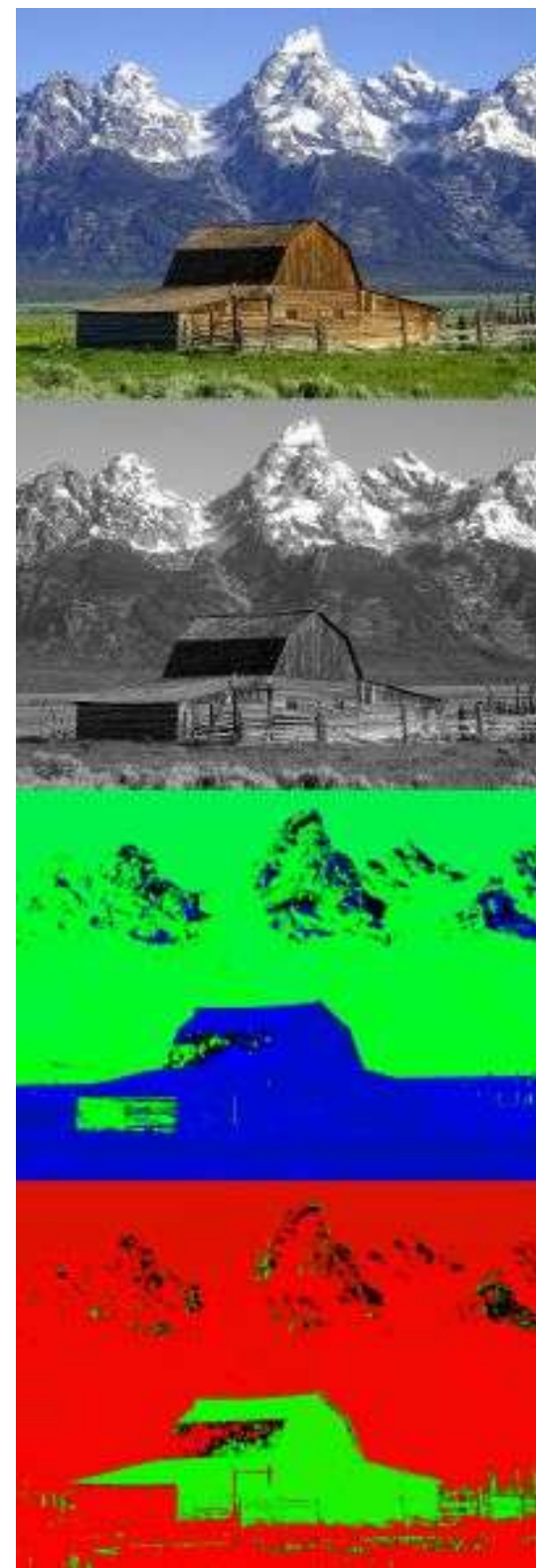
Focal Points?

