

Variables, Data Types, and a bit of Math



Slides courtesy of Dr. Abdallah Mohamed.

Announcements

Objectives

- You should be able to:
 - Define value, variable, and memory location
 - Create and use variables of different data types
 - recognize the naming rules and guidelines for variables.
 - List and compare the data types in processing.
 - Define and use the "color" type.
 - Creating and using constants.
 - Properly use math operators.
 - Evaluate math expressions.





Values, Variables, and Locations

- A value is a data item that is manipulated by the computer.
- A variable is the name that the programmer uses to refer to a location in memory.
- A location has an address in memory and stores a value.



Values, Variables, and Locations

Let's say we want to store a number that represents the age.
 Step #1: Declare variable by giving it a <u>NAME</u> and a <u>TYPE</u>.

int age; // age can only store integers
The computer allocates space for the variable in memory (at some memory address). Every time we give the name age, the computer knows what data item we mean.

Step #2: Initialize the variable to have a starting value. E.g., age = 21;

Step #3: Value stored in a location can be changed throughout the program to whatever we want using assignment ("=" symbol). age = age + 3;

16

20

24

Variable	Name Lookup	Table
Name	Location	Type
age	16	int



Variable Name

A variable must have a <u>NAME</u> and a <u>TYPE</u>.

- Names (aka *identifiers*):
 - are case sensitive (B is not the same as b)
 - can be a sequence of characters that include only *letters*, *digits*, *underscores* (__), and *dollar signs* (\$).
 - must start with a letter, an underscore (_), or a dollar sign (\$).
 - cannot start with a digit.
 - can *not* be a reserved word.
 - E.g. cannot be called double, true, false, or null.
 - Naming guidelines
 - can be of any length, but reasonable (readable) length is preferred.
 - should start with a lower case letter.
 - if more than one word, remove the spaces and capitalize all words after the first one (e.g. my first car → myFirstCar)

Variable Type

A variable must have a *name (identifier)* and a <u>type</u>. Each type has a valid range of values and uses a different amount of memory space.

	Туре	Size in memory	Range
	byte	8 bits	-2 ⁷ to 2 ⁷ -1 (-128 to 127)
whole numbers	short	2 bytes	-2 ¹⁵ to 2 ¹⁵ -1 (-32768 to 32767)
	int	4 bytes	-2 ³¹ to 2 ³¹ -1
	long	8 bytes	-2 ⁶³ to 2 ⁶³ -1
real	float	4 bytes	e.g. 17.345f
numbers	double	8 bytes	e.g. 12452.212 (more accurate)
characters	char	2 bytes	e.g. 'a', '1'and '?'
boolean	boolean	1 byte	true or false

Note: Unlike JavaScript, where you don't specify a type (i.e. just use var), in Java (and Processing) you must specify the variable type.

The String Type

Strings are sequences of characters inside double quotes (i.e. text in double quotes).

Example:

String personName = "Abdallah Mohamed";
personName = "John Smith";

 The first statement creates (defines) a variable and initializes its value to "Abdallah Mohamed".

- The second statement is assigns a new value to existing variable.
- The concatenation operator is used to combine two strings into a single string. The notation is a plus sign '+'.

String firstName = "Abdallah", lastName = "Mohamed";
String fullName = firstName + lastName;

The Color Type

Processing introduces a new data type called color which stores color information. The value of a color variable can be set by the color() function.

```
color red = color(255,0,0);
                                    // red in RGB mode
color navy = color(#443F76); // navy in hex notation (RGB)
colorMode(HSB, 360, 100, 100);
color green = color(128,100,100);// green in HSB mode
color blue = color(#0011FF); // blue in hex notation (RGB)
background (navy);
                        // navy background
                                              red uses RGB as it was defined
fill(red);
                                              before changing the color mode
rect(10, 10, 40, 40);
                        // red square
fill(green);
rect(50,50,40,40);
                        // green square
                                          blue uses RGB even though it was defined
                                          after setting the color mode because blue
fill(blue);
                                          was defined using hex notation
ellipse(75,25,30,30); // green square
```

