KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Computer graphics - Practice

Phạm Thanh Tùng - Võ Thế Hào

Week 02

Install basic 2D geometry drawing algorithm

1. Summary of theory

- **Line drawing algorithm**

The Mid-point algorithm for straight lines needs to handle the following 4 cases:

- $0 \le m \le 1$ <=> $0 \le |Dy| \le |Dx|$ and Dy, Dx have the same sign
- $m \ge 1$ <=> $0 \le |Dx| \le |Dy|$ and Dy, Dx have the same sign
- $-1 \le m \le 0$ <=> $0 \le |Dy| \le |Dx|$ and Dy, Dx have opposite signs
- $m \le -1$ <=> $0 \le |Dx| \le |Dy|$ and Dy, Dx have opposite signs

Refer to the pseudocode handling case t in Section 3

- **Ellipse drawing algorithm**

Suppose we need to draw an ellipse (E) with the following equation:

$$\left(\frac{x}{A}\right)^2 + \left(\frac{y}{B}\right)^2 = 1$$

$$\Leftrightarrow B^2x^2 + A^2y^2 = A^2B^2$$

$$\Leftrightarrow B^2x^2 + A^2y^2 - A^2B^2 = 0$$

Let $F(x, y) = B^2x^2 + A^2y^2 - A^2B^2$. We have the following comments:

Comment 1:

$$F(x,y)\begin{cases} < 0, & \text{if } (x,y) \text{ inside ellipse} \\ = 0, & \text{if } (x,y) \text{ above ellipse} \\ > 0, & \text{if } (x,y) \text{ outside ellipse} \end{cases}$$

Comment 2: to draw an ellipse, we only need to draw a quarter of the ellipse, then make it symmetrical about the major and minor axes. To simplify the problem, we choose to draw the ellipse arc in the first quadrant, that is, $x \geq 0$ and $y \geq 0$

Comment 3: In the first quadrant, we have:

$\frac{|dx|}{|dy|} \geq 1$ with $0 \leq x \leq x_0$ (equivalent to $B \geq y \geq y_0$)

$0 \leq \frac{|dx|}{|dy|} \leq 1$ with $0 \leq y \leq y_0$ (equivalent to $A \geq x \geq x_0$)

with $x_0 = \frac{A^2}{\sqrt{A^2+B^2}}$, $y_0 = \frac{B^2}{\sqrt{A^2+B^2}}$. We call the point (x_0, y_0) the change state point

Case 1:

$0 \leq x \leq x_0$ (equivalent to $B \geq y \geq y_0$) Thus, we can draw an ellipse arc in the range (0, B) -> ($x_0 = \frac{A^2}{\sqrt{A^2+B^2}}$, $y_0 = \frac{B^2}{\sqrt{A^2+B^2}}$).

At this time, x changes faster than y.

Suppose we have drawn the ith point with integer coordinates (x_i, y_i)

We need to choose the i+1 point as one of the following two integer points: $S(x_i+1, y_i)$ or $P(x_i+1, y_i-1)$. So,

$x_{i+1} = x_i + 1$ and $y_{i+1} \in \{y_i, y_i - 1\}$

We have:

MidPoint = $(x_i + 1, y_i - \frac{1}{2})$

At this point, the selection of points S and P above is taken to consider the sign of the expression:

$p_i = 4F(\text{MidPoint}) = 4F(x_i + 1, y_i - \frac{1}{2})$

If $p_i \leq 0$, the MidPoint is inside the ellipse. At this time, the real point Q is above the MidPoint point, so we choose S, which means $y_{i+1} = y_i$

If $p_i \geq 0$, MidPoint is outside the ellipse. At this time, the real point Q is below the MidPoint point, so we choose P, which means $y_{i+1} = y_i - 1$

On the other hand

$$p_{i+1} - p_i = 4F\left(x_{i+1}+1, y_{i+1} - \frac{1}{2}\right) - 4F\left(x_i+1, y_i - \frac{1}{2}\right).$$

Since $x_{i+1} = x_i + 1$, we have

$$p_{i+1} - p_i = 4\left(B^2(2x_i+3) + A^2(y_{i+1} - y_i)(y_{i+1} + y_i + 1)\right)$$

