Welcome to Kickstart

Welcome to Computer Science and the programming kickstart course! The exercises below will quickly get you up and running with programming P5 JavaScript. We have intentionally left some things unexplained as we want you to experiment on your own.

Resources

There is a world of information and resources to help you learn p5.js. Check out the following:

Book McCarthy, Lauren, Casey Reas, and Ben Fry. Getting started with P5. JS: Making interactive graphics in JavaScript and Processing. Maker Media, Inc., 2015.

p5.js Web site The official website

Happy Coding A fun set of tutorials and examples for p5.js: Happy Coding

Kickstart Absalon You can find the book and other materials on the LMS.

Have Fun

DIKU
ds@di.ku.dk

First Program

Type the following example in the Processing editor and press -button:

```
function setup() {
  //set the canvas and background color
  createCanvas(400, 400);
  background(220);

  // Tree
  rect(55, 50, 10, 20);
  ellipse(60, 35, 30, 40);
}

function draw() {}
```

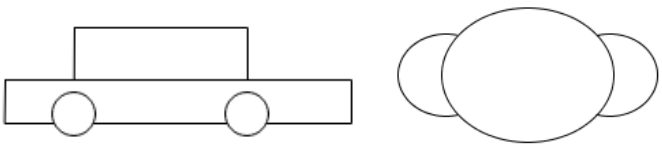
Then add the following into the `setup()` function:

```
  // powerplant
  rect(120, 50, 60, 30);
  rect(160, 20, 10, 30);
  triangle(120, 50, 136, 40, 136, 50);
  triangle(136, 50, 152, 40, 152, 50);
```

And:

```
  // windmill
  line(300, 50, 320, 51);
  line(300, 50, 289, 67);
  line(300, 50, 291, 32);
  line(300, 50, 300, 90);
```

Now try drawing a car and a cloud (psst, use the documentation):



Documentation

Documentation is a manual describing how a specific part of a programming language works. This is an easy way to learn what all the numbers refer to in `triangle(120, 50, 136, 40, 136, 50)`.
Use the **documentation** on p5.js. For instance, search for *triangle* and press the reference `triangle()`. Then you can read what each of the parameters (the numbers in paremthesis) given to `triangle()` do.

DIKU
ds@di.ku.dk

Colors

You are now going to colour the shapes. To do this, use the `fill(r, g, b)` function, which selects which colour to use for fill and text colour. It's all about calling `fill` in the right places! As arguments, you specify the amount of red (0-255), blue (0-255), and green (0-255). Here are some basic colors to experiment with:

```
fill(255, 0, 0);  // red
fill(0, 255, 0);  // green
fill(0, 0, 255);  // blue
```

```
fill(0, 0, 0);       // black
fill(255, 255, 255); // white
fill(255, 255, 0);   // yellow
```

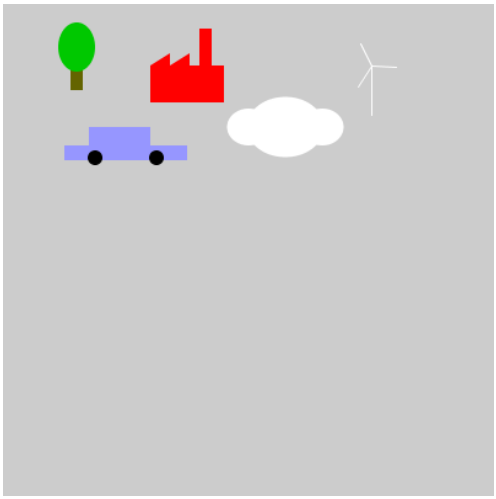
Optionally, find colours using an online color picker or RGB colour table. For example, search for "RGB color picker".

Lines and outlines

To specify the colour of strokes (e.g. `line`) and outlines, use `stroke(r, g, b)`. Also try the `noStroke()` function, to turn off outline drawing.