Declaring and Initializing Variables

// Declaring Variables

double a; // Declare a to be a double variable
int x, y; // Declare x and y as integer variables

// Assignment Statements

а	=	7.1;	//	Assign	7	.1	to	a;
X	=	1 + 3;	//	assign	4	to	х;	
у	=	x + 2;	//	assign	6	to	у;	

// Declaring and Initializing in ONE Step
double a2 = 7.1;
int x2 = 1, y2 = 2;

Using Variables

Here are two more examples of two variables x and y

<pre>int x;</pre>	// declare a variable
x = 5;	<pre>// initialize a variable - what is assignment '='?</pre>
int y = 10;	<pre>// declare and initialize a variable</pre>
<pre>println(x);</pre>	// print value of x
x = 10;	// overwrite old value
<pre>println("x " + x);</pre>	//what is the output?

int x = 10, y;	//	У	has no values yet
y = x;	//	У	is 10 now
y = y + 1;	//	=	does not mean equal, it means assignment.
<pre>println("x + y = " +</pre>	(x	+	y)); //notice the output

Constants

- Constants are similar to variables except that once initialized they cannot change.
- To create a constant, use the keyword final before your variable declaration.

final double PI = 3.14159;
final int SIZE = 3;

Naming Convention:
 Capitalize all letters in constants
 e.g. MAX, PI, SIZE
 Use underscores for multiple words.
 e.g. MAX_VALUE





The Assignment Statement

An assignment statement changes the value of a variable.

- The variable on the left-hand side of the = is assigned the value from the right-hand side.
- The value may be changed to a constant, to the result of an expression, or to be the same as another variable.
- The values of any variables used in the expression are always their values before the start of the execution of the assignment.
- Example:

int A, B; A = 5; B = 10; A = 10 + 6 / 2; B = A; A = 2*B + A - 5;

Question: What are the values of A and B?

Expressions

An expression is a sequence of operands and operators that yield a result. An expression contains:
 operands - the data items being manipulated in the calculation

 e.g. 5, "Hello, World", myDouble
 operators - the operations performed on the operands

 e.g. +, -, /, *, % (modulus or remainder after integer division)

An operator can be:

unary - applies to only one operand

e.g. d = -3.5; // "-" is a unary operator, 3.5 is the operand
binary - applies to two operands

• e.g. d = 3 * 5.0; // "*" is binary operator, 3 and 5.0 are operands

Integer Division:

5 / 2 if both operands are integers, the output is an integer 2

■ 5.0 / 2 if at leas one operand is float, output is float 2.5

Division Operator

- What is the result of 25 / 4?
- How would you rewrite the expression if you wished the result to be a floating-point number?

Are the following statements correct? If so, show the output.

println("25 / 4 is " + 25 / 4);
println("25 / 4.0 is " + 25 / 4.0);
println("3 * 2 / 4 is " + 3 * 2 / 4);
println("3.0 * 2 / 4 is " + 3.0 * 2 / 4);

The Remainder Operator (%)

- The % operator returns the remainder of two numbers.
- Examples:

<u>Operation</u>

- a) 14 % 6
- b) -34 % 5

c) -34 % -5

d) 34 % -5

e) 5 % 1

f) 1 % 5

g) 3 % 0

<u>Result</u>

2

- 4

4

0

- 4 (matches numerator sign)

1 runtime error. Can't divide by zero

COSC 123 – 18

Operator Precedence

- Each operator has its own priority similar to their priority in regular math expressions:
 - Any expression in parentheses is evaluated first starting with the inner most nesting of parentheses.
 - 2. Unary + and unary have the next highest priorities.
 - **3.** Multiplication and division (*, /, %) are next.
 - 4. Addition and subtraction (+,-) are then evaluated.

The ++ and -- Operators

- It is very common to subtract 1 or add 1 from the current value of an integer variable.
- There are two operators which abbreviate these operations:
 - add one to the current integer variable
 - subtract one from the current integer variable
- Example:
 - var j=0;
 - j++; // j = 1; Equivalent to j = j + 1;
 - **j--;** // j = 0; Equivalent to j = j 1;

Augmented Assignment

The operators +, -, *, /, and % can be combined with the assignment operator = to form augmented operators.

X	+=	5;	<pre>//Equivalent</pre>	to	X	=	X	+	5
X	-=	5;	<pre>//Equivalent</pre>	to	X	=	X	-	5
X	*=	5;	<pre>//Equivalent</pre>	to	X	=	X	*	5
X	/=	5;	<pre>//Equivalent</pre>	to	X	=	X	/	5
X	%=	5;	<pre>//Equivalent</pre>	to	X	=	X	%	5

Summary

- The pre-class materials covered the following:
 - Variables and Data types
 - Primitive types: byte, short, int, long, float, double,
 - char, boolean
 - New types: color
 - String type
 - Naming rules and guidelines (for variable and constants)
 - Math operators and expressions.
 - Binary operators: +, -, *, /, %
 - Unary operators: -3, x++, y--
 - Augmented assignment: -=, +=, /=, *=, %=

Variable Scope

Variable Scope

- The scope of a variable is the part of the program where you can access or use the variable.
- Local variables are those defined in functions and can be only accessed inside this function.
- Global variables are those defined outside functions all functions can access global variables.

Variable Scope

This program increases the rectangle size every time a key is pressed.

```
int size = 10; -
void setup(){
  size(200,200);
void draw(){
  int x = 100, y = 100; •
  rectMode(CENTER);
  rect(x,y,size,size);
}
void keyPressed(){
  size += 5;
```

Global variable: Declared outside all functions. ALL functions can access size

Local variables:
Declared within a function.
Only this function can access x and y

size increases when a key is pressed, then draw uses the new value of size.

Question

Variable Scope

Why does this code have a compile error?

- A. We cannot have a variable called size
- B. We need to specify a type for the variable y (i.e. int y = 100)
- C. draw() and keyPressed() cannot access x or y.
- D. The variable size must be defined as global variable.
 E. Something else.

```
void setup(){
  int x = 10, y = 10;
  size(200,200);
}
void draw(){
  int size = 50;
  rect(x, y, size, size);
}
void keyPressed(){
     × += 20;
     v += 20;
}
```

Controlling Animations with Variables

Controlling Animations with Variables (1)

- Variables can be used to control many aspects of the your animation.
- The key idea is to store some *attributes* of your sketch in global variables and update them:
 - (a) every frame (e.g. x++ in the draw() method), and/or
 - **(b)** whenever an event happens (e.g. x=0 whenever a key is pressed)

Controlling Animations with Variables (2)

- Example attributes include
 - Position
 - Use variables, e.g. x and y, to store the position.
 - Angle
 - Use a variable to store an angle. Use that variable to transform the shape coordinates using the rotate() function.
 - Scale
 - IDEA1: use a variable to store scale, and use it to transform coordinates
 - IDEA2: use variables to store scale and use it as a multiplicand to control the size of the shape/item.
 - Color
 - Use variables to store color components.
 - ...etc

(a) updating attributes every frame

Moving Square

- Here, we use a global variable x to control the x-position of the square. The variable is incremented in every frame.
- Question:
 - What happens if we declare and initialize x in
 - A) setup() ?
 - B) draw() ?

```
int x=0;
void setup(){
  size(200,200);
}
void draw(){
  background(100);
  rect(x, 75, 50, 50);
  X++;
}
```

Revolving Wheel

- Previously, you created the design below.
- We can animate the angle of the rotation using a variable dr that is updated in every frame.



