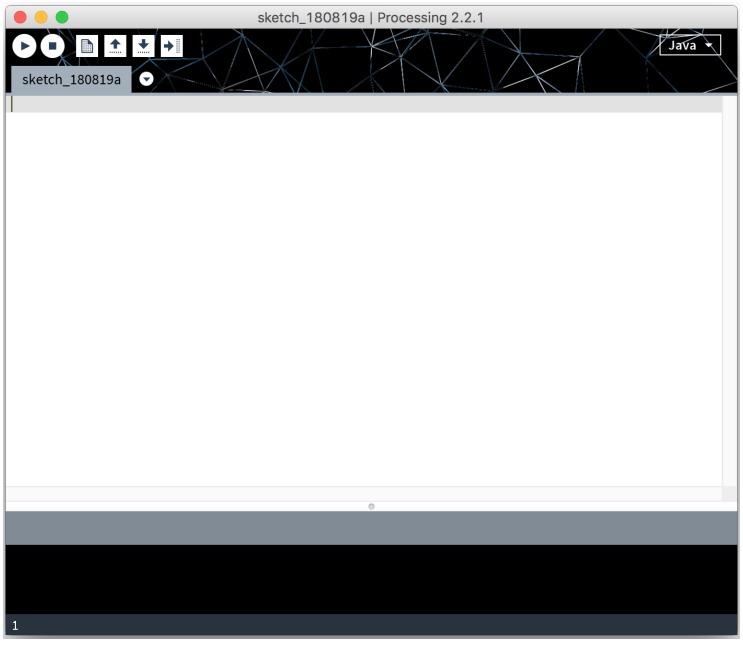
Processing tutorial

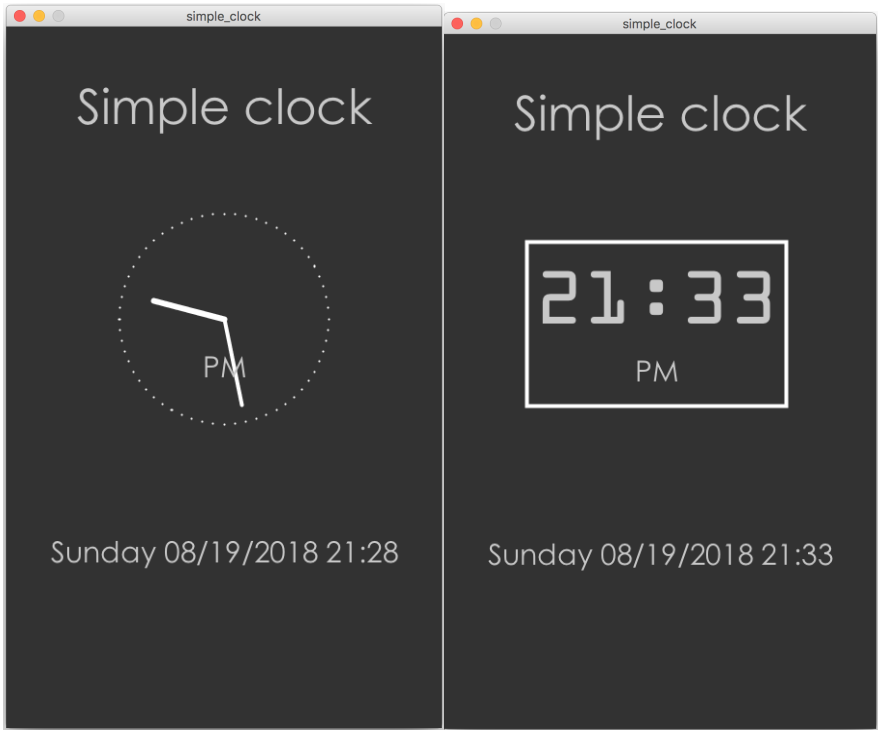
Processing is a programming language and environment based on Java. Processing provides a library to display visuals into the screen in an easy and quick way.

The processing IDE is simple but provides enough functionality to run simple to complex programs.



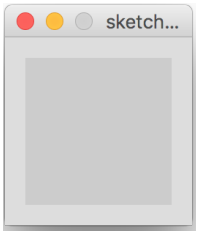
It has a text editor, where the code goes, a play and stop button to run and stop the execution of the program your developing.

In this tutorial we're going to build an analog and digital clock.



With this I will go from basic to more complex concepts regarding animation and interactivity.

Processing is a ready-to-be-used language. So much so that we can already run it without writing any code and we'll already get displayed an empty canvas:



Let's change the size of this canvas:

```
size(450, 720);
```

The first number represents the width and the second the height.

