Monash University • Clayton's School of Information Technology

CSE3313 Computer Graphics

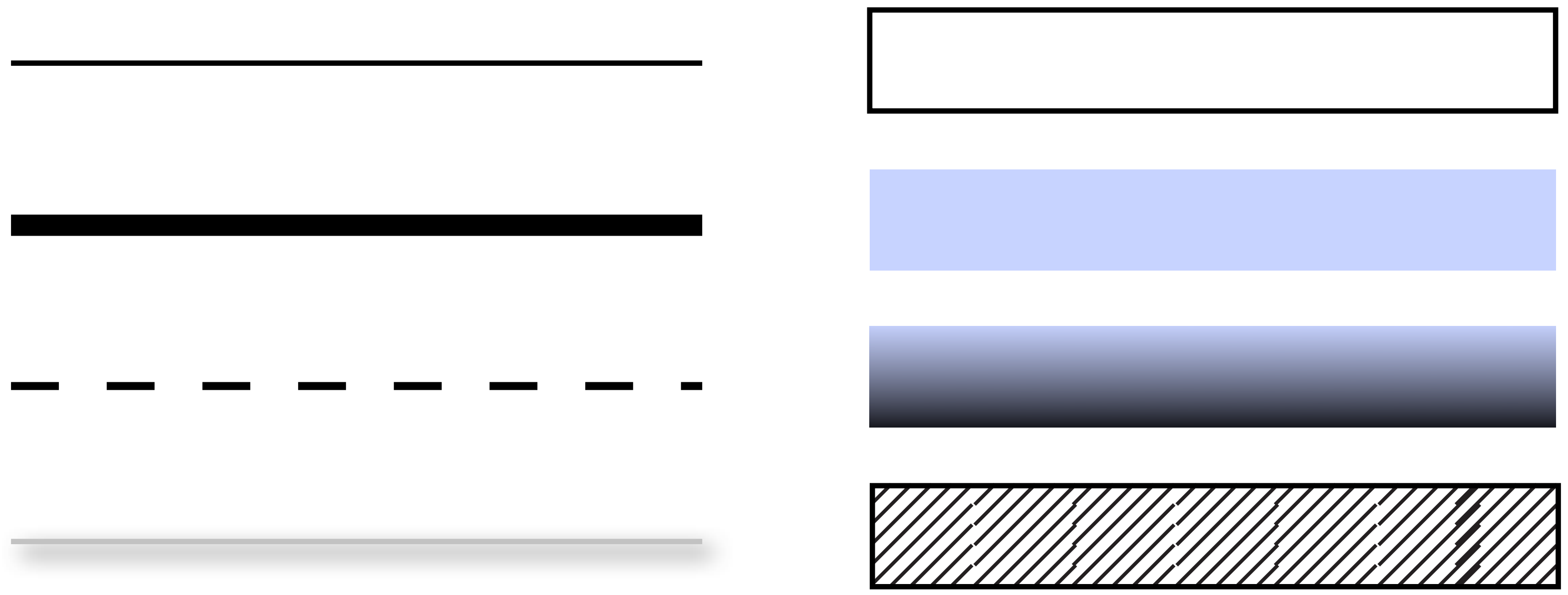
Lecture 5: Open GL Part 2: Attributes

Attributes

- Graphics primitives may be of the same type but displayed differently.

- An **attribute** is any property that determines how a geometric primitive is to be rendered. E.g.:
  - colour;
  - thickness of line;
  - pattern used to fill a polygon.

- Attributes can be **bound** with primitives at various points in the rendering pipeline.



Attributes for lines and polygons

Object Display

- OpenGL puts the emphasis on **immediate mode**.
- Primitives are not stored in the system but are passed through the graphics pipeline for possible display as soon as they are defined.
- When a primitive is defined, the present state (values) of the attributes are used. There is no memory of the primitive in the system.
- The primitive's image appears on the display. Once the image is erased, there is no memory of the primitive. To display it again it must be defined again.
- In order to allow objects to be re-displayed objects must be kept in memory via **display lists**.

Attributes

- Points have colour and size attributes
- Line segments have colour, thickness, type (solid, dashed, or dotted).
- Filled primitives have a variety of attributes:
  - fill with solid colour or a pattern;
  - display only the edges;
  - display edges in a different colour from the interior.
- Stroke text may have a variety of attributes:
  - direction of the text string;
  - path followed by successive characters;
  - height and width of characters;
  - font;
  - style (bold, italic, underlined).

