Key Points So Far

- a program is a series of *instructions* (called *statements*) – each statement specifies an action

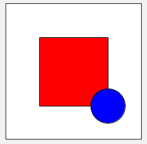
```
rect(100,100,50,200);   // draw a 50x200 rectangle at (100,100)
fill(255,0,0);          // set the current fill color to (255,0,0)
background(255);        // clear the background to white
```

- we've seen statements for –
  - opening a window
  - drawing rectangles, ellipses, lines, points
  - clearing the background to a particular color
  - setting the current fill and stroke colors, used by shape-drawing instructions
  - setting the current "position mode" for rectangles and ellipses (center or upper left corner), used by `rect` and `ellipse`

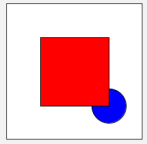
Key Points So Far

- the order of instructions matters
  - "executing a statement" means performing the action
  - statements are executed in order from top to bottom

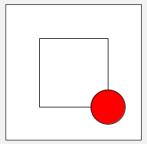
```
size(200,200);
background(255);
rectMode(CENTER);
ellipseMode(CENTER);
fill(255,0,0);
rect(100,100,100,100);
fill(0,0,255);
ellipse(150,150,50,50);
```



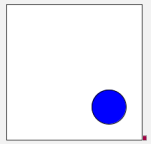
```
size(200,200);
background(255);
rectMode(CENTER);
ellipseMode(CENTER);
fill(0,0,255);
ellipse(150,150,50,50);
fill(255,0,0);
rect(100,100,100,100);
```



```
size(200,200);
background(255);
rectMode(CENTER);
ellipseMode(CENTER);
rect(100,100,100,100);
fill(255,0,0);
ellipse(150,150,50,50);
fill(0,0,255);
```



```
size(200,200);
rectMode(CENTER);
ellipseMode(CENTER);
fill(0,0,255);
ellipse(150,150,50,50);
background(255);
fill(255,0,0);
rect(100,100,100,100);
```



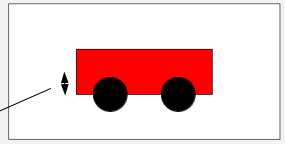
Key Points So Far

- current settings (fill color, stroke color, rect/ellipse mode) stay in effect until changed – they do not need to be set for every shape drawn

```
size(400,200);
background(255);
rectMode(CENTER);
ellipseMode(CENTER);
// the body of the car
fill(255,0,0);
rect(200,100,200,200/3);
// wheels  (left and right)
fill(0);
ellipse(150,100+200/6,50,50);
ellipse(250,100+200/6,50,50);
```

- compute exact positions and sizes when you can instead of guessing

the centers of the wheels line up with the bottom of the rectangle – the y coordinate is half the rectangle's height below the center of the rectangle



```
rect(200,100,200,200/3);
ellipse(150,100+200/6,50,50);
```


Interaction

Static vs Active, Interaction vs Animation

In a *static sketch*, nothing changes over time.

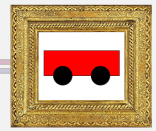
In an *active sketch*, things change (or have the potential to change) over time.

In *interaction*, the changes over time are driven by user actions.

- mouse or keyboard actions

In *animation*, the changes over time are driven (only) by the passage of time.

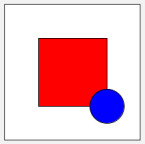
Static Sketches



- a static sketch in Processing is just a list of statements, executed from top to bottom
  - each statement is executed once

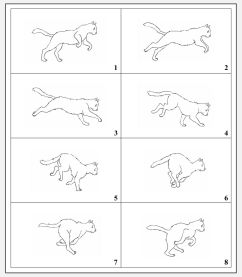
template → example

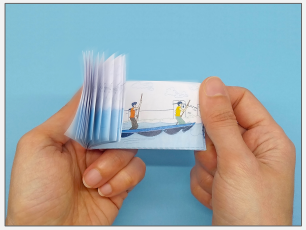
```
// open window          → size(200,200);
// clear the background → background(255);
// draw stuff           → rectMode(CENTER);
                          ellipseMode(CENTER);
                          fill(255,0,0);
                          rect(100,100,100,100);
                          fill(0,0,255);
                          ellipse(150,150,50,50);
```



