Computer Creativity

Randomness and Images

Slides courtesy of Dr. Abdallah Mohamed.

Announcements

Useful Functions

Slides courtesy of Dr. Abdallah Mohamed.

Key Points

1) Three range functions:
   - `map()` map's a value from [range1] → [range2]
   - `norm()` map's a value from [range] → [0,1]
   - `constrain()` forces a value to stay within a [range]

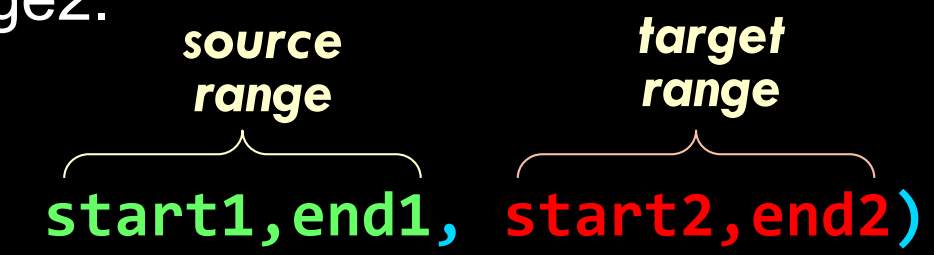
2) Two random functions:
   - `random()` generates a random value within a [range]
   - `noise()` returns the Perlin noise value.

3) Value conversion
   - `int()` converts a float to int
   - `float()` converts an int to float

Range functions

The `map()` function

- The `map()` function maps a given number from a given range1 to another target range2.
- Syntax:



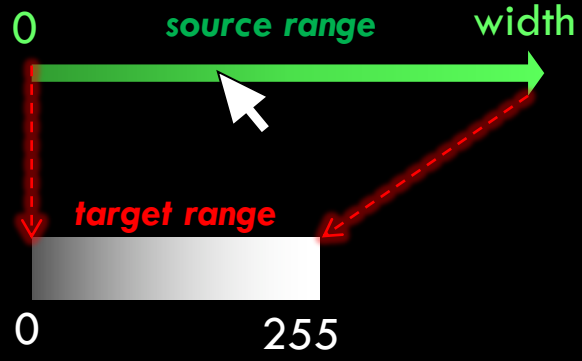
  - `value`: given number to be mapped
  - `start1,end1`: the lower and upper bounds of the source range.
  - `start2,end2`: the lower and upper bounds of the target range.

```
float value = map(25, 0,50, 100,200);
print(value);  // output 150
```

```
print(map(0,  0,50, 100,200));  //100
print(map(25, 0,50, 100,200));  //150
print(map(50, 0,50, 100,200));  //200
```

Mouse Controlling the Color (2)

- In this example, the color of a square changes from black to white as the mouse moves from left to right (i.e. based on mouseX).
  - When mouseX = 0 → color is black
  - When mouseX = width → color is white
- This means, the range of mouseX [0,width] should be mapped to the proper range of color which is [0,255]
  - `map()` can be used to map `mouseX` to the range from 0 to 255 suitable for the color range.



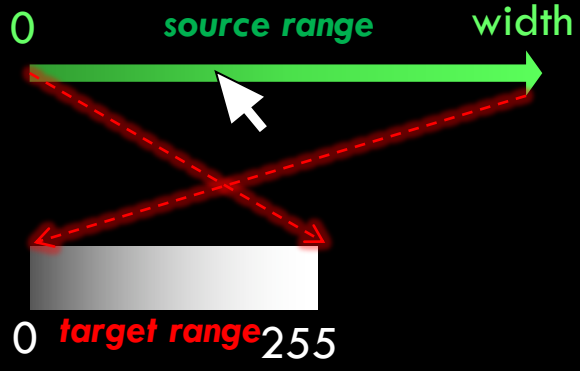

```
void draw() {
  background(0);
  float c = map(mouseX, 0, width, 0, 255);
  fill(c);
  rect(10, 10, 80, 80);
}
```

source range

target range

Mouse Controlling the Color

- This example is the same as the previous one, except that the mapping is reversed:
  - mouseX = 0 → white
  - mouseX = width → black
- This means, the range of mouseX [0,width] should be mapped to the proper range of color which is [255,0]

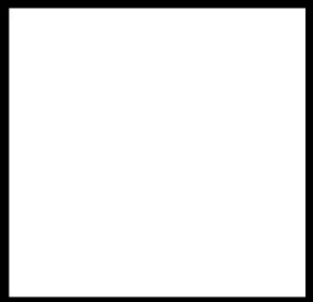



```
void draw() {
  background(0);
  float c = map(mouseX, 0, width, 255, 0);
  fill(c);
  rect(10, 10, 80, 80);
}
```

Mouse Controlling Size and Color of 2 Square

```
void setup() {
  size(640, 360);
  noStroke();
  rectMode(CENTER);
}
void draw() {
  background(0);
  translate(width/2,height/2);

  float c1 = map(mouseX, 0, width, 0, 255);
  float c2 = 255-c1;

  fill(c1);
  rect(c1/2, 0, c1, c1);
  fill(c2);
  rect(-c2/2, 0, c2, c2);
}
```

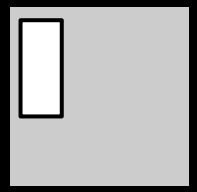
- This example reads `mouseX` and uses it to change the color, size, and location of two squares.
- `map()` is used to map `mouseX` to the range from 0 to 255 suitable for the color range.

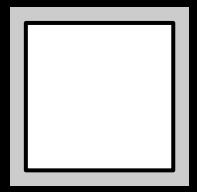


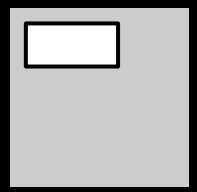
Example source: processing.org/examples/mouse1d.html

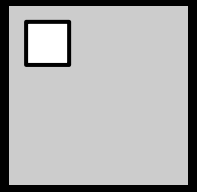
map()

what is the output of this code?