Example

Here's an example of what it might look like after colouring:

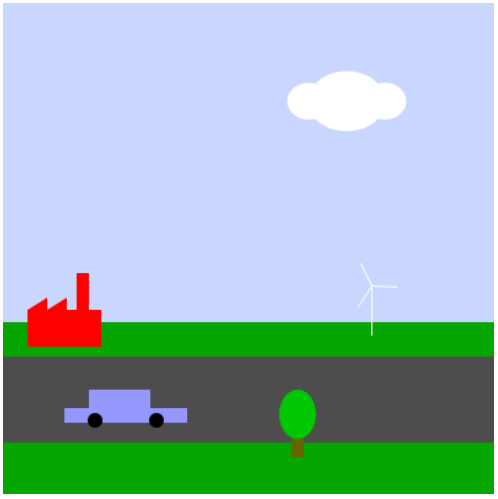


DIKU
ds@di.ku.dk

Green City

Use what you've learned to make it a slightly nicer scene, with background, foreground and the extra details you think should be there. For example, I've drawn a road and moved the objects around.

Tip: To change the background color from gray you can use `rect` which you already know, but you can also use the `background(r, g, b)` command. It will delete everything and fill the screen with the specified colour.



Comments

A comment is any line starting with `//`. The line won't be interpreted as code by the program. A comment can be used to "remove" a piece of code temporarily for debugging or for helping yourself understand the code with an easy to understand label, for example:

```
// the next line is a comment, but the last line is code
// fill(255, 0, 0);
fill(255, 0, 0);
```


DIKU
ds@di.ku.dk

Making an Aquarium

Create a new project ("File" -> "New") and immediately save it the project. Call it "Aquarium".

Type in this piece of code:

```
function setup() {
  createCanvas(400, 400);
  background(220);

  //basic fish shape
  fishX = 150;
  ellipse(fishX, 200, 120, 75);
  triangle(fishX - 60, 200, fishX - 90, 170, fishX - 90, 230);
}

function draw() {

}
```

Try changing 150 to a different number in the `fishX` specification.
Now add the following:

```
  eyeSize = 15;
  ellipse(fishX + 30, 190, eyeSize, eyeSize);
```

Try changing the value of `eyeSize`.

Variables

A variable is a piece of data linked to a name. This could be a number, a piece of text, or one of the many other data types.

It is defined with the following syntax:

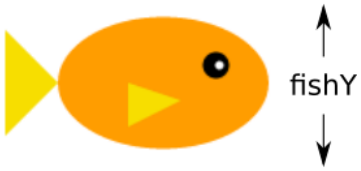
```
x = 1;
y = 2;
z = x + y;
```


DIKU
ds@di.ku.dk

Tasks

You have now added a fish that can be moved just by changing one value.

- Color the fish
- Give the fish a fin that moves when you change `fishX`.
- Give the fish a pupil that moves when you change `fishX`.
- Create a new variable, `fishY`, that controls the y-position of the fish



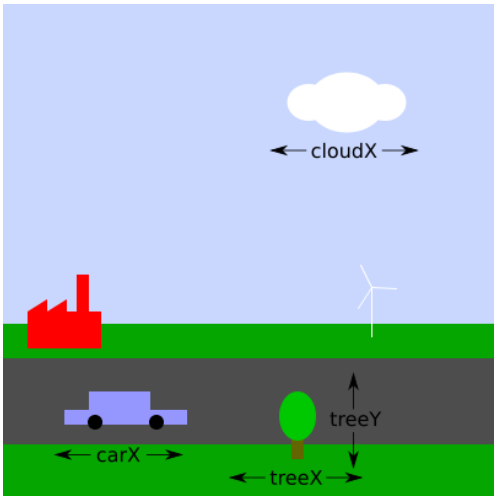
Remember to save the project. We'll be working on it later.

DIKU
ds@di.ku.dk

Green City continues

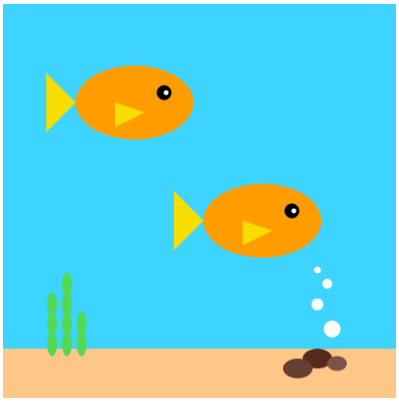
Switch to the electric car project and introduce variables to specify the location of the objects so that we can later animate these objects.