Colour Space Models

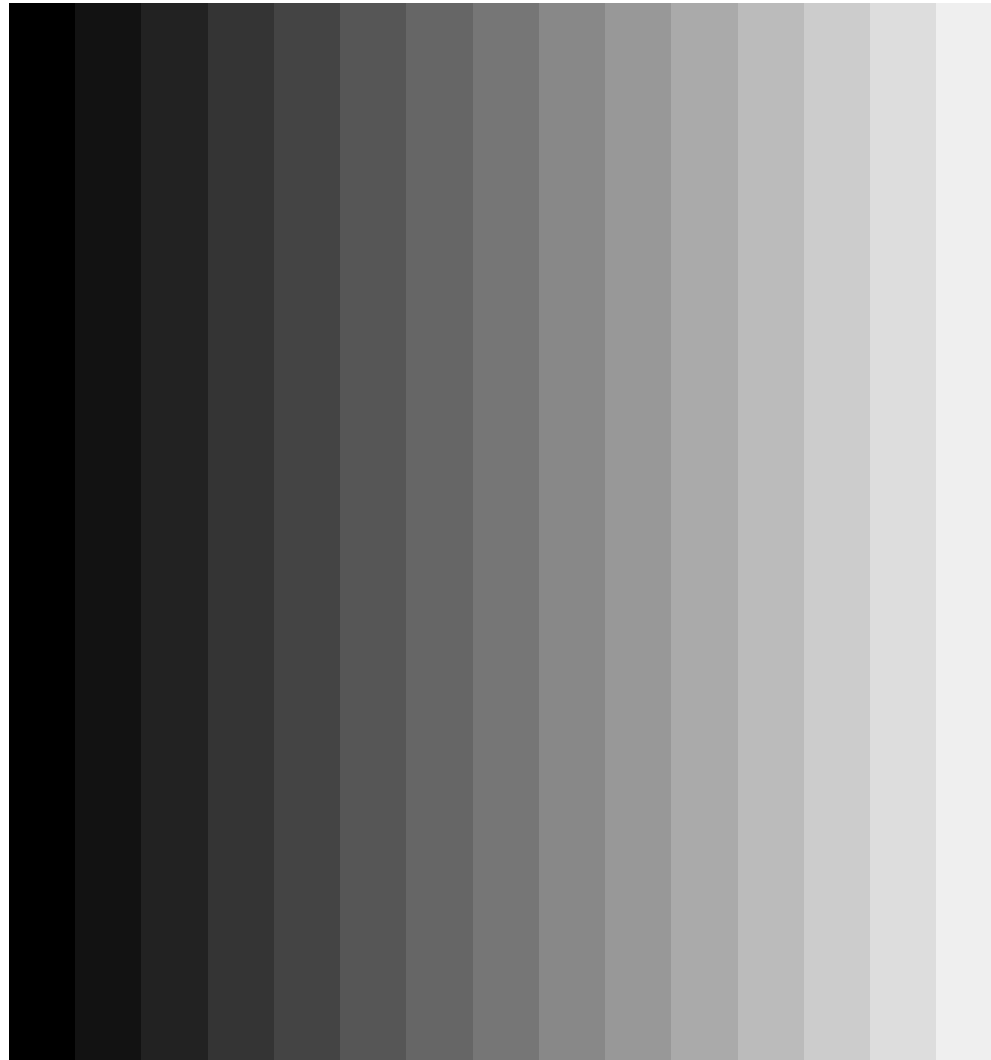
- RGB
- CMYK
- YUV
- Grey scale

YUV

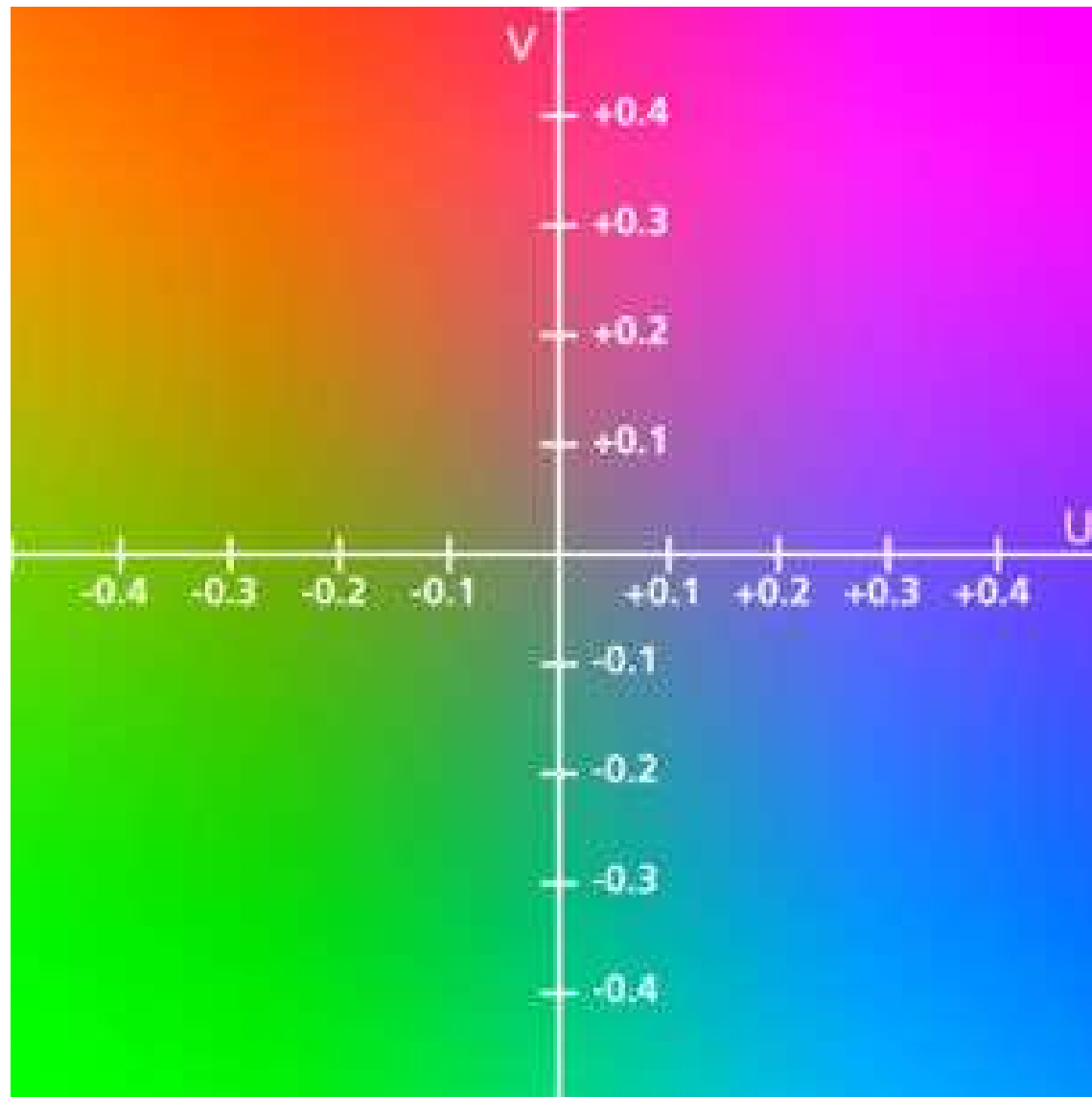
- The YUV model defines a colour space in terms of one **luminance** and two **chrominance** components. YUV is used in the **analogue** variant of the PAL system of television broadcasting, which is the standard in much of the world.
- YUV models human perception of colour more closely than the standard RGB model used in computer graphics hardware.
- Y stands for the luminance component (the brightness) and U and V are the chrominance (colour) components.



