# Lab Week 3. Events and Animation

### Last Week

* Modular code
  + Procedures
  + Functions (see videos and walkthrough in week 2)
  + Passing parameters
* Local variables, global variables

## Learning Objectives

* Events
* Animation
* IF statement – conditional branching
* User interaction
* Pong Game

You should have completed all last week’s practical lab exercises. How are you progressing with the formative tests?

[Note : you may need to click on the drawn screen to interact with it – via mouse or keyboard]

## Resources

* Lecture Notes & moodle – some references on conditionals (**if** statements) on moodle
* Processing website - reference
* <http://www.cs.sfu.ca/CourseCentral/166/tjd/first_program.html>

[Pay, particular attention to the exercises at the bottom of the page above – which you should attempt in your own time – some of these exercises come from that source]

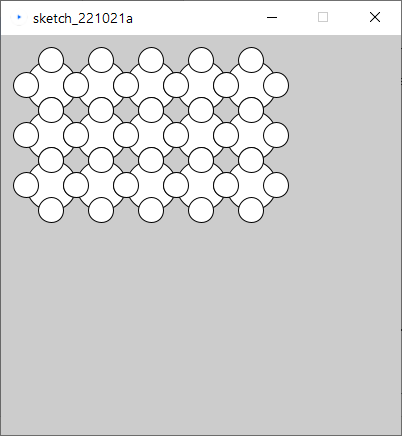
1. **Debugging Exercise**

Code below, should draw the pattern (right). Fix the code and improve its readability (refactor):

void setup()

{

size(400,400);

 Pattern(50,50.5,3);

}

void pattern(int x,int y, int s, int r, int c

[

int left = x;

for(int i=0,i<c,i++);

{

for(int j=0;j<r;j++)

{

int half=s/2;

ellipse(x,y, s,s);

eclipse(x+half,y, half,half);

ellipse(x-half,y, hal,half);

ellipse(x,y+half, half);

ellipse(x,y-half, half,half);

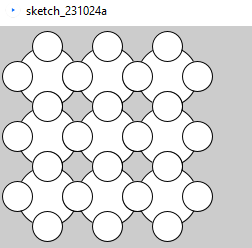
x=x+s

}

x=left;

y=y+s;

}



void setup()

{

size(400,400);

pattern(50,50,60,3,3);

}

void pattern(int x,int y, int size, int row, int collumn)

{

int left = x;

for(int i=0;i<collumn;i++)

{

for(int j=0;j<row;j++)

{

int half=size/2;

ellipse(x,y, size,size);

ellipse(x+half,y, half,half);

ellipse(x-half,y, half,half);

ellipse(x,y+half, half,half);

ellipse(x,y-half, half,half);

x=x+size;

}

x=left;

y=y+size;

}

}

1. **Group Exercise**

Let’s create an animation to draw a circle that grows to 200 pixels wide then shrinks to 0, before starting the cycle again. How will we achieve this?

**3. Bouncing ball**. Earlier we started thinking about making a ball bounce off the left and right edge of the screen. We now want to extend this so the ball moves up and down (with bounces from top and bottom) as well.

Think about the extra variables we will need. Hint everything we tackled for the **x** movement needs duplicating for the **y** movement.

//Global variables

int x; //ball x position

int speedX = 5; //ball x direction is right, step 5

void **setup**() //runs once at start

{

**size**(500,250);

}

void **draw**() //runs repeatedly

{

**background**(200); //clear screen RGB = 200 (grey)

//draw and update ball position

**ellipse**(x,125,10,10);

x = x + speedX; //move ball x right

//Collision Detection

//collide right hand edge?

if (x>=500)

speedX = -speedX; //reverse x direction

//collide left hand edge?

//collide top edge?

//collide bottom edge?

}

Think about refactoring code above, moving code to a separate procedure **updateBall()**, **bounceOffWalls()** as another.

**Ex 4.** Write a program which allows the UP and DOWN keys to move a rectangle (**rect** command) up and down the screen (draw it near the left hand edge). We covered this event in the slides (see <https://processing.org/reference/keyCode.html> for an example). Add **if** statements to prevent it disappearing off the top and bottom edge (try to keep the rectangle completely within the screen).

What do we need to store to know where to draw the rectangle?

**Ex 5.** Modularise code and add a Bat

Earlier, we wrote a program that moved a rectangle up and down using the arrow keys. Add this code to your ball program and add in collision detection function for the bat. Disable bouncing from the left hand edge and you have a Pong style squash game.

**Extension exercises**

Using the Processing web pages for reference you can extend this to a complete game by keeping a score (displayed on screen, **text** command) of the number of shots in a rally, add in lives, 3 misses and game over etc.

* Extend your game to make it two player.
* Add a second ball to the game.
* Once completed try to write a simple algorithm to allow computer control of one of the players (simple Artificial Intelligence – how can we achieve this with the techniques we have met so far).

**EtchySketch**

The code below allows the user to draw a continuous picture on the screen, using the arrow keys. This code can be adapted to become a **light cycles game (**[**https://www.greenfoot.org/scenarios/1556**](https://www.greenfoot.org/scenarios/1556)**)** . We would need to test the colour of the new position before the point is drawn there. The get() function (see <https://processing.org/reference/get_.html>) returns the colour of a pixel on the screen, allowing you to check whether the new position is already occupied or a background colour. Normally the outer edge of the screen would also have coloured walls to crash into as well.

int x,y; //position

int speedX=0; //ready for liteCycles

int speedY=0;

void setup()

{

size(400,400);

x=width/2;

y=height/2;

}

void draw()

{

point(x,y);

}

void keyPressed()

{

if(key==CODED) //if arrow cluster pressed

{

if (keyCode==UP)

y=y-1;

if (keyCode==DOWN)

y=y+1;

if(keyCode==LEFT)

x=x-1;

if(keyCode==RIGHT)

x=x+1;

}

}

CodingBat – function writing practice

<https://codingbat.com/java/Logic-1>

pick up where you left off with the problems in the link above, there are also more complex problems here:

<https://codingbat.com/java/>