A. 

B. 

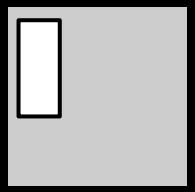
C. 

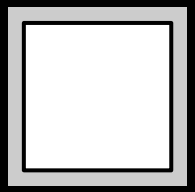
D. 

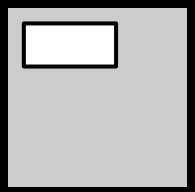
```
size(100,100);

float w = map(10, 0,10, 0,80);

float h = map(50, 0,50, 0,80);

rect(10,10,w,h);
```

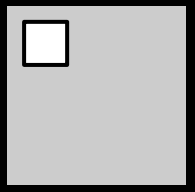
map()

what is the output of this code?

A. 

B. 

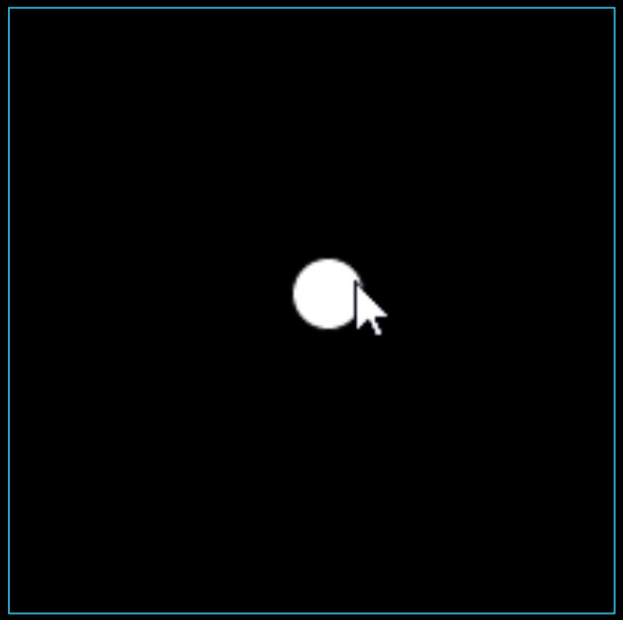
C. 

D. 

```
size(100,100);

float w = map(10, 0,10, 0,80);

float h = map(10, 10,50, 80,0);

rect(10,10,w,h);
```

Using the map() function

- Using the map() function, draw a circle that follows the mouse cursor "*to some extent*" as in the shown animation.
- To do this, you need to map:
  - mouseX to from range (0 to window width) to range (25% to 75% of the width), and
  - mouseY from range (0 to window height) to the range (25% to 75% of the height).



The `norm()` function

- The `norm()` function normalizes a given number from the current range of the number to 0…1.
  - This is the same as **map(value, low, high, 0, 1)**.
- Syntax: `norm(value, start1, end1)`
  - `value`: given number to be normalized
  - `start1,end1`: the lower and upper bounds of the current range.
- Note: `norm(v, a, b)` is equivalent to `map(v, a, b, 0, 1)`

```
float value = 25;
//both value2 and value3 have same value
float value2 = norm(value, 0, 50);
float value3 = map(value, 0, 50, 0, 1);
println(value2); // output is "0.5"
println(value3); // output is "0.5"
```

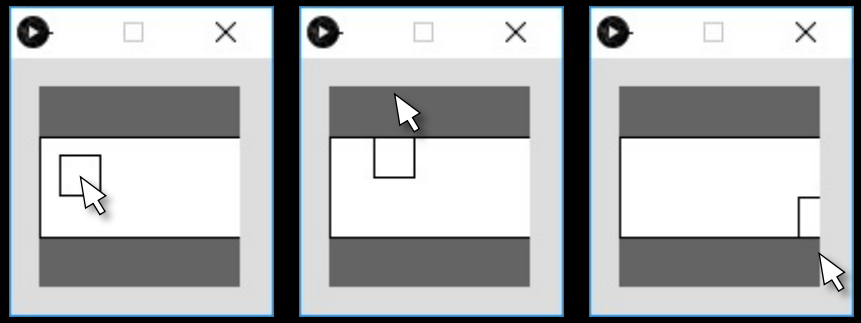
The `constrain()` function

- The `constrain()` function forces a value within a specific range [low,high].
- Syntax: `constrain(value, low, high)`
  - Example 1:

```
print(constrain(10, 50, 100));  // prints 50
print(constrain(60, 50, 100));  // prints 60
print(constrain(130, 50, 100)); // prints 100
```

  - Example 2: constraining a circle within a specific area

```
void draw(){
  background(100); rectMode(CENTER);
  rect(50,50,100,50);
  int x = mouseX;
  int y = constrain(mouseY, 35, 65);
  rect(x,y,20,20);
}
```



Use of constrain()

We want background to gradually change from black to white. Which code is better?

(1)

```
int shade = 0;
void draw(){
  background(shade);
  shade = shade+1;
}
```

(2)

```
int shade = 0;
void draw(){
  background(shade);
  shade = constrain(shade+1,0,255);
}
```

A. (1) is better than (2)

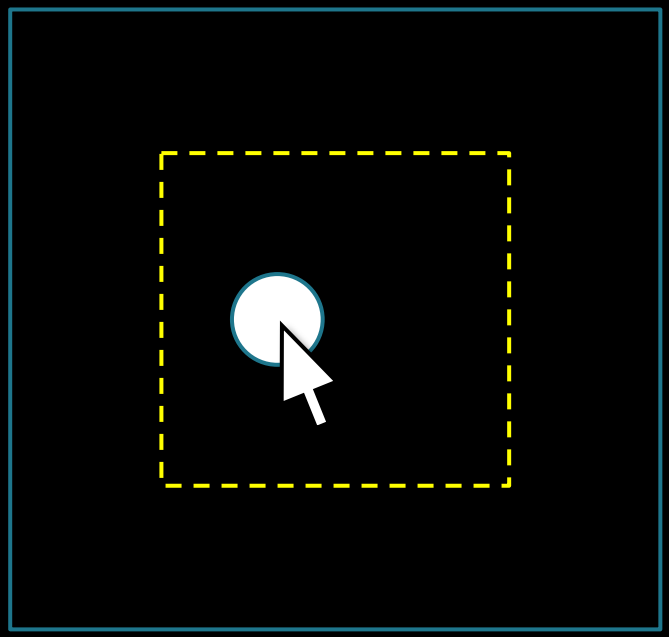
B. (2) is better than (1)

