ControlP5 tutorial

This tutorial covers the basics of using a GUI library called ControlP5. This library provides functionality for user interface elements such as buttons, checkboxes, sliders, etc. This library is not required to be used in assignments but a GUI library is recommended for user interaction and ControlP5 is one of them.

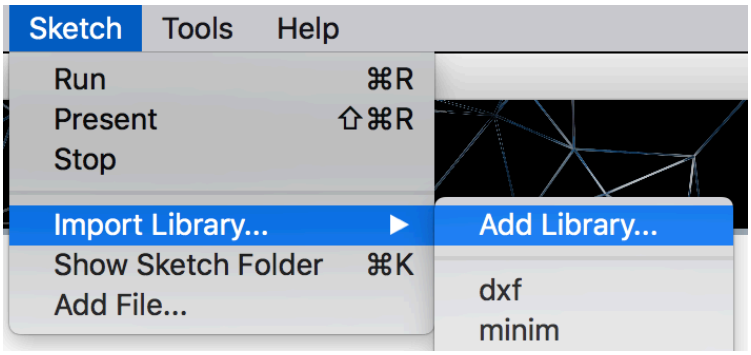
We'll start with a slightly modified version of the simple clock used in the previous tutorial, also uploaded in the course page:

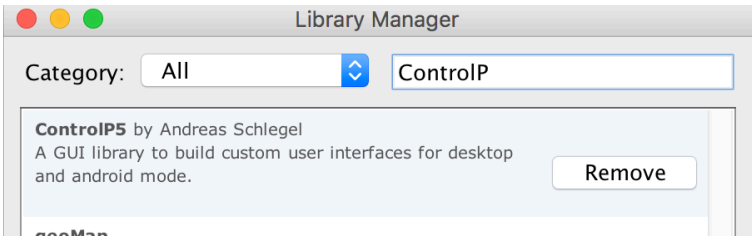
```
import java.util.*;
import java.text.*;

int cx, cy;
int minutes;
String minsString;
long currentTime;
boolean digital;

int radius;
float minutesHand;
float hoursHand;

int interval;

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

  cx = 225;
  cy = 300;
  minutes = 0;

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

  radius = 120;
  minutesHand = radius * 0.80;
  hoursHand = radius * 0.65;

  interval = 1000;
}

void draw(){
  background(50);

  Date time = new Date(currentTime);

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

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

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

```
  else{
    // Analog clock drawn here
    int minutes = (int)((currentTime / 1000) / 60) % 60;
    int hours = (int)(((currentTime / 1000) / 60) / 60) % 24;

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

    // drawing minute and hour clock hands
    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);

    // drawing clock 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();
  }

  // Getting the AM or PM text
  DateFormat apFormat = new SimpleDateFormat("a");
  String ap = apFormat.format(time);

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

  // Displaying header and footer texts
  PFont header = createFont("Century Gothic", 50);

  textFont(header);
  textAlign(CENTER);
  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);

  currentTime += interval;
}
```

This gives us the following interaface:

Now let's first install the ControlP5 library using the processing interface:





Now let's add it to our project, declare it and initialise it:

```
import java.util.*;
import java.text.*;
import controlP5.*;

...
```

```
int interval;

ControlP5 cp5;

...


  interval = 1000;

  cp5 = new ControlP5(this);

...
```

We can now add elements. Let's add two buttons to change between the digital and analog clock:

```
  ...

  cp5 = new ControlP5(this);
  cp5.addButton("digital")
   .setPosition(50, 650)
   .setSize(150, 50)
   ;
  cp5.addButton("analog")
   .setPosition(250, 650)
   .setSize(150, 50)
   ;

  ...
```

The buttons are now rendered into the interface:

This adds the buttons to the interface but they don't do anything yet. Let's add handlers to those buttons. By default, ControlP5 looks for a function with the name of the button to trigger click events. At the bottom of the code, add:

```
...

public void digital() {
  digital = true;
}

public void analog() {
  digital = false;
}
```

Now our buttons change the clock to either digital or analog.

Let's dynamically change the speed of our clock. For that let's first add a speed variable and initialise it:

```
...

int interval;
int speed;

ControlP5 cp5;

...

  interval = 1000;
  speed = 0;

...
```

We will multiply this speed to the interval being used. At the end of the draw method:

```
...

  currentTime += interval * speed;
}

...
```

Let's add a slider to handle this. After the buttons:

```
  ...

  cp5.addButton("analog")
   .setPosition(250, 650)
   .setSize(150, 50)
```

```
    ;
  cp5.addSlider("speed")
    .setPosition(50,140)
    .setSize(20,100)
    .setRange(0,10)
    .setNumberOfTickMarks(10)
    ;

...
```

This slider takes a variable of the same name and modifies it according to user input. This is how it looks so far:

The colours of these elements seem out of place, let's change them:

```
...

ControlP5 cp5;
CColor controlsColours;

...

  speed = 0;
  controlsColours = new CColor(0x99ffffff, 0x55ffffff, 0xffffffff, 0xffffffff,
0xffffffff);

...
```

This is how ControlP5 handles a colour palette for element styles. This hold information regarding the colour for background, text, border, pressed, etc. Let's add this colours to our elements:

```
...

  cp5.addButton("digital")
   .setPosition(50, 650)
   .setSize(150, 50)
   .setColor(controlsColours)
   ;
  cp5.addButton("analog")
   .setPosition(250, 650)
   .setSize(150, 50)
   .setColor(controlsColours)
   ;

   cp5.addSlider("speed")
   .setPosition(20,250)
   .setSize(20,100)
   .setRange(0,10)
   .setNumberOfTickMarks(10)
   .setColor(controlsColours)
   ;

...
```

Much better! Now let's add a slider that lets us control time.

```
...

  cp5.addSlider("speed")
   .setPosition(20,250)
   .setSize(20,100)
   .setRange(0,10)
   .setNumberOfTickMarks(10)
   .setColor(controlsColours)
   ;

  cp5.addSlider("timeline")
   .setPosition(50,600)
   .setSize(350,20)
   .setRange(0,1000)
   .setColor(controlsColours)
   ;

...
```

Let's make it move as time goes by. We'll use a variable called startTime. I'm addint a variable previousUpdate which I'll explain in a second.

```
...

int interval;
int speed;
long startTime;
long previousUpdate;

...

  interval = 1000;
  speed = 0;
  startTime = currentTime;
  previousUpdate = 0;

...
```

To update the slider:

```
...

  currentTime += interval * speed;
  cp5.getController("timeline").setValue((currentTime - startTime) / 60000);
}
```

```
...
```

Let’s not update it at every draw. Let’s make it update only every passing “minute”:

```
...

  currentTime += interval * speed;
  if(currentTime - previousUpdate > 60000){
    cp5.getController("timeline").setValue((currentTime - startTime) / 60000);
    previousUpdate = currentTime;
  }

...
```

The slider now moves as time passes by. Now let’s control time using the slider:

```
...

public void timeline(int value){
  currentTime = (value * 60000) + startTime;
}

...
```

Final product: