2D Graphics

Lecture 4
Section 2.7

Robb T. Koether
Hampden-Sydney College

Wed, Sep 2, 2009
Outline

1. The Viewport
2. The Pixels
3. Resizing the Window
   - Distorting the Scene
   - Fixing a Corner
   - Fixing the Center
4. Assignment
In two-dimensional graphics, objects are drawn as 3D objects, except:

- The \(z\)-coordinate is ignored (normally \(z = 0\)).
- Objects are projected orthographically onto the \(xy\)-plane.
- Depth testing is turned off.
- The last object drawn appears on top.
Since there is no perspective viewing, we need only keep track of the $x$ and $y$ world coordinates at the edges of the viewport.

We will call these $x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, \text{and } y_{\text{max}}$.

The function call will be

$$\text{gluOrtho2D}(x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}});$$
Example (The Reshape Function)

```cpp
void reshape(int w, int h)
{
    if (w > 0 && h > 0)
    {
        xmin = ... // Calculate new bounds
        xmax = ... // if necessary
        ymin = ...
        ymax = ...
        screenWidth = w;
        screenHeight = h;
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(xmin, xmax, ymin, ymax);
        glViewport(0, 0, w, h);
        glutPostRedisplay();
    }
    return;
}
```
The viewport is the part of the window in which the scene is displayed.

Normally, this is the entire window.

Typically, we initialize the window size to $640 \times 480$.

The function call

```c
glViewport(0, 0, w, h);
```

establishes the viewport on the screen.
Viewport Coordinates

- The coordinates are based on the “gridlines” between the pixels.
The coordinates are based on the “gridlines” between the pixels.

- (0, 0)
- (640, 0)
- (0, 480)
- (640, 480)

pixel
Pixel \((i, j)\) is located between gridlines \(x = i\) and \(x = i + 1\), and \(y = j\) and \(y = j + 1\).
Viewport Coordinates

- The corner pixels.

Pixel coordinates:
- pixel(0, 0)
- pixel(0, 479)
- pixel(639, 0)
- pixel(639, 479)
World Coordinates

- To make the two-dimensional world coordinates identical with the viewport coordinates, write

```plaintext
xmin = 0;
ymin = 0;
xmax = screenWidth;
ymax = screenHeight;
```
More typically, we place the origin in world coordinates at the center of the screen.

Then we assign new boundaries accordingly.

For example,

\[
\begin{align*}
\text{xmin} &= -4.0; \\
\text{xmax} &= 4.0; \\
\text{ymin} &= -3.0; \quad \text{// Aspect ratio 4/3} \\
\text{ymax} &= 3.0;
\end{align*}
\]
Resizing the Window

- When we resize the window, the two most common ways to redraw the scene are
  - Distort the scene proportionally.
  - Keep the scene a constant size and keep the upper-left corner fixed.
Distorting the Scene

- If we change the viewport without changing the world-coordinates boundaries, then the scene will be distorted.
- That is, if we do not recompute $x_{\text{min}}, x_{\text{max}}, y_{\text{min}},$ and $y_{\text{max}},$ the scene will be distorted.
Distorting the Scene

The scene will be stretched or contracted in both the $x$ and $y$ directions.
Distorting the Scene

Example (Distorting the Scene)

- The code.
- The executable.
We may want to keep the objects in the window the same size whether the window size is increased or decreased.

We simply show more or less of the scene.

The question is, what part of the scene should be added or deleted?
For simplicity, assume that the window border is pulled down and to the right.
Many applications keep the upper-left corner of the scene fixed.

Thus, $x_{\text{min}}$ and $y_{\text{max}}$ remain unchanged.

$x_{\text{max}}$ and $y_{\text{min}}$ must be recomputed.
Example (Constant Size, Fixed Corner)

- The code.
- The executable.
In the $x$-direction, the extent has been changed by the fraction $w/screenWidth$.

Therefore,

$$xmax = xmin + \left(\frac{w}{screenWidth}\right) \times (xmax - xmin);$$

Similarly,

$$ymin = ymax - \left(\frac{h}{screenHeight}\right) \times (ymax - ymin);$$
Sometimes we may want to expand the scene equally in all directions, keeping the center point in the center.
Constant Size, Fixed Center

The diagram shows a 2D graphics representation with the following variables:

- `screenWidth`
- `screenHeight`
- `(xmin, ymin)`
- `(xmax, ymax)`
- `w`
- `h`

The diagram illustrates the relationship between the viewport and the pixels, focusing on constant size and fixed center scenarios.
Constant Size, Fixed Center

Example (Constant Size, Fixed Center)

- The code.
- The executable.
The old center was at
\[(\frac{x_{\text{min}} + x_{\text{max}}}{2}, \frac{y_{\text{min}} + y_{\text{max}}}{2})\]

The new screen width is
\[w / \text{screenWidth} \times (x_{\text{max}} - x_{\text{min}})\]
Half of this amount should be added to and subtracted from the center to give the new $\texttt{xmax}$ and $\texttt{xmin}$. 

$$\texttt{xmin} = (\texttt{xmax} + \texttt{xmin})/2 - \left(\frac{\texttt{w}}{\texttt{screenWidth}}\right) \times (\texttt{xmax} - \texttt{xmin})/2;$$

$$\texttt{xmax} = (\texttt{xmax} + \texttt{xmin})/2 + \left(\frac{\texttt{w}}{\texttt{screenWidth}}\right) \times (\texttt{xmax} - \texttt{xmin})/2;$$

Similar calculations give $\texttt{ymin}$ and $\texttt{ymax}$. 
Homework

Read Section 2.7 – the `glViewport()` function.