Let's change the colour of the background:

```
background(50);
size(450, 720);
```

When we only pass one number to the background function, we are using the scale of grays. The range goes from 0 to 255 shades of gray, with 0 being black and 255 white.

If we want to add different colours we can pass three numbers to this function. For example:

```
background(255, 0, 0); // red
background(180, 140, 100); // brownish
```

Let's begin drawing the digital clock. For this let's first draw the rectangle where the numbers will be:

```
background(50);
size(450, 720);

rect(100, 200, 270, 170);
```

Where the first two arguments are the x and y coordinate, respectively, of the top left point of the rectangle.

Let's change the colour of the rectangle using the fill function:

```
fill(0, 255, 0);
rect(100, 200, 270, 170);
```

When defining the colour we can also define the alpha component, as a fourth argument to the fill function. Let's compare the two:

```
fill(0, 255, 0);
rect(100, 200, 270, 170);
```

```
fill(0, 255, 0, 50);
rect(100, 400, 270, 170);
```

We can change the border thickness:

```
fill(0, 255, 0, 50);
strokeWeight(4);
rect(100, 200, 270, 170);
```

And colour as well:

```
fill(0, 255, 0, 50);
stroke(255);
strokeWeight(4);
rect(100, 200, 270, 170);
```

For our clock I actually only want the border with no fill. For that I set the fill's alpha as 0:

```
fill(0, 0);
stroke(255);
strokeWeight(4);
rect(100, 200, 270, 170);
```

There are different ways to tell the rect function how to draw our rectangle by using the rectMode function:

```
fill(0, 0);
strokeWeight(4);

stroke(255);
rectMode(CENTER);
rect(100, 200, 270, 170);

stroke(255, 0, 0);
rectMode(CORNER);
rect(100, 200, 270, 170);

stroke(0, 0, 255);
rectMode(CORNERS);
rect(100, 200, 270, 170);
```

For our digital clock we'll use the CENTER mode:

```
fill(0, 0);
stroke(255);
strokeWeight(4);
rectMode(CENTER);
```

```
rect(225, 300, 270, 170);
```

Let's add some text:

```
text("10:55", 225, 300);
```

Notice that the fill function from the rectangle applies to the text as well. So we should call fill again:

```
fill(200);
text("10:55", 225, 300);
```

Let's increase the size and change the font:

```
fill(200);
PFont digitalFont = createFont("OCRAStd", 70);
textFont(digitalFont);
text("10:55", 225, 300);
```

Let's align the text to the center:

```
textAlign(CENTER);
fill(200);
PFont digitalFont = createFont("OCRAStd", 70);
textFont(digitalFont);
text("10:55", 225, 300);
```

Let's add the AM or PM text:

```
PFont legendFont = createFont("Century Gothic", 30);
textFont(legendFont);
text("AM", 225, 360);
```

So far the code looks like this:

```
background(50);
size(450, 720);

fill(0, 0);
stroke(255);
strokeWeight(4);
rectMode(CENTER);
rect(225, 300, 270, 170);

textAlign(CENTER);
fill(200);
PFont digitalFont = createFont("OCRAStd", 70);
textFont(digitalFont);
```

```
text("10:55", 225, 300);

PFont legendFont = createFont("Century Gothic", 30);
textFont(legendFont);
text("AM", 225, 360);
```

And our clock like this:

Now if I want to move the clock to the right:

```
rect(300, 300, 270, 170);
```

So it's a good idea to start using variables to define locations in the canvas for a group of elements:

```
int cx = 225;
int cy = 300;

fill(0, 0);
stroke(255);
strokeWeight(4);
rectMode(CENTER);
rect(cx, cy, 270, 170);

textAlign(CENTER);
fill(200);
PFont digitalFont = createFont("OCRAStd", 70);
textFont(digitalFont);
text("10:55", cx, cy);

PFont legendFont = createFont("Century Gothic", 30);
textFont(legendFont);
```

```
text("AM", cx, cy + 60);
```

So now if I change `cx = 300`, everything will move.

Let's make it dynamic, so that it keeps increasing as time goes by. For this we need to define two functions that processing uses to initialise and to render: **setup** and **draw**. Setup is in charge of initialisation and configurations, and draw is in charge of rendering at each time step as it will be called continuously.

