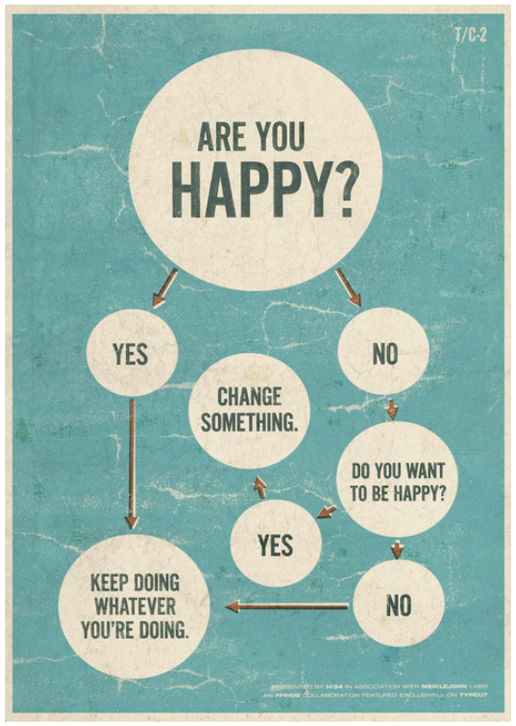
09.10.18 //

PROCESSING III

DART 631
RILLA KHALED

CONDITIONALS



Now we turn to the idea of controlling the flow of a program

Which is getting pretty sophisticated, and pretty awesome.

TRUE OR FALSE

In Processing, there are certain expressions that *evaluate* to be true or false, mostly based on math:

`10 < 20` is `true`

`1 + 1 == 3` is `false`

We can check on the truth or falsity of expressions like this in order to choose what to do in our programs.

It works even better if we use variables…

`avatarX < width` is … well, we don't know right now!

CONDITIONAL OPERATORS

Here are the major conditional operators:

`1 < 2` (less than)

`2 > 1` (greater than)

`1 <= 2` (less than or equal to)

`2 >= 2` (greater than or equal to)

`1 != 2` (does not equal)

`1 == 1` (equals)

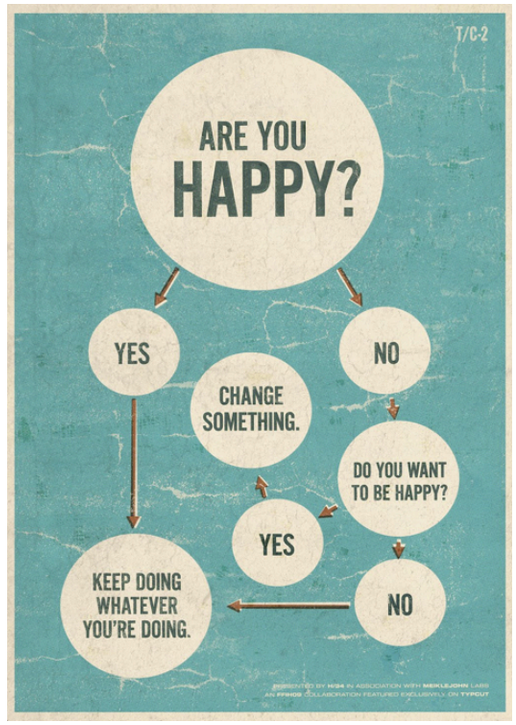
All of those are… **`true`**.

THINGS ARE GETTING A LITTLE IFFY...

So how do we actually check these conditions in some useful way in our programs?

We use `if` statements.

An `if` statement checks on whether a condition is true or not, and then based on that will either do something or not.



THINGS ARE GETTING A LITTLE IFFY...

```
if ( condition )
{
     // Do something
}
```

THINGS ARE GETTING A LITTLE ELSEY...

```
if ( condition )
{
        // Do something (condition is true)
}
else
{
        // Do something else (condition is false)
}
```

We can use this extra “else” and curly brackets if we want to tell Processing what to do when the condition turned out to be `false`.

THINGS ARE GETTING A LITTLE ELSEIFFY...

```
if ( condition )
{
        // Do something (condition is true)
}
else if ( anotherCondition )
{
        // Do something (condition is false)
        // (But anotherCondition is true)
}
```

We can even specify another condition to check after we found out that `condition` is `false`.

THINGS ARE GETTING A LITTLE NESTY...

```
if ( condition )
{
        if ( anotherCondition )
        {
                // Do something
                // (condition and anotherCondition are both true)


        }
}
```

We can even *nest* conditionals to check if multiple things are true at the same time!

Because sometimes you need to know more than one thing

THINGS ARE GETTING A LITTLE LOGICAL...

