Then a full refresh takes \((0.01 \times 1000) = 10\) seconds.

100 MHz PCI bus, so at least 10,000ns are needed per 32 bit transfer.

But this is very optimistic. Suppose the video card is on an (old) 1152 x 864 x 8bit x 24bit color buffer. How long will it take to fill the above frame buffer?

Number of refreshes/second = \(1/0.01\) refresh/second = 100 refreshes/second.

\[ 1152 \times 864 \times 24 \times 8 \times \frac{1}{100} = 3.377 MB/sec \] seconds.

What is the maximum number of refreshes/second approximated?

What is the maximum number of refreshes per second that the frame buffer can receive from the frame buffer?

**Suppose 10 nanoseconds per refresh (100 refreshes/sec).**

\[ 10/10 = 0.1 \text{Billionth sec}. \] average.

**1. How many bytes of memory is used by the frame buffer of a 1152x864 color display when the refresh rate is 80Hz?**

\[ 3,981,312 \times 312/1024 = 3.979 MB \]

**2. How many Megabytes per second are transmitted from the frame buffer to the display when the refresh rate is 80Hz?**

\[ 3.979 MB/\text{sec} \times 80 \text{Hz} = 310.3 \text{MB/sec} \]

**3. Suppose 10 nanoseconds per refresh (100 refreshes/sec).**

\[ 10/10 = 0.1 \text{Billionth sec}. \] average.

**What is the maximum number of refreshes per second that the frame buffer can receive from the frame buffer?**

**Calculation problems:**

\[ \text{CSI 422/502: Lecture 04 Computer Science Dept., University at Albany} \]
Printing devices? (1 whole frame per refresh).

How does hardware retrieve and translate and transmit frame buffer data to the display or printer?

Frame buffer? (SW+HW+Graph. accelerator HW).

How do graphics applications store data into the frame buffer?

SW+network?+Graph. accelerator HW.

(1) How do graphics applications store data into the frame buffer?

Display or printer memory

2-dimensional array

Computer, graphics card or printer memory

864 x 1152

Buffer

(or refresh)

(or color)

Scan Line

Pixel
satisfies $0 < (h', x) \mathcal{A}$ above the line; $(h', x) \mathcal{A} > 0$ classes are on a line.

The truth of $\mathcal{A}$, or equivalently, $h' = 0$, $M = h' \mathcal{A}$, determine

$\mathcal{A}$.

2. **Convert any graphic symbol into a corresponding odd integer.**

1. **Transform or map** something: $\mathcal{A} = 2n + 1$ transforms each integer

Mathematical functions, two uses:

- Raise the line: Compute which pixels to modify.
- In a modern system, transform endpoint data to graphics device.

2. Each endpoint.

1. **Translate world coordinates to device (or pixel coordinates) for

   $\text{flush}(\text{end})$;

   $\text{end}(\text{vertex21} (10, 145)$;

   $\text{vertex21} (180, 15)$;

   $\text{begin(cl_lines)}$;

   

   **Begin to answer (1) Given OpenGL line primitive, what happens?**
2. A screen viewport in screen coordinates

*** TO ***

1. A clipping window in world coordinates

We go directly from

now—matrices are in YOUR future!

We refer to HB 6-I, 6-2 and 6-3 but use formulas without matrices for

Introductory example of Transformations
will determine the clipping window in the world coordinate system.

\begin{figure}[h]
\centering
\includegraphics{figure1.png}
\caption{Example of a 3D scene rendered with OpenGL.}
\end{figure}

\begin{listing}[h]
\begin{verbatim}
\begin{verbatim}
\end{verbatim}
\end{verbatim}
\caption{OpenGL code snippet for rendering a cube.}
\end{listing}

In our first OpenGL example,
where within the world coordinate system, the two Ratios determine the location of a point inside the clipping window.

\[
\text{RATIO of x-offset of point in world coods: (\(x_w, y_w\)) from left window border to window width equals:}
\]

\[
\frac{x_w - x_w_{\text{min}}}{x_w_{\text{max}} - x_w_{\text{min}}}
\]
The analogous ratio for the $y$ position is

\[
\left(\frac{\text{max}_y - \text{min}_y}{\text{max}_y - \text{min}_y}\right)
\]
\[ 69 = x_{\text{max}} y \]
\[ 0 = y_{\text{min}} y \]
\[ 699 = x_{\text{max}} x \]
\[ 0 = x_{\text{min}} x \]

determines the screen viewport as \([400, 300] \)?