Luminance Component (Y)



Chrominance Component (UV)



Data Compression

- Lossless Compression

?

- Lossy Compression

?

Lossless Compression

- Many lossless compression programs use a variation of the **LZ adaptive dictionary-based algorithm** to shrink files.
- **Lempel and Ziv**, the algorithm's creators, and "dictionary" refers to the method of **cataloguing** pieces of data.
- The system for arranging dictionaries varies, but it could be as simple as a numbered list. When we go through text, we pick out the words that are repeated and put them into the numbered index. Then, we simply write the number instead of writing out the whole word.
- Keep in mind that we need to **save the dictionary itself along with the file**.

File-reduction Ratios for Lossless Compression

- The **file-reduction ratio** depends on a number of factors, including file type, file size and compression scheme.
- **Text files** compress very well. A reduction of 50 percent or more is typical for a good-sized text file.
- **Programming languages** are also very redundant because they use a relatively small collection of commands, which frequently go together in a set pattern.
- Files that include a lot of unique information, such as **Graphics** or **Audio** files, cannot be compressed much with this system because they don't repeat many patterns.



Huffman Compression Algorithm



- The Huffman compression algorithm is named after its inventor, David Huffman, formerly a professor at MIT.
- Huffman compression is a lossless compression algorithm that is ideal for compressing text or program files.
- "Huffman Codes" are used in many applications that involves the compression and transmission of digital data, such as fax machines, modems and computer networks.
- BUT it is not so good applied directly to graphics files, especially photographs. The best known lossless compression methods can compress such data about 2:1 on average.

Lossy Compression

- **Lossy compression** works very differently. This method of compression eliminates "unnecessary" bits of information, changing the file so that it is smaller.
- This type of compression is used a lot for reducing the file size of bitmap pictures, which tend to be fairly bulky.
- Lossy Compressions file types:
 - GIF
 - PNG
 - JPEG

JPEG Compression

- JPEG can typically achieve 10:1 to 20:1 compression **without visible loss**, bringing the effective storage requirement down to 1 to 2 bits/pixel.
- 30:1 to 50:1 compression is possible with small to moderate defects,
- Very-low quality purposes such as previews or archive indexes, 100:1 compression is quite feasible.
- JPEG is inefficient at compressing black and white and grey scale images.

How does it do it?

JPEG - 6 Step Process

1. Transform the colour space
2. Downsample the image
3. Frequency Mapping
4. Quantizing
5. Encode (lossless compression)
6. Add headers to the file

1. Transform the colour space

- For colour images you generally want to transform RGB into a luminance/chrominance color space (YUV).
- The reason for doing this is that you can afford to lose a lot more information in the chrominance components than you can in the luminance component.
- The human eye is not as sensitive to high-frequency chroma info as it is to high-frequency luminance.
- Colour space transformation is slightly lossy due to round-off error, but the amount of error is much smaller than later stages.