You try it now!
 No need to submit anything yet to Canvas

float dr = 0; void setup(){ size(300,300); strokeWeight(2);

void draw(){

background(0); translate(150,150); // move origin to center // outer rings

fill(0,0,150);stroke(0,0,255);ellipse(0,0,180,180); fill(0); stroke(0,0,255); ellipse(0,0,160,160); // green rectangles and red ellipses noFill(); rotate(dr); stroke(0,255,0); rect(0,0,80,40); stroke(255,0,0); ellipse(0,0,80,30); rotate(PI/2); stroke(0,255,0); rect(0,0,80,40); stroke(255,0,0); ellipse(0,0,80,30); rotate(PI/2); stroke(0,255,0); rect(0,0,80,40); stroke(255,0,0); ellipse(0,0,80,30); rotate(PI/2); stroke(0,255,0); rect(0,0,80,40); stroke(255,0,0); ellipse(0,0,80,30); dr += 0.02;

Moving Objects at Given Speed

- A good idea to show moving objects is to use two variables x and y for the location, and add to them a small displacement, speedX and speedY, every frame.
- In the example, the object moves to the right only.
 - **Q1: WHY?**
 - Q2: make the object move upwards only.
 - Q3: make the object move diagonally.
- We will see later how to change the speed in the runtime.

```
float x, y, diam = 16;
float speedX = 1, speedY = 0;
```

```
void setup(){
    size(200,200);
    x = 0;
    y = height/2;
}
```

```
void draw(){
    background(0);
    ellipse(x, y, diam, diam);
    x = x + speedX;
    y = y + speedY;
}
```

Animating Size and Opacity

- Here, we control the circle using four variables.
 - radius and opacity change at the end of each frame, causing the next frame to appear differently.
 - location (x,y) is not changed.



 Question: modify the code so that the circle moves from the top-left corner to the bottom-right corner.



```
float radius = 10, opacity = 255;
float x = 75, y = 75;
```

```
void setup() {
   size(150, 150);
```

```
noStroke();
```

```
}
```

}

```
void draw() {
   background(0);
   fill(255, opacity);
   ellipse(x, y, radius, radius);
   radius++;
   opacity--;
```

(b) updating attributes with events

Controlling Speed with Mouse & Key Events

- This examples initially draws a rectangle in the middle of the sketch
- Clicking the mouse button will cause it to start moving left.
- Pressing a key on the keyboard will cause it to start moving right.

```
int x = 75, speedX = 0;
void setup(){
  size(400,200);
}
void draw(){
  background(100);
  rect(x, 75, 50, 50);
  x += speedX;
}
void keyPressed(){
  speedX = 2;
}
void mousePressed(){
  speedX = -2;
}
```

Controlling Colors with Mouse & Key Events

- This examples initially draws a black circle on a white background.
- Clicking the mouse button brightens a circle.
- Pressing a key on the keyboard dims the background.



```
int background = 255, foreground = 0;
```

```
void setup() {
    size(100, 100);
}
```

```
void draw() {
   background(background);
   fill(foreground);
   ellipse(50,50,80,80);
```

```
}
```

}

```
void keyPressed() {
   background -= 10;
}
```

```
void mousePressed() {
```

Panning the Sketch with the Mouse

- When mouse is *dragged*, x and y are updated with the relative mouse displacement.
 - i.e. the difference between current mouse location and the previous mouse location.



```
int x=0, y=0;
void setup(){
  size(200,200);
}
void draw(){
  background(100);
  translate(x,y);
  rect(0, 0, 50, 50);
  ellipse(100,100,30,30);
  line(10,100,100,50);
void mouseDragged(){
  x += mouseX - pmouseX;
  y += mouseY - pmouseY;
```

Example 8 (Aside)

also Zooming...!

