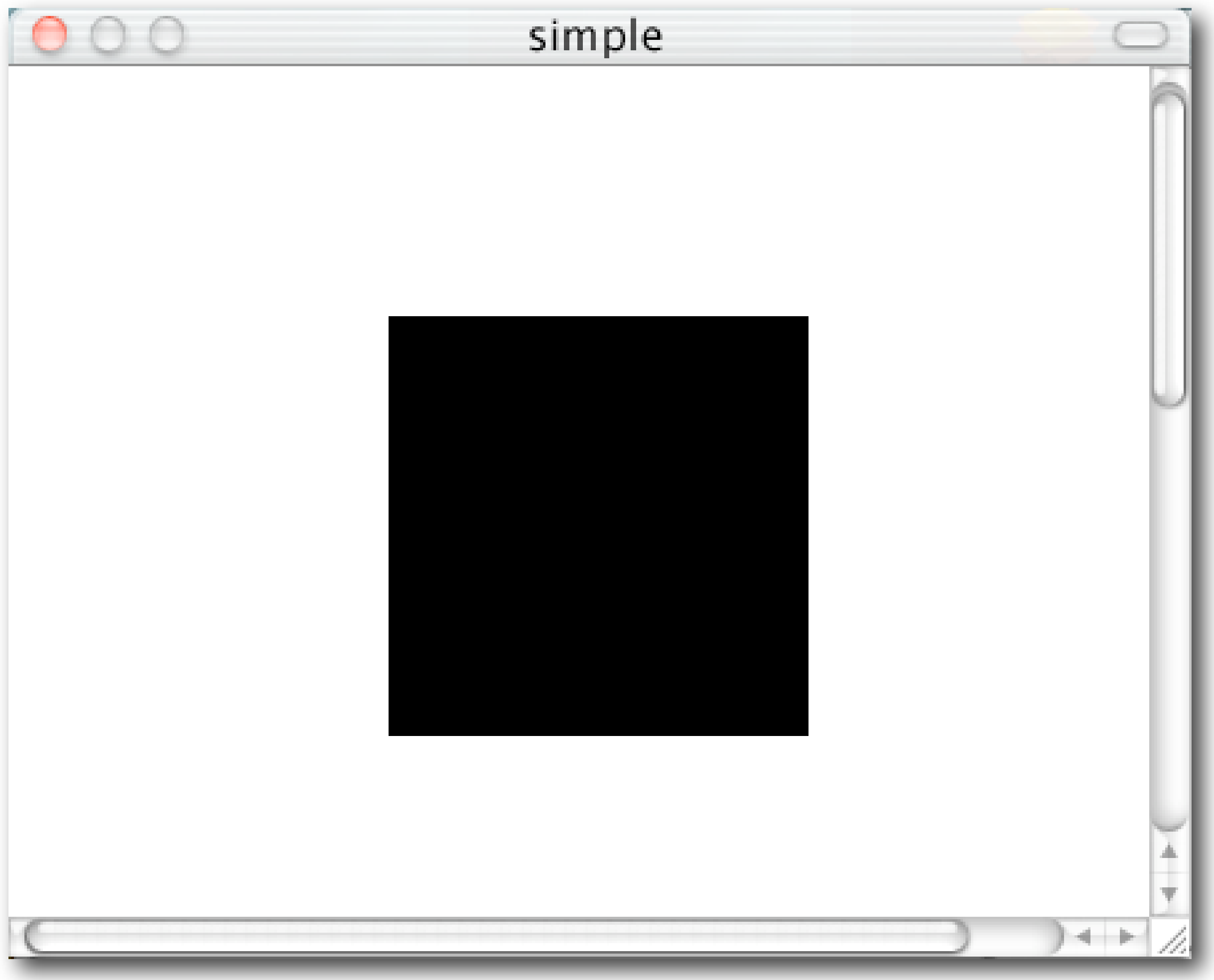

Monash University • Clayton's School of Information Technology

CSE3313 Computer Graphics

Lecture 6: OpenGL Part 3: Control of Windows and Viewports

Control Functions

- As well as the screen coordinates, **window coordinates** contain depth information of the object that maps to that pixel on the screen. This depth information is used in *hidden surface elimination.*

- References to the window that contains the output of the graphics program are relative to one corner of the window. Usually the lower left corner is the origin and has window coordinates (0,0).

- The window need not take up all the display space. For example, the screen may be 1280 x 1024 but the window might be 640 x 480.

- In GLUT information to the windowing system is first passed via

  ```
  glutInit( int *argcp, char ** argv )
  ```

  The two arguments are for command line arguments like those in `main`.

Windows

- An OpenGL window can then be opened via

  ```
  glutCreateWindow( char * title )
  ```

  Here title might be a string that appears at the top of the window.

- GLUT functions can be used to specify the window size, position on the screen and the way it used RGB colour.

  ```
  glutInitDisplayMode (
            GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE );
  glutInitWindowSize( 480, 640 );
  glutInitWindowPosition( 0, 0 );
  ```

  This specifies a 480 x 640 window in the top left corner of the display. In this example we specify RGB colour rather than CLUT colour, a depth buffer for hidden surface removal and double buffering for smoother animation.

- The **aspect ratio** is the ratio of the rectangle's height to width.

Aspect Ratio Mismatch

- If the aspect ratio of the viewing rectangle and display window are different, then objects may appear distorted on the screen.

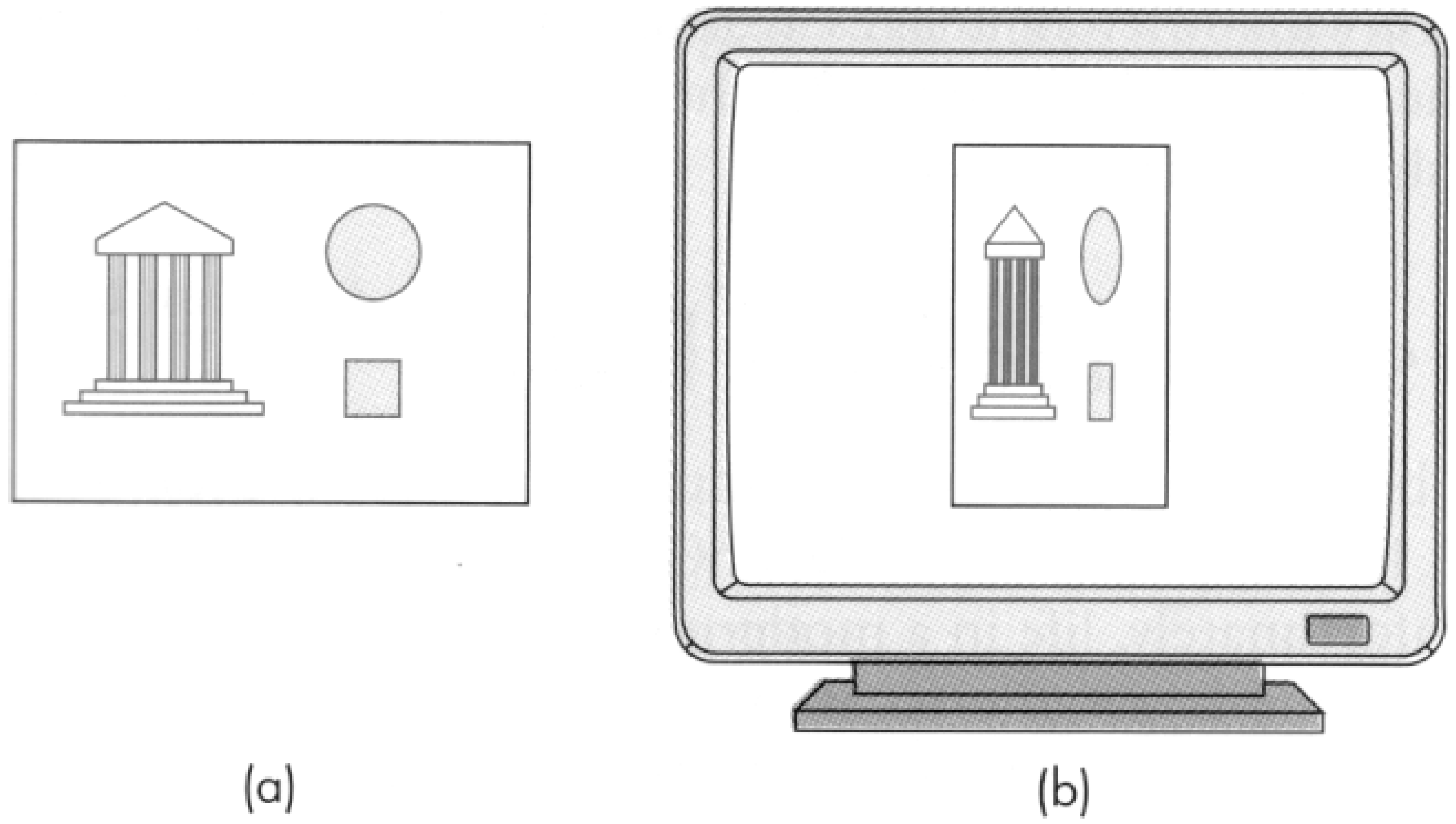

Figure 2.32 Aspect-ratio mismatch. (a) Viewing rectangle. (b) Display window.

Viewports

- A **viewport** is a rectangular area of the display window.
- By default it is the entire window.
- A viewport which does not take up the entire display window can be specified via the function:

  ```
      void glViewport( GLint x, GLint y, GLsizei w,
  GLsizei h )
  ```

  where `(x, y)` is the lower left corner of the viewport (measured relative to the lower left corner of the display window, in pixels). `w` and `h` give the width and height of the viewport.

Viewports (cont.)