Time in Processing

- time in the real world is continuous
- time in Processing is discrete
  - an active sketch consists of a series of *frames* drawn one after the other





https://www.printablee.com/post_printable-flip-books_134250/
https://www.basementanimation.com/flipbook/

Active Sketches

- an active sketch in Processing has two lists of statements
  - things that happen once when the sketch first starts
  - things that happen to draw one frame
- template

```
void setup () {
  // open window
  // other things done only once at the beginning
}
void draw () {
  // draw one frame
}
```

- when the sketch is run, first all of the statements in `setup()` are executed once, then the statements in `draw()` are executed over and over and over and …

```
// open window
// other things done only once at the beginning
// draw one frame
// draw one frame
// draw one frame
// draw one frame
// draw one frame
// …
```

Static vs Active Sketches

static

active

sketch (code)

```
size(200,200);
background(255);
rectMode(CENTER);
ellipseMode(CENTER);
fill(255,0,0);
rect(100,100,100,100);
fill(0,0,255);
ellipse(150,150,50,50);
```

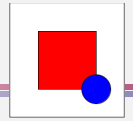
what happens when the sketch is run

```
size(200,200);
background(255);
rectMode(CENTER);
ellipseMode(CENTER);
fill(255,0,0);
rect(100,100,100,100);
fill(0,0,255);
ellipse(150,150,50,50);
```

sketch (code)

```
void setup () {
  size(200,200);
}

void draw () {
  background(255);
  rectMode(CENTER);
  ellipseMode(CENTER);
  fill(255,0,0);
  rect(100,100,100,100);
  fill(0,0,255);
  ellipse(150,150,50,50);
}
```



what happens when the sketch is run

```
size(200,200);
background(255);
rectMode(CENTER);
ellipseMode(CENTER);
fill(255,0,0);
rect(100,100,100,100);
fill(0,0,255);
ellipse(150,150,50,50);
background(255);
rectMode(CENTER);
ellipseMode(CENTER);
fill(255,0,0);
rect(100,100,100,100);
fill(0,0,255);
ellipse(150,150,50,50);
background(255);
rectMode(CENTER);
ellipseMode(CENTER);
fill(255,0,0);
```

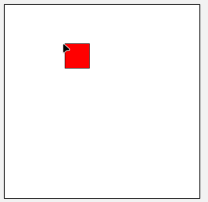

Interaction – Mouse Position

The current position of the mouse is represented by the *system variables* `mouseX` and `mouseY`.

- variables capture quantities that always exist but whose values (may) change
- the current value for the quantity is used whenever a statement containing a variable is executed

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

void draw () {
  background(255);
  rectMode(CORNER);
  fill(255, 0, 0);
  // draw a rectangle whose upper left corner
  //  is at the mouse's current position
  rect(mouseX, mouseY, 50, 50);
}
```

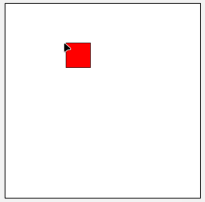


Interaction – Mouse Position

sketch (code)

what happens when the sketch is run

each time a rect is drawn, the current values of `mouseX` and `mouseY` are used



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

void draw () {
  background(255);
  rectMode(CORNER);
  fill(255, 0, 0);
  // draw a rectangle whose upper left corner
  //  is at the mouse's current position
  rect(mouseX, mouseY, 50, 50);
}
```

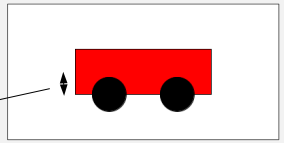
```
size(400, 400);
background(255);
rectMode(CORNER);
fill(255, 0, 0);
rect(mouseX, mouseY, 50, 50);
background(255);
rectMode(CORNER);
fill(255, 0, 0);
rect(mouseX, mouseY, 50, 50);
background(255);
rectMode(CORNER);
fill(255, 0, 0);
rect(mouseX, mouseY, 50, 50);
background(255);
rectMode(CORNER);
fill(255, 0, 0);
rect(mouseX, mouseY, 50, 50);
background(255);
rectMode(CORNER);
fill(255, 0, 0);
rect(mouseX, mouseY, 50, 50);
background(255);
rectMode(CORNER);
fill(255, 0, 0);
rect(mouseX, mouseY, 50, 50);
background(255);
```