- This is the same code as before except that another variable is used to zoom in and out (scale the sketch).
- This variable is updated based on the mouse wheel rotation.
 - The mouseWheel() function:
 - This function is automatically called whenever the mouse wheel rotates.
 - The e.getCount() returns 1 or -1 every time the mouse wheel is rotated up or down.

```
int x=0, y=0;
float scl = 1.0;
void setup(){
  size(200,200);
void draw(){
  background(100);
  translate(x,y);
  scale(scl);
  rect(0,0,50,50);
  ellipse(100,100,30,30);
  line(10,100,100,50);
}
void mouseDragged(){
  x += mouseX - pmouseX;
  y += mouseY - pmouseY;
void mouseWheel(MouseEvent e){
  scl -= e.getCount()/10.0;
```

Moving/Dragging Items

- In this example, we have 4 variables to store the location of the mouse.
- "moving" text is using the first two which are set whenever the mouse is moved
- "dragging" text is using the dragX and dragY which are set whenever the mouse is dragged.



```
int moveX = 50, moveY = 50;
int dragX = 50, dragY = 50;
void setup() {
    size(100, 100);
    textAlign(CENTER);
```

}

}

```
void draw() {
```

```
background(0);
fill(255,125,0);
text("moving", moveX, moveY);
fill(0,255,0);
text("dragging", dragX, dragY);
```

```
void mouseMoved() { // Move the text "Moving"
  moveX = mouseX;
  moveY = mouseY;
```

```
}
void mouseDragged(){ // Move the text "Dragging"
    dragX = mouseX;
    dragY = mouseY;
```

End of Tuesday's Class



Quick Tutorial on Animations



Slides courtesy of Dr. Abdallah Mohamed.

Key Idea 1: How to Animate "Things"

- A question is: *how to animate "Things" in your sketch?* "Things" are basically the attributes of different items such as the color, location, transformation, size, etc.
- Here are the process you need to follow:
 (1) Identify which attributes you want to animate (e.g. size, color, location, etc.)
 - you may want to use the PDE's Tweak tool to help you identify the right attribute.
 - (2) for each attribute you want to animate, create and initialize a global variable.
 - (3) In draw(), use the global variables to represent the attributes.
 - (4) Change the value of your global variables either:
 - in draw()
 - For continuous animation (e.g. falling rain drops).
 - in an event function (e.g. keyPressed())
 - For interactive animations (e.g. controlling character position with keyboard)

Animation Tutorial 1

Let's say you have a white circle on a dark background.



- And let's say we want two things to be animated:
 - The background to get lighter
 - The ball to move from left to right

Animation Tutorial 1, cont'd

Step (1) Identify the attributes that we should change, i.e. the background color and the x-location of the ball.

void setup() { size(600, 200); void draw() { background(80); ellipse(25,100,50,50);



Step (2) create two global variables, one for each attributes.



Step (3) use the variables for the attributes values



Step (4) Update your variables

int c = 80, x = 25;void setup() { size(600, 200); void draw() { background(c); ellipse(x,100,50,50); x = x + 2;c = c + 1;

Key Idea 2: Controlling Speed

Let's have a look at the following code below. Obviously, the x's increment value controls the speed of the ball.
 The speed is just another attribute in your sketch.



Key Idea 2: Controlling Object's Speed

Now that we have the speed attribute stored in a variable, you can control it by updating its value either in draw() or other event functions.

```
float x = 25, speedX = 0;
void setup() {
   size(800, 200);
}
void draw() {
   background(80);
   ellipse(x,100,50,50);
   //update variables
   x = x + speedX;
   speedX += .4;
}
```

Ball gradually increase its speed

```
float x = 25, speedX = 0;
void setup() {
   size(800, 200);
}
void draw() {
   background(80);
   ellipse(x,100,50,50);
   //update variables
   x = x + speedX;
   speedX = 0.01 * mouseX;
}
```

speedX is controlled by mouseX

```
float x = 25, speedX = 0;
void setup() {
    size(800, 200);
}
void draw() {
    background(80);
    ellipse(x,100,50,50);
    //update variables
    x = x + speedX;
}
void mousePressed() {
    speedX = 3;
}
void mouseReleased() {
    speedX = 0;
}
```

Ball only moves when mouse is clicked

Exercise

Move Your Wheel!