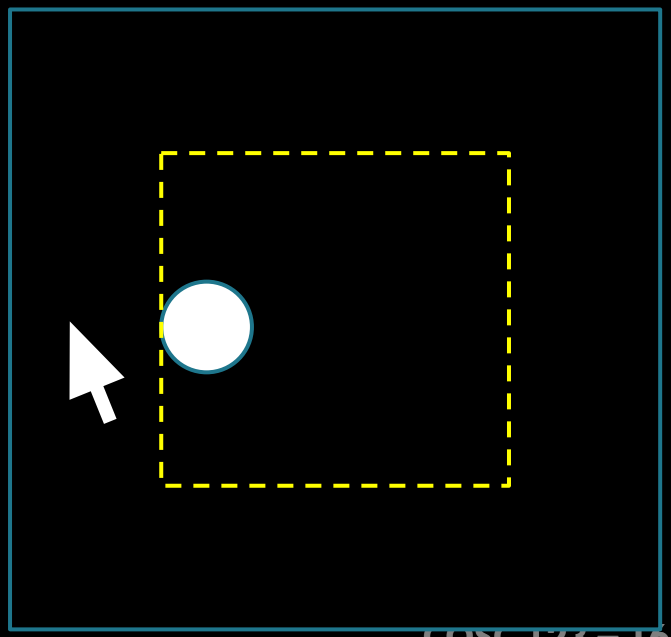
C. They are both the same

D. I don't understand what you are talking about.

Using the constrain() function

- Repeat ***Exercise 1*** but with the ball exactly following the mouse cursor as long as the mouse is within the dotted box (below). The ball cannot follow the mouse outside the box.
- To do this, replace the map() function with a constrain() function.
- (***you don't have to draw the dotted box***)





Key Points

1) Three range functions:
   - `map()` map's a value from [range] → [range2]
   - `norm()` map's a value from [range] → [0,1]
   - `constrain()` forces a value in a [range]

2) Two random functions:
   - `random()` generates a random value within a [range]
   - `noise()` returns the Perlin noise value.

3) Value conversion
   - `int()` converts a float to int
   - `float()` converts an int to float

Can we do fine without these functions?

- map()

- constrain()

Adding Randomness to Your Sketch

The `random()` function

- You can use the random() function to generate a random number to use in your sketch.
  - `random(high)` returns a random float in range [0, `high`[.
  - `random(low,high)` returns a random float in range [`low`, `high`[

- Note that you can convert a float to an integer using the `int()` function
  - `i.e. int(random(high)) returns an integer in the range from 0 to high`
  - `int(3.5)` is 3

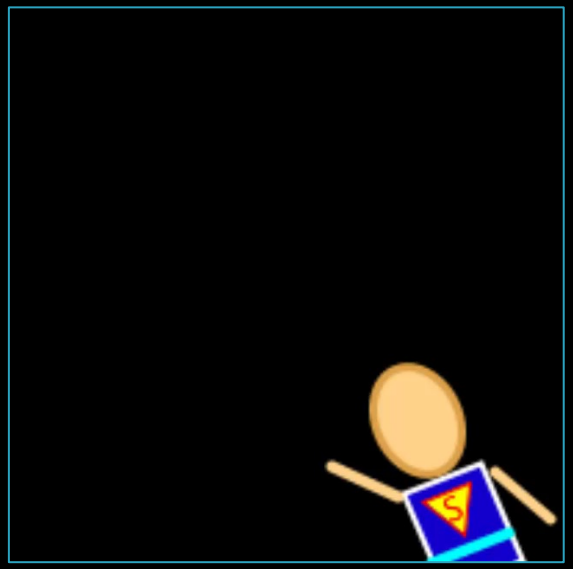
Example 2

Randomness (Source: Shiffman textbook)

```
float r, g, b, a, x, y, diam;
void setup() {
  size(300, 300);
  background(0);
  noStroke();
}

void draw() {
  // Fill all variables with random values
  r = random(255);   g = random(255);
  b = random(255);   a = random(255);
  diam = random(30);
  x = random(width); y = random(height);

  // Use values to draw an ellipse
  fill(r, g, b, a);
  ellipse(x, y, diam, diam);
}
```

Is it a bird? Is it a plane? No it is…!

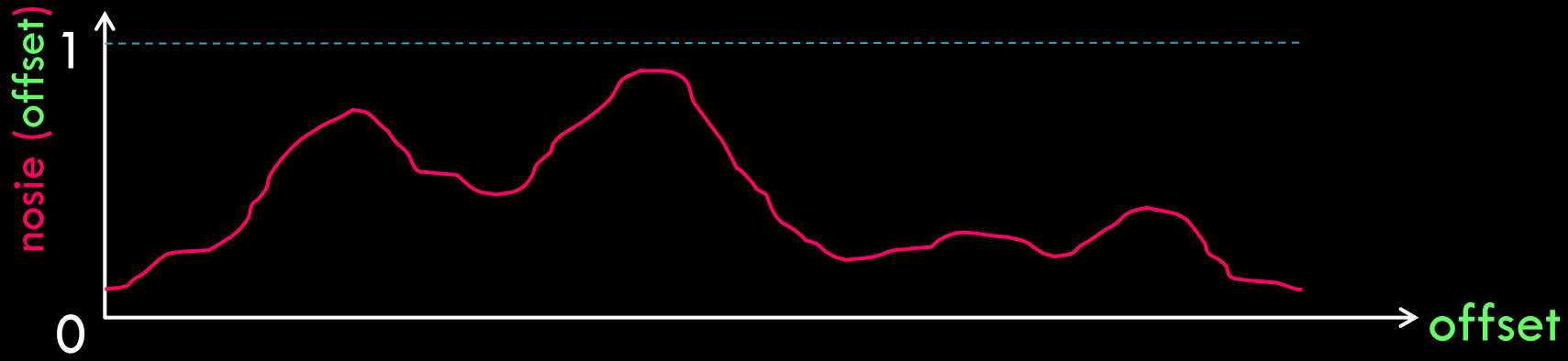
- Make your character fly from the bottom right corner of the screen to the top left corner as the animation advances. At the same time, make the limbs shake to give the illusion of fast flying.