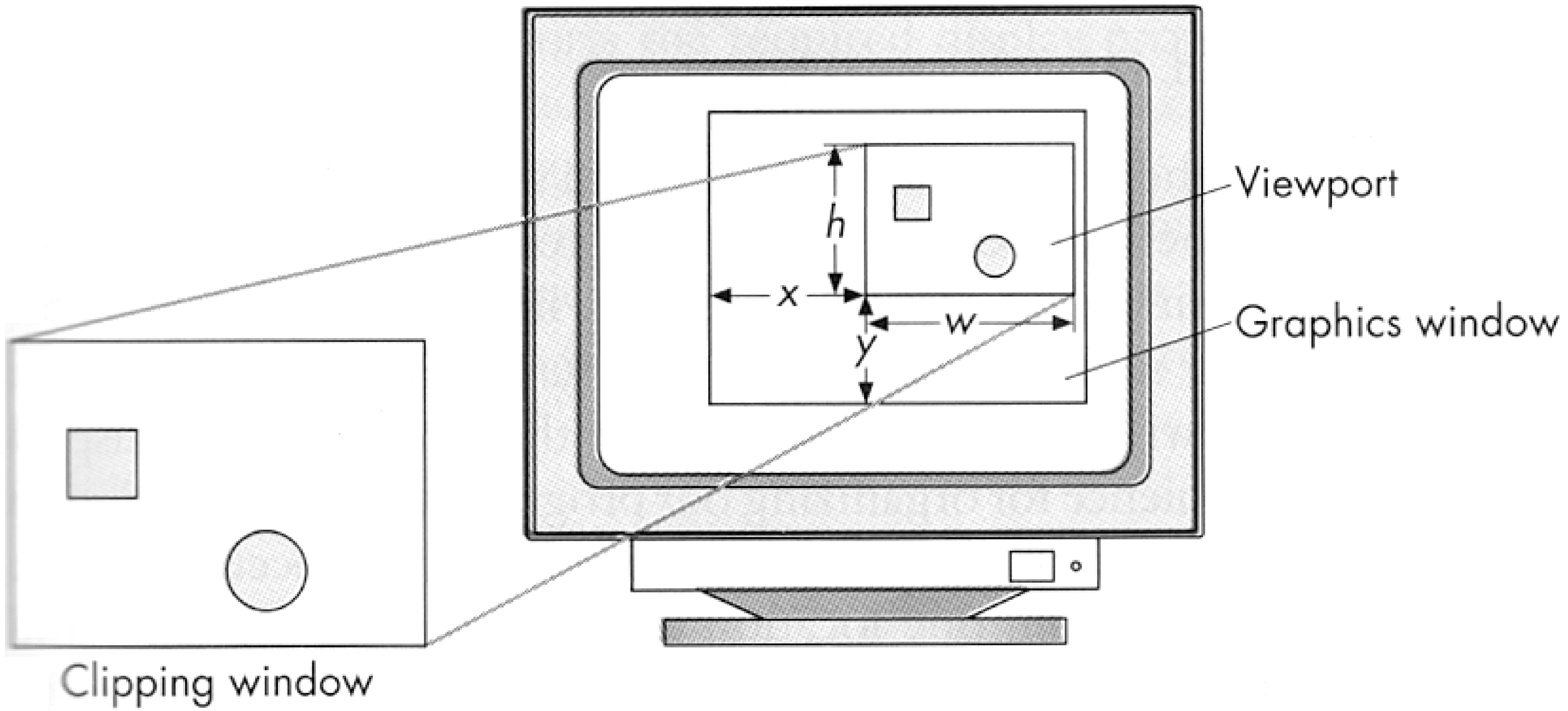

Figure 2.33 A mapping to the viewport.

The main, display and myInit functions

- In immediate mode graphics output is written as soon as possible. If the program just produces a picture as output, the program terminates and the display is then cleared. We might not even see the picture!
- In order to force the program to pause we could make the program go into a waiting loop.
- The program could have a window that is connected to `stdin`, `stdout` and `stderr`.
- We can force a pause by displaying a prompt in this window and waiting for the user to enter something from the keyboard before the program recommences.

Display callbacks

- With interactive graphics we can do a similar thing by putting the program into an event loop. If we specify no events the program will hang until the user closes the window manually. This can be done via

```
void glutMainLoop ( void )
```

- Graphics are sent to the screen via a function called the **display callback**.

```
void glutDisplayFunc( void (* func)(void) )
```

- `func` is a pointer to a function. It does not have any arguments and does not return any results. It gets called every time the OpenGL window needs to be redisplayed.

- This may be because the window is first opened, or because some window in front of the OpenGL window has been moved.

Display callbacks (cont.)

- Because the display callback function has no parameters, all information to this function has to be via global variables.