2. Downsample the image

- The luminance component is left at full resolution, while the chroma components are downsampled by averaging together groups of pixels
- Often reduced 2:1 horizontally and either 2:1 or 1:1 vertically. In JPEG-speak these are called 2h2v and 2h1v sampling, or "411" and "422" sampling.
- This reduces the data volume by one-half or one-third. It is highly lossy, but for most images it has almost no impact on perceived quality, because of the eye's poorer resolution for chroma info.
- Note that downsampling is not applicable to grayscale data; this is one reason colour images are **more compressible** than grayscale.

3. Frequency Mapping

- Group the pixel values for each component (Luminance and Chrominance) into 8x8 blocks.
- Transform each 8x8 block through a discrete cosine transform (DCT). The DCT is a relative of the Fourier transform and gives a frequency map for each 8x8 block.
- You now have numbers representing the average value in each block and successively higher-frequency changes within the block.
- Note that higher frequencies are less visible to the eye.

4. Quantizing

- You can now throw away high-frequency information without affecting low-frequency information.
- In each block, divide each of the 64 frequency components by a "quantization coefficient", and round the results to integers. **This is the fundamental information-losing step.** The larger the quantization coefficients, the more data is discarded. Note that even the minimum quantization coefficient, 1, loses some info, because the exact DCT outputs are typically not integers.
- Higher frequencies are always quantized less accurately (larger coefficients) than lower.
- Luminance data is typically quantized more accurately than the chroma data.

5. Encode (lossless compression)

- Encode the reduced coefficients using either Huffman or arithmetic coding.
- Baseline JPEG only allows Huffman coding; arithmetic coding is an optional extension.
- Notice that this step is lossless, so it doesn't affect image quality.
- Most existing implementations support only the Huffman code, so as to avoid license fees.

6. Add headers to the file

- Add on appropriate headers to the compressed file and output the result.
- In a normal JPEG file, all of the compression parameters are included in the headers so that the decompressor can reverse the process.
- These parameters include the quantization tables and the Huffman coding tables. These add several hundred bytes to the file size.

For more detailed information: Wallace, Gregory K. "The JPEG Still Picture Compression Standard", *Communications of the ACM*, April 1991 (vol. 34 no. 4), pp. 30-44.

