Computer Creativity

Conditionals and a little about Images

Slides courtesy of Dr. Abdallah Mohamed.

Objectives

- This is a pre-class reading materials. After reading them, you should be able to:
  - Write and evaluate conditions. This includes using:
    - relational operators (==, >=, etc.)
    - AND, OR, and NOT operators (to write complex conditions).
    - nested conditional statements.
  - Make decisions in your program using `if-else` statement
  - Understand and use the “IDEAS” presented in this set of notes.

- *Today is a revision on conditionals* **+** *useful ideas for your animation*
  - *This pre-class reading is basically a revision on topics discussed in COSC 111,122*
  - *The* ***only*** *addition is the application of these topics in the context of our course (Processing).*

- *However, this pre-class reading is expected take a bit longer time than previous ones*

Reference Material
Conditionals I

Making Decisions

- ***Decisions*** are used to allow the program to perform different actions in certain conditions.
  - For example, if a person applies for a driver's license and is not 17, then the computer should not give them a license.

- To make a decision in a program we must do several things:
  - 1) Determine the ***condition*** used to make the decision.
    - E.g. in a website for applying for a driving license, what is the appropriate age to for an applicant to be eligible?
  - 2) Tell the computer what to do if the condition is true or false.
    - A decision always has a Boolean value or true/false answer.
    - E.g. in the above example, if age > 16, allow user to apply.
- There are several keywords that can be used to make a decision such as the ***if*** statement.

Making Decisions: Comparisons

- ***Relational operators*** compare two items called operands.
  - Syntax: `operand1 operator operand2`

- Comparison operators:
  - `>` - Greater than
  - `>=` - Greater than or equal
  - `<` - Less than
  - `<=` - Less than or equal
  - `==` - Equal (Note: not "=" which is used for assignment)
  - `!=` - Not equal

- The result of a comparison is a ***Boolean value*** which is either `true` or `false`.

Using Boolean Variables

```
int j=25, k = 45;
double d = 2.5, e=2.51;
boolean result;

result = (j == k);        // false
result = (j <= k);        // true
result = (d != e);        // true
result = (k >= 25);       // true
result = (25 == j);       // true
result = (j > k);         // false
result = (e < d);         // false
j = k;
result = (j == k);        // true
```

Making Decisions: `if` Statement

- To make decisions with conditions, we use the `if` statement.
  - If the condition is `true`, the statement(s) after `if` are executed otherwise they are skipped.
  - If there is an `else` clause, statements after `else` are executed if the condition is `false`.
- Syntax:

```
if (condition)
    statement;
```

OR

```
if (condition)
    statement;
else
    statement;
```

- Example

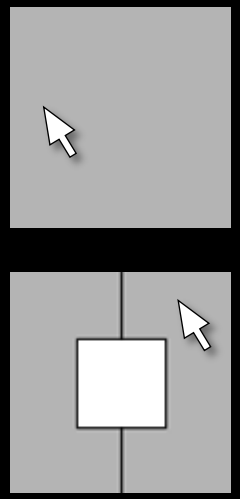
```
//draw rectangle only when
//mouse is in right half of window
void draw() {
  background(180);
  if(mouseX > width/2)
    rect(30, 30, 40, 40);
}
```

```
//draw rectangle or ellipse based
//on where the mouse is
void draw() {
  background(180);
  if(mouseX > width/2)
    rect(30, 30, 40, 40);
  else
    ellipse(50, 50, 40, 40); }
```

Making Decisions: Block Syntax

- If there are multiple statements in the if or else parts, we must use the block syntax. A block starts with "{" and ends with "}".
  - All statements inside the brackets are grouped together.
- Example:

```
//draw a rectangle and a line only when
//mouse is in right half of window
void draw() {
  background(180);
  if(mouseX > width/2) {
     line(width/2, 0, width/2, height);
     rect(30, 30, 40, 40);
  }
}
```

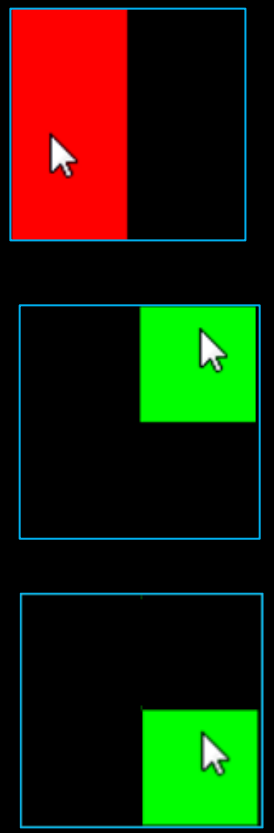


- We will use block statements in many other situations as well.

Nested `if` Statement

- We nest if statements for more complicated decisions.

```
void draw() {
  background(0);
  if (mouseX < width/2) {
    fill(255,0,0);
    rect(0,0,width/2,height);
  }else{
    fill(0,255,0);
    if (mouseY < height/2)
      rect(width/2, 0, width/2, height/2);
    else
      rect(width/2, height/2, width/2, height/2);
  }
}
```



Dangling `else` Problem

- The dangling else problem occurs when a programmer mistakes an else clause to belong to a different if statement than it really does.
- You can use blocks (brackets) to determine which statements are grouped together
- ***Example***: we want `else` to belong to **first** `if`.

Wrong

```
if (grade >= 50))
     if (grade >= 95)
        state = "With Honours";
else // Belongs to 2nd if
        state = "Fail";
```

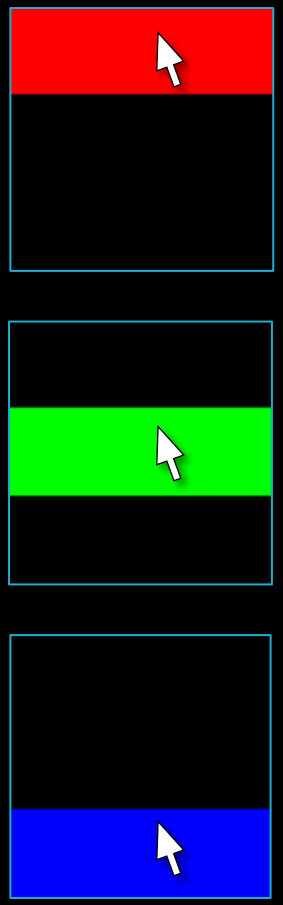
Correct

```
if (grade >= 50){
     if (grade >= 95)
        state = "With Honours";
} else // Belongs to 1st if
     state = "Fail";
```

Multi-part if statement

- You can test multiple conditions using "`else if`" as in the example below

```
void draw() {
  background(0);
  noStroke();
  if (mouseY < height/3) {
    fill(255,0,0);
    rect(0,0,width,height/3);
  } else if (mouseY < 2*height/3){
    fill(0,255,0);
    rect(0,height/3,width,height/3);
  } else {
    fill(0,0,255);
    rect(0,2*height/3,width,height/3);
  }
}
```



Boolean Expressions

- A Boolean expression is a sequence of conditions combined using AND (`&&`), OR (`||`), and NOT (`!`).
  - Allows you to test more complex conditions
  - Group subexpressions using parentheses
- Syntax:
  - `(expr1) && (expr2)` - expr1 AND expr2
  - `(expr1) || (expr2)` - expr1 OR expr2
  - `!(expr1)` - NOT expr1
- Examples:

```
boolean b;
b = (x > 10) && !(x < 50);
b = (month == 1) || (month == 2) || (month == 3);
if (day == 28 && month == 2);
if !(num1 == 1 && num2 == 3);
b = ((10 > 5 || 5 > 10) && ((10>5 && 5>10)); // False
```

IDEA1: Deciding based on System Variables

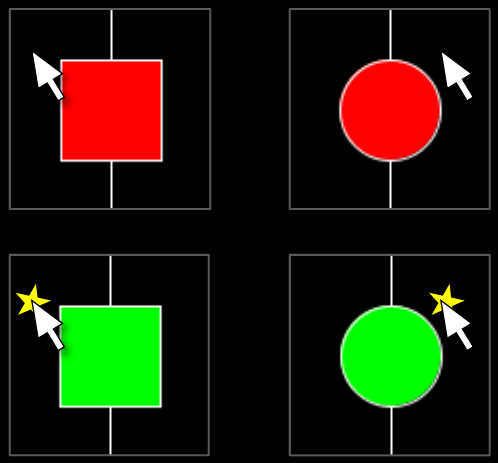
Draw Only When Mouse is Pressed

- You previously wrote a program to draw a continuous line using the *current* and *previous* mouse coordinates.
- The previous program can be modified so that the mouse draws only IF the mouse is pressed (using the system variable `mousePressed`).

```
void setup() {
      size(400,400); background(255);
}
void draw() {
  if(mousePressed)
       line(pmouseX,pmouseY,mouseX,mouseY);
}
```

Changing Color Only when Mouse is Pressed

- The system variable `mousePressed` is used to determine the fill color.
  - "*if mouse is pressed, color is **red** otherwise color is blue*"
- `mouseX` is used to determine which shape to draw
  - "*if mouse is on right, draw a circle otherwise draw a rect*"

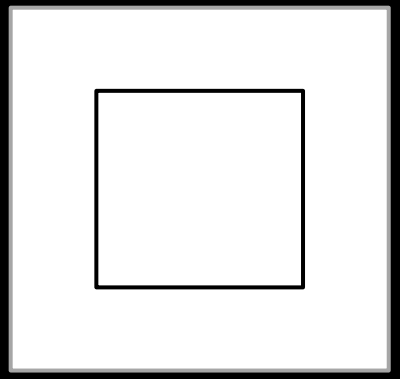


```
void setup() {
  rectMode(CENTER);
  stroke(255);
}
void draw() {
  background(0);
  line(width/2, 0, width/2, height);

  //Set Color based on whether mouse is pressed
  if (mousePressed)
    fill(0, 255, 0);
  else
    fill(255, 0, 0);

  //Draw Shape based on mouse position
  if (mouseX > width/2)
    ellipse(width/2,height/2,50,50);
  else
    rect(width/2,height/2,50,50);
}
```

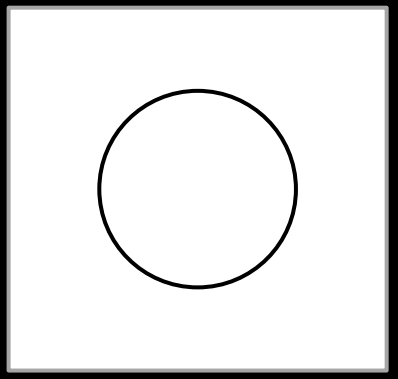
Changing Shape if Key is Pressed

- If any key is pressed, draw a rectangle.
- Otherwise, draw a circle

```
void draw() {
  background(255);
  if (keyPressed)
     rect(20, 20, 60, 60);
  else
     ellipse(50, 50, 80, 80);
}
```



keyPressed = true



keyPressed = false

IDEA2: Moving Objects with Arrow Keys

Moving Objects with Arrow Keys Version1

The key idea is as follows:

- Draw an item at (x,y). Update x & y with speedX and speedY in every frame.
- When an arrow key is pressed:
  - Set speedX/Y to non-zero values.
- When key is released
  - Set speedX/Y to 0

Try this code → on your computer!!

```
float x=50, y=50, speedX=0, speedY=0;
void draw() {
  background(0);
  ellipse(x,y,30,30);
  x += speedX;
  y += speedY;
}
void keyPressed(){
  if (keyCode == LEFT)  speedX = -5;
  if (keyCode == RIGHT) speedX = 5;
  if (keyCode == UP)    speedY = -5;
  if (keyCode == DOWN)  speedY = 5;
}
void keyReleased(){
  if(keyCode==LEFT||keyCode==RIGHT) speedX = 0;
  if(keyCode==DOWN||keyCode==UP)    speedY = 0;
}
```

Moving Objects with Arrow Keys Version2

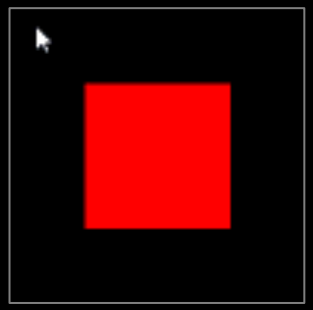
This is the same code as the previous example except we use the `constrain` function to keep the object within the boundaries of the sketch.

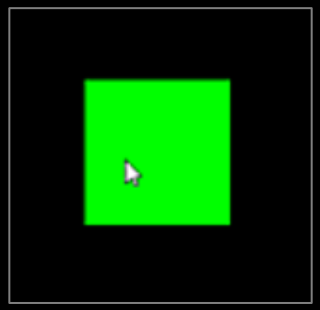
```
float x=50, y=50, speedX=0, speedY=0;
void draw() {
  background(0);
  ellipse(x,y,30,30);
  x = constrain(x+speedX,0,width);
  y = constrain(y+speedY,0,height);
}
void keyPressed(){
  if (keyCode == LEFT)  speedX = -5;
  if (keyCode == RIGHT) speedX = 5;
  if (keyCode == UP)    speedY = -5;
  if (keyCode == DOWN)  speedY = 5;
}
void keyReleased(){
  if(keyCode==LEFT||keyCode==RIGHT) speedX = 0;
  if(keyCode==DOWN||keyCode==UP)    speedY = 0;
}
```

IDEA 3: Detecting Mouse Movement Over Objects

Detecting Mouse Over a Rectangle

- The example shows how to detect the mouse moving over a shape (rectangle).
- When the mouse moves over the shape, the color changes to green. When the mouse is off the shape, the color changes back to red.



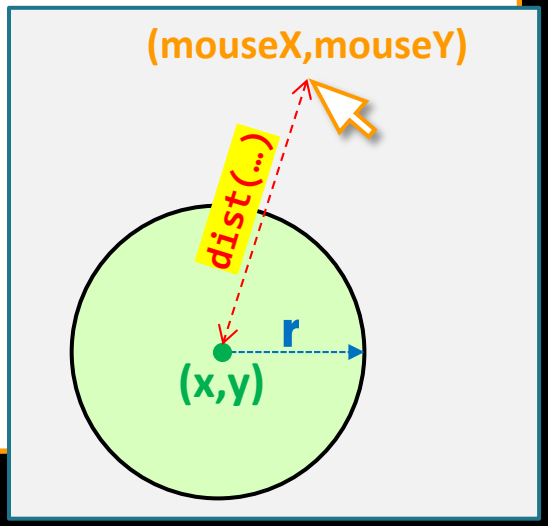


```
float x = 50, y = 50, w = 100, h = 100;
void setup(){
  size(200,200);
}
void draw(){
  background(0);
  //set color based on mouse position
  if(mouseX > x && mouseX < x+w &&
     mouseY > y && mouseY < y+h)
      fill(0,255,0);
  else
      fill(255,0,0);
  //draw rectangle
  rect(x,y,w,h);
}
```

Detecting Mouse Over a Circle

- This code is similar to previous code except that it detects the mouse presence over a ***circular*** area.
- The `dist()` function is used to check the distance between the mouse location and the circle center (i.e. to check if mouse is within the circumference of the circle.)

```
float x = 100, y = 100, r = 50;
void setup(){
  size(200,200);
}
void draw(){
  background(0);
  //set color based on mouse position
  if(dist(mouseX,mouseY,x,y) < r)
    fill(0,255,0);
  else
    fill(255,0,0);
  //draw circle
  ellipse(x,y,2*r,2*r);
}
```



Multiple Rollover

- In this code, we use if statement to highlight the one quarter of the window that is under the mouse.

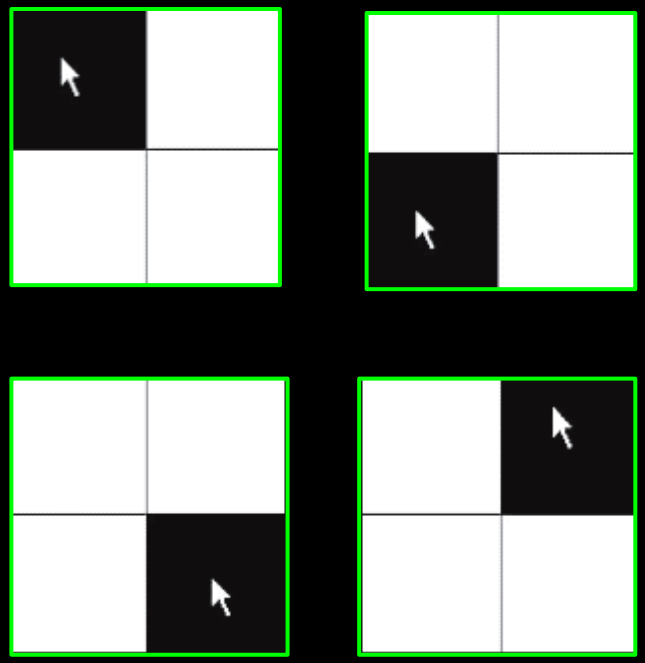


Figure 5.7

```
void setup() {
  size(200, 200);
}
void draw() {
  background(255);
  stroke(0);
  line(100, 0, 100, 200);
  line(0, 100, 200, 100);
  // Fill a black color
  noStroke();
  fill(0);
  if (mouseX < 100 && mouseY < 100)
    rect(0, 0, 100, 100);
  else if (mouseX > 100 && mouseY < 100)
    rect(100, 0, 100, 100);
  else if (mouseX < 100 && mouseY > 100)
    rect(0, 100, 100, 100);
  else if (mouseX > 100 && mouseY > 100)
    rect(100, 100, 100, 100);
}
```

Source: Shiffman

IDEA 4: Clickable Buttons

Designing a Clickable Button version1

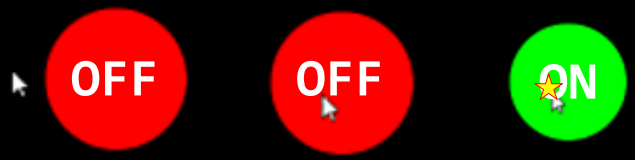
- This code is similar to previous example except that the circle (button) turns green if two conditions are satisfied:
  - Mouse is over the button
  - Mouse is pressed
- This simulates the visuals of a button being clicked

```
float x = 100, y = 100, r = 50;
void setup(){
  size(200,200);
}
void draw(){
 background(0);
 //set color based on mouse position
 if(dist(mouseX,mouseY,x,y)<r && mousePressed)
    fill(0,255,0);
 else
    fill(255,0,0);
 //draw circle
 ellipse(x,y,2*r,2*r);
}
```

Designing a Clickable Button version2

- This code is the same as the previous example except with additional visual components:
  - ON/OFF text on the button
  - Button gets smaller when clicked.

