COMP1000 Lecture – Week 2

Greg Baker

5th August 2025

Learning Objectives

- Make some animations
- Connect variables to problem solving and algorithm development
- Access built-in variables like `width`, `height`, `mouseX`, and `mouseY`
- Declare global variables and update them in `draw()` to animate shapes
- Apply `random()` for positions or colours
- Handle mouse clicks using `mousePressed()`
- Use comments

Recap: Week 1 Growing Circle

Listing 1: GrowingCircleWeek1.pde

```
void setup() {
  size(500,500);
  background(0);
}

void draw() {
  fill(255,0,0);
  circle(250, 250,
     second());
}
```

- How do we make it grow faster? Is that related to the frame rate?
- Why does it "ripple"?
- What else could we animate?

Faster Growth

- Multiply the value from `second()` to make the circle expand faster
- Parameters can be numbers, variables, or expressions

Faster Growth

- Multiply the value from `second()` to make the circle expand faster
- Parameters can be numbers, variables, or expressions
  - Anywhere you see a number you can put a variable or an expression

Expressions

- Combinations of values, variables, and operators that produce a result
- Basic arithmetic: +, −, *, /, and %
- Can mix numbers and variables to compute new values
- But // means a comment – ignore the rest of the line
- /* *everything here is also a comment* */

Expression Evaluation

- Follows operator precedence: multiplication before addition
- Parentheses change the order of evaluation
- The final value can be assigned or used directly

Listing 2: ExpressionsExample.pde

```
void setup() {
  println(5 * (2 + 3)); // evaluates to 25;
}
```

Clearing the Background – the Rippling

- `draw()` runs repeatedly
- Without `background()` the circle draws over itself

Listing 3: GrowingCircleBackground.pde

```
void setup() {
  size(500,500);
}

void draw() {
  background(0);
  fill(255,0,0);
  circle(250, 250,
     10*second());
}
```

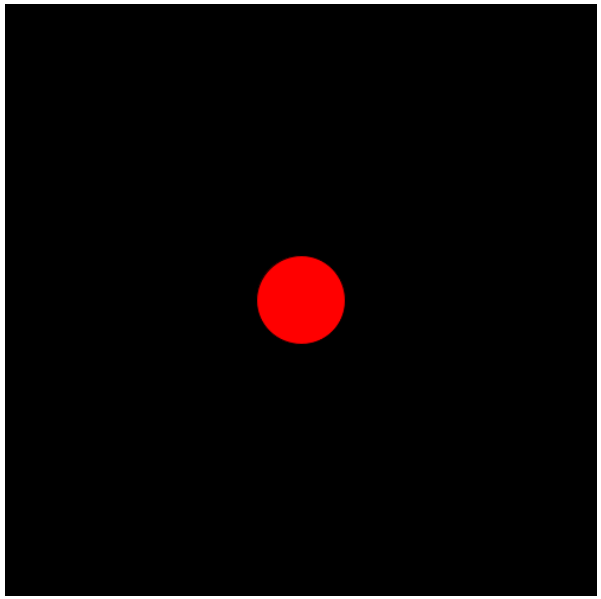
Smoother Growth with `millis()`

- `millis()` counts milliseconds since the sketch started
- Using it gives fractional sizes for smooth animation

Listing 4: GrowingCircleSmooth.pde

```
void setup() {
  size(500, 500);
  frameRate(60);
}

void draw() {
  background(0);
  fill(255, 0, 0);
  circle(250, 250, millis()
      * 0.05);
}
```



Cycling animations

Eventually `millis()` gets too big. How can we cycle?

- Modulo operator % – the remainder when you divide – is never bigger than the divisor
- Remember `sin()` and `cos()` from high school

Capped Growing Circle

- Use % to keep the circle's size below 400
- `millis()` keeps growing, but the remainder resets

Listing 5: GrowingCircleModulo.pde

```
void setup() {
  size(500, 500);
  frameRate(60);
}

void draw() {
  background(0);
  fill(255, 0, 0);
  circle(250, 250, (millis() * 0.05) % 400);
}
```

Cycling Background Colour

- Multiply `millis()` then take `%` 256 for RGB values
- Gives a repeating rainbow without any variables

Listing 6: CyclingBackground.pde

```
void setup() {
  size(500, 500);
  frameRate(60);
}

void draw() {
  background((millis() * 0.1) % 256,
             (millis() * 0.2) % 256,
             (millis() * 0.3) % 256);
}
```

Line Around a Circle

- `sin()` and `cos()` give positions on a circle
- `TWO_PI` radians equals one full turn

Listing 7: RotatingLine.pde

```
void setup() {
  size(400, 400);
}

void draw() {
  background(255);
  line(0, 0,
       200 + cos(TWO_PI * millis() /
          1000.0) * 100,
       200 + sin(TWO_PI * millis() /
          1000.0) * 100);
}
```

What is a Variable?