- Create an animation where the speed and rotation of the wheel are controlled by mouseX.
 - When mouseX is 0, the wheel freezes.
 - The higher value mouseX is, the faster the wheel moves to the right while rotating.
- When any key is pressed, the wheel's location is reset (far left) and it stops rotation.
- Idea: Reuse the code of the wheel presented in last lecture's notes:
 - declare two variables: x and dr and initialize both to 0.
 - Use the two variables to *transform* your wheel.
 - Both variables should be updated every frame using mouseX.
 - Whenever a key is pressed, both variables should be set to 0.



System Variables

System Variables

We have seen some system variables, such as mouseX and mouseY, that hold useful values. Here is a longer list of commonly used system variables:

<pre>mouseX,mouseY</pre>	Mouse location
width,height	Sketch size (in pixels)
displayWidth, displayHeight	Entire screen size (in pixels)
frameCount	Number of frames displayed so far.
frameRate	the current frame rate.
key	The most recent key used on the keyboard. e.g. 'a', 'b', '!', ' ',
keyCode	The code of the most recent key used on the keyboard. Useful for special keys, e.g. UP, LEFT, SHIFT, ALT,
keyPressed	true or false based on whether a key is pressed.
mouseButton	LEFT, RIGHT, or CENTER, based on which mouse button is pressed
mousePressed	true or false based on whether mouse is pressed.

System Variables to Control Animation

- This program draws a circle at the sketch center (determined by width & height) and writes some text on the top-left corner.
- The color and transparency are gradually changed using framecount.
- The program also reads the user's input using key and keyPressed().

```
String msg = "You wrote: ";
                                           You wrote: abcd
void setup() {
  size(400, 200);
  colorMode(HSB);
  textSize(26);
  noStroke();
}
void draw() {
  background(0);
  fill(frameCount % 255 , 255 , 255 ); //color changes
  text(msg, 10 ,30);
  fill(255,255,255,frameCount);
                                 //smooth appearance
  ellipse(width/2, height/2, 80, 80); //center of sketch
}
void keyPressed() {
  msg += key;
                                       //read user's input
}
                                                                COSC 123 – 51
```

System Variables to Make Decisions

You wrote code for the following program in a previous class using mouse-event methods. The same output can be produced using system variables (simpler code).



Example 1 (cont'd)

System Variables to Make Decisions, cont'd

```
void setup(){
  size(200,200);
  fill(0,255,0);
  stroke(255);
  strokeWeight(2);
void draw(){
  background(0);
  ellipse(mouseX,mouseY,40,40);
void mousePressed(){
  fill(255,0,0);
  stroke(255,255,0);
  strokeWeight(4);
void mouseReleased(){
  fill(0,255,0);
  stroke(255);
  strokeWeight(2);
```

```
void setup() {
  size(200,200);
void draw() {
  background(0);
  if (mousePressed) {
    fill(255, 0, 0);
    stroke(255, 255, 0);
    strokeWeight(4);
  } else {
    fill(0, 255, 0);
    stroke(255);
    strokeWeight(2);
  ellipse(mouseX, mouseY, 40, 40);
}
```

Simpler code.

Built-in Math Functions

Mathematical Functions

You have used the math function abs() before to produce an absolute value of a given number. Here is a list of more functions that you can use in your computations.

Function	Example	Function	Example
abs()	d = abs(-4); // 4	max()	d = min(2,7); // 7
round()	d = round(2.6); // 3	min()	d = min(2,7); // 2
<pre>floor()</pre>	d = floor(2.9); // 2	sin()	d = sin(PI/6); // 0.5
<pre>ceil()</pre>	d = ceil(2.3); // 3	cos()	d = cos(PI/3); // 0.5
pow()	d = pow(2,3); // 8	tan()	d = tan(PI/4); // 1
sq()	d = sq(-3); // 9	asin()	d = asin(0.5); // $\pi/6$
sqrt()	d = sqrt(9); // 3	acos()	d = $acos(0.5); // \pi/3$
dist()	d=dist(0,0,3,4);// 5	atan()	d = atan(1.0); // $\pi/4$

This one is particularly useful when developing games. We won't use it today, but just remember it for future lectures!