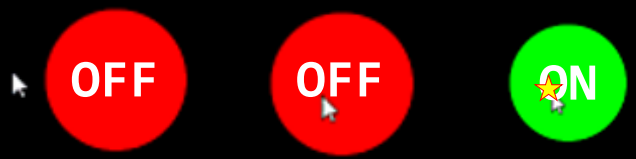
see orange code on the right




```
float x = 100, y = 100, r = 50; String s = "OFF";
void setup(){
  size(200,200);
  textSize(28); textAlign(CENTER,CENTER);
}

void draw(){
  background(0);
  //set color based on mouse position & if clicked
  if(dist(mouseX,mouseY,x,y)<r && mousePressed){
    fill(0,255,0);
    s = "ON";
    r = 45;
  }else{
     fill(255,0,0);
     s = "OFF";
     r = 50;
  }
  //draw button
  ellipse(x,y,2*r,2*r);
  fill(255);
  text(s,x,y);
}
```

Designing a Clickable Button version3

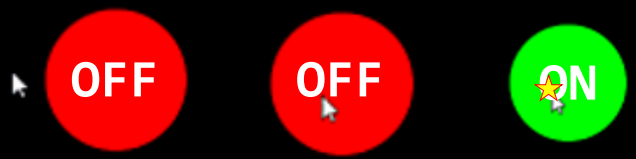
- This is the same as version2 of this example except a **Boolean variable** is used to "remember" whether the button is clicked.




```
float x = 100, y = 100, r = 50; String s = "OFF";
boolean clicked = false;
void setup(){
  size(200,200);
  textSize(28); textAlign(CENTER,CENTER);
}
void draw(){
  background(0);
  //set clicked
  if(dist(mouseX,mouseY,x,y)<r && mousePressed)
     clicked=true;
  else
     clicked=false;
  //set button attributes
  if(clicked){
    fill(0,255,0);  s = "ON";    r = 45;
  }else{
    fill(255,0,0); s = "OFF";  r = 50;
  }
  //draw button
  ellipse(x,y,2*r,2*r);
  fill(255);
  text(s,x,y);
}
```

Designing a Clickable Button version4

- Here is another way that produces the same output as the previous example, but using ***boolean variable*** that is set in two methods: `mousePressed` and `mouseReleased`.
- The ***benefit of using a Boolean variable*** is that its can be accessed in different methods.




```
float x = 100, y = 100, r = 50; String s = "OFF";
boolean clicked = false;
void setup(){
  size(200,200);
  textSize(28); textAlign(CENTER,CENTER);
}
void draw(){
  background(0);
  if(clicked){
    fill(0,255,0);  s = "ON";   r = 45;
  }else{
    fill(255,0,0); s = "OFF";  r = 50;
  }
  ellipse(x,y,2*r,2*r);
  fill(255);
  text(s,x,y);
}
void mousePressed(){
 if(dist(mouseX,mouseY,x,y)<r && mousePressed)
       clicked=true;
 else
       clicked=false;
}
void mouseReleased(){
       clicked=false;
}
```

IDEA 5: Toggle Buttons

Designing a Toggle Button

- A toggle button holds one of two states. Clicking the button alternates the state.
- The state of the button can be stored in a ***Boolean variable***, and the looks of the button as well as other decisions can be determined based on that variable.

```
boolean buttonActive = false;    // initial button state
int x = 50, y = 50, r = 40;      // button attributes
void setup() {
  size(200, 200);    strokeWeight(5);
  textSize(25);      textAlign(CENTER,CENTER);
}
void draw() {
 background(0);
 if(buttonActive) { // make decisions based on button state
  fill(0, 200, 0);  stroke(0,255,0); ellipse(x,y,2*r,2*r);
  fill(200,255,200);text("ON",x,y);
 }else{
  fill(180, 0, 0);  stroke(255,0,0); ellipse(x,y,2*r,2*r);
  fill(100,0,0);    text("OFF",x,y);
 }
}
void mousePressed() {
 if(dist(mouseX,mouseY,x,y)<r)
     buttonActive = ! buttonActive; // change button state
}
```

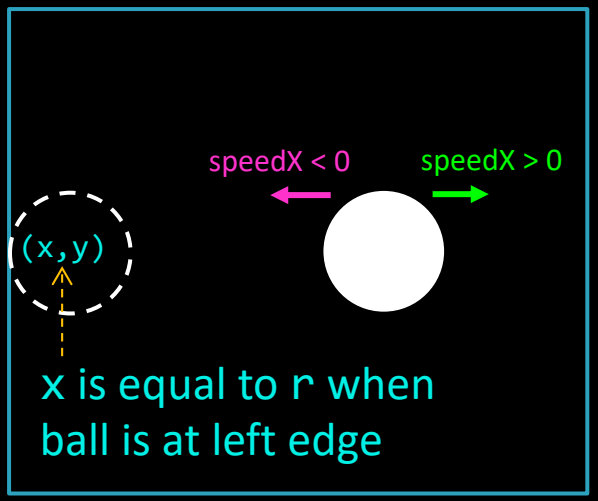
IDEA 6: Bouncing “Attributes”

Bouncing Ball version 1

The code shows a ball bouncing off the edges. For simplicity, the ball moves only horizontally.

IDEA:

- increment the x position by speedX.
- **IF** the ball reaches the right **OR** left edge, reverse the motion direction (by changing the sign of speedX).

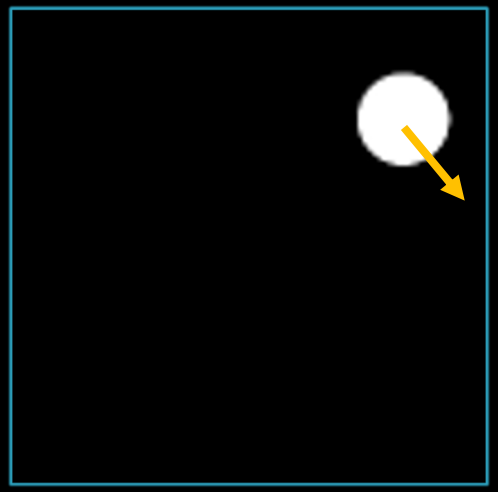



```
float speedX = 3;
float x=20, y=100, r = 20;
void setup(){
  size(200,200);
}
void draw(){
  background(0);
  ellipse(x,y,2*r,2*r);
  //increment x
  x += speedX;
  //if ball at edge, reverse direction
  if( x > width-r || x < r )
      speedX = -speedX;
}
```

Bouncing Ball version 2

This is the same as previous example except that the ball moves in all directions.

- i.e. y and speedY also change.



```
float speedX = 1, speedY = 2;
float x=20, y=100, r = 20;
void setup(){
  size(200,200);
}
void draw(){
  background(0);
  ellipse(x,y,2*r,2*r);
  //increment x,y
  x += speedX;
  y += speedY;
  //if ball at edge, reverse direction
  if( x > width-r || x < r )
      speedX = -speedX;
  if(y > height-r || y < r)
      speedY = -speedY;
}
```

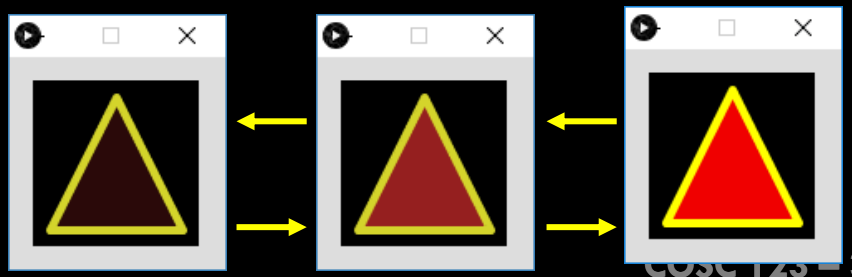
Bouncing Other Attributes

The logic used in the Bouncing Ball example can be applied to bounce *any attribute* once it reaches its limits.

Example: Flashing Color

- In this example, we ***bounce the fill color*** once its value reaches some limits (50 to 245). The output is a flashing triangle that looks like an alert signal.

```
float c = 100;     //color attribute
float speedC=10;   //speed of color change
void setup() {
  size(100, 100);
  strokeWeight(5);
  stroke(255,255,0);
  strokeJoin(ROUND);
}
void draw() {
  background(0);
  fill(c,0,0);
  triangle(50,10,10,90,90,90);
  c += speedC;
  if(c<50 || c>245)
    speedC = - speedC;
}
```



Using Conditionals in Sketches

Let's discuss the most confusing ideas

- IDEA1: Deciding based on system variables
  - mousePressed, keyPressed, mouseX, …etc
- IDEA2: Moving objects with arrow keys
  - keyPressed() and keyReleased() along with keyCode
- IDEA3: Detecting mouse movement over objects
  - comparing mouseX,mouseY to object location
  - The use of dist() function
- IDEA4 & 5: Creating Buttons (clickable and toggle)
  - Using a Boolean variable to "remember" the status of the button
- IDEA6: Bouncing Attributes
  - Reverse how the attribute changes once a limit is reached.

Example on Bouncing Attributes

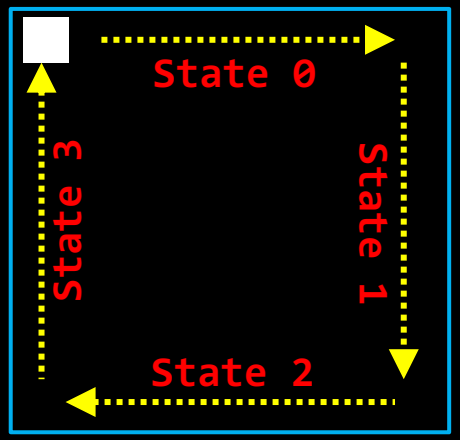
- This code bounces four attributes:
  - x location
  - y location
  - Shade (color)
  - Size (diameter)

```
float x =100, speedX = 4;
float y =100, speedY = 3;
float sh =100, speedSh = 20;
float d =35, speedD = 5;
void setup(){size(200,200);}
void draw(){
  background(0);
  //1) USE variable attributes to draw
  fill(sh);
  ellipse(x,y,d,d);
  //2) UPDATE attributes
  x += speedX;
  y += speedY;
  sh += speedSh;
  d += speedD;
  //3) BOUNCE attributes that reach their limits
  //once you hit an edge, then reverse the speed
  if(x>=width-d/2 || x<=d/2)  //did the ball hit a vertical edge?
    speedX = -speedX;
  if(y>=height-d/2 || y<=d/2) //did the ball hit a horizontal edge?
    speedY = -speedY;
  //once you reach the limits of [0,255], reverse the speedSh
  if(sh>=255 || sh<=0)
    speedSh = -speedSh;
  //once size is outside limits, reverse speedD
  if(d>=60 || d<=30)
    speedD = -speedD;
}
```

IDEA 7: Designing Complex Motion Paths

Designing Complex Motion Paths

- One can design complex motion paths using conditionals.
- For example, assume we want to move an item along the dotted path as in the shown figure (i.e. around the window).
- One way to solve this problem is to
  - Create a variable "state" that 'remembers' which direction the rectangle is going.
    - We have 4 rectangle states: 0 to 3. Each state will represent the motion in one directions.
  - use the variable to decide how the rectangle moves.
- The code for doing this is on the next slide.



