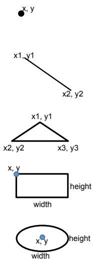
Obamicon

Review

- What is Computing?
- What can be Programmed?
- Creative Computing
- Processing
- Downloading Processing
- Dropbox
- Primitive Shapes
  - point
  - line
  - triangle
  - quad
  - rect
  - ellipse
- Processing Canvas
- Coordinate System
- Shape Formatting
  - Colors
  - Stroke
  - Fill

Drawing Primitives

- `point(x, y);`
- `line(x1, y1, x2, y2);`
- `triangle(x1, y1, x2, y2, x3, y3);`
- `rect(x, y, width, height);`
- `ellipse(x, y, width, height);`

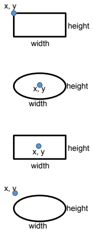


Modes

- `rect(x, y, width, height);`

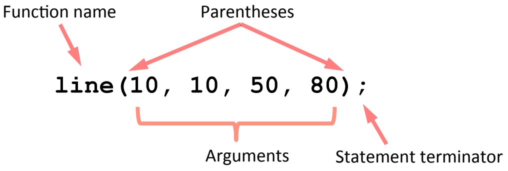
  `ellipse(x, y, width, height);`
- `rectMode(CENTER);`

  `ellipseMode(CORNER);`



Programming Principle

- Syntax is important!



```
line(10, 10, 50, 80);
```

Odds and Ends

- Processing programs carry the extension `.pde`
- must be in a folder with the same name
  - `myProgram.pde` must be inside a folder called `myProgram`
- Code block
  - The curly braces { }
- Comments
  - //
  - /* and */
- Naming convention

Basic Processing Program

```
void setup() {
  // Called once when program starts
}

void draw(){
  /* Called repeatedly
     while program runs */
}
```

The Event Loop

- Any code in **`draw()`** is executed 60 times per second
- Put code that you only want executed once in **`setup()`**
  - defaults
- **`noLoop()`**
- **`loop()`**

Mouse Interaction

- Built-in predefined variables that hold the mouse X and Y locations
  - current **`mouseX mouseY`**
  - previous (last) **`pmouseX pmouseY`**
  - 0 if mouse is not in window

More Graphics Primitives

`arc(…)`

`curve(…)`

`bezier(…)`

`shape(…)`

Arcs

`arc(x, y, width, height, start, stop);`

An arc is a section of an ellipse

`x, y, width, height`

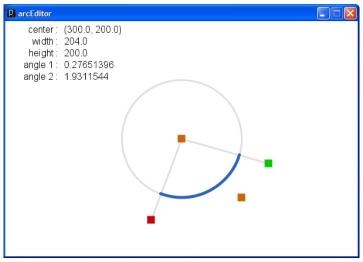
location and size of the ellipse

`start, stop`

arc bounding angles (in radians)

arcEditor

`arc(x, y, width, height, angle1, angle2);`



```
 center :  (300.0, 200.0)
  width :  204.0
 height :  200.0
angle 1 :  0.27651396
angle 2 :  1.9311544
```

Spline Curves

```
curve(x1, y1, x2, y2, x3, y3, x4, y4);
```

<u>spline:</u> A smooth curve drawn defined by four points

`x2, y2` *and* `x3, y3`

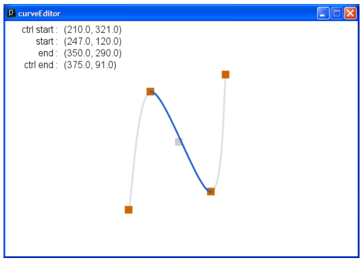
beginning/end points of visual part of curve

`x1, y1` *and* `x4, y4`

control points that define curve curvature

curveEditor

```
curve(x1, y1, x2, y2, x3, y3, x4, y4);
```



```
ctrl start :  (210.0, 321.0)
     start :  (247.0, 120.0)
       end :  (350.0, 290.0)
  ctrl end :  (375.0, 91.0)
```

Bézier Curves

```
bezier(x1, y1, cx1, cy1, cx2, cy2, x2, y2);
```

A smooth curve defined by two <u>anchor points</u> and two <u>control points</u>

`x1, y1` *and* `x2, y2`

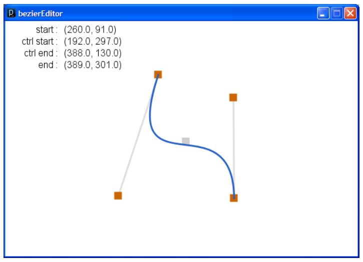
anchor points of bézier curve

`cx1, cy1` *and* `cx2, cy2`

control points that define curvature

bezierEditor

```
bezier(x1, y1, cx1, cy1, cx2, cy2, x2, y2);
```



```
     start :  (260.0, 91.0)
ctrl start :  (192.0, 297.0)
  ctrl end :  (388.0, 130.0)
       end :  (389.0, 301.0)
```