- ***The Idea***:
  - use two variables x,y to control the location of the character. Decrement both every frame.
  - Add some randomness to the position of lines representing the limbs.
  - Rotate the scene using coordinate transformation.

Slides for your reference (fun things)

The `noise()` function



- Takes one input (for 1D noise) and returns a value in range [0,1].
- If input is a value that is slightly incremented over subsequent frames, output follows a Perlin curve (similar to the above).
- If **same input** is given more than once, **same output is returned**.
- Different Perlin curves are used for different program executions

```
println( noise(0.0) );  // 0.2292526
println( noise(0.1) );  // 0.2751146
println( noise(0.2) );  // 0.3123877
println( noise(0.3) );  // 0.3495249
```

You can see some sort of a relationship between these values

How to use `noise()` to Animate?

- Step1: Create and initialize an offset global variable.

```
float offset = 0.0;
```

- Step2: Use the offset to compute an attribute value in draw().

For example, to compute a random y location based on noise, use this code:

```
y = noise(offset) * height;
```

- Step3: Slightly change the offset in draw()

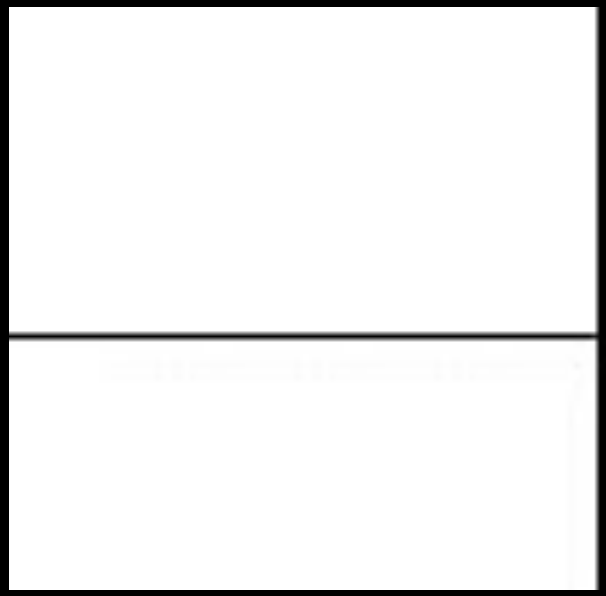
```
offset += 0.01;
```

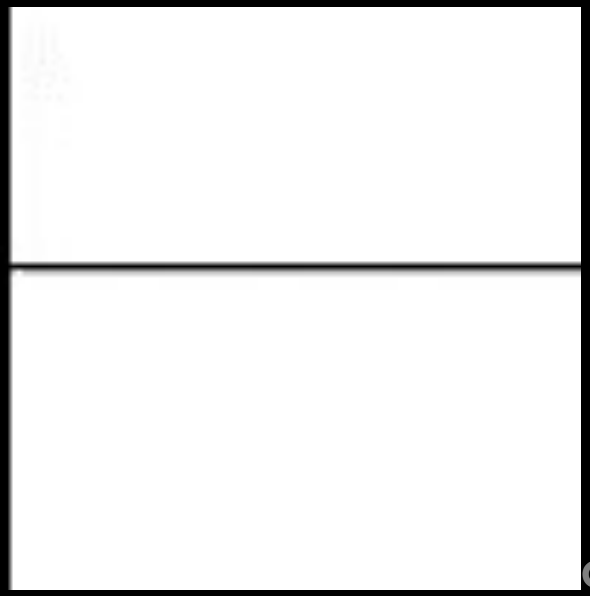
Full code on next slide.

noise() vs. random()

```
float offset = 0.0;
void draw() {
  background(255);
  float y = noise(offset) * height;
  offset  += 0.01;
  line(0, y, width, y);
}
```

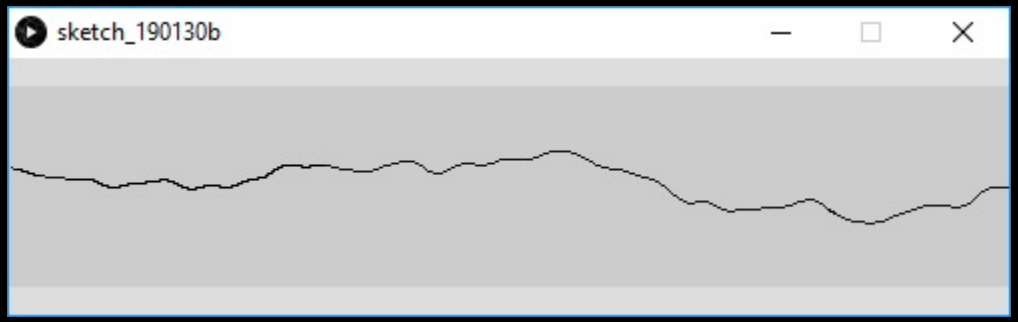
```
void draw() {
  background(255);
  float y=random(height);
  line(0, y, width, y);
}
```