JPEG Compression

Bitmap 72KB



JPEG High 29KB



JPEG Medium 24KB



JPEG Low 21KB



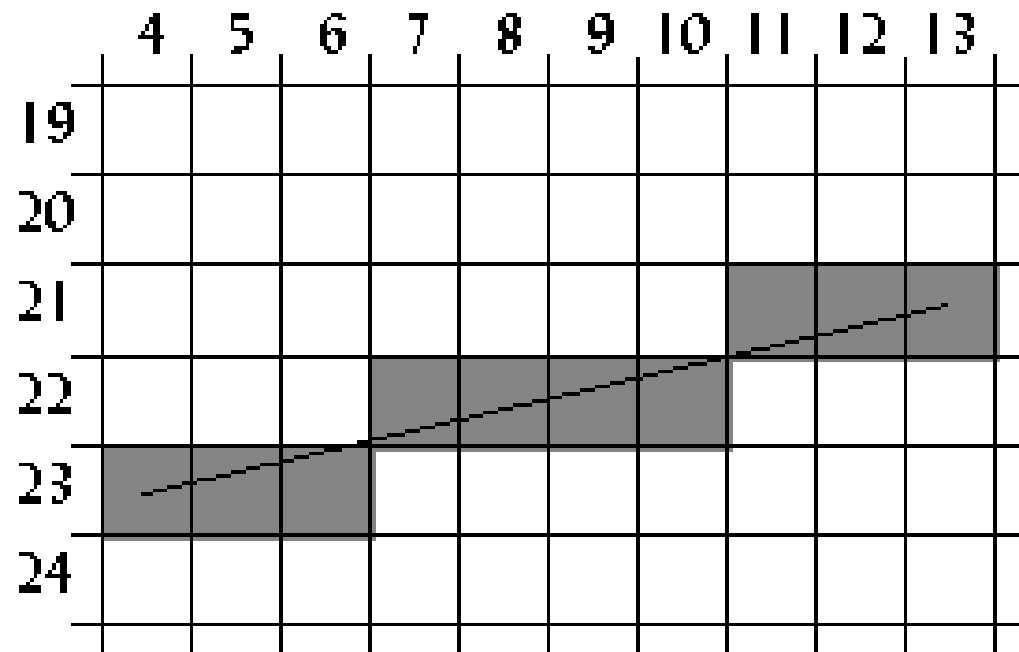
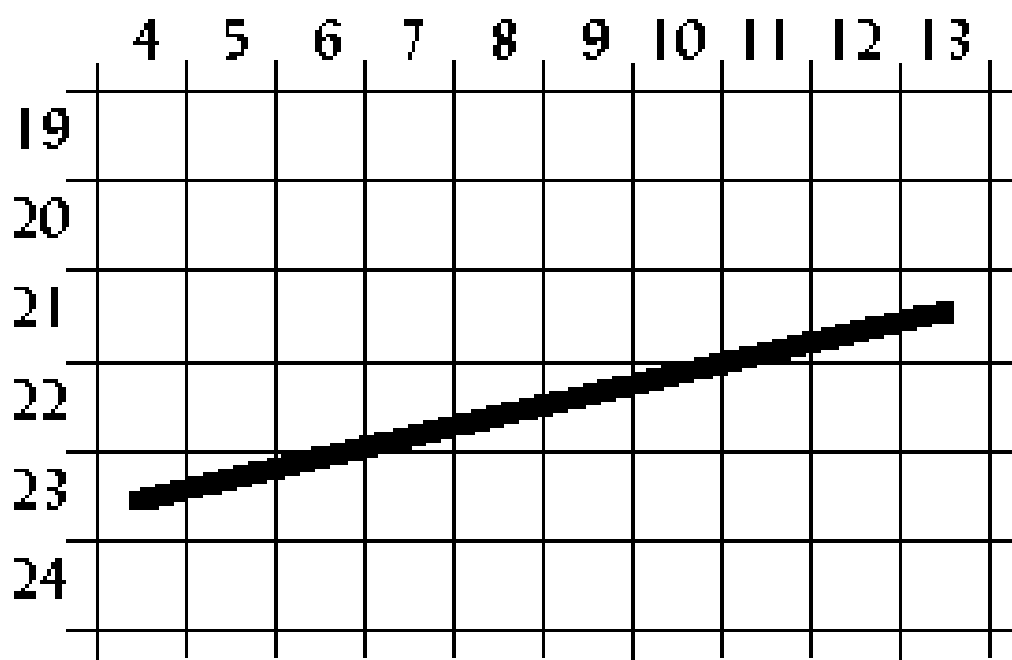
Primitive Shapes & Lines

Four basic drawing elements:

- lines
- rectangles
- circles
- arcs

Problems with lines

- Lines can be defined by the two end points ... however..
- A computer screen is arranged in a grid of horizontal and vertical lines. A diagonal line drawn on the screen will cross both horizontal and vertical grid lines



Drawing lines

- One way to draw a line is to first calculate the slope of the line, then plot a pixel at each specified step along the major axis. To do this, we use a form of the point-slope equation of a line, which is

$$y = \text{slope} (x-x_1) + y_1$$

- Assuming the x axis is the major axis (the line is closer to horizontal than vertical in the previous slide) then the slope is

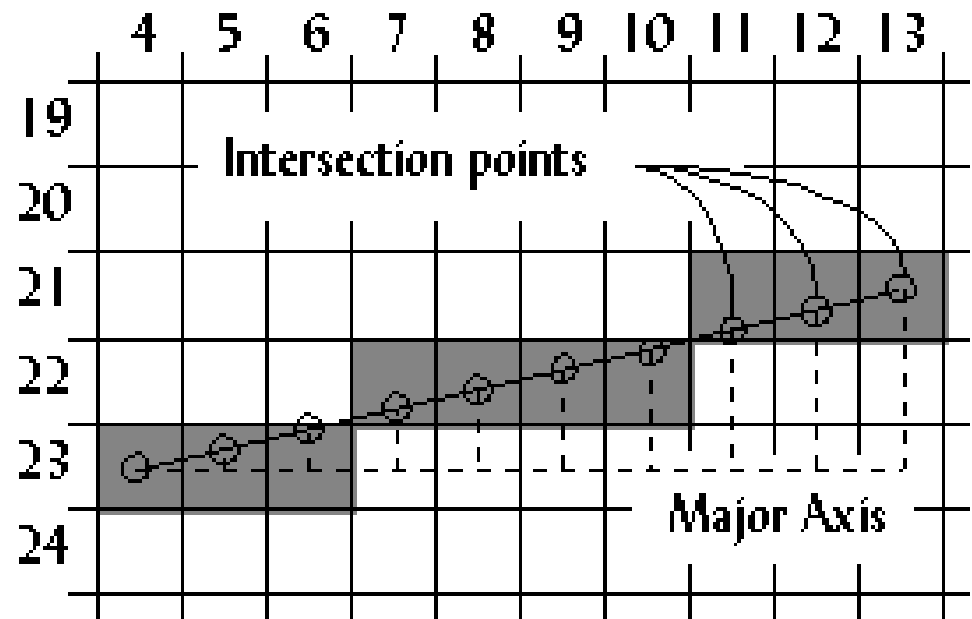
$$\text{slope} = \frac{y_2 - y_1}{x_2 - x_1}$$