In our first OpenGL example,
same as the corresponding ratios for the point in the
screen viewport.
determining ratios for the point in the world clipping
window are the

Goal: Figure out the screen coordinates of the point so the position

Entire Screen

(point in screen coords: (x\text{\_viewport}, y\text{\_viewport})

Entire Screen

min _ x _ max _ x

min _ y _ max _ y

min _ x _ max _ x

min _ y _ max _ y
Screen coordinates, both measured in pixel units.

\[
\frac{y_{\text{max}} - y_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} = \frac{y_{\text{max}} - y}{x_{\text{max}} - x}
\]

Figure out \( x_v \) and \( y_v \) to make corresponding ratios be equal.

\[
\frac{y_{\text{max}} - y_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} = \frac{y_{\text{max}} - y_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}, \quad \frac{x_{\text{max}} - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} = \frac{x_{\text{max}} - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}
\]

\( y_{\text{max}} \), \( x_{\text{max}} \), \( x_{\text{min}} \), \( y_{\text{min}} \), \( y_{\text{max}}, x_{\text{min}}, x_{\text{max}}, y_{\text{min}} \), and \( y_{\text{max}} \) are known values, \( x_v \) and \( y_v \) to solve equations.

\( y_{\text{max}} \) is the unkown \( x_v \) and \( y_v \).

To solve the problem: With known

1. Transform the world coordinates \( (x', y') \) to screen coordinates \( (x_v, y_v) \) of this point by

2. Compute the screen coordinates \( (x_v, y_v) \) of this point by

Entire Screen

(x', y')

Point in screen coods:

(screen viewport)

(x', y')
\[
\frac{\text{un}_\text{max} x - x_{\text{un} \text{min}}}{x_{\text{un} \text{max}} - x_{\text{un} \text{min}}} = \frac{x_{\text{max}} x_{\text{un} \text{min}} - x_{\text{un} \text{max}} x}{x_{\text{max}} - x_{\text{un} \text{max}}}
\]

Write equation that says these ratios are equal, and solve for \( x \). (ditto for \( y \).)
We added \( x_{\min} \) to both sides.

\[
(\underbrace{u_{\min}x - x_{\text{max}}x}_{\text{both sides}}) \times \frac{u_{\min}x - x_{\text{max}}x}{u_{\min}x - mx} + u_{\min}x = \omega x
\]

We multiplied both sides by \((u_{\min}x - x_{\text{max}}x)\)

\[
(\underbrace{u_{\min}x - x_{\text{max}}x}_{\text{both sides}}) \times \frac{u_{\min}x - x_{\text{max}}x}{u_{\min}x - mx} = u_{\min}x - \omega x
\]

Solve the equation for \(x\):

\[
\frac{u_{\min}x - x_{\text{max}}x}{u_{\min}x - mx} = \frac{u_{\min}x - x_{\text{max}}x}{u_{\min}x - \omega x}
\]

Problem: Given equation with unknown variable \(x\),
without changing its size or orientation.

Technically, move here means translate, which means move something becomes \( \Delta \) pixels units right of the left viewport boundary. The position \( \Delta \) moves position \( \Delta \) to the right, so the position

\[
\text{viewport width} \times \frac{\text{viewport width}}{\text{screen}} = \Delta = \frac{\text{viewport width} \times \text{viewport width}}{\text{screen}}
\]

or the clipping window width: The result is the ratio which is between 0 and 1.

By scaling \( \Delta \) by the factor of screen \( \text{viewport width} \times \frac{\text{viewport width}}{\text{screen}} = \Delta = \frac{\text{viewport width} \times \text{viewport width}}{\text{screen}} \)

By moving \( \Delta x \) to the left, to get value \( \Delta \).

By moving \( \Delta y \) to the right, to get value \( \Delta \).

By moving \( \Delta z \) to the back, to get value \( \Delta \).

\[
(u_{min}x - x_{max}x) \times \frac{u_{min}x - x_{max}x}{u_{min}x - x} + u_{min}x = x
\]

Geometric significance of the solution formula:
in world coordinates.

Basic Line Drawing: Begin with a clipping window plus one line primitive.
Given a viewport in screen coordinates and the endpoints (x₀, y₀), (xₙ, yₙ) of a line, transform the line endpoints into screen coordinates using the view

(19, 8) (20, 10) (30, 18) (34, 19)

Screen Viewport

(xₙ, yₙ) (x₀, y₀) (xₙ, yₙ) (x₀, y₀)

(xₙ, yₙ) (x₀, y₀) (xₙ, yₙ) (x₀, y₀)
\[
0 = (\mathbf{z}) (0, 0) + 8 = (\mathbf{z}) (0, 0) + (0, -1) + 8 = 0 \nu \mathbf{h}
\]
\[
0 = 0 + 19 = (\mathbf{z}) (0, 3) + 19 = (\mathbf{z}) (0, 0) + (0, 3) + (0, -2) + 19 = 0 \nu \mathbf{x}
\]
\[
(\mathbf{z}) (0, 0) + (\nu \mathbf{x}) + 19 = 0 \nu \mathbf{x}
\]
\[
(\mathbf{z}) (0, 0) + (\nu \mathbf{x}) + 19 = 0 \nu \mathbf{x}
\]
\[
(\mathbf{z}) (0, 0) + (\nu \mathbf{x}) + 19 = 0 \nu \mathbf{x}
\]
\[
(\mathbf{z}) (0, 0) + (\nu \mathbf{x}) + 19 = 0 \nu \mathbf{x}
\]
\[
(\mathbf{z}) (0, 0) + (\nu \mathbf{x}) + 19 = 0 \nu \mathbf{x}
\]
\[
(\mathbf{z}) (0, 0) + (\nu \mathbf{x}) + 19 = 0 \nu \mathbf{x}
\]
\[
(\mathbf{z}) (0, 0) + (\nu \mathbf{x}) + 19 = 0 \nu \mathbf{x}
\]
\[
(\mathbf{z}) (0, 0) + (\nu \mathbf{x}) + 19 = 0 \nu \mathbf{x}
\]
\[
(\mathbf{z}) (0, 0) + (\nu \mathbf{x}) + 19 = 0 \nu \mathbf{x}
\]
Line Rasterization Problem: Given pixel coordinates of the endpoints (x₀, y₀), (xₙ, yₙ) of a line, which pixels should be colored in order to draw the line?
compute them.

This defines which pixels should be colored, but is a very inefficient way to

\[ 2. \text{ For } x = x_1, \ldots, x_n, \text{ compute } y_i = \text{round}(y_i^\text{true}) + B. \]

\[ 1. \text{ Determine } \mathcal{W} \text{ and } B \text{ in the line's equation } \mathcal{W}x + B. \]

Simplified solution (for the 1 case):

\[ |\mathcal{W}| \geq 1 \]
\[(h', x) \text{ above the line; line is between \((h', x)\) and only if } 0 < (h', x)A\]
\[(h', x) \text{ below the line; line is between \((h', x)\) and only if } 0 > (h', x)A\]
\[
(h' - (B + xW))(x \nabla z) = (h', x)A
\]
\[
\hat{h} - (B + xW) \text{ multiple of } \hat{W}
\]
\[
\text{Since is not an integer except when } W = 0 = W
\]
\[
\frac{\cdot 0x - ux}{\cdot \hat{h} - u\hat{h}} = \frac{x \nabla}{\hat{h} \nabla} = W \quad \text{Slope}
\]
\[
(0\hat{h} - u\hat{h}) = \hat{h} \nabla, (0x - ux) = x \nabla \quad \text{Differences}
\]
\[
\text{The number of pixels is also } + \frac{0x - ux}{h + 1}.
\]
\[
\text{Details given only for slope between } 0 \text{ and } 1. \quad \text{First make } x^0 \text{ and } 1.
\]
\[
\text{Pascal's algorithm:}\]
\[
\text{Bresenham's algorithm:}\]
\[
\text{Problem: \text{Rastorize the line from the pixel given these pixels.}}
\]
3. Store values $x^0$ and $y^0$ for future use.

2. Set pixel $(x^0, y^0)$.

Evaluate and store for future use:

$$\text{dual} = \text{dual} + (\frac{\gamma}{I} + 0y, 1 + 0x) \text{ or } (0y, 1 + 0x)$$

This helps because the sign of this number determines whether the dual pixel is $\text{dual} = \text{dual} + (\frac{\gamma}{I} + 0y, 1 + 0x) \text{ or } (0y, 1 + 0x)$.

Algorithm steps and their justification:
$p_0$ is called the initial decision parameter.

\[ x \nabla - \psi \nabla z = p_0 \]

which we compute using the simple but equivalent formula

\[
\left( \frac{\xi}{I} - \frac{x \nabla}{k \nabla} \right)(x \nabla z) = \\
\left( \frac{\xi}{I} - w \right)(x \nabla z) = \\
\left( \left( \frac{\xi}{I} + 0h \right) - w + (0h) \right)(x \nabla z) = \\
\left( \left( \frac{\xi}{I} + 0h \right) - w + (q + 0xw) \right)(x \nabla z) = \\
\left( \left( \frac{\xi}{I} + 0h \right) - q + (1 + 0x)w \right)(x \nabla z) = \\
\left( \frac{\xi}{I} + 0h, 1 + 0x \right) \mathcal{H}
\]
The next pixel is at most 1 pixel above the previous because the slope is

\[
\begin{aligned}
0 \geq \gamma d & \quad \text{if } I + \gamma h \\
0 > \gamma d & \quad \text{if } I
\end{aligned}
\]

i = I + \gamma h

justification of step 4: Remember, \( \gamma d \) is the decision parameter for the