Custom Shapes

- Composed of a series of vertexes (points)
  - Vertexes may or may not be connected with lines
  - Lines may join at vertexes in a variety of manners
  - Lines may be straight, curves, or bézier splines
- Shape may be closed or open

Custom Shapes

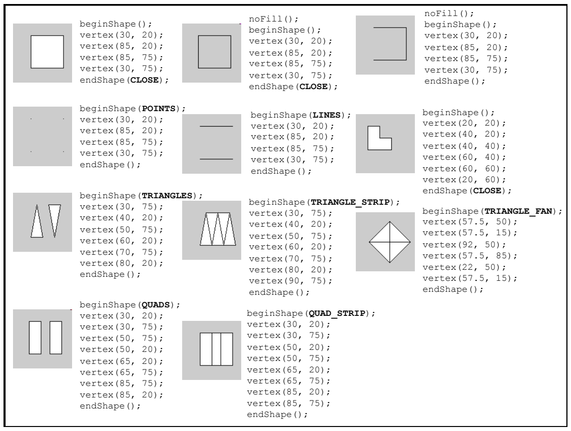
```
beginShape([option]);

     vertex(x, y);

     curveVertex(x, y);

     bezierVertex(cx1, cy1, cx2, cy2, x, y);

endShape([CLOSE]);
```



```
beginShape();
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape(CLOSE);
```

```
noFill();
beginShape();
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape(CLOSE);
```

```
noFill();
beginShape();
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape();
```

```
beginShape(POINTS);
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape();
```

```
beginShape(LINES);
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape();
```

```
beginShape();
vertex(20, 20);
vertex(40, 20);
vertex(40, 40);
vertex(60, 40);
vertex(60, 60);
vertex(20, 60);
endShape(CLOSE);
```

```
beginShape(TRIANGLES);
vertex(30, 75);
vertex(40, 20);
vertex(50, 75);
vertex(60, 20);
vertex(70, 75);
vertex(80, 20);
endShape();
```

```
beginShape(TRIANGLE_STRIP);
vertex(30, 75);
vertex(40, 20);
vertex(50, 75);
vertex(60, 20);
vertex(70, 75);
vertex(80, 20);
vertex(90, 75);
endShape();
```

```
beginShape(TRIANGLE_FAN);
vertex(57.5, 50);
vertex(57.5, 15);
vertex(92, 50);
vertex(57.5, 85);
vertex(22, 50);
vertex(57.5, 15);
endShape();
```

```
beginShape(QUADS);
vertex(30, 20);
vertex(30, 75);
vertex(50, 75);
vertex(50, 20);
vertex(65, 20);
vertex(65, 75);
vertex(85, 75);
vertex(85, 20);
endShape();
```

```
beginShape(QUAD_STRIP);
vertex(30, 20);
vertex(30, 75);
vertex(50, 20);
vertex(50, 75);
vertex(65, 20);
vertex(65, 75);
vertex(85, 20);
vertex(85, 75);
endShape();
```

```
void mousePressed() {
  // Called when the mouse is pressed
}

void mouseReleased() {
  // Called when the mouse is released
}

void mouseClicked() {
  // Called when the mouse is pressed and released
  // at the same mouse position
}

void mouseMoved() {
  // Called while the mouse is being moved
  // with the mouse button released
}

void mouseDragged() {
  // Called while the mouse is being moved
  // with the mouse button pressed
}
```

```
void keyPressed() {
  // Called each time a key is pressed
}

void keyReleased() {
  // Called each time a key is released
}

void keyTyped() {
  // Called when a key is pressed
  // Called repeatedly if the key is held down
}
```

keyCode vs. key

key

– A built-in variable that holds the character that was just typed at the keyboard

keyCode

– A built-in variable that hold the code for the keyboard key that was touched

All built-in keyboard interaction functions …

- Set *keyCode* to the integer that codes for the keyboard key
- Set *key* to the character typed
- All keyboard keys have a *keyCode* value
- Not all have a *key* value

ASCII - American Standard Code for Information Interchange

	0	1	2	3	4	5	6	7	8	9
30				!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~		€	
130	‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹
140	Œ		Ž			‘	’	“	”	•
150	–	—	˜	™	š	›	œ		ž	Ÿ
160		¡	¢	£	¤	¥	¦	§	¨	©
170	ª	«	¬	-	®	¯	°	±	²	³
180	´	µ	¶	·	¸	¹	º	»	¼	½
190	¾	¿	À	Á	Â	Ã	Ä	Å	Æ	Ç
200	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ
210	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û
220	Ü	Ý	Þ	ß	à	á	â	ã	ä	å
230	æ	ç	è	é	ê	ë	ì	í	î	ï
240	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù
250	ú	û	ü	ý	þ	ÿ				

Example Sketches...

– LadyBug

– Monster

– Ndebele

– Penguin

– SouthParkCharacter

– Sushi

– GiorgioMorandi