We can also use logic to make more complicated conditionals, i.e. we can combine conditions. The logical operators are:

`&&` - and

`||` - or

`!` – not

So how does that work?

THINGS ARE GETTING A LITTLE LOGICAL...

`condition1 && condition2`

Is `true` if both `condition1` and `condition2` are `true`.

`condition1 || condition2`

Is `true` if *either* `condition1` or `condition2` are `true`.

`!condition1`

Is only `true` if `condition1` is `false`.

THIS SORT OF THING MIGHT BE USEFUL...

```
if (avatarX > width || avatarX < 0)
{
    // The avatar is off the screen!
}
```

So we can check important conditions in our program and then react appropriately.

```
if (avatarX > width || avatarX < 0)
{
        // The avatar is off the screen!
}

Notice that this is the same as:

if (avatarX > width)
{
        // The avatar is off the screen!
}
if (avatarX < 0)
{
        // The avatar is off the screen!
}

But better, because we only have to react one time!
```

WHAT WILL THIS DO?

```
if (2 > 0 || 10 < 9)
{
  println(“Squeezing blood from a stone!”);
}
else if (10 < 20 && 9 <= 9)
{
  println(“When pigs fly!”);
}
if (!(10 > 0 && 9 < 10))
{
  println(“When hell freezes over!”);
}
```

```
if (2 > 0 || 10 < 9)
{
  println("Squeezing blood from a stone!");
}
else if (10 < 20 && 9 <= 9)
{
  println("When pigs fly!");
}
if (!(10 > 0 && 9 < 10))
{
  println("When hell freezes over!");
}
```

BUT, AGAIN, IT'S BETTER WITH VARIABLES...

Here's something a bit more complicated…

In fact, it's the beginning of some not amazing physics…

```
int avatarX = 0; // Avatar location on X
int avatarY = 0; // Avatar location on Y
int avatarSize = 10; // Avatar size (will be a square)
int avatarVelocityX = 5; // Number of pixels avatar should move each frame on X

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

void draw()
{
  background(255); // Fill the background to create animation
  avatarX += avatarVelocityX; // Add the velocity to the avatar's location so it moves
  rect(avatarX,avatarY,avatarSize,avatarSize); // Draw the avatar in its new location

  // Now check if the avatar is going off the screen
  if (avatarX > width || avatarX < 0)
  {
    // If it is, then reverse its velocity!
    avatarVelocityX = -avatarVelocityX;
  }
}
```

AND IN FACT, IT CAN BE EVEN BETTER…

Here's something even more complicated…

In fact, it's the beginning of some crappy controls…

Check the code/L6/AvatarMovement folder on GitHub for code.

MORE COMPLEX

Let's have the walls constrain the avatar.

Check the code/L6/AvatarMovementMoreComplex folder on GitHub for code.

ONE LAST SAVING GRACE!

Processing has a variable *type* which you can use to store the results of conditions in, it's called `boolean`.

The value of a `boolean` variable will be either `true` or `false`.

```
int meaningOfLife = 42;
boolean lifeHasMeaning = (meaningOfLife == 42);
if (lifeHasMeaning)
{
      // Do something!
}
```

Notice how we can use a boolean variable in the same places we would use a conditional!

ANOTHER EXAMPLE OF IF


```
boolean myButtonPressed = false;

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

void draw()
{
  if (myButtonPressed)
  {
    background(255);
  }
  else
  {
    background(0);
  }
}

void mouseReleased()
{
  myButtonPressed = !myButtonPressed;
}
```

EXERCISE.

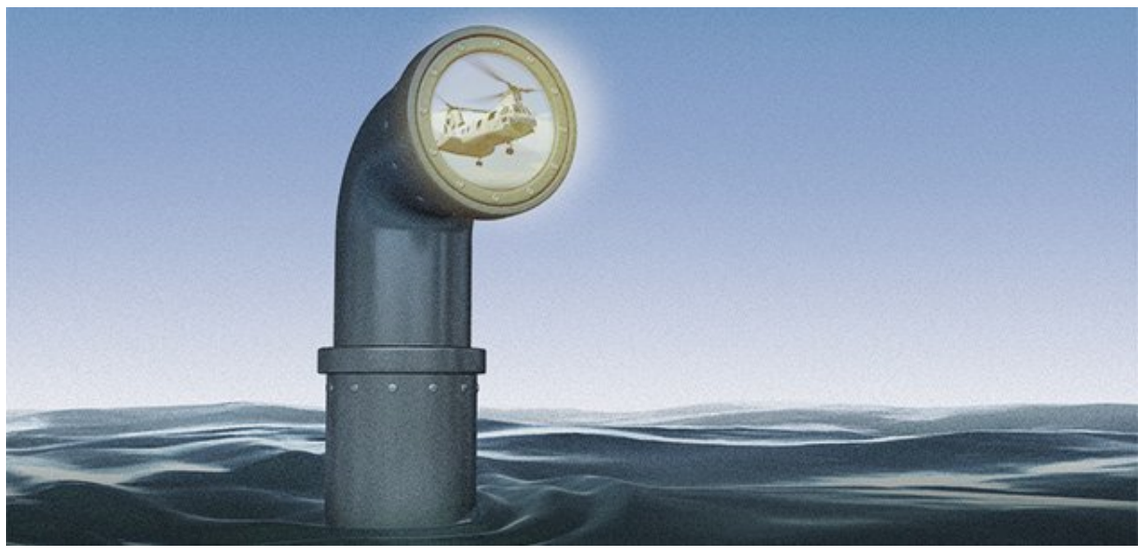
Modify or create a new scene to use conditionals in reacting to player input or anything else.