5. Perform step 4. \( u - x \nabla = 1 - n \) in total.

\[
\begin{aligned}
\text{We add a negative value to positive or } 0 \gamma d \quad & \\
\text{otherwise, set } \gamma h = I + \gamma h \quad & \\
\text{We add positive } 2 \nabla & \\
\text{and plot pixel } (I + \gamma h, I + \gamma h) \quad & \\
\text{If } \gamma x < 0, \text{ then set } \gamma x = 1 + \gamma x \quad & \\
\text{At each pixel, } \gamma x \quad &
\end{aligned}
\]

\[
\begin{aligned}
\gamma h + 1 & \\
\gamma h + 1 & \\
\gamma h + 1 & \\
\gamma h + 1 &
\end{aligned}
\]
\[
\left(\frac{\partial}{\partial x} + \frac{\partial}{\partial y} + \partial_3\right) H = \partial_3 \frac{\partial}{\partial x}
\]

You can derive this last formula using algebra with

\[
0 \leq \partial_3 \Rightarrow \left\{ \begin{array}{c}
\partial_3^2 - \frac{\partial}{\partial x} \partial_3 \partial_3^2 + \partial_3 \partial_3^3 \\
\partial_3^2 - \frac{\partial}{\partial x} \partial_3 \partial_3^2 \\
\end{array} \right\} + \partial_3 = \left(\frac{\partial}{\partial x} + \frac{\partial}{\partial y} + \partial_3\right) H = 1 + \partial_3
\]

\(\partial_3 + 1\) affects the decision parameter calculation for the pixel \((\gamma + 2)\).
\( (\hat{y} \nabla z) - (\bar{y} \nabla z) \) is a total of \( \hat{y} \) coefficient plus the \( \hat{y} \)-coefficient; a total of \( \bar{y} \)-coefficient, when both \( x \) and \( y \) increase by \( I \), the value of \( \mathcal{H} \) increases by \( \hat{y} \) is when \( x \) increases by \( I \), the value of \( \mathcal{H} \) increases by \( \hat{y} \)-coefficient which is a linear function of \( x \) and \( y \) plus a constant.

\[
\mathcal{H} (x \nabla z) + \hat{y} (x \nabla z) - x (\bar{y} \nabla z) = (\hat{y} , x) \mathcal{H} \\
(\mathcal{H} + \hat{y} - x (\frac{x \nabla}{\bar{y} \nabla})) (x \nabla z) = (\hat{y} , x) \mathcal{H}
\]

\[
\frac{x \nabla}{\bar{y} \nabla} = \mathcal{W} \quad \text{and} \quad (\hat{y} - (\mathcal{H} + x \mathcal{W})) (x \nabla z) = (\hat{y} , x) \mathcal{H}
\]

Analyze the formulas:

\[
0 \leq \gamma d \iff x \nabla z - \bar{y} \nabla z \left\{ \begin{array}{c} x \nabla z - \bar{y} \nabla z \end{array} \right\} + \gamma d = \left( \frac{z}{I} + 1 + \gamma \bar{y} , 1 + 1 + \gamma x \right) \mathcal{H} = 1 + \gamma d
\]

Laziest (smart[1]) person's way to derive
Bresenham's rule

We did not have to figure out the constant coefficient \( P(x) \nabla \frac{d}{dx} \) to derive

\[
(x \nabla \frac{d}{dx})(y \nabla \frac{d}{dx}) \frac{d}{dx}
\]

By \( y \) and increasing \( y \) by \( 1 \) also. Hence we add \( y \) and increasing \( y \) from increasing \( x \).

In the other case, the \( \frac{d}{dx} \) value is for the point resulting from increasing \( x \).

Therefore, the case when \( \frac{d}{dx} > 0 \), the \( \frac{d}{dx} \) value is for a point obtained by
circle and other curve rasterization.

We will apply them to run Bresenham-like algorithms, so the main CPU doesn't need to. But the

Nowadays, hardware accelerators built into graphics cards

problem.

Such a demonstration may be a quiz problem, and will be a midterm

algorithm for slopes $m > 1$ and for slopes $-1 > m > 0$ (negative slopes).

Ex. 3-12, do another example, set up and demonstrate Bresenham's

HOMEWORK (to be assigned): Reproduce the calculations of HB