In a small program, it's a good practice to declare the variables outside the functions, initialise them in setup and use them in draw, like so:

```
int cx, cy;
PFont digitalFont, legendFont;

void setup(){
  background(50);
  size(450, 720);

  cx = 225;
  cy = 300;
}

void draw(){
  fill(0, 0);
  stroke(255);
  strokeWeight(4);
  rectMode(CENTER);
  rect(cx, cy, 270, 170);

  textAlign(CENTER);
  fill(200);
  digitalFont = createFont("OCRAStd", 70);
  textFont(digitalFont);
  text("10:55", cx, cy);

  legendFont = createFont("Century Gothic", 30);
  textFont(legendFont);
  text("AM", cx, cy + 60);
}
```

We still haven't given it anything to change in draw. Let's declare a variable that holds the time:

```
int cx, cy;
PFont digitalFont, legendFont;
int minutes;
String minsString;

void setup(){
```

```
  background(50);
  size(450, 720);

  cx = 225;
  cy = 300;

  minutes = 0;
}

void draw(){
  fill(0, 0);
  stroke(255);
  strokeWeight(4);
  rectMode(CENTER);
  rect(cx, cy, 270, 170);

  minsString = String.format("%d:%d", minutes / 60, minutes % 60);

  textAlign(CENTER);
  fill(200);
  digitalFont = createFont("OCRAStd", 70);
  textFont(digitalFont);
  text(minsString, cx, cy);

  legendFont = createFont("Century Gothic", 30);
  textFont(legendFont);
  text("AM", cx, cy + 60);

  minutes++;
}
```

We're drawing on top of what we drew the last step. For this, we need to reset our canvas by simply calling the background in draw rather than setup.

```
...
void draw(){
  background(50);
...
```

It's going too fast, let's make it go slower. For the actual clock we'll use actual time objects but for the moment let's slow down the frequency of the draw function using the **frameRate** function.

```
...
void setup(){
  size(450, 720);
  frameRate(1);
...
```

This function tells processing how many frames per second we want to draw.

Numbers look weird right, they should be 0-padded:

```
...
minsString = String.format("%02d:%02d", minutes / 60, minutes % 60);
...
```

Now let's use dates:

```
import java.util.*;
import java.text.*;

int cx, cy;
PFont digitalFont, legendFont;
long currentTime;
String minsString;

void setup(){
  size(450, 720);
  //frameRate(1);

  cx = 225;
  cy = 300;

  currentTime = new Date().getTime();
}

void draw(){
  background(50);

  fill(0, 0);
  stroke(255);
  strokeWeight(4);
  rectMode(CENTER);
  rect(cx, cy, 270, 170);

  // minsString = String.format("%02d:%02d", minutes / 60, minutes % 60);

  Date time = new Date(currentTime);
  DateFormat ddf = new SimpleDateFormat("HH:mm");
  String digitalDate = ddf.format(time);

  textAlign(CENTER);
  fill(200);
  digitalFont = createFont("OCRAStd", 70);
  textFont(digitalFont);
  text(digitalDate, cx, cy);
```

```
  legendFont = createFont("Century Gothic", 30);
  textFont(legendFont);
  text("AM", cx, cy + 60);

  currentTime += 1000;
}
```

Default frame rate is 60, so in each execution second, we are adding a second to currentTime, so there we add 60 seconds (one minute) in each second. Let's decide how much to add at each time step:

```
import java.util.*;
import java.text.*;

int cx, cy;
PFont digitalFont, legendFont;
long currentTime;
String minsString;
int interval;

void setup(){
  size(450, 720);
  //frameRate(1);

  cx = 225;
  cy = 300;

  currentTime = new Date().getTime();
  interval = 500;
}

void draw(){
  background(50);

  fill(0, 0);
  stroke(255);
  strokeWeight(4);
  rectMode(CENTER);
  rect(cx, cy, 270, 170);

  // minsString = String.format("%02d:%02d", minutes / 60, minutes % 60);

  Date time = new Date(currentTime);
  DateFormat ddf = new SimpleDateFormat("HH:mm");
  String digitalDate = ddf.format(time);

  textAlign(CENTER);
  fill(200);
```

```
  digitalFont = createFont("OCRAStd", 70);
  textFont(digitalFont);
  text(digitalDate, cx, cy);

  legendFont = createFont("Century Gothic", 30);
  textFont(legendFont);
  text("AM", cx, cy + 60);

  currentTime += interval;
}
```

AM is fixed right now, let's change that:

```
  ...
  DateFormat ddf = new SimpleDateFormat("HH:mm");
  String digitalDate = ddf.format(time);

  DateFormat apFormat = new SimpleDateFormat("a");
  String ap = apFormat.format(time);

  textAlign(CENTER);
  fill(200);
  ...
  textFont(legendFont);
  text(ap, cx, cy + 60);

  currentTime += interval;
  ...
```

Let's add interactivity. I want the clock to move where I click:

```
void mouseClicked() {
  cx = mouseX;
  cy = mouseY;
}
```