What would happen if you used an if statement with `random()`? (Hint: think about probability…)

What if you checked the values of `mouseX` and `mouseY` when the player clicked the mouse button?

What if you checked combinations of conditions, like where the mouse is and what key is being pressed?

SCOPE



One more thing about all this!

about what can be seen by the program

SHORT (AND CURLY) STORY.

Your variables will only work inside the curly brackets they were declared in.

The only exception (for now) is when you declare them outside any curly brackets at all.

So…

SCOPE

```
void setup()
{
	int avatarX = 0;
	int avatarY = 0;
}

void draw()
{
	rect(avatarX,avatarY,10,10);
}
```

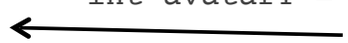
```
void setup()
{
        int avatarX = 0;
        int avatarY = 0;
}

void draw()
{
        rect(avatarX,avatarY,10,10);
}
```

SCOPE

```
void setup()
{
        int avatarX = 0;
        int avatarY = 0;
}

void draw()
{
        rect(avatarX,avatarY,10,10);
}
```



avatarX stopped existing here!

GLOBAL SCOPE

```
int avatarX = 0;
int avatarY = 0;

void setup()
{
}

void draw()
{
        rect(avatarX,avatarY,10,10);
}
```

This will work because variables defined outside everything can be used everywhere.

But it’s not the best practice usually, so use it only cautiously and think about why you’re doing it!

MORE SCOPE

```
if (9 < 10)
{
	boolean nineIsLessThanTen = true;
}

if (nineIsLessThanTen)
{
	println(“Yaaaaay!”);
}
```


```
if (9 < 10)
{
	boolean nineIsLessThanTen = true;
}

if (nineIsLessThanTen)
{
	println(“Yaaaaay!”);
}
```

MORE SCOPE

```
if (9 < 10)
{
        boolean nineIsLessThanTen = true;
}


if (nineIsLessThanTen)
{
        println(“Yaaaaay!”);
}
```

nineIsLessThanTen stopped existing here!

IMAGES

you can load in images

IMAGES

Images in Processing have a special data type: the `PImage`

Amongst other things, you can use a `PImage` to load an existing image file:

```
PImage cat;
cat = loadImage(“kasper.jpg”);
image(cat, 0, 0);
```

Note: the image file *must* be in your Processing project’s “data” folder. You can drag it over your sketch or add it from the Sketch >> Add file menu option.

In GitHub code look at:

L6/ImageDrawing1

L6/ImageDrawing2

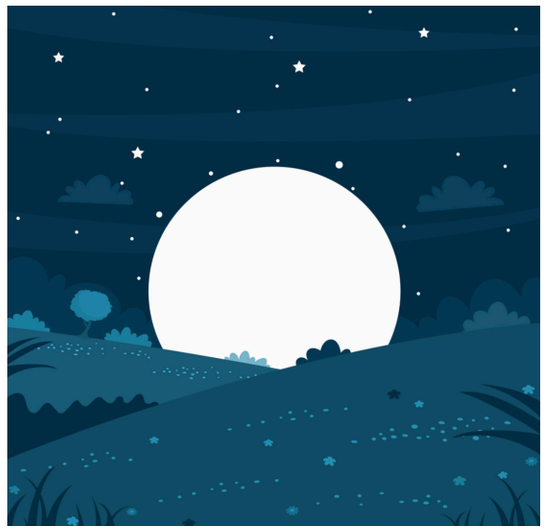
L6/ImageAvatar

L6/ImageAvatarHitBox

Chapters 1 — 6 if you want to brush up on what we learned last week too.

EXERCISE.

Using everything that we have learned so far, make an interactive scene that can represent day or night.



The scene should use

- random()
- mouseX and mouseY
- at least 3 images
- Conditional statements: if, else if, else (etc.)