Attributes (cont.)

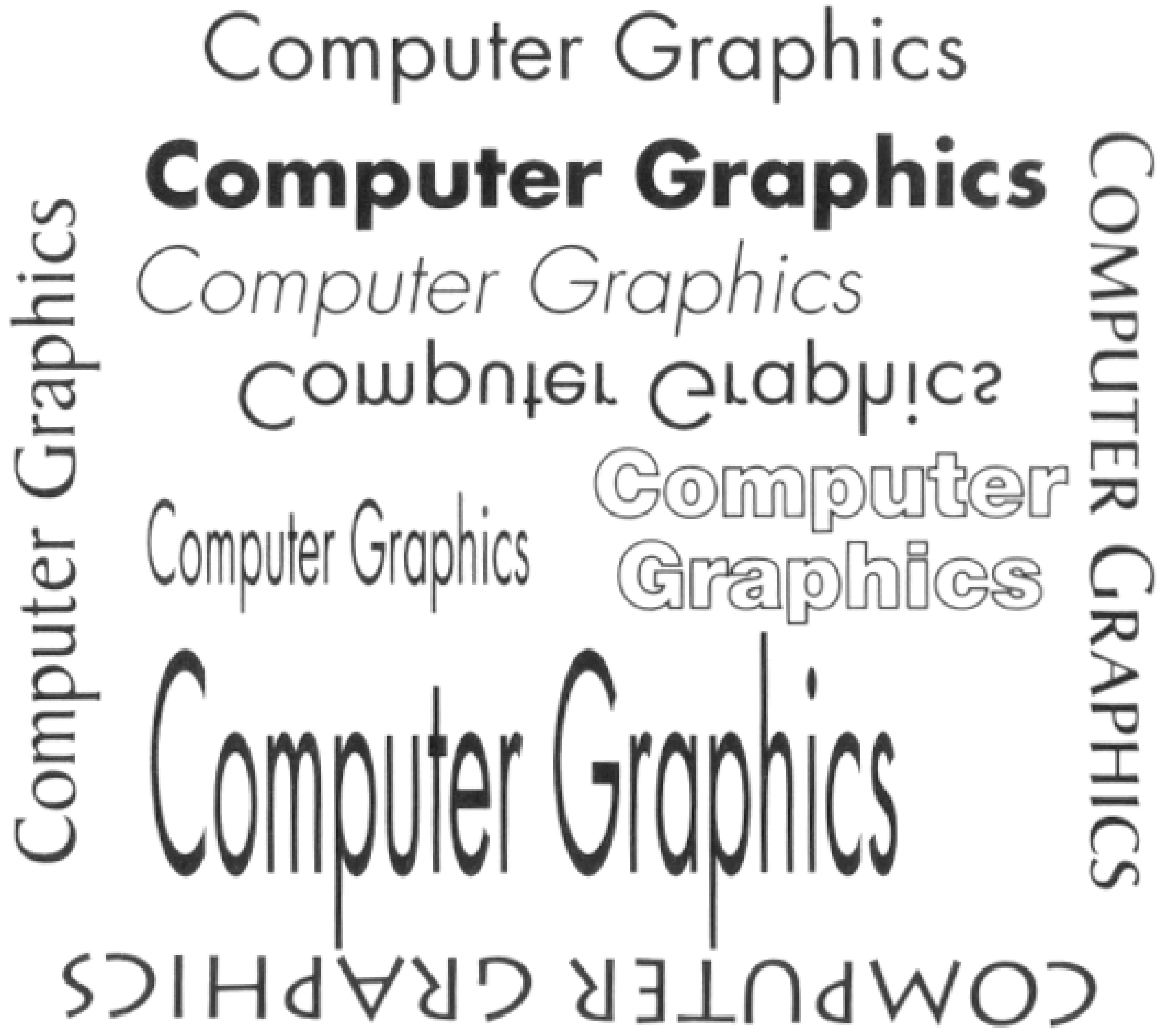


Figure 2.20 Stroke-text attributes.