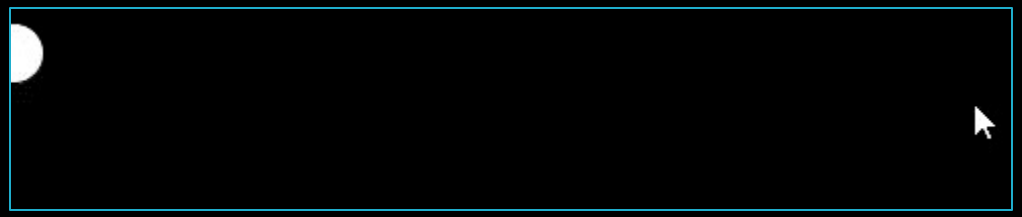
Draw the Perlin curve

```
float offset = 0.0;  //x location
void setup(){
  size(500,100);
}
void draw() {
  float y = noise(offset) * height;
  offset  += 0.01;
  point(offset*100, y);
}
```



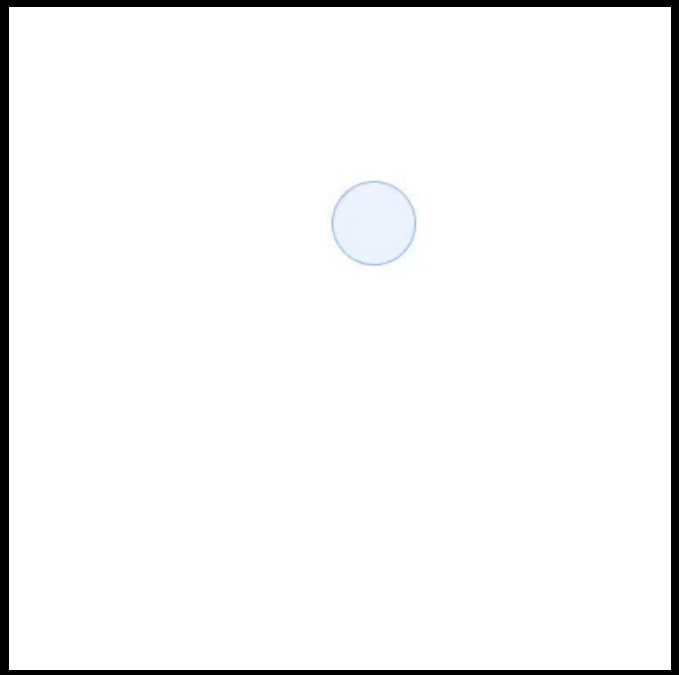
Draw the Perlin curve

```
float x_off = 0.0;  //offset
void setup(){
  size(500,100);
}
void draw() {
  background(0);
  x_off  += 0.01;
  float y = noise(x_off) * height;
  float x = x_off * 150;
  ellipse(x, y, 30, 30);
}
```



Wandering Cell – move ball along Perlin path

- This example shows a cell (circle) that moves randomly on the sketch.
  - In practice, it moves along a path defined by Perlin curves along x and y.



```
float x_off = 0.0, y_off = 50.0;
void setup(){
  size(400,400);
  fill(0,100,255, 20);
  stroke(0,120,255,120);
}
void draw() {
  background(255);

  // Compute random (x,y)
  x_off += 0.005;
  float x = noise(x_off)*width;

  y_off += 0.005;
  float y = noise(y_off)*height;


  //Draw the cell
  ellipse(x, y, 50, 50);
}
```

Adding Randomness to Your Animation

Which is better for adding randomness to your animation: `random()` or `noise()`?

A. `random()` is better than `noise()`

B. `noise()` is better than `random()`

C. It depends on the situation

D. I don't understand what you are talking about.

Bubble!

- Create a sketch in which a bubble (ellipse) moves from the bottom of the screen to the top while being blown by some imaginary, low-turbulence wind. The size should also slightly increase as the bubble moves up.
- The idea:
  - Start with the bubble below the window bottom (y>height) – decrement y by 1 and increment radius by a small amount every frame.
  - Use Perlin noise to determine the x location

Objectives

- These are notes. 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**

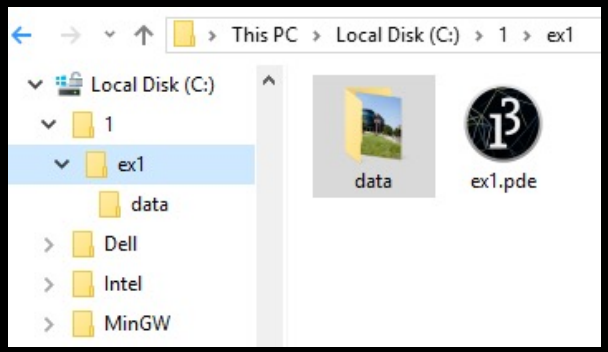
2) Declare a variable of the type `PImage`.
   - `PImage` is a Processing data type that can store image information.

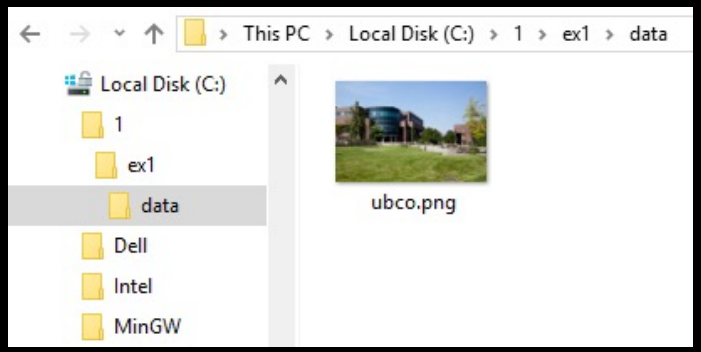
3) Load the image into your variable using `loadImage(filename)`.