Designing Complex Motion Paths (2)

```
float d = 20, x = d, y = d; // location of ball
float speed=5, state=0;     // initial speed and state of square
void setup() {size(200, 200); noStroke(); fill(255); }

void draw() {
  background(0);
  ellipse(x, y, d, d);
  if(state==0){             // RIGHT state
      x += speed;           // keep moving right
      if (x > width-d) {    // when the square reaches the right edge:
        state = 1;          //   a) switch to DOWN state
        x = width-d;        //   b) align square for accuracy
      }
  }else if(state==1){       // DOWN state
      y = y + speed;
      if (y > height-d) {y=height-d; state=2;}
  }else if(state==2){       // LEFT state
      x = x - speed;
      if (x < d) {x=d; state=3;}
  }else if(state==3){     // UP state
      y = y - speed;
      if (y < d) {y=d; state=0;}
  }
}
```

Exercise

Again..

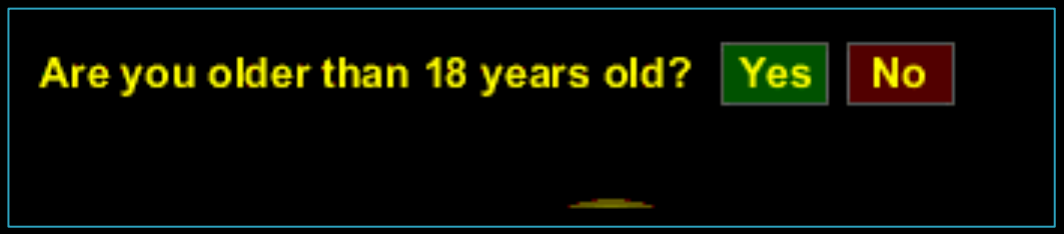
- Same as previous example but we are controlling other aspects of the animation based on the object state.

```
int x = 15, y = 15, r = 15, speed = 2;
int state = 0;

void setup(){
  size(300,300);
  rectMode(CENTER);
}
void draw(){
  //control background COLOR based on object state
  if(state==0)  background(0);
  else if(state ==1) background(255,0,0);
  else if(state ==2) background(0,255,0);
  else background(0,0,255);
  //control object SHAPE and SIZE based on state
  if(state == 0 || state == 2)
    ellipse(x, y, 2*r+random(10), 2*r+random(10));
  else
    rect(x,y,2*r,2*r);
  // control object PATH based on state
  if(state == 0){
    x += speed;
    if(x >= width - r) state = 1; //switch state when reaching edge
  } else if(state == 1){
    y += speed;
    if(y >= height - r) state = 2;
  } else if(state == 2){
    x -= speed;
    if(x <= r) state = 3;
  } else if(state == 3){
    y -= speed;
    if(y <= r) state = 0;
  }
}
```

Using Buttons to Get User's Input

Getting Input from the User

- We can use the IDEA 4/5 (buttons) to read input from user.
- The code on the next page displays a question to the user with two possible answers, Yes and No.
- Whenever the user clicks a button, the button brightness increases and a message is displayed.
  - "Yes" displays in green "Congrats! You can get a driving license"
  - "No" displays in red "Sorry! You are too young to get a driving license"



Getting Input from the User, cont'd

- The idea:
  - Draw two toggle buttons using the code from IDEA5 in the pre-class readings
    - Note that you will need two sets of variables, one for each button. i.e. we need `x1` for button *Yes* and `x2` for button *No*, and so on.
  - Create one variable, `answer`, that "remembers" which button is clicked.
    - e.g. `answer` should be 1 for Yes, 2 for No, and -1 otherwise
  - For each button, create a `color` variable.
  - When mouse is clicked, set `answer` to one of three values:
    - 1 if Yes is clicked
    - 2 if No is clicked
    - -1 if mouse is clicked away from the two buttons.
  - Use the value of `answer` to decide how to **draw ALL items** on the sketch (e.g. Yes should be bright green if answer is 1)

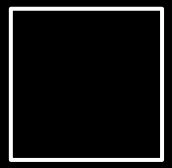
```
int answer = -1;                        // 1 for YES, 2 for NO, -1 for neither
final int x1 = 237, y1 = 5, x2 = 279, y2 = 5, W = 35, H = 20;  //buttons dimensions
int color1, color2, colorMsg;           //buttons & msg colors
String msg = "";

void setup() {  size(350, 60);  stroke(128);  textFont(createFont("Arial Bold", 14)); }

void draw() {
  background(0);
  // set drawing attributes based on answer
  if (answer == 1) {          // YES button is active
    color1 = color(0,255,0);    color2 = color(90,0,0);    colorMsg = color1;
    msg = "Congratulations! You can get a driving license";
  } else if (answer == 2) { // NO button is active
    color1 = color(0,90,0);    color2 = color(255,0,0);    colorMsg = color2;
    msg = "Sorry! You are too young to get a driving license";
  } else {                    // neither button is active
    color1 = color(0,90,0);    color2 = color(90,0,0);
    msg = "";
  }
  // Draw buttons and put text using above attributes
  fill(color1);    rect(x1,y1,W,H);
  fill(color2);    rect(x2,y2,W,H);
  fill(255,255,0); text("Are you older than 18 years old?    Yes     No  ", 10, 20);
  fill(colorMsg);  text(msg,10, 50);
}

void mouseReleased() {
  if (mouseX>x1 && mouseX<x1+W && mouseY>y1 && mouseY<y1+H)      answer = 1;
  else if (mouseX>x2 && mouseX<x2+W && mouseY>y2 && mouseY<y2+H) answer = 2;
  else                                                          answer = -1;
}
```

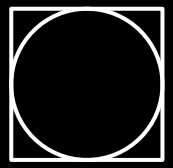
Live Class (Fridays)
Conditionals I

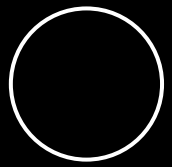
Making Decisions

What is the output of this code?

```
noFill();  rectMode(CENTER); stroke(255);

int num = 10;

if (num > 10)
  rect(50,50,50,50);
else
  ellipse(50,50,50,50);
```

A. 

C. 

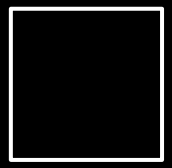
B. 

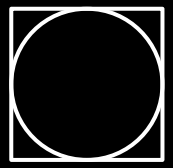
D. Something else

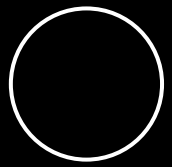
Making Decisions

What is the output of this code?

```
noFill();  rectMode(CENTER); stroke(255);

int num = 9;

if (num != 10)
  rect(50,50,50,50);
ellipse(50,50,50,50);
```

A. 

C. 

B. 

D. Something else

Testing Multiple conditions

What is the output?

```
int grade = 90;
if (grade > 50)       print("D");
else if (grade > 60) print("C");
else if (grade > 70) print("B");
else if (grade > 85) print("A");
else                  print("F");
```

A. A

B. DCBA

C. DCBAF

D. D

E. Something else

Question

Testing Multiple conditions

What is the output?

```
int grade = 90;
if (grade > 85)      print("A");
else if (grade > 70) print("B");
else if (grade > 60) print("C");
else if (grade > 50) print("D");
else                 print("F");
```

A. A

B. ABCDF

C. ABCD

D. F

E. Something else

Question

Boolean Expressions

Is **result** `true` or `false`?

```
int x = 10, y = 20;
boolean result = (x > 10) || (y < 20);
println(result);
```

A. `true`

B. `false`

Question

Boolean Expressions

Is **result** `true` or `false`?

```
int x = 10, y = 20;
boolean result = !(x != 10) && (y == 20);
println(result);
```

A. `true`

B. `false`

Question

Boolean Expressions

Is **result** `true` or `false`?

```
int x = 10, y = 20;
boolean result = (x >= y) || (y <= x);
println(result);
```

A. `true`

B. `false`

Making Decisions

What is the output of this code?

```
int num=12;
if (num >= 8)
   print("big");
  if (num == 10)
     print("ten");
else
   print("small");
```

A. big

B. small

C. bigsmall

D. ten

E. bigten

Reference Material
Conditionals II

Objectives

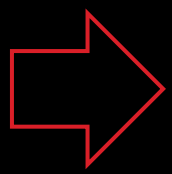
- This is a pre-class reading materials. After reading them, you should be able to:
  - Use the `switch` statement.
  - Compare Strings and objects.

The `Switch` Statement

Making Decisions: the `Switch` Statement

- There are cases where you want to compare a single value against many alternatives. One way of doing this is to use a multi-part *if* statement. Another ways is to use `switch`.