These expressions are getting complicated! Can we give parts of these expressions names?

- A named storage location for a value
- Lets programs remember information between steps
- Variables are like your phone's memory—they remember things so you don't have to
- The type of a variable determines what kind of data it can hold
- Some variables are built-in, some you create
  - You declare the type of the variable, and the name (and optionally, an initial value)

Built-in Variables

- `width` and `height` give the size of the window
- `mouseX` and `mouseY` track the mouse position
- Useful for positioning shapes relative to the screen or cursor

Example: Line to the Mouse

Listing 8: LineToMouseXMouseY.pde

```
void setup(){
  size(400,300);
}

void draw(){
  background(255);
  line(0,0,mouseX,mouseY);
  println(width,height,mouseX,mouseY);
}
```

Standard Types in Processing

- `int` for whole numbers
- `float` for numbers with decimals
- `boolean` for true or false values
- `String` for text
- One day we'll define our own types

The Assignment Statement

- Uses the = operator to store a value in a variable
- The expression on the right is evaluated first
- The result replaces the previous value on the left

Listing 9: VariableAssignment.pde

```
int score = 0;

void setup() {
  size(200, 200);
  score = score + 10; // adds 10 to score
  println(score);
}
```

Polling: What will this code print?

```
int score = 0;

void setup() {
  size(200, 200);
  score = score + 10; // adds 10 to score
  println(score);
}
```

- What value will appear in the console?

Polling: Answer

The console prints 10.

Try This Now

Change the initial `score` to 50 and add 25 instead of 10.

Problem Solving with Variables

- Imagine a game character starts with 100 health points
- A variable `health` lets us update this value as the game runs
- Variables help algorithms track and modify game state

Global Variables for Animation

Listing 10: RedCircleFallingDown.pde

```
int yPos=0; //global variable

void setup(){
  size(300,700);
}

void draw(){
 background(255);
 fill(255,0,0);
 circle(width/2, yPos, 50);
 yPos = (yPos + 2) % height;
}
```

Common Variable Mistakes (which the compiler will complain about)

- Using a variable before it is declared
- Storing the wrong type of data in a variable: when you store a `float` in an `int`

String Variables

- Use `String` to store words and sentences
- Combine strings with + to build messages
- Handy for labels, names, and debugging output

Drawing Text

- `text(string, x, y)` displays words on the screen
- `textSize()` and `textAlign()` control appearance
- Useful for labels, instructions, or debugging values

Listing 11: DrawText.pde

```
void setup() {
  size(200, 200);
  textSize(32);
  textAlign(CENTER, CENTER);
}

void draw() {
  background(200);
  text("Hello", width/2, height/2);
}
```

Displaying a Number

Listing 12: RedCircleFallingDownText.pde

```
int yPos = 0; // global variable

void setup() {
  size(300, 700);
}

void draw() {
  background(255);
  fill(255, 0, 0);
  circle(width/2, yPos, 50);
  fill(0);
  text(yPos, 10, 20);
  yPos = (yPos + 2) % height;
}
```

Randomness and Pseudo-Randomness

- Computers use algorithms to generate pseudo-random numbers
- Given the same starting seed – `randomSeed()` – they produce the same sequence
- `random()` gives values that seem unpredictable
- Good enough for animations and simple games
- For real randomness in cryptography, computers use CPU entropy and timing events

Random Numbers

Listing 13: RandomBubbles.pde

```
void setup(){
  size(400,400);
  frameRate(4);
}

void draw(){
  background(0);
  fill(255,0,0);
  circle(random(width),random(height),50);
}
```

Try This Now

Use `random()` so bubble sizes vary between 10 and 100.

Event Handlers

- Special functions that run in response to user actions
- Examples: `mousePressed()`, `keyPressed()`, `mouseMoved()`
- `draw()` checks input state and calls the right handler

Responding to Mouse Presses

Listing 14: GrowingCircles.pde

```
int dia=10;
void setup(){
  size(400,400);
}
// draw() calls mousePressed(),
// if you miss draw() mouse pressed will be ignored
void draw(){
}

void mousePressed(){
 fill(random(255),random(255),random(255));
 circle(mouseX,mouseY, dia);
 dia = (dia+10) % 200;
}
```

Try This Now

Create a variable called `clicks`. Increase it in `mousePressed()` and display the count.

State and Variables

- Variables store everything needed to rebuild the sketch's state
- Imagine recording where every item belongs in your house
- Programs do the same for positions, colours, and more

What Can You Build Now?

- A clock that updates every second
- A drawing pad that leaves trails where the mouse moves
- A bouncing ball that resets when it hits the bottom
- A bubble generator using `random()` for colours and size

Summary

- Use built-in and global variables to store values
- `random()` creates variety in your sketches
- Respond to events like mouse clicks for interaction