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

Drawing an Image in Static Program

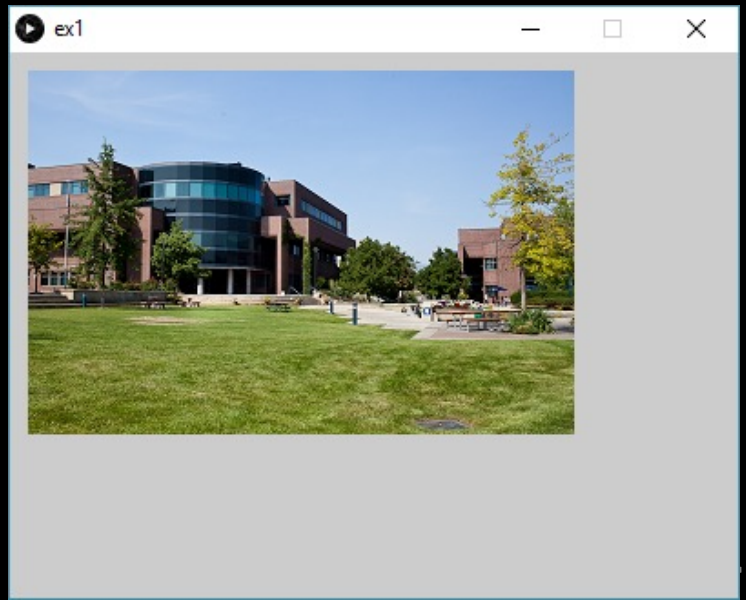
Step 1: put image, e.g. ubco.png, in the data folder of the current sketch






```
size(400,300);

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

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

top-left corner at (10,10)

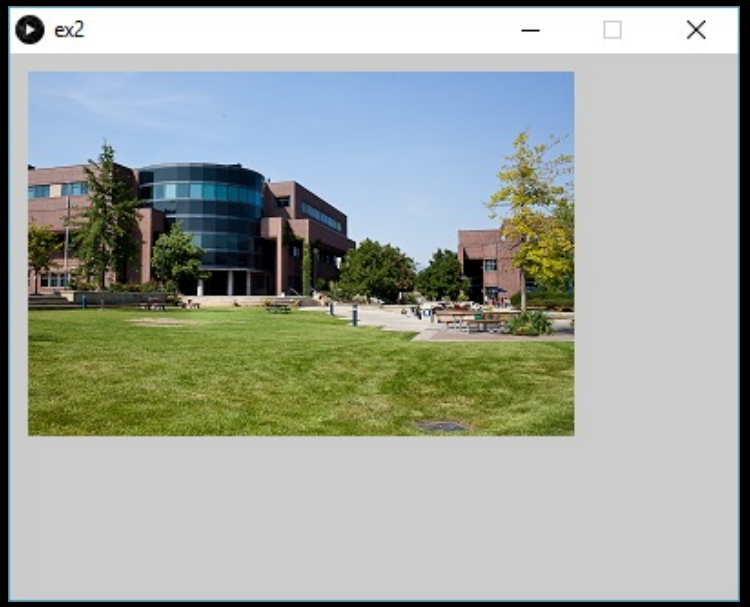


Drawing an Image in Active Program

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()`

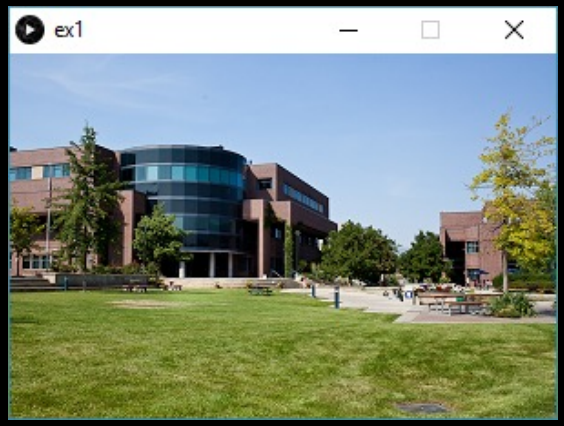
```
PImage img;

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**.