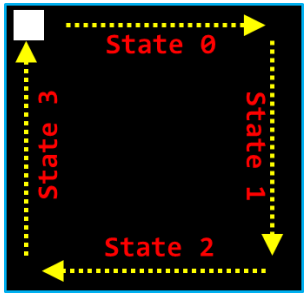
```
if (x == value1) {
    statements;
}
else if (x == value2) {
    statements;
}
...  //other else-if
else {
    statements;
}
```



```
switch (x) {
    case value1:
        statements;
        break;
    case value2:
        statements;
        break;
    ...  //other cases
    default:
        statements;
}
```

Complex Motion Paths Using if

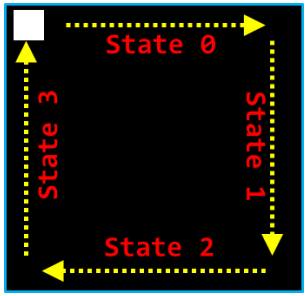
```
int x = 0, y = 0, speed=5, state=0;
void setup() { size(200, 200);   noStroke();  fill(255); }
void draw() {
  background(0);
  rect(x, y, 10, 10);

  if(state == 0)           // RIGHT state
  {   x += speed;
      if (x > width-10)    { x = width-10;  state = 1; }
  }
  else if(state == 1)      // DOWN state
  {   y = y + speed;
      if (y > height-10)   { y = height-10; state = 2; }
  }
  else if(state == 2)      // LEFT state
  {   x = x - speed;
      if (x < 0)           { x = 0;        state = 3; }
  }
  else if(state == 3)      // UP state
  {   y = y - speed;
      if (y < 0)           { y = 0;        state = 0; }
  }
}
```



Code by Shiffman

Complex Motion Paths Using switch

```
int x = 0, y = 0, speed=5, state=0;
void setup() { size(200, 200);   noStroke();  fill(255); }
void draw() {
  background(0);
  rect(x, y, 10, 10);
  switch(state) {
  case 0:                    // RIGHT state
      x += speed;
      if (x > width-10)    { x = width-10;  state = 1; }
    break;
  case 1:                    // DOWN state
      y = y + speed;
      if (y > height-10)   { y = height-10; state = 2; }
    break;
  case 2:                    // LEFT state
      x = x - speed;
      if (x < 0)           { x = 0;        state = 3; }
    break;
  case 3:                    // UP state
      y = y - speed;
      if (y < 0)           { y = 0;        state = 0; }
  }
}
```



Code by Shiffman

Switch Statement

```
int num = 2;

switch (num){
  case 1:  text("Sunday", 10, 10);    break;
  case 2:  text("Monday", 10, 10);    break;
  case 3:  text("Tuesday", 10, 10);   break;
  case 4:  text("Wednesday", 10,10);  break;
  case 5:  text("Thursday", 10, 10);  break;
  case 6:  text("Friday", 10, 10);    break;
  case 7:  text("Saturday", 10, 10);  break;
  default: text("Invalid day!",10,10);break;
}
```

Output: Monday

Ordering the Cases

In most situations, Order doesn't matter!

```
int num = 2;

switch (num){
  case 3:  text("Tuesday", 10, 10);   break;
  case 1:  text("Sunday", 10, 10);    break;
  case 4:  text("Wednesday", 10,10);  break;
  case 6:  text("Friday", 10, 10);    break;
  default: text("Invalid day!",10,10);break;
  case 2:  text("Monday", 10, 10);    break;
  case 7:  text("Saturday", 10, 10);  break;
  case 5:  text("Thursday", 10, 10);  break;
}
```

Output: Monday

The use of break

- Each ***case*** usually ends with a `break` statement which means: "exit the switch block right after finishing this case".
  - Execution of each case continues until the **`break`** statement. If you don't put a break statement, the next case will be executed.

```
int x = 0;
switch (x){
  case 0:  print("0"); break;
  case 1:  print("1"); break;
  default: print("X"); break;
}
```

Output: 0

```
int x = 0;
switch (x){
  case 0:  print("0");
  case 1:  print("1");
  default: print("X");
}
```

Output: 01X

Limitations

- You can use switch statement ***only when***:

  1) You are comparing values for equality
     - Can't compare using >, <, <=, etc

  2) The values are ***integers*** or ***Strings***.

Comparing Strings and Objects

Comparing Strings and Objects

- Comparing strings and objects is different than numbers.
  - Operators such as <, > are not useful for strings and objects.
  - Operator "==" can be used but it is not very useful.
    - The "==" operator compares if two string/object references refer to the same object NOT if the string/object has the same value.
      - We will discuss more about this later

- To compare strings, use the equals() method:

```
String st1 = "Hello", st2="He", st3="Test", st4 ="He";

println(st1.equals(st2));              // false
println(st2.equals(st4));              // true

st4 = st4.toUpperCase();               // st4 is now HE
st2 = st2.toLowerCase();               // st2 is now he
println(st2.equals(st4));              // false
println(st2.equalsIgnoreCase(st4));    // true
```

Live Class (Fridays)
Conditionals II

Question

Switch Statement

What is the output of this code?

```
int num=2;
switch (num){
  case 1: print("one");       break;
  case 3: print("three");     break;
  case 2: print("two");       break;
  default:print("other");     break;
}
```

A. one

B. two

C. three

D. other

Switch Statement

What is the output of this code?

```
int num=2;
switch (num){
  case 1: print("one");
  case 2: print("two");
  case 3: print("three");     break;
  default:print("other");
}
```

A. one

B. two

C. twothree

D. onetwothree

E. other

Question

Switch Statement

What is the output of this code?

```
int num=2;
switch (num){
  case 1: print("one");
  case 3: print("three");     break;
  case 2: print("two");
  default:print("other");     break;
}
```

A. three

B. two

C. twothree

D. twoother

E. other

Question

String Comparisons

What is the output of this code?

```
String str1 = new String("abc");
String str2 = new String("abc");

print(str1.equals(str2));
```

This is another way of creating strings that ensures new string objects have their own memory space.

A. true

B. false

Question

String Comparisons

What is the output of this code?

```
String str1 = new String("abc");
String str2 = new String("abc");

print(str1 == str2);
```

A. true

B. false

Physics 101

Physics 101

- We know from before that we update the location of an object every frame with its speed:

```
y = y + speedY
```

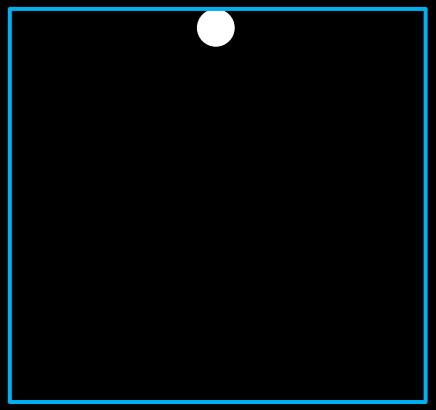
- Now, let's try to code "***gravity***". Gravity is the *rate of change of the speed* (i.e. the acceleration). This means, to code gravity we need to ***also change the speed every frame*** by some amount

```
speedY = speedY + gravity;      // e.g. gravity = 0.1
```

- Now let's see how the results look like with and without gravity

Bouncing Ball with Gravity (version 1)

```
float x = 150, y = 10, r = 15;  // location of ball
float speedY = 0;               // speed of ball
float gravity = 0.2;
void setup() {
  size(300, 300);
}
void draw() {
  background(0);
  ellipse(x, y, 2*r, 2*r);
  // update ball's speed and location
  speedY += gravity;
  y += speedY;
  // reverse speed when ball hits the bottom
  if (y > height-r){
     y = height-r;      // ball can't penetrate ground
    speedY = -speedY;  //not so natural looking!!
}}
```



comment this out to see how it looks without gravity

Based on Shiffman

Physics 101, cont'd

- The motion in the previous example doesn't look very natural.
- When the ball bounces off the ground there will be some loss of energy (speed) – this is known as ***dampening effect***.
- This means, we need to replace:

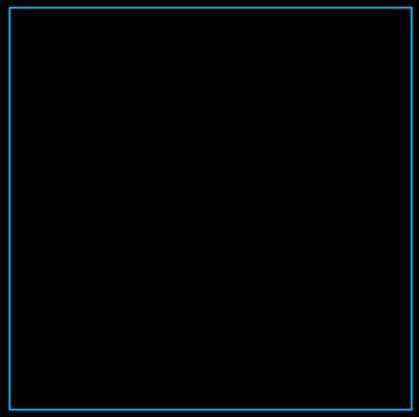
```
Speed = -speed
```

With

```
speed = -0.9 * speed; // or another dampening factor
```

Bouncing Ball with Gravity (version 2)

```
float x = 150, y = 10, r = 15;  // ball location,size
float speedY = 0;               // speed of ball
float gravity = 0.2;
void setup() {
  size(300, 300);
}
void draw() {
  background(0);
  ellipse(x, y, 2*r, 2*r);
  // update ball's speed and location
  speedY += gravity;
  y += speedY;
  // reverse speed when ball hits the bottom
  if (y > height-r){
     y = height-r;     // ball can't penetrate ground
    speedY = - 0.9 * speedY; //natural looking
}}
```

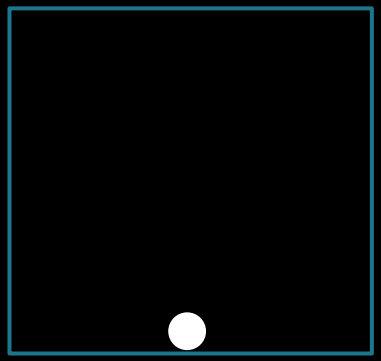


Jumping Ball

In this example, pressing *space bar* causes the ball to jump.

The boolean variable `isJumping`

- used to track the state of the ball. The space bar sets this variable to true. The variable is set back to false when the ball lands on floor.
- The jumping action only happens in the draw() method when `isJumping` is true.



```
float x=150,y=285,speedY=0,gravity=0.15;
boolean isJumping = false; int r = 15;
void setup() { size(300, 300);}

void draw() {
  background(0);
  ellipse(x, y, 2*r, 2*r);
  if (isJumping) {
    speedY += gravity;
    y += speedY;
    if (y >= height-r) { //ball lands on floor
      speedY = 0; y = height-r;
      isJumping = false;
    }
  }
}

void keyPressed() {
  if (key == ' ' && !isJumping) {
    isJumping = true;
    speedY = -6;  //go up
  }
}
```

Jumping Ball Step-by-Step

```
float r=20, x=200, y =400-r;
float speedY=-6, gravity=0.2;

void setup(){
  size(400,400);
}
void draw(){
  background(0);
  ellipse(x,y,2*r,2*r);

  speedY += gravity;
  y += speedY;

}
```

Initial code – ball is thrown upwards from bottom edge

We need to stop it once it hits the ground again

```
float r=20, x=200, y =400-r;
float speedY=-6, gravity=0.2;
boolean jumping = true;
void setup(){
  size(400,400);
}
void draw(){
  background(0);
  ellipse(x,y,2*r,2*r);
  if(jumping){
    speedY += gravity;
    y += speedY;
    if(y>height) jumping=false;
  }
}
```

Now ball stops at bottom edge

We need to add user interaction (jump only when spacebar is pressed)

```
float r=20, x=200, y =400-r;
float speedY = 0, gravity=0.2;
boolean jumping = true;
void setup(){
  size(400,400);
}
void draw(){
  background(0);
  ellipse(x,y,2*r,2*r);
  if(jumping){
    speedY += gravity;
    y += speedY;
    if(y>height) jumping=false;
  }
}
void keyPressed(){
  if(key == ' '){
    jumping = true;
    speedY = -6;
  }
}
```

Done!