If $p_i < 0$, we will choose the integer point S, that is $y_{i+1} = y_i$. Then, we have it

$$p_{i+1} - p_i = 4B^2(2x_i+3)$$

If $p_i \geq 0$, we will choose the integer point S, that is, $y_{i+1} = y_i - 1$. At that time, we have

$$p_{i+1} - p_i = 4\left(B^2(2x_i+3) - A^2(2y_i - 2)\right)$$

We calculate the value p_0 corresponding to the initial point

$$p_0 = 4F(\text{MidPoint}) = 4F\left(0+1, B - \frac{1}{2}\right) = 4B^2 + A^2 - 4A^2B$$

Case 2: consider similarly to case 1

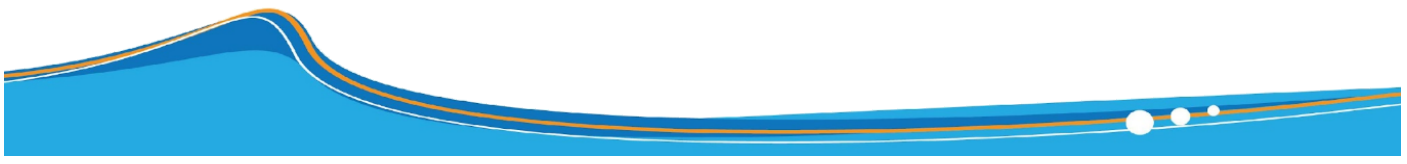
2. Case Studies

- Based on the basic program of Week 01 and the theory of 2D geometric drawing algorithms, complete the installation of manual drawing functions for these objects.
- Learn about simple input operations with mouse clicks and command prompts on the Web / WebGL / Canvas interface
- Expanded content for further reading
  - https://webglfundamentals.org/webgl/lessons/webgl-points-lines-triangles.html

3. Specific instructions

- **Pseudocode 1 line drawing algorithm 1 case**

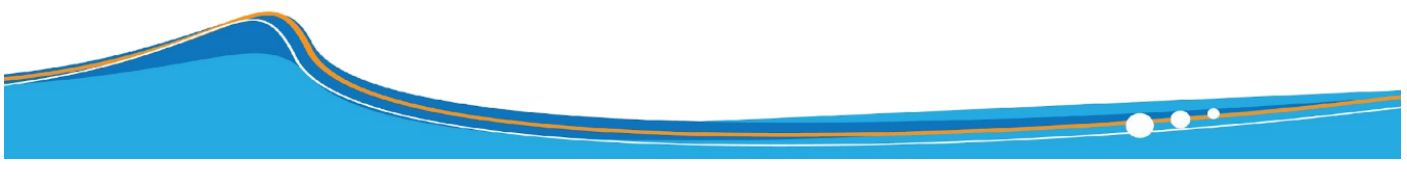
```
void LineMidPoint(int x1, int y1, int x2, int y2) //Case 1
{
      int Dx, Dy, p, Const1, Const2;
      int x,y;
      Dx = x2-x1;
      Dy = y2-y1;
      p = 2*Dy-Dx;
      Const1=2*Dy;
      Const2=2(Dy-Dx);
```



```
	x=x1;
	y=y1;

	putpixel(x, y, Color);
	for (i=x1; i<x2; i++)
	{
		if (p<0)
		p+=Const1;
		else. else
		{
			p+=Const2;
			y++;
		}
		x++;
		putpixel(x, y, Color);
	}
}
```