Colour

- Computer graphics uses **three colour theory**.
- In additive colour, we think of forming a colour by adding basic colours, known as **primaries**, together. e.g. Red, Green and Blue.
- Since different systems can display different amounts of colour we want the programmer to be able to specify the colour in a hardware independent way and lets the graphics system map the requested colour to the closest available colour in the hardware.
- In OpenGL we can set the current colour via:

  ```
      glColor3f(1.0, 0.0, 0.0);
  ```

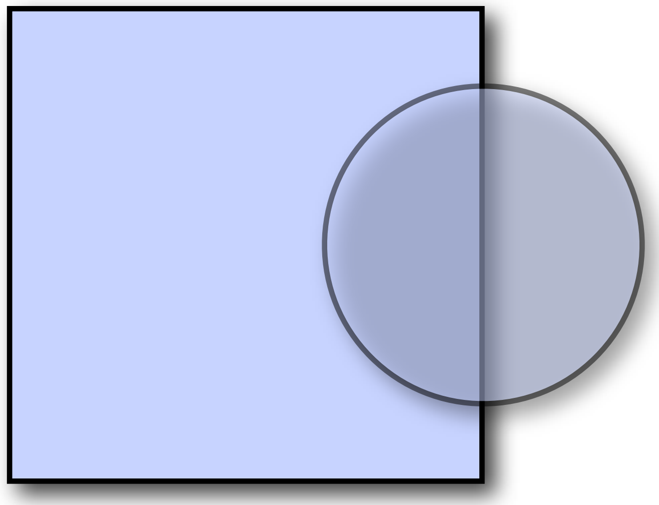
  where the three numbers represent the Red, Green and Blue intensities.
- 0.0 represents no colour component and 1.0 represents the maximum intensity for the colour component. Each value is **normalized.** The `3f` part follows the standard OpenGL conventions.

Colour (cont.)

- In **RGBA** colour, the **Alpha** value is treated by OpenGL as an **opacity** or **transparency** value. No light passes through an opaque object, whereas all light passes through a fully transparent object.
- By using RGBA we can create effects where the drawing window interacts with other windows that may be underneath.
- E.g.:

  ```
  glClearColour(1.0, 1.0, 1.0, 1.0)
  ```

  defines a four colour clearing colour that is white and opaque. This function call makes the window on screen solid and white.



Solid and partially transparent (0.5) shapes

Indexed Colour

- Some systems do not store 3 or 4 bytes of colour information per pixel. Instead they store a **colour number** or **index**. A one byte colour index can store 256 distinct colours. The hardware uses this number as an index to a colour lookup table (CLUT).

- In the past it was common to have 1 byte colour numbers and 256 entry CLUTs where each colour in the CLUT was a 24 bit RGB triple.

- OpenGL supports both `GLUT_RGB` and `GLUT_INDEX` colour modes. In colour index mode the colour gets selected from the CLUT via

  ```
  glIndexi( element )
  ```

  Entries in the CLUT are set via

  ```
  glutSetColour(GLint colour, GLfloat r,
       GLfloat g, GLfloat b)
  ```

Viewing

- OpenGL uses a synthetic camera model to view the world. The programmer specifies 3D objects which he places in a 3D world and the graphics program shows what this would look like to a synthetic camera placed at a particular position in the 3D world.
- OpenGL handles 2D graphics as a special case of 3D graphics. However, it is possible to begin with 2D viewing before progressing to 3D viewing.
- 2D viewing is based on taking a rectangular area of a 2D world and *mapping* its contents to an area of a 2D display.
- The area of the 2D world that appears in the image is known as the **viewing rectangle** or **clipping rectangle**. Objects outside the rectangle are not displayed.
- Objects which straddle the border of the viewing volume get clipped.

Viewing (cont.)