COSC 123 - 55

Angles in Processing

- Many Processing functions, e.g. trigonometric sin() & cos() and transformation's rotate() functions, take an angle argument *in radians*.
 - sin(PI/6) is 0.5
 - rotate(PI/4) rotates the coordinates by 45 degrees.
- degrees() and radians() can be used to convert from degrees to radians and vise versa.
 degrees(PI/2) is 90.
 radians(90) is PI/2

Note: the inverse of the trigonometric functions, i.e. asin(), acos(), atan(), take any value from -∞ to ∞ and output the corresponding angle in radians.
 e.g. asin(0.5) is PI/6 (i.e. π/6) cosc 123-56

Computing Motion Paths

This code moves a ball from left to right along a sinusoidal path.

The idea:

- 1) x is incremented by 1 after every frame, moving the ball from left to right.
- 2) for every x value, y is computed using the following expression which produces 2 full sinusoidal waves:

$$y = 30 \cdot \sin(\frac{x}{sketch \, width} \cdot 4 \cdot \pi)$$

To make things easier, the coordinate is translated so that the origin (0,0) is at the middle of the left edge

```
float x=0, y=0;
void setup(){
    size(200,200);
    background(0);
    noStroke();
}
void draw(){
    translate(0,height/2);
    y = 30 * sin(x*4*PI/width);
    ellipse(x, y, 5, 5);
    x++;
}
```



Nice Idea for Games!

- The motion paths for different items in a game could be computed using the same idea presented in the previous example.
- The code shown is very similar to the previous slide, except that it moves 3 triangles (could be enemy ships in a side-scrolling game) along the same sinusoidal path.



```
float x1=0,y1=0,x2=-20,y2=0,x3=-40,y3=0;
void setup(){
  size(400,200); noStroke(); fill(255);
}
void draw(){
  background(0);
  translate(0, height/2);
  //first spaceship
  y1 = 30 * sin(x1*6*PI/width);
  triangle(x1, y1-5, x1, y1+5, x1+10, y1);
  x1++;
  //second spaceship
  y_{2} = 30 * sin(x_{2}*6*PI/width);
  triangle(x2, y2-5, x2, y2+5, x2+10, y2);
  x2++;
  //third spaceship
  y3 = 30 * sin(x3*6*PI/width);
  triangle(x3, y3-5, x3, y3+5, x3+10, y3);
  x3++;
```

Exercise 1

Paths Defined by an Equation

Do you understand this animation?



float x1=0,y1=0,x2=-20,y2=0,x3=-40,y3=0; void setup(){ size(400,200); noStroke(); fill(255); } void draw(){ background(0); translate(0,height/2); //first spaceship y1 = 30 * sin(x1*6*PI/width);triangle(x1, y1-5, x1, y1+5, x1+10, y1); x1++; //second spaceship y2 = 30 * sin(x2*6*PI/width); triangle(x2, y2-5, x2, y2+5, x2+10, y2); x2++; //third spaceship y3 = 30 * sin(x3*6*PI/width); triangle(x3, y3-5, x3, y3+5, x3+10, y3); x3++;

What if we want to have another group of ships going in the opposite direction or following a different path? How about you also add the player's own spaceship which is controlled by the mouse?



Using System Variables (based on the textbook)

- Write code to produce the following sketches so that the code for all three is EXACTLY the same except for the size() statement.
- That is, the shapes must resize themselves relative to the window size. No matter what you specify for size(), the result should look identical!
- Hint: use width and height to determine the shapes location and





size(100,200)



size(300,200)

Lecture Activity Task

Analogue Clock

Write code to animate the seconds hand in a clock

$$\theta = \left(\text{second}() * \frac{PI}{30}\right) - \frac{PI}{2}$$

How about we now add the minutes and the hours hands?