See the next slide for it in practice

Drawing Lines

$$x \quad y = \text{slope}(x-x_1)+y_1$$

4	$-2/9(4-4) + 23 = 23$
5	$-2/9(5-4) + 23 = 22.778 = 23$
6	$-2/9(6-4) + 23 = 22.556 = 23$
7	$-2/9(7-4) + 23 = 22.333 = 22$
8	$-2/9(8-4) + 23 = 22.111 = 22$
9	$-2/9(9-4) + 23 = 21.889 = 22$
10	$-2/9(10-4) + 23 = 21.667 = 22$
11	$-2/9(11-4) + 23 = 21.444 = 21$
12	$-2/9(12-4) + 23 = 21.222 = 21$
13	$-2/9(13-4) + 23 = 21$



Circles

- If the circle is centred at the origin $(0, 0)$, then its formula can be expressed as

$$x^2 + y^2 = r^2$$

- Expressed in parametric equations, (x, y) can be written using the trigonometric functions sine and cosine as

$$x = a + r \cos(t)$$

$$y = b + r \sin(t).$$

Arcs

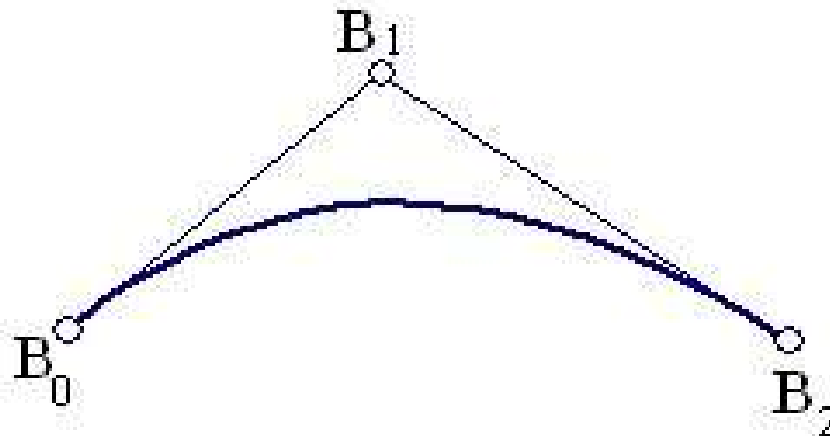
- *Bezier-arcs* are curves defined by parametric equations of the form

$$P(t) = \sum_{i=0}^n B_i J_{n,i}(t) \quad 0 \leq t \leq 1$$

- where B_0, \dots, B_n are points of the plane and $J_{n,i}(t)$ are the so-called Bezier or Bernstein or blending functions

$$J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i} = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}.$$

- The curve has coordinate functions that are polynomial of degree n . The n points B_0, \dots, B_n are called *control points* of the arc.