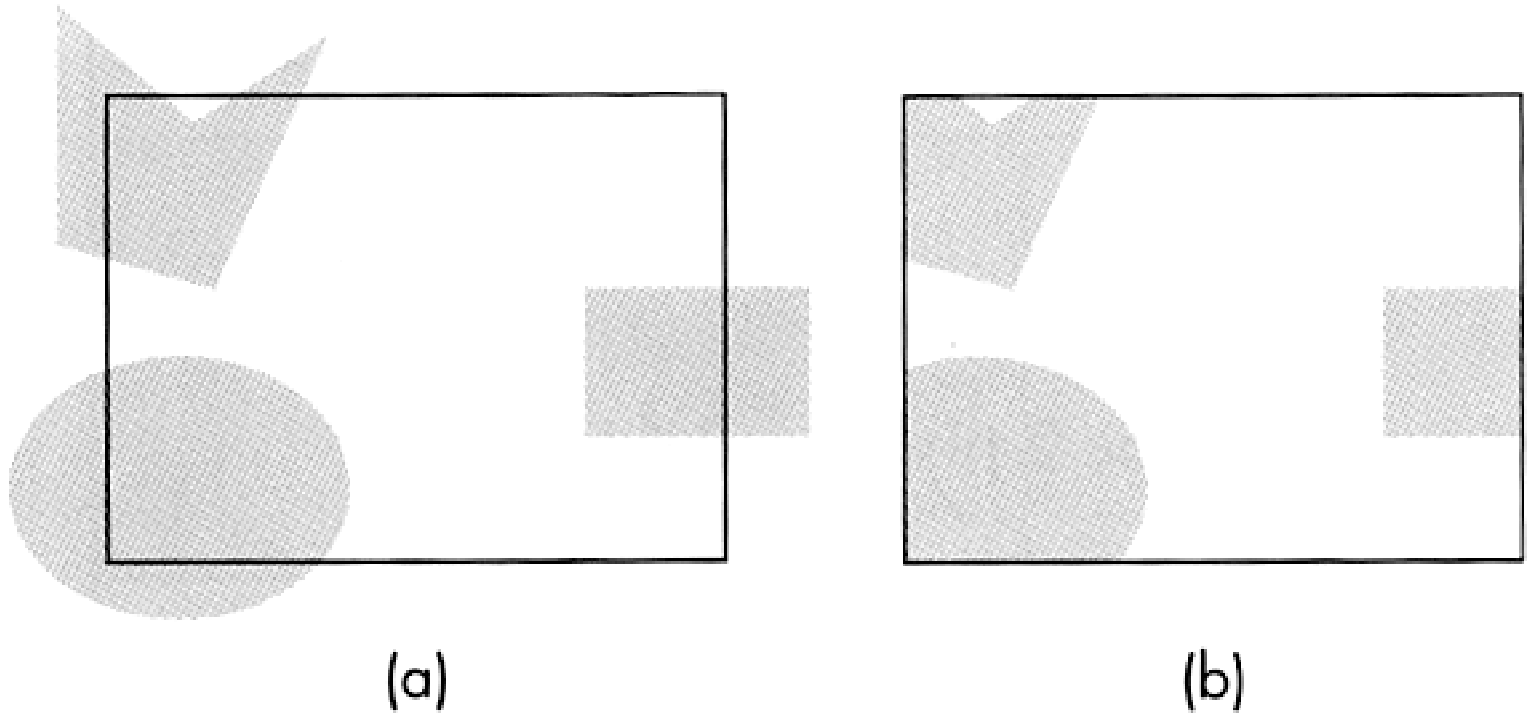


Figure 2.28 Two-dimensional viewing. (a) Objects before clipping. (b) Image after clipping.

Viewing (cont.)

- In OpenGL 2D viewing is a special case of 3D viewing. In 3D objects within a view volume are displayed on a 2D display surface.
- The viewing volume is a 2 x 2 x 2 cube with its centre at the origin of the coordinate system. To convert the 3D coordinates to 2D, an **orthographic** projection is carried out.
- The projection just basically sets the *z* coordinate to 0.
- In OpenGL this orthographic projection is specified via:

```
void glOrtho( GLdouble left, GLdouble right,
              GLdouble bottom, GLdouble top,
              GLdouble near, GLdouble far );
```

- If the *z* coordinate is less than `near` or greater than `far` the point will not be in the standard viewing volume and its image will not appear on the display.

Orthographic Camera

- The utility library gives a special function for defining a 2D clipping window.

```
void gluOrtho2( GLdouble left, GLdouble right,
    GLdouble bottom, GLdouble top )
```

This function is equivalent to `glOrtho` with `near` and `far` set to -1 and 1 respectively.