Note: this code is a bit different from previous slide – it allows for multi-jumping (see condition in keyPressed)

Aside: Android Mode

- If you are interested in converting your animation/game to an Android app, do the following:

1) Save your sketch (give it a meaningful name as this will be your app name).

2) Switch to Android mode from your PDE (top-right corner)

3) If asked, accept installing “Android Mode” and “SDK”

4) Connect your Android device using a USB cable –allow “USB Debugging” if asked on your phone.

5) Run your sketch (press Run button) – after a few seconds, your sketch will appear as an app on your phone.

Aside: Android Mode – try it yourself

- Here is the same code we saw before after modifying it to work on fullscreen.

```
float r,x,y;
float speedY=0, speed, gravity;
boolean jumping = true;

void setup(){
  fullScreen();
  r = width/20;
  x = width/2;
  y = height-r;
  speed = -height/30;
  gravity = height/1000;
}
```

```
void draw(){
  background(0);
  ellipse(x,y,2*r,2*r);
  if(jumping){
    speedY += gravity;
    y += speedY;
    if(y>height-r){
      jumping = false;
      y = height-r;
    }
  }
}
void mousePressed(){
    jumping = true;
    speedY = speed;
}
```

Aside: Android Mode – Choosing App ICON

If you want to change the app icon, design/download an icon of your choice and put it under the sketch folder (no the data folder). Your icon's name should be

icon-48.png.

You can generate other sizes and save them in the same folder to be used if your phone launcher needs to use other icon sizes. Your icon files should be named:

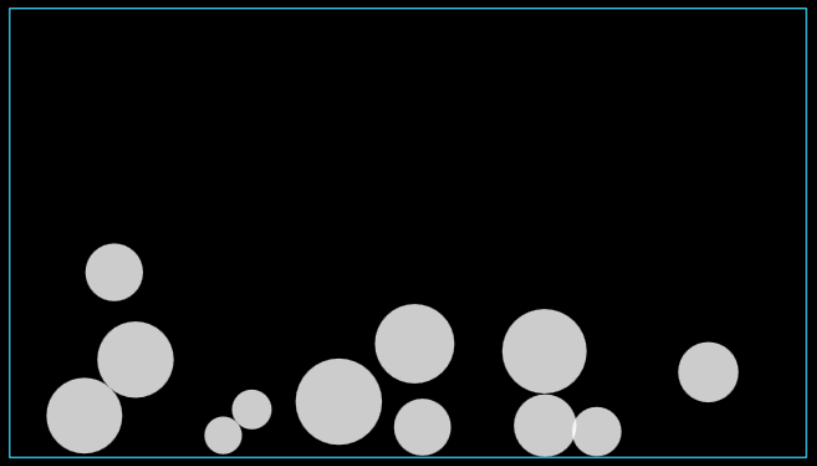
icon-36.png
icon-48.png
icon-72.png
icon-96.png
icon-180.png
icon-192.png

You can find free icon files only e.g. iconfinder.com

Aside: Advanced Physics

- We will not do advanced physics in this course, but if you wish to learn more, use Shiffman's book: The Nature of Code.

- (***Optional***) Here is another advanced example of several bouncing balls which also collide.
  - https://processing.org/examples/bouncybubbles.html



Objectives

- Define: Boolean, condition
- List and use the comparison operators.
- Construct and evaluate Boolean expressions using AND, OR, NOT.
- Write and use decisions using the if/else and switch statements
- Use conditionals with common animation themes such as bouncing attributes, complex paths, buttons, etc.
- Use conditionals to react to user response to our questions (e.g. Yes/No questions)
- Explain the dangling else problem.
- Explain how to compare Strings/Objects and how why cannot use == with them.

Live Class (Fridays)
Intro to Images

Computer Creativity

Pre-Class Reading

Images

Slides courtesy of Dr. Abdallah Mohamed.

Objectives

- This is a pre-class reading materials. After reading them, you should be able to:
  - Draw an image on the sketch (static or active modes)
    - New type: `PImage`
    - New functions: `loadImage()`, `image()`, `imageMode()`
  - Use an image as the sketch background.
  - Read the image `width`, `height`
  - Resize an image
    - Either using width and height
    - Or by defining the upper-left and lower-right corners.
  - Change the tint and opacity of an image.
  - Animating an image (e.g. position, size, opacity, etc.)
  - Drawing only part of the image

How to Draw Images on Your Sketch?

- You can add an image to your sketch and change its ***position***, ***size***, ***color*** and ***opacity***.
- To do so, you need to do the following:

1) Put the image in the "data" folder of the current sketch.
   - You can use several image formats including: **gif**, **png**, **jpg, tga, bmp**

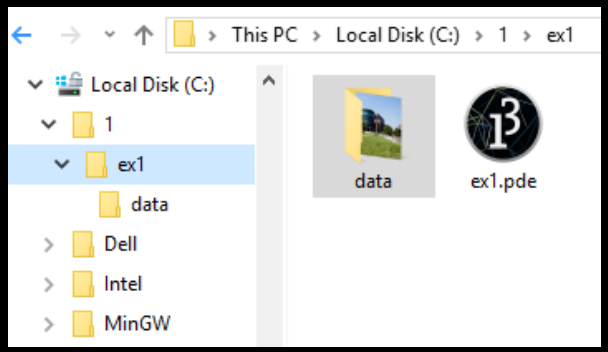
2) Declare a variable of the type `PImage`.
   - `PImage` is a Processing data type that can store image information.

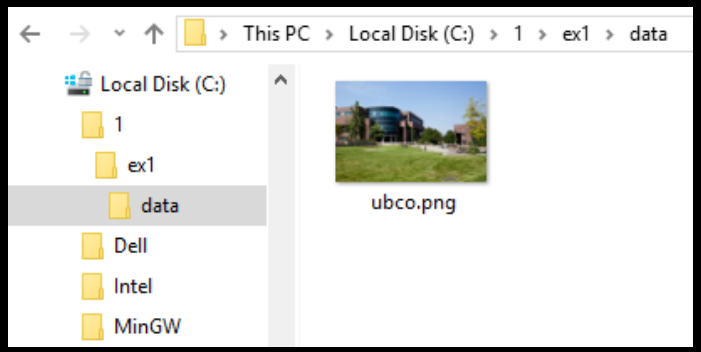
3) Load the image into your variable using `loadImage(filename)`.

4) Draw the image on your sketch using `image()`.

Drawing an Image in Static Program

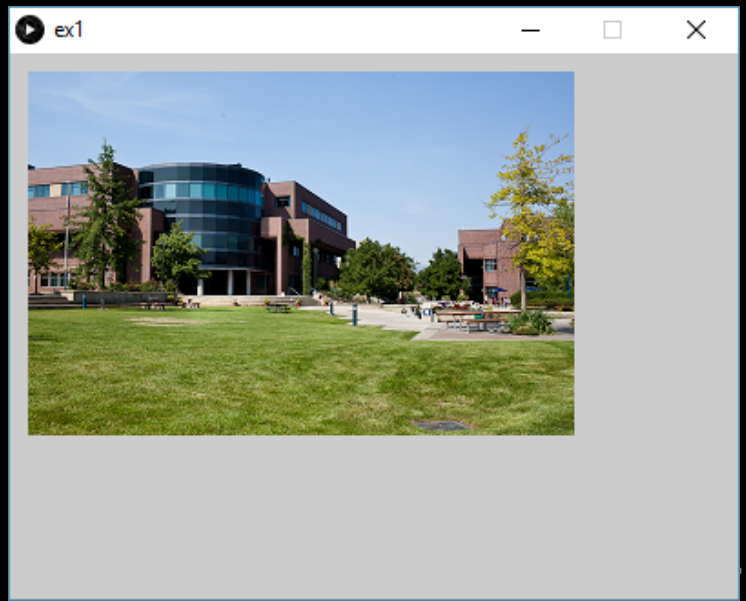
Step 1: put image, e.g. ubco.png, in the data folder of the current sketch






```
size(400,300);

//Step 2,3
PImage img = loadImage("ubco.png");

//Step 4
image(img,10,10);
```

top-left corner at (10,10)

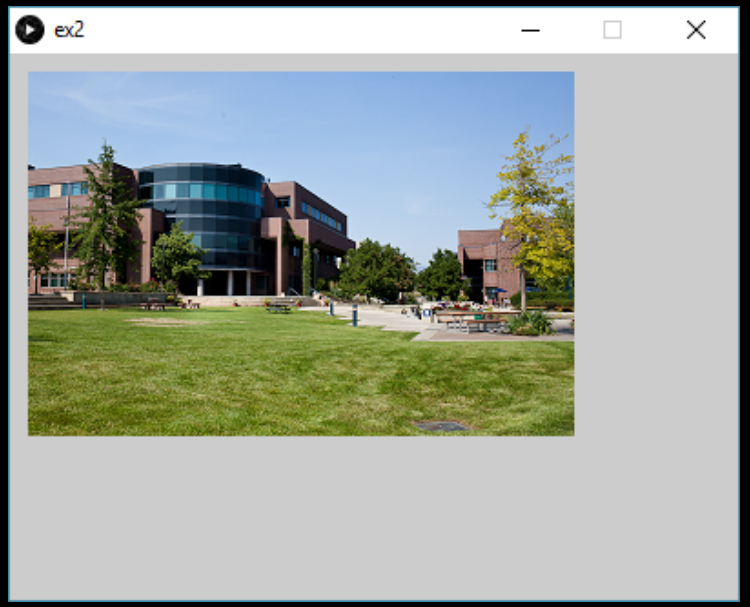


Drawing an Image in Active Program

If you draw an image in Active mode (i.e. you're using draw()), you should:

- Declare your `Pimage` variable as **global**
- **load** the image from within `setup()`, ***not*** from draw().
  - ***Can you explain why?***
- Draw your image in `draw()`

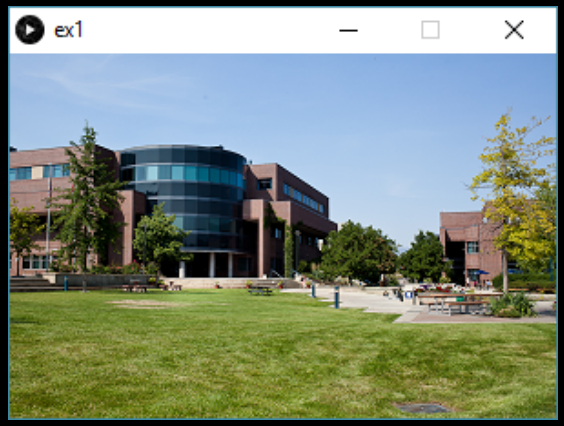
```
PImage img;

void setup(){
    size(400,300);
    img = loadImage("ubco.png");
}

void draw(){
    image(img,10,10);
}
```



Using an Image as Background

- You can use the image as the sketch background using `background(img)`.
  - Restriction: the **sketch size must match the image size**.