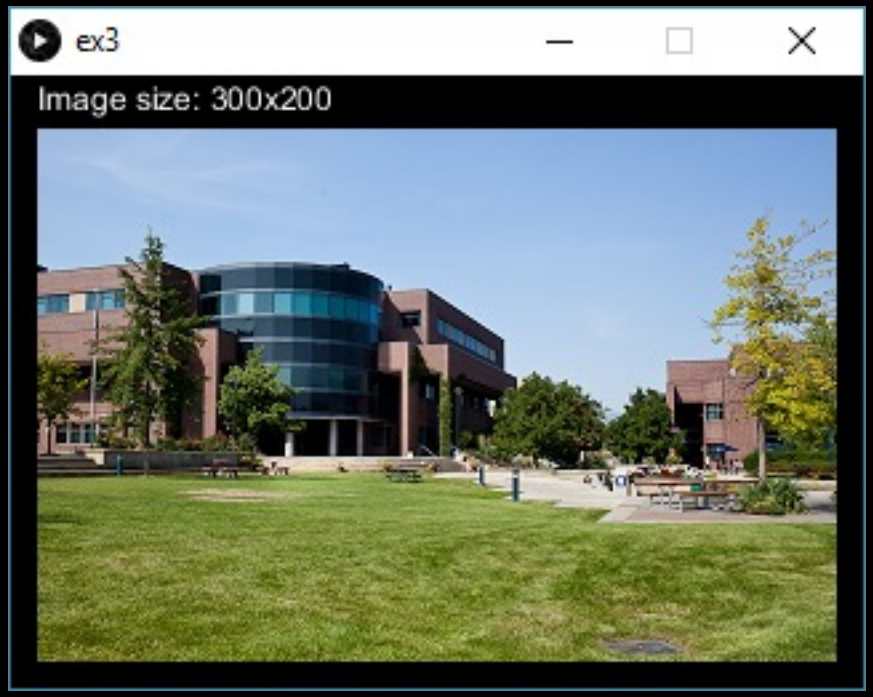
```
PImage img;

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:

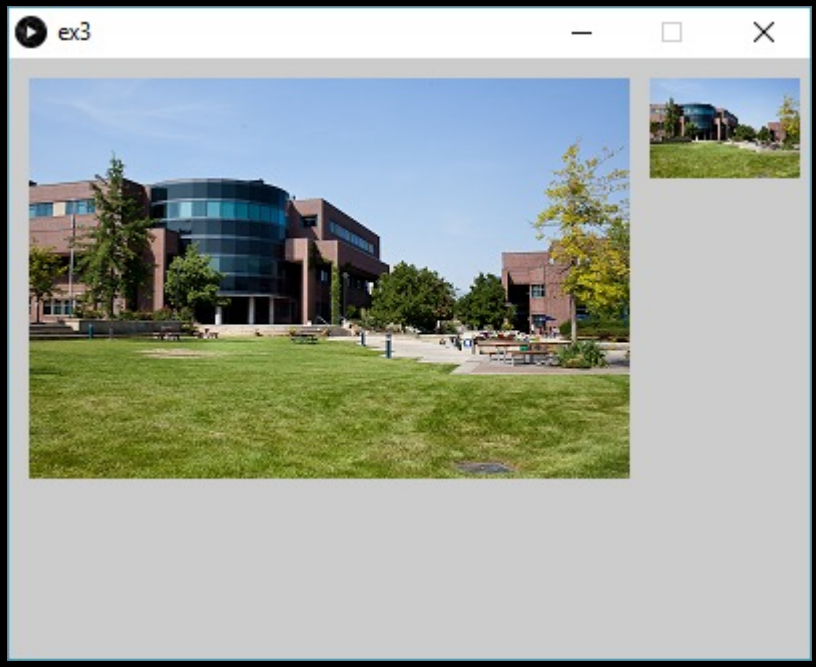
```
image(img, x, y [, w, h])
```

- 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“).

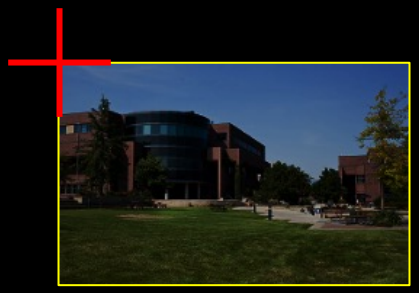
- Two ways to add images to the data folder of your sketch:
  - ***Without PDE***: create a folder called data, then copy your images to that folder using any file manager program (e.g. Windows Explorer)
  - ***Using PDE***: `Sketch` menu `-> Add File`.

- 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!

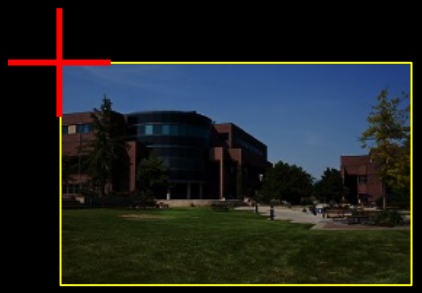
The `imageMode()` function

- *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:

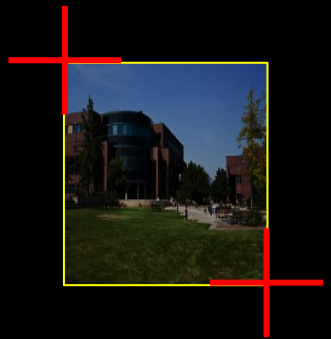


image(img,x,y);



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.

Drawing an image on the sketch

Which of the following is a function that we use to draw an image on the sketch (assume the image is already loaded into a an image variable):

A. PImage

B. loadImage

C. image

D. imageMode

E. tint

Images as Background

Assuming ubco.png is 400x300 pixels, is the following code is correct?

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

A. Yes

B. No

Controlling Image Opacity

Which of the following is a statement that we can use to control set the transparency of an image to 50%?

A. `tint(255);`

B. `tint(128);`

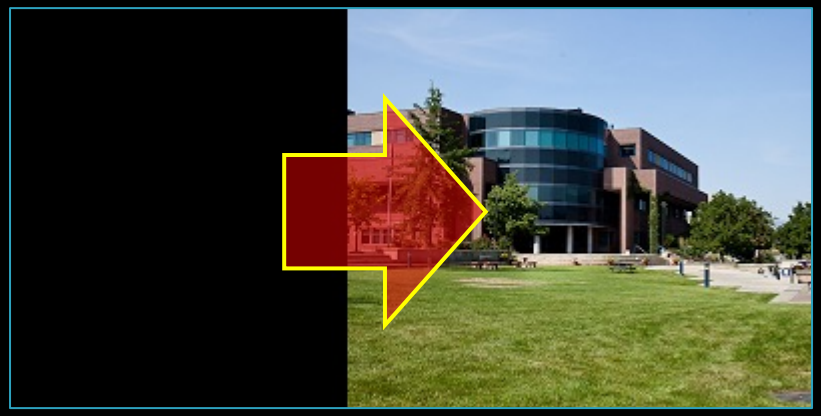
C. `tint(255,128)`

D. `opacity(128);`

E. `transparency(50)`

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)

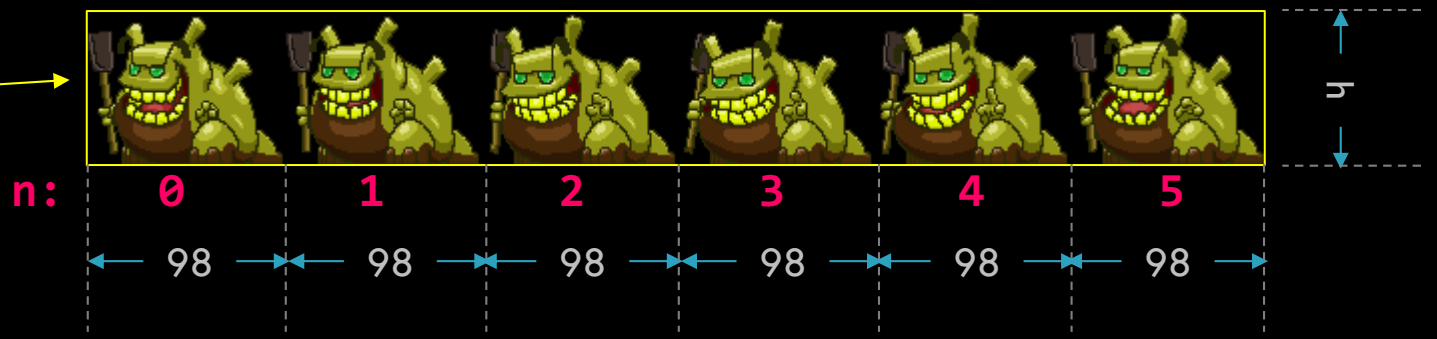
```
image(img, x, y, w, h, x1, y1, x2, y2)
```

- 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()

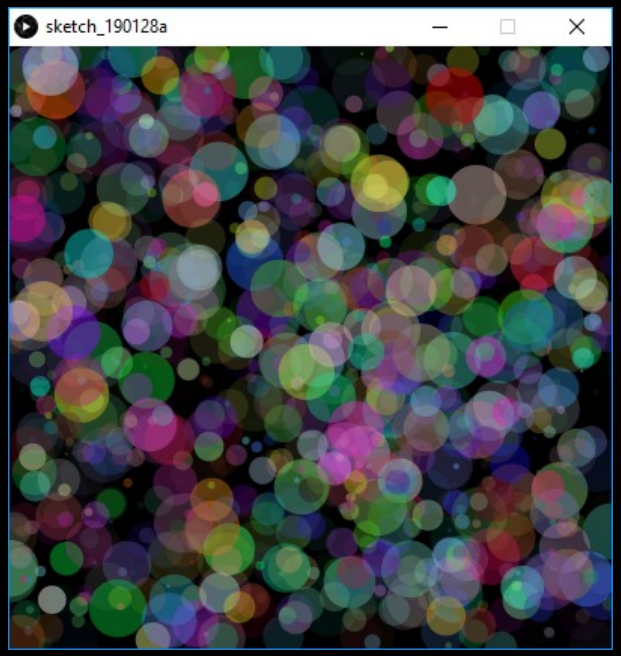
```
PImage img = loadImage("myImage.jpg");