Arithmetic Expressions

- arithmetic expressions involve the operators +, -, *, /
  - * is multiply
  - / is divide
- can be used anywhere a value is needed

the rectangle's height is one third the window's height

the centers of the wheels line up with the bottom of the rectangle – the y coordinate is half the rectangle's height below the center of the rectangle



```
rect(200,100,200,200/3);
ellipse(150,100+200/6,50,50);
```

Arithmetic Expressions

```
rectMode(CORNER);
// draw a rectangle whose upper left corner
//  is at the mouse's current position
rect(mouseX, mouseY, 50, 50);
```

```
rectMode(CORNER);
// draw a rectangle where the middle of the
//  left side is at the mouse's current
//  position
rect(??, ??, 50, 50);
```

- the upper left corner is half the rectangle's height above the middle

```
rectMode(CORNER);
// draw a rectangle where the middle of the
//  left side is at the mouse's current position
rect(mouseX, mouseY-25, 50, 50);
```


Interaction – Events

Mouse clicks and key presses are *events* – user-initiated things that happen at some moment in time.

Fill in the body of an *event handler* to specify what to do when a given kind of event occurs.

- system calls the appropriate handler when that type of event happens
- statements in the handler body are executed from top to bottom

```
void mouseClicked () {
 …
}
void mousePressed () {
 …
}
void mouseReleased () {
 …
}
```

```
void keyPressed () {
 …
}
```


Interaction – Events

sketch (code)

what happens when the sketch is run

the statements in `draw()` are repeated over and over

when the mouse is clicked, the statements in `mouseClicked()` are executed once after the current repeat of `draw()` finishes

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

void draw () {
  background(255);
  rectMode(CORNER);
  fill(255, 0, 0);
  rect(mouseX, mouseY, 50, 50);
}

void mouseClicked () {
  background(0);
}
```

```
size(400, 400);
background(255);
rectMode(CORNER);
fill(255, 0, 0);
rect(mouseX, mouseY, 50, 50);
```

```
background(255);
rectMode(CORNER);
fill(255, 0, 0);
rect(mouseX, mouseY, 50, 50);
```

```
background(255);
rectMode(CORNER);
fill(255, 0, 0);
rect(mouseX, mouseY, 50, 50);
```

mouse click!

```
background(255);
rectMode(CORNER);
fill(255, 0, 0);
rect(mouseX, mouseY, 50, 50);
```

background(0);

```
background(255);
rectMode(CORNER);
fill(255, 0, 0);
rect(mouseX, mouseY, 50, 50);
```

mouse click!

```
background(255);
rectMode(CORNER);
fill(255, 0, 0);
rect(mouseX, mouseY, 50, 50);
```

background(0);

mouse click!

```
background(255);
rectMode(CORNER);
fill(255, 0, 0);
rect(mouseX, mouseY, 50, 50);
```

background(0);

```
background(255);
rectMode(CORNER);
fill(255, 0, 0);
rect(mouseX, mouseY, 50, 50);
```

```
background(255);
```


Pattern Summary

- nothing changes (static picture)
  → static mode sketch
- something changes based on the mouse position
  → active mode sketch (`setup()`, `draw()`) using `mouseX`, `mouseY`
- do something when the mouse is clicked (or pressed or released) or a key is pressed
  – → active mode sketch (`setup()`, `draw()`) with event handlers

```
size(200,200);
background(255);
rectMode(CENTER);
ellipseMode(CENTER);
fill(255,0,0);
rect(100,100,100,100);
fill(0,0,255);
ellipse(150,150,50,50);
```

```
void setup () {
  size(400, 400);
}
void draw () {
  background(255);
  rectMode(CORNER);
  fill(255, 0, 0);
  rect(mouseX, mouseY, 50, 50);
}
```

```
void setup () {
  size(400, 400);
}
void draw () {
  background(255);
  rectMode(CORNER);
  fill(255, 0, 0);
  rect(mouseX, mouseY, 50, 50);
}
void mouseClicked () {
  background(0);
}
```