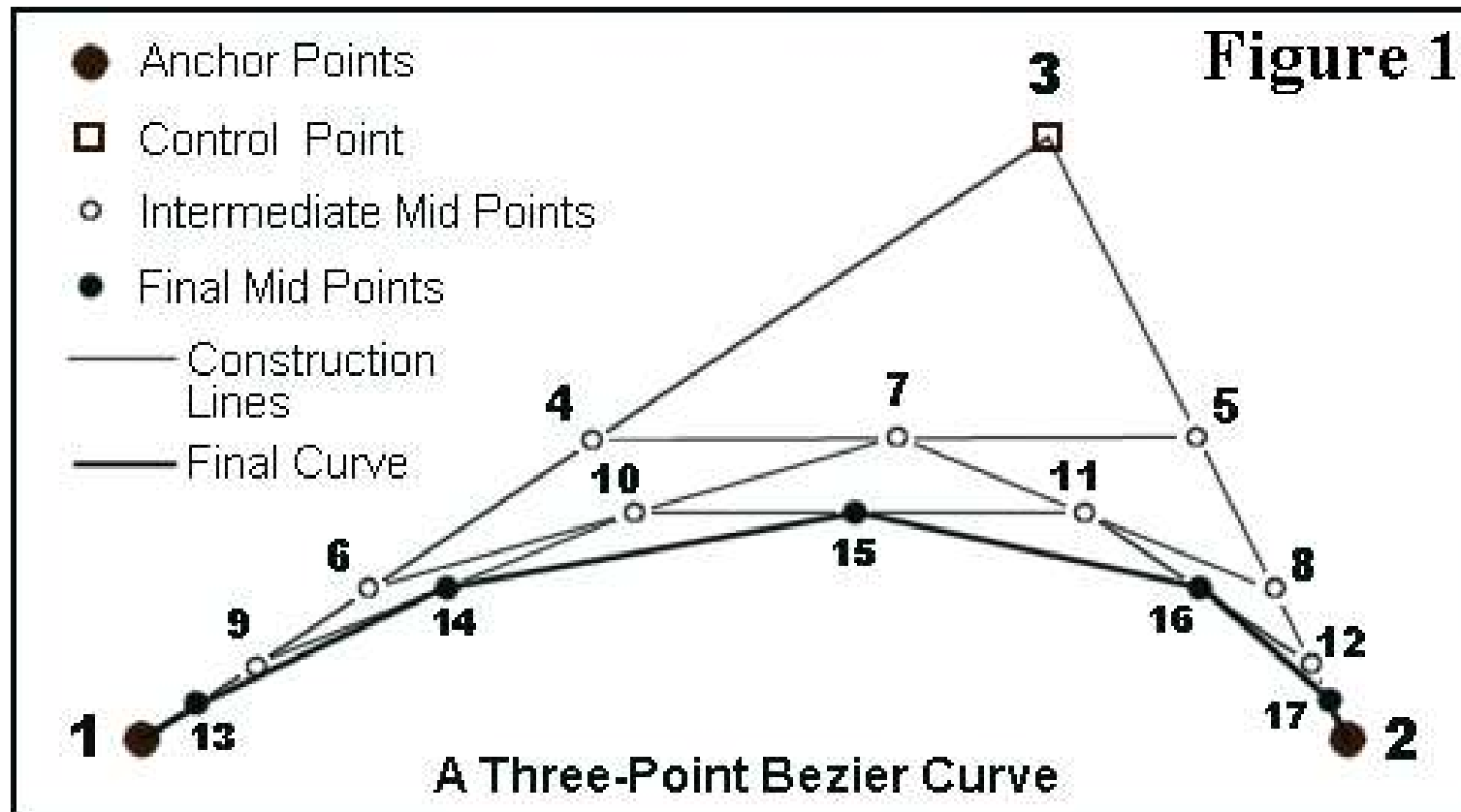




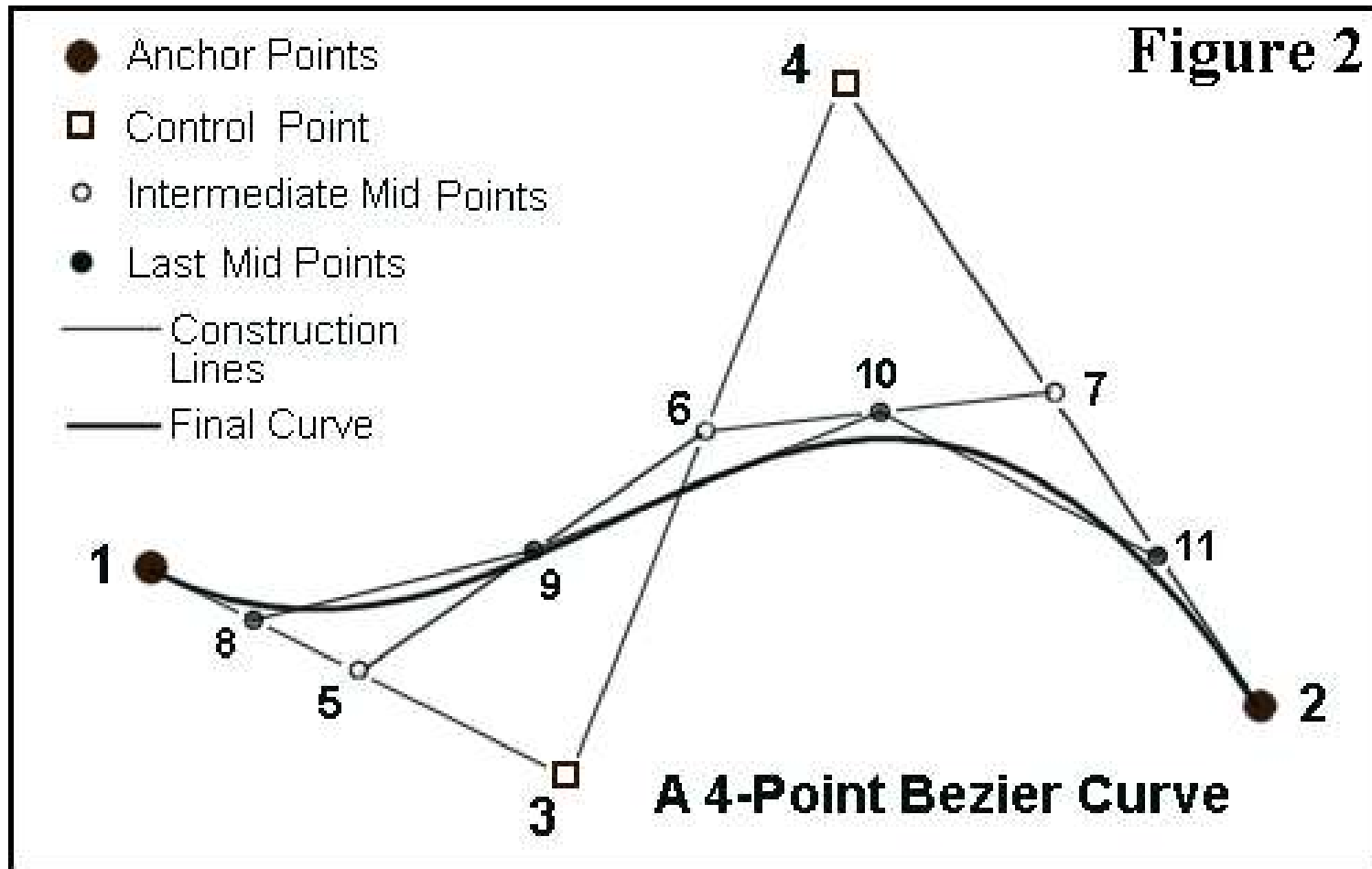
Pierre Bézier



- Pierre Bézier worked for the French Renault car manufacturing corporation in the 1960's and designed a way to describe a versatile variety of mathematical curves.



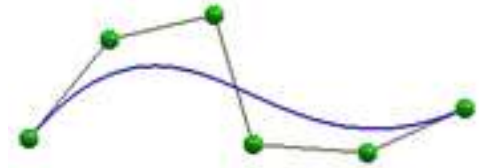
4-point Bézier curve



These curves are also called Splines



NURBS



Non-Uniform, **R**ational **B**-Splines are a generalization of Bézier curves.

NURBS are industry standard tools for the representation and design of geometry. Some reasons for the use of NURBS are, that they:

- offer one common mathematical form for both, standard analytical shapes and free form shapes;
- describe 2D and **3D curved surfaces**
- can be evaluated reasonably fast by numerically stable and accurate algorithms;
- are generalizations of non-rational B-splines and non-rational and rational Bezier curves and surfaces.

Drawing Primitive Shapes

- Many programming languages have built in functions that create these basic shapes.
- For example to draw a line you would only specify the end coordinates not the individual pixels.

2D shapes:

- Line
- Rectangle / square
- Oval / Circle
- Arc or curves

2D Transforms

- A 2D Transform is a mathematical operation that transforms a shape to a new form.
- An **Affine Transform** is a common set of transformations in 2 dimensions. It performs a linear mapping from 2D coordinates to other 2D coordinates that preserves the "**straightness**" and "**parallelness**" of lines.
- Affine transformations are translations, scales, flips, rotations, and shears.

...to be continued