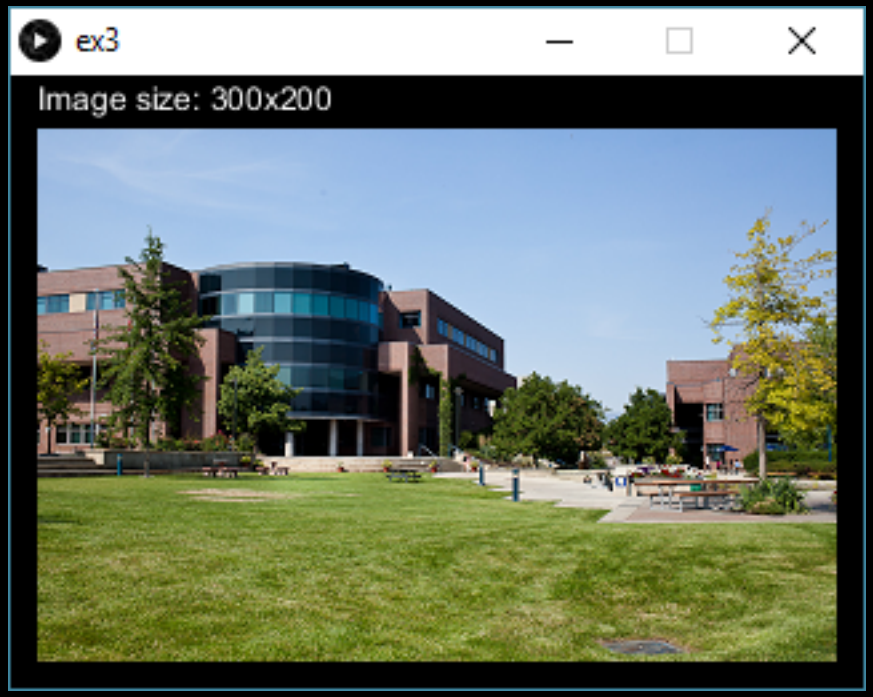
```
PImage img;

void setup(){
    size(300,200);
    img = loadImage("ubco.png");
}

void draw(){
    background(img);
}
```



Getting the Image Width and Height

- You can read the image width and height using **width** and **height** attributes of the **PImage** variable.

```
size(320,230);
background(0);
PImage img = loadImage("ubco.png");
int w = img.width;
int h = img.height;
image(img,10,20);
text("Image size: "+w+"x"+h,10,13);
```



The `image()` function

- The `image()` function is used to ***draw*** and ***resize*** an image stored in a `PImage` variable at any ***position*** on the sketch:

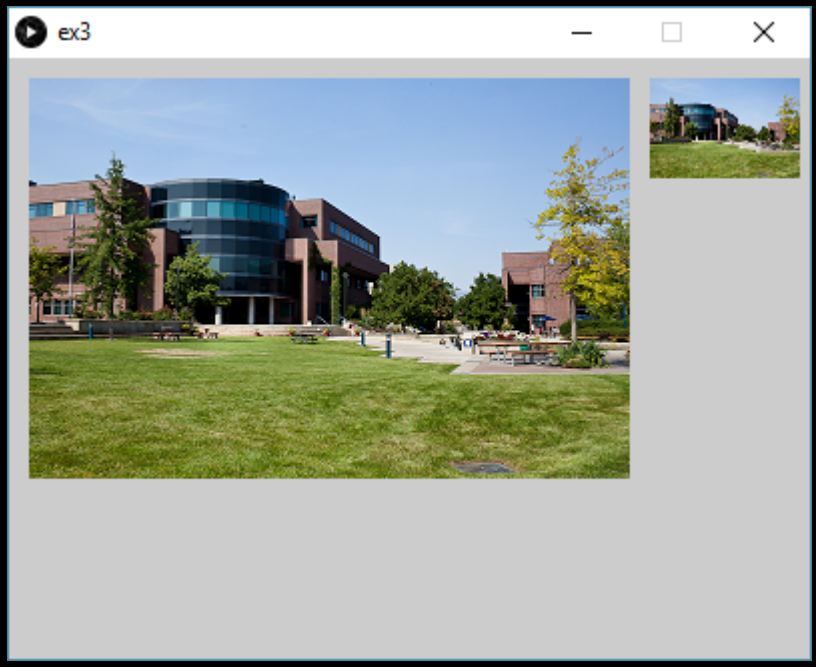
```
image(img, x, y [, w, h])
```

- Where:
  - `img` is a *`PImage`* variable
  - `x` and `y` : position of the image on the sketch.
  - `w` and `h` : ***optional*** width and height for ***resizing the image***.

***Note**: if `imageMode(CORNERS)` is used (discussed later), (w,h) would represent the lower-right corner of the image, which could also be used to resize the image.*

Resizing an Image

```
size(400,300);
PImage img = loadImage("ubco.png");

// original size.
// top-left corner at (10,10)
image(img,10,10);

// quarter size
// top-left corner at (230,10)
int w = img.width;
int h = img.height;
image(img, w+20, 10, w/4, h/4);
```



Notes

- If the image is not in the data folder, then you must use **the absolute path** of the image, e.g., loadImage("c:/my folder/image.gif“).

- Two ways to add images to the data folder of your sketch:
  - ***Without PDE***: create a folder called data, then copy your images to that folder using any file manager program (e.g. Windows Explorer)
  - ***Using PDE***: `Sketch` menu `-> Add File`.

- Sequence of images
  - You can display a sequence of images (as an animated image) by loading a series of images into an array, then drawing them in successive frames (by incrementing the array index).
    - We’ll discuss more about this later!

The `imageMode()` function

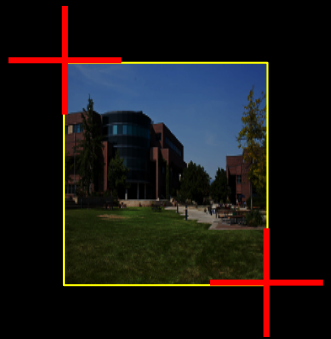
- *Similarly* to `rectMode()` and `ellipseMode()`, **`imageMode()`** is used to set the reference point from which the image is drawn.

- By default, CORNER is used. This can be changed as follows:

image(img,x,y);

imageMode(CORNER);
image(img, x, y);

imageMode(CENTER);
image(img, x, y);



imageMode(CORNERS);
image(img,x1,y1,x2,y2);

This one may change aspect ratio

Tint and Opacity

- `tint()` function
  - used to ***color*** an image and set its ***opacity***.
    - `tint()` for images **is similar to `fill()` for shapes**.
  - takes color parameter(s) first, then optionally an additional parameter for opacity.
- `noTint()` function removes current tint values.

- Examples:
  - `tint(255);` use original colors (***same as `noTint()`***)
  - `tint(128);` darken image
  - `tint(255,0,0);` Red tint (if RGB color mode is used)
  - `tint(255,192);` 75% opacity (image is semi transparent).
  - `tint(255,0,0,64)` Red tint with 25% opacity

Using tint() and noTint() functions

```
background(80); size(700,400);
PImage img = loadImage("c:/1/ubco.png");

image(img,0,0); //original image

tint(100);      //darken the image
image(img,100,50);

noTint();//remove tint - same as tint(255)
image(img,200,100);

tint(255,0,0);  //red tent
image(img,300,150);

tint(255,196);  // 75% opacity
image(img,400,200);
```

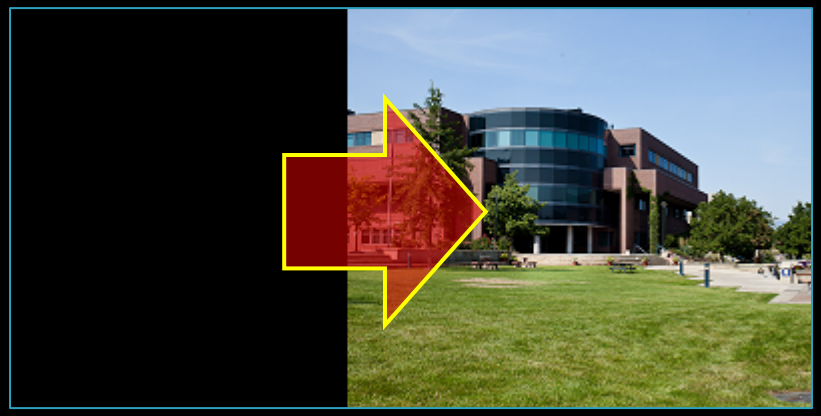
Notes

- The current color mode, e.g. RGB or HSB, is used by the `tint()` function.
  - Remember: RGB mode is used by default.