- **Pseudo code for an ellipse drawing algorithm**

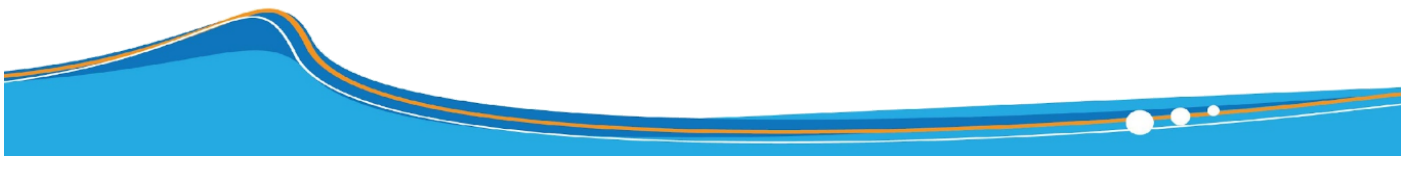
```
void DrawEllipse(long A, long B)
{
	long A2, B2, p, Const1, Const2, Delta1, Delta2, x, y, MaxX, MaxY;
	A2 = A*A;
	B2 = B*B;
	MaxX = floor(A2/sqrt(A2+B2));
	MaxY = floor(B2/sqrt(A2+B2));
	// (0, B) -> (MaxX, MaxY)
	p = long(B2-A2*B+A2/4);
	Const1 = 2*B2;
	Const2 = 2*A2;

	x = 0;
	y = B;
	Delta1 = B2*(2*x+3);
	// Delta1 = B2*3 since x=0
	Delta2 = 2*A2*(1-y)+B2*(2*x+3);
	// Delta2 = 2*A2*(1-B)+Delta1 since x=0, y=B
	Put4Pixel(x,y);
	while (x<MaxX)
	{
		if (p>=0)
		{
			p += Delta2;
			Delta2 += Const2;
			y--;
		}
		else. else
		p += Delta1;
		Delta2+=Const1;
		Delta1+=Const1;
		x++;

		Put4Pixel(x,y);
	}
```



```
      // (A, 0) -> (MaxX, MaxY)
      p = long(A2-A*B2+B2/4);
      Const1 = 2*A2;
      Const2 = 2*B2;
      x = A;
      y = 0;
      Delta1 = A2*(2*y+3);
      // Delta1 = A2*3 since y=0
      Delta2 = 2*B2*(1-x)+A2*(2*y+3);
      // Delta2 = 2*B2*(1-A)+A2*3 since x=A, y=0
      Put4Pixel(x,y);
      while (y<MaxY)
      {
            if (p>=0)
            {
                  p +=Delta2;
                  Delta2+=Const2;
                  x--;
            }
            else. else
            p+=Delta1;

            Delta2+=Const1;
            Delta1+=Const1;
            y++;
            Put4Pixel(x,y);
      }
}
```

- **Method to get mouse click coordinates when interacting with Canvas card**

```
<canvas width="200" height="200"
style="background-color: green">
</canvas>

<script type="text/javascript">
function getMousePosition(canvas, event) {
let rect = canvas.getBoundingClientRect();
let x = event.clientX - rect.left;
let y = event.clientY - rect.top;
console.log("Coordinate x: " + x,
"Coordinate y: " + y);
}

let canvasElem = document.querySelector("canvas");

canvasElem.addEventListener("mousedown", function(e)
{
getMousePosition(canvasElem, e);
});
</script>
```



- **Simple method to get value through window.prompt**

```
<button onclick="clickHandle()">Try it</button>

<script>
function clickHandle() {
let age = window.prompt("Please enter your age", "12");
if (age != null) {
window.alert("Your age is " + parseInt(age));
}
}
</script>
```

4. Exercise

Based on the sample program provided last week. Which sets the following drawing functions

Initializes a Canvas screen that allows users to interact with a predefined size of 800x600

Allows users to interact with the interface to perform drawing of basic objects

1. Draw straight lines
    a. Allows users to enter coordinates of 2 points with mouse click
2. Draw Ellipse
    a. Allows users to enter center coordinates with a mouse click
    b. Enter the long and short radius in prompt
3. Drawing Parabola (learn the algorithm yourself)
    a. Allows users to enter vertex coordinates with a mouse click
    b. Enter the remaining parameters using prompt
4. Drawing Hyperbola (learn the algorithm yourself)
    a. Allows users to enter vertex coordinates with a mouse click
    b. Enter the remaining parameters using prompt

5. Note

Homework submitted in the following format

- 01 text file containing the link of the program written directly on the website jsfiddle.net
- Required name: **DHMT_TH_02_<MSSV>.txt**
    - Write in correct capitalization
    - Do not add any other characters
    - Do not compress into zip, rar, tar files...