- myInit() is used to set the OpenGL state variables dealing with viewing and parameters. Here is an example:

```
void myInit( void ) {
    /* set clear colour to black */
    glClearColor( 0.0, 0.0, 0.0, 1.0 );
    /* set drawing/fill colour to white */
    glColour3f( 1.0, 1.0, 1.0);
    /*set up standard orthogonal view with clipping*/
    /* this is the default so it could be removed */
    gluOrtho2D( -1.0, 1.0, -1.0, 1.0 );
}
```

Simple main program

- The following is a main program that works with most non-interactive graphics applications.

```
#include <glut.h>

main(int argc, char * * argv) {
/* initialize mode and open a window in the upper */
/* left corner of the screen. Window title is */
/* the program name */
    glutInit( &argc,argv );
    glutInitDisplayMode( GLUT_SINGLE | GLUT_RGB );
    glutInitWindowSize( 500, 500 );
    glutInitWindowPosition( 0, 0 );
    glutCreateWindow( argv[0] );
    glutDisplayFunc( display );
    myInit();
    glutMainLoop();
}
```

Display callback function

- A sample display callback function could be:

```
void display ( void ) {
    /* clear the window */
    glClear( GL_COLOR_BUFFER_BIT );
    /* define a unit square */
    glBegin( GL_POLYGON );
      glVertex2f( -0.5, -0.5 );
      glVertex2f( -0.5, 0.5 );
      glVertex2f( 0.5, 0.5 );
      glVertex2f( 0.5, -0.5 );
    glEnd();
    /* flush GL buffers */
    glFlush();
}
```

- The call to `glFlush()` will cause the output to be displayed as soon as possible.