//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)

Random colors

- Now let's do something simpler
- Lets display circles at random locations with random colors and transparencies.
  - Don't clear the background every frame.

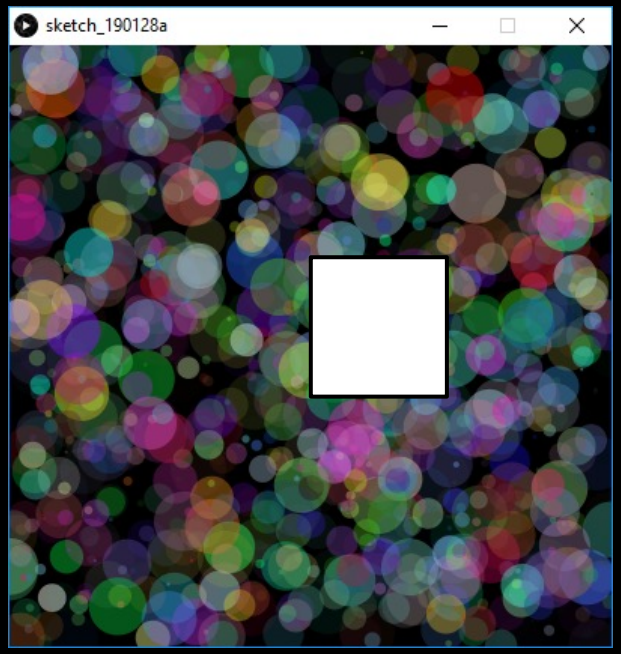


Advanced: pGraphics

- This is used to draw into an off-screen graphics buffer.
- A `PGraphics` object can be constructed using the `createGraphics()` function.
- The `beginDraw()` and `endDraw()` methods can be used to indicate the beginning and the end of the buffer processes.
- Use `image()` to place the `PGraphics`

Advanced: pGraphics Example

- We can have the background with the random circles drawn on a PGraphics object, then place it as an image.
- Then draw on the foreground anything else. For example, let's try to draw a white rectangle that follows the mouse as it moves while the background is what we did in Exercise 4.



Advanced: pGraphics Example, cont'd

```
PGraphics pg;
float r, g, b, a, diam, x, y;
void setup() {
  size(400,400);
  pg = createGraphics(400,400);
}
void draw() {
  background(0);
  r = random(255); g = random(255);
  b = random(255);  a = random(128);
  diam = random(40);  x = random(width);  y = random(height);
  pg.beginDraw();
  pg.noStroke();
  pg.fill(r,g,b,a);
  pg.ellipse(x,y,diam,diam);
  pg.endDraw();
  image(pg,0,0);
  rect(mouseX,mouseY,50,50);
}
```

Draw an Image

- Find an image that you like and draw it on a new sketch. Do the following:
  1) Load image.
  2) Set sketch size to image size and background to black.
  3) Set image mode to CENTER.
  4) Set image opacity to 50%.
  5) Draw image at the center of the sketch.
  6) Set image opacity to 100%.
  7) Draw the image again at the center but at 50% of its original size.

Animate Image Transformation and Opacity

- In the following exercises, use any image of your choice.
  1) Animate the transparency of your image so that it appears as if it is fading out.
  2) Animate the coordinates' scale and rotation so that the image appears as if it is moving away (inwards) while rotating.

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 →