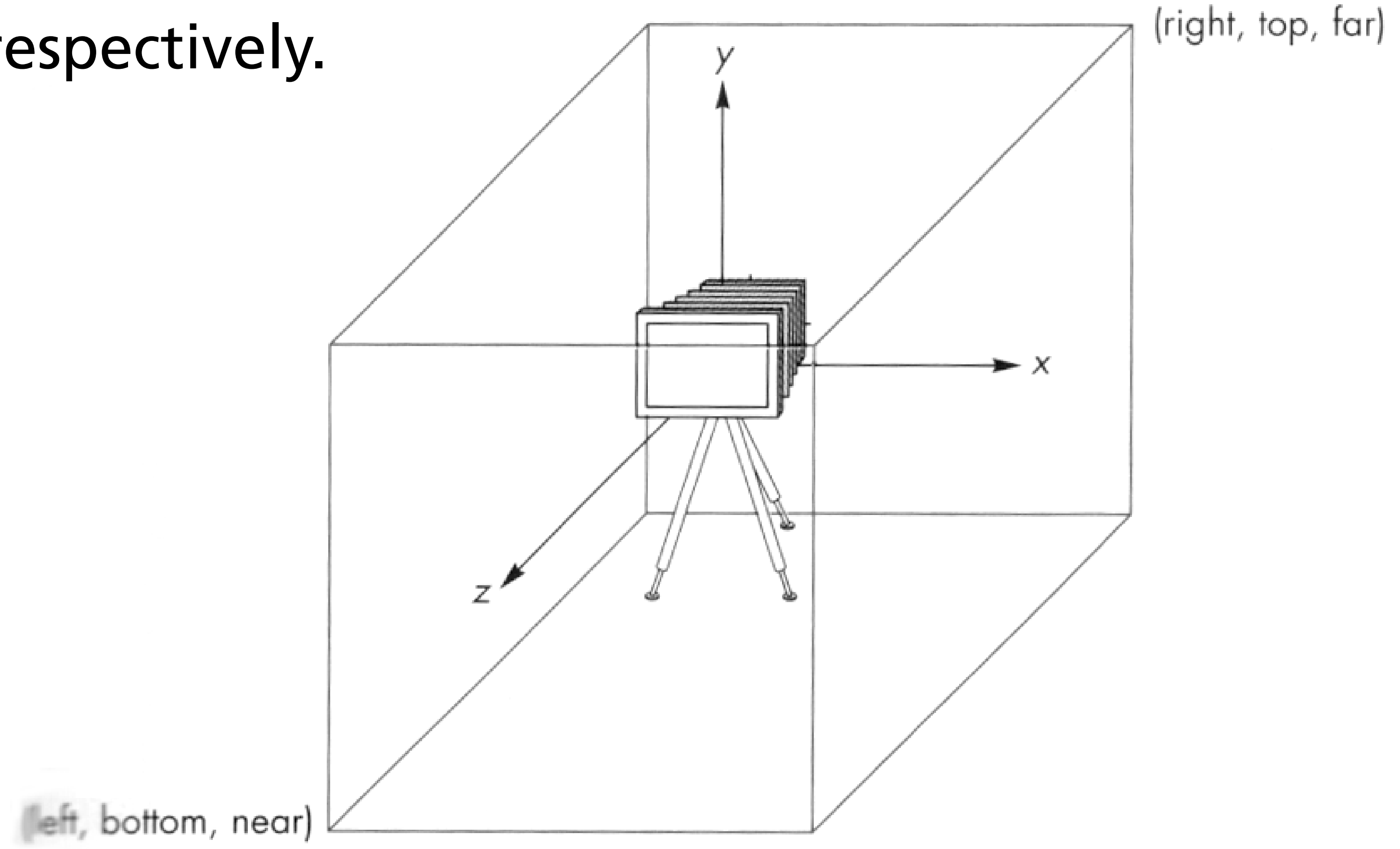


Figure 2.31 The default camera and an orthographic view volume.

Control Functions

- In OpenGL it is assumed that output will appear in a window which is being managed by a windowing system.
- GLUT is a library of functions that provides a simple interface between the graphics programming and windowing system.
- Applications programs that use GLUT should run under multiple windowing systems.
- OpenGL uses the term *window* (or *screen window*) to denote a rectangular area of a display. A window has a width and a height.
- Positions in the window are measured in **window** or **screen** coordinates, where the units are pixels.
- Screen coordinates are 2D;
  World coordinates are 3D.