- Create a variable `carX` so the car can move forward and back
- Create a variable `cloudX` so the cloud can move back and forth
- Create a variable `treeX` so the tree can be moved horizontally
- Create a variable `treeY` so the tree can be moved vertically



Aquarium Continues

Use what you've learned to expand your aquarium project; here's an example, but feel free to use your imagination! For example, one variable is used for the x-coordinate of the seaweed plant and another variable for the x-coordinate of the entire group of rocks (as a unit). An additional set of variables `fish2X`/`fish2Y` to control the location of the additional fish.



Functions

Functions allow you to reuse the same code in multiple places, name entire blocks, and add structure to code.

Open the Aquarium project. Add the following function to draw a fish:

```
function setup() {
  // Set up the canvas
  createCanvas(400, 400);

  // Set the background color to light grey
  background(220);
}

function draw() {
  // Draw the first fish at x-coordinate 100
  drawFish(100);

  // Draw the second fish at x-coordinate 280
  drawFish(280);
}

function drawFish(fishX) {
  // Draw the basic fish shape
  // Draw fish body  (fishX, 200) with a width of 120 and height of
      75
  ellipse(fishX, 200, 120, 75);

  // Draw the fish tail using a triangle
  // The triangle points are at (fishX - 60, 200), (fishX - 90,
      170),
  //and (fishX - 90, 230)
  triangle(fishX - 60, 200, fishX - 90, 170, fishX - 90, 230);

  // Draw the fish eye
  // The eye is a small ellipse located at (fishX + 30, 190)
  //with a size of 15x15
  eyeSize = 15;
  ellipse(fishX + 30, 190, eyeSize, eyeSize);
}
```

Now it's much faster to fill the aquarium with fish and we avoid copying code.

Functions

A function in programming is like a function in math. It takes one or more arguments in like x, which could be a number (Int) type. It then utilizes the input to do or compute something, which can be returned as a result.

An example of a function that returns an output is `plus2`. `plus2` takes a number and returns the number plus 2:

```
function plus2(a){
    return a + 2
}
```

In the example above, `a` is an input/parameter to the function that needs to be a number when calling the function. The function returns a number that can be used or bound elsewhere in the code.

A function does not need to return anything in p5.js. An example of this is the drawFish function above, which draws the fish without any returns.

Task

Create your own `drawFish(x, y)` function that draws your entire fish with colour, fins and eyes.

Notes and Extra

- Try changing 50 to another number
- Try changing the line `x = x + 1` to `x = x - 1` or to `x = x + 5`
- Try moving the call to `background` from `draw` to `setup` - what happens?

BE AWARE

When using setup/draw, call draw functions outside of setup and draw are not allowed. Everything must be moved into the two functions.

DIKU
ds@di.ku.dk

Green City continued

Over in the electric car project, you can also try writing a function to draw trees:

```
 function setup() {
  createCanvas(400, 400);
  background(220);
  drawTree(160);
}

function draw() {
  // we are going to use the draw function soon!
}

function drawTree(treeX) {
  fill(100, 100, 0);
  rect(treeX - 5, 350, 10, 20);
  fill(0, 200, 0);
  ellipse(treeX, 335, 40, 50);
}
```

Structure your electric car project code with functions:

- Write a `drawCloud(x)` function that draws a cloud
- Extend the `drawTree(x)` function to also accept a y-coordinate
- Write a `drawCar(x)` function that draws a car
- Write a `drawPowerplant()` function and a `drawWindmill()` function that draws the power plant and the windmill, respectively. We won't need to move them around, so they don't need to not take coordinates as an argument.

Call all the functions in the `setup()` for now. For example:

```
drawTree(150, 235);
drawTree(240, 335);
drawPowerplant();
drawWindmill();
drawCar(50);
drawCloud(280);
```