- GIF and PNG retain their alpha channel – that is, if there are transparent areas in a GIF or PNG image, they will remain transparent on your sketch.

Animating Image Position

You can animate the position of the image using the same techniques we discussed before (use variables for position and update them every frame)



```
PImage img;
float  x, speedX = 1;

void setup() {
  size(400, 200);
  img = loadImage("ubco.png");
  x = -img.width; //initial location on left
}

void draw() {
  background(0);
  image(img, x, 0);
  x = x + speedX;
}
```

The `image()` function Again!

- You can use the image() function to ***crop*** the image (i.e. draw part of the image)

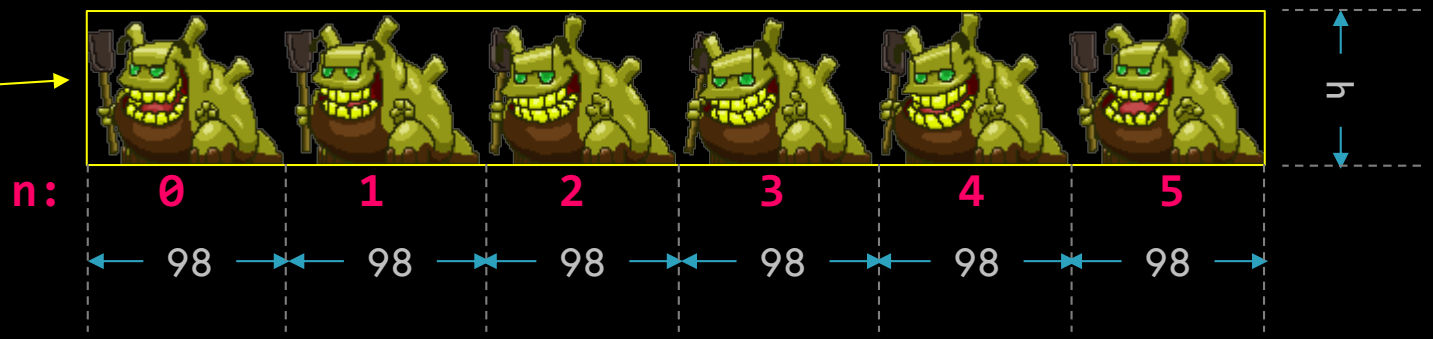
```
image(img, x, y, w, h, x1, y1, x2, y2)
```

- Where:
  - `x`, `y`, `w`, `h`: are the coordinates and size on the sketch (as above)
  - `x1`, `y1`, `x2`, `y2`: are the top-left and bottom-right corners of the part of the image that you want to draw

Showing Part of an Image

- This code displays part of the image strip below that shows 6 frames of the character animation. Each frame is 98x76 pixels. The integer **n** is used to determine which frame to display.

```
Image img = loadImage("s_crapmunch_idle_strip6.png");
int h = img.height;

int n = 1;        //get second sprite (pose)
image(img, 0, 0, 98, h, 98*n, 0, 98*(n+1), h);
```



Output:

This is frame 1 in the image above

Summary of the use of image()

```
PImage img = loadImage("myImage.jpg");

//top-left corner at (x,y)
image(img, x, y);

//top-left corner at (x,y), resized to w x h
image(img, x, y, w, h);

//area identified by (x1,y1)to(x2,y2) from image is
//displayed at (x,y) in the sketch and resized to w x h
image(img, x, y, w, h, x1, y1, x2, y2);
```

Where to Get Images?

- Of course you can create and use your own images. However, if you decide to use online images, make sure you read and understand their ***copyrights***.
- Here are some online sources with good images, many of them are free to use (*read the copyrights on each site*):
  - Game Art
    - kenney.nl/assets?q=2d
    - opengameart.org
    - www.gameart2d.com/freebies.html
    - opengamegraphics.com
    - This article refers to several websites with great free game art.
  - Others
    - openclipart.org
    - pixabay.com
    - search.creativecommons.org
    - Google Images (has the option to search for royalty-free images only)

Draw a Game Platform

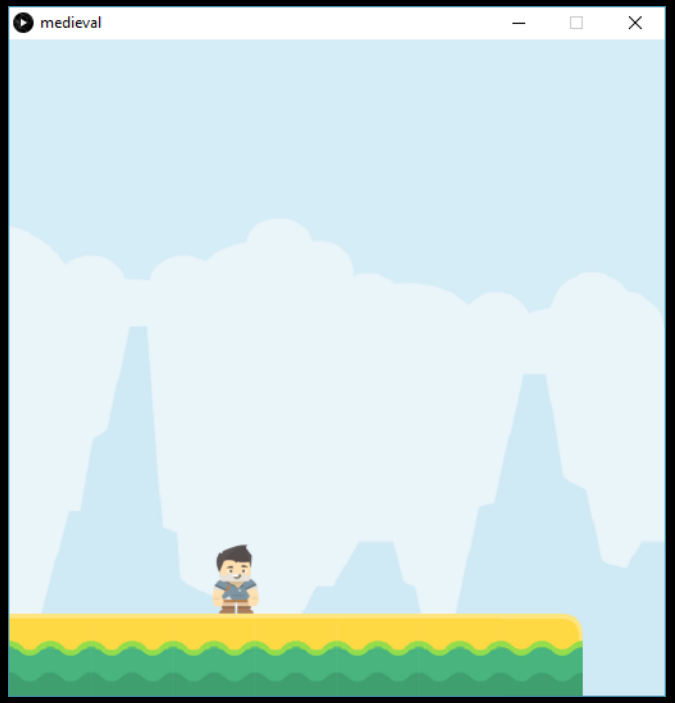
In this exercise, we will build a game platform using a few tiles that can be used as "Lego pieces".

tiles from kenney.nl

- All tiles have the same width & height (64 x 64)
- Tiles need to be loaded into separate variables (e.g. img1, img2, …), and then placed one by one on the sketch

Steps:

1) Download the starter code.
   Unzip and open in Processing.
2) Write the missing code as per the instructions in starter code.
   The output should be similar to →



+3 points

Move Your Player

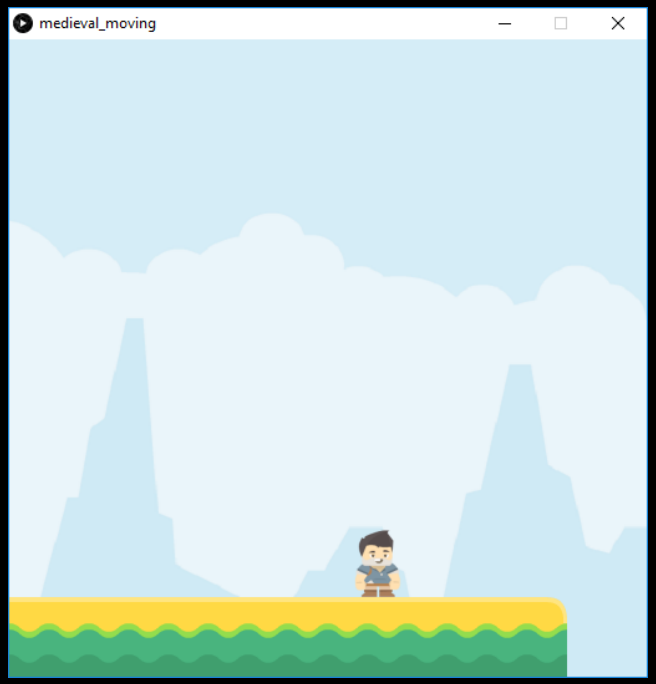
Previously, you created the game platform. For today, do the following:

1) Use your solution from the previous exercise .

2) Update the code so that the player ***moves left or right with the arrow keys***.

The player should be moving as long as an arrow key is pressed and should stop moving if the key is released.

Hint: use *IDEA2*



Tiles from kenney.nl

Grading:

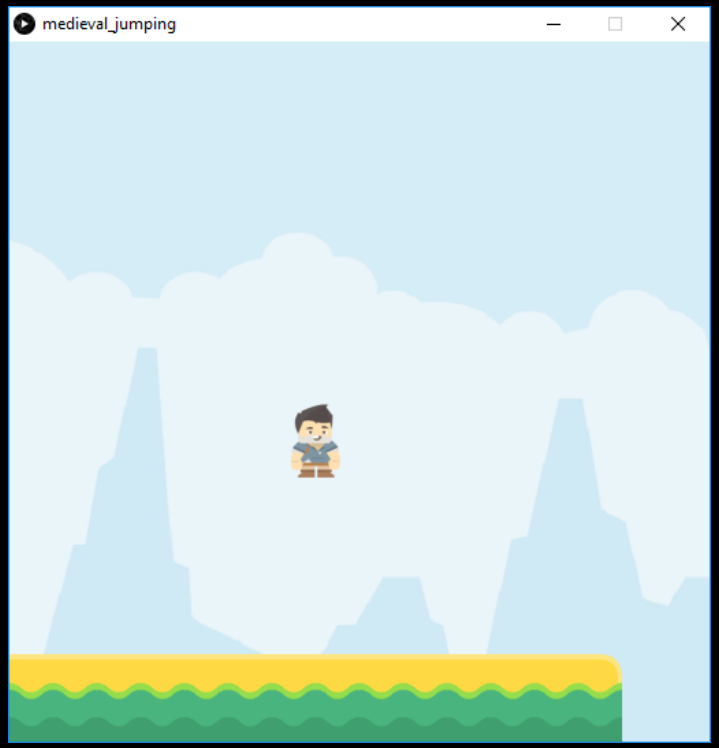
+2 moving the character as indicated
+1 bonus for 'protecting' character from falling off the cliff (i.e. stop by cliff even if → is pressed)

Jump.. Jump.. High Up in the Sky!!

Previously, you created the game platform and added code to ***move the player left or right with the arrow keys***.

1) Open your solution from the previous exercise .

2) Add more code to make your player jump when SPACE is pressed. Note that a player cannot jump if already jumping



Tiles from kenney.nl

Marking: +2 points