Now I want to drag it:

```
void mouseDragged() {
  cx = mouseX;
  cy = mouseY;
}
```

Let's start with the analog clock:

```
import java.util.*;
import java.text.*;

```

```
int cx, cy;
PFont digitalFont, legendFont;
long currentTime;
String minsString;
int interval;
boolean digital;

void setup(){
  size(450, 720);
  //frameRate(1);

  cx = 225;
  cy = 300;

  currentTime = new Date().getTime();
  interval = 500;
  digital = true;
}

void draw(){
  background(50);


  // minsString = String.format("%02d:%02d", minutes / 60, minutes % 60);

  Date time = new Date(currentTime);
  DateFormat ddf = new SimpleDateFormat("HH:mm");
  String digitalDate = ddf.format(time);

  DateFormat apFormat = new SimpleDateFormat("a");
  String ap = apFormat.format(time);

  if(digital){
    fill(0, 0);
    stroke(255);
    strokeWeight(4);
    rectMode(CENTER);
    rect(cx, cy, 270, 170);

    textAlign(CENTER);
    fill(200);
    digitalFont = createFont("OCRAStd", 70);
    textFont(digitalFont);
    text(digitalDate, cx, cy);
  }
  else{
    ellipse(cx, cy, 200, 200);
  }
```

```
  legendFont = createFont("Century Gothic", 30);
  textFont(legendFont);
  text(ap, cx, cy + 60);

  currentTime += interval;
}

void mouseClicked() {
  cx = mouseX;
  cy = mouseY;
}
void mouseDragged() {
  cx = mouseX;
  cy = mouseY;
}
void keyPressed() {
  digital = !digital;
}
```

Then:

```
import java.util.*;
import java.text.*;

int cx, cy;
PFont digitalFont, legendFont;
long currentTime;
String minsString;
int interval;
boolean digital;
int radius;
float minutesHand;
float hoursHand;

void setup(){
  size(450, 720);
  //frameRate(1);

  cx = 225;
  cy = 300;

  currentTime = new Date().getTime();
  interval = 2500;
  digital = true;

  radius = 120;
  minutesHand = radius * 0.80;
```

```
  hoursHand = radius * 0.65;
}

void draw(){
  background(50);

  // minsString = String.format("%02d:%02d", minutes / 60, minutes % 60);

  Date time = new Date(currentTime);
  DateFormat ddf = new SimpleDateFormat("HH:mm");
  String digitalDate = ddf.format(time);

  DateFormat apFormat = new SimpleDateFormat("a");
  String ap = apFormat.format(time);

  if(digital){
    fill(0, 0);
    stroke(255);
    strokeWeight(4);
    rectMode(CENTER);
    rect(cx, cy, 270, 170);

    textAlign(CENTER);
    fill(200);
    digitalFont = createFont("OCRAStd", 70);
    textFont(digitalFont);
    text(digitalDate, cx, cy);
  }
  else{
    // get the minutes
    DateFormat minsFormat = new SimpleDateFormat("m");
    String mins = minsFormat.format(time);
    int minsint = Integer.parseInt(mins);

    // get the hours
    DateFormat hoursFormat = new SimpleDateFormat("h");
    String hours = hoursFormat.format(time);
    int hoursint = Integer.parseInt(hours);

    float m = map(minsint, 0, 60, 0, TWO_PI) - HALF_PI;
    float h = map(hoursint + norm(minsint, 0, 60), 0, 24, 0, 2 * TWO_PI) - HALF_PI;

    // Draw the hands of the clock
    stroke(255);
    strokeWeight(4);
    line(cx, cy, cx + cos(m) * minutesHand, cy + sin(m) * minutesHand);
    strokeWeight(6);
    line(cx, cy, cx + cos(h) * hoursHand, cy + sin(h) * hoursHand);
```

```
  // Draw the minute ticks
  strokeWeight(2);
  beginShape(POINTS);
  for (int a = 0; a < 360; a+=6) {
    float angle = radians(a);
    float x = cx + cos(angle) * radius;
    float y = cy + sin(angle) * radius;
    vertex(x, y);
  }
  endShape();

}
...
```

Additions:

```
cx = 225;
cy = 300;

digitalFont = createFont("OCRAStd", 70);
legendFont = createFont("Century Gothic", 30);
header = createFont("Century Gothic", 50);

currentTime = new Date().getTime();
interval = 2500;
digital = true;

...

// at end of draw()
textFont(header);
text("Simple clock",  225, 100);

DateFormat df = new SimpleDateFormat("EEEE MM/dd/yyyy HH:mm");
String reportDate = df.format(time);
textFont(legendFont);
text(reportDate, 225, 550);
```