Práctica 8 :
Introducción a `p5.js`

David Alberto Medina Medina
Dr. Modesto Fernando Castrillon Santana

8 de abril de 2019

Índice general

Introducción

P5.js es una librería de `Javascript` que, al igual que *Processing*, es utilizada para la creación artística y diseño tecnológico en soporte web.

En este documento se exponen los detalles de implementación de una aplicación web de dibujo que hemos denominado `SketchJS`. Esta experiencia nos permitirá utilizar las primitivas básicas que incluye esta librería librería.

En concreto, se utilizará la librería básica `p5.js` para implementar la parte gráfica de la aplicación y `p5.dom` para la generación de los elementos `DOM` que componen el panel que controla los parámetros de pincel.

Método y materiales

2.1. Materiales

El desarrollo de este proyecto se ha llevado a cabo utilizando el IDE de desarrollo de aplicaciones web de *JetBrains*, *WebStorm*, y las siguientes librerías:

Librería `p5.min.js` Contiene las primitivas básicas que nos permiten implementar los elementos gráficos de la aplicación web (P5.js, s.f.-b).

Librería `p5.dom.min.js` Utilizada para generar los elementos `DOM` que conforman el panel de control de los parámetros del pincel (P5.js, s.f.-a).

2.2. Método

El archivo `sketch.js` contiene todas las rutinas que hacen posible la implementación de esta aplicación de dibujo. Según la función que desempeñan, podemos clasificarlas en:

- Rutina de inicialización y bucle principal
- Rutinas de generación del panel de control utilizando `DOM`
- Eventos

2.2.1. Rutina de inicialización y bucle principal

El conjunto de rutinas que forman parte de este grupo son:

- `setup()`
- `draw()`

Método `setup()`

La librería gráfica `p5.js` exige que se declare un método `setup()` en donde se establecerán las configuraciones iniciales que se establecerán antes de pintar sobre el lienzo.

En primer lugar, se genera el panel de control que presenta al usuario los parámetros configurables del pincel, tal como el color y grosor del mismo. El panel de control se genera al llamar al método `generatePanel()`.

Posteriormente, se ajusta el tamaño del lienzo a `600x600` píxeles al llamar a la primitiva `createCanvas()` y se pinta en blanco el fondo del lienzo.

```
function setup()
{
    generatePanel();

    createCanvas(600,600);
    background(255);

    myRGB = color('black');
}
```

Método `draw()`

El método `draw()` cambia el icono del ratón por un pincel redondo llamando a la primitiva `cursor()`.

```
function draw()
{
    cursor("icon/brush.cur", 17, 16);
}
```

2.2.2. Rutinas de generación del panel de control utilizando `DOM`

Se presentan a continuación las rutinas responsables de la creación del panel:

- `generatePanel()`
- `createThicknessController()`
- `createColorController()`

Método `generatePanel()`

Se crea el contenedor principal con etiquetas `<div>`. En este método se generan:

- El *slider* mediante el cual el usuario podrá seleccionar el grosor del pincel.
- La paleta de selección de colores
- Un botón de *reset* mediante el cual se limpia completamente el lienzo de cualquier trazo que el usuario haya podido realizar

Una vez creado todos estos elementos, se anidan todos estos elementos dentro del contenedor `<div>` con la primitiva `child()` de la librería `p5.dom.min.js`.

Finalmente, se establece el estilo del contenedor utilizando `CSS` y la primitiva `style()` que ofrece la librería esta misma librería.

Puede observarse como se genera un evento que llama al método `clear()` cuando se realiza click sobre el botón `Clear`. La primitiva que acciona este evento es `mouseClicked()`.

```
function generatePanel()
{
    panel = createDiv("<h1 style='text-align: center'>Control Panel</h1
    >");
```

```
4
5       let thicknessDiv = createThicknessController();
6
7       let colorPickerDiv = createColorController();
8
9       let clearButton = createButton("Clear");
10      clearButton.position(30, 125);
11      clearButton.mouseClicked(clear);
12
13      panel.child(thicknessDiv);
14      panel.child(colorPickerDiv);
15      panel.child(clearButton);
16
17      panel.style('color', '#D0EEFB');
18      panel.style('background-color', '#0088C2');
19      panel.style('position', 'absolute');
20      panel.style('width', '600px');
21      panel.style('height', '25%');
22  }
```

Método `createThicknessController()`

Este método genera y posiciona, mediante manipulaciones `DOM`, el `<div>` los elementos que forman parte del control del grosor del pincel.

Cada vez que se modifica el valor del *slider* se llama a la primitiva `input()` que acciona un evento que llamará al método `updateThicknessVal()`.

Puede observarse, además, cómo al llamar a la primitiva `attribute()` se han añadido atributos a la etiqueta `<input>` que se genera en el documento `HTML` cuando se llama a la primitiva `createSlider()`.

```
1   function createThicknessController()
2   {
3       let thicknessDiv = createDiv("<label for='thicknessSlider'>
        Thickness : </label>");
4
5       thicknessSlider = createSlider(1, 100, 5);
6       thicknessSlider.attribute("id", "thicknessSlider");
7       thicknessSlider.attribute("name", "thicknessSlider");
8       thicknessSlider.position(90, -2);
9       thicknessSlider.input(updateThicknessVal);
10
11      thickness = thicknessSlider.value();
12      thicknessSpan = createSpan(thicknessSlider.value());
13      thicknessSpan.position(230, 0);
14      thicknessSpan.style('font-weighht', 'bold');
15
16      thicknessDiv.child(thicknessSlider);
17      thicknessDiv.child(thicknessSpan);
18      thicknessDiv.child(thicknessSpan);
19
20      thicknessDiv.position(30, 75);
21      return thicknessDiv;
22  }
```

Método `createColorController()`

Este método genera el botón que abre la paleta de selección de colores. Este botón se genera con la llamada a `createColorPicker()`. Como puede

observarse, se establece el color negro por defecto.

Cada vez que se cambie el valor de este botón, se acciona un evento que llama al método `changeColor()`. Este evento se produce al llamar a la primitiva `input()`.

```
function createColorController()
{
    let colorPickerDiv = createDiv('<span">Choose a color : </span>');
    colorPicker = createColorPicker('#000000');
    colorPicker.position(125, -5);
    colorPicker.input(changeColor);

    colorPickerDiv.child(colorPicker);
    colorPickerDiv.position(325, 75);
    return colorPickerDiv;
}
```

2.2.3. Eventos

Los eventos que aquí se han implementado se pueden clasificar en los siguientes grupos:

- Eventos relacionados con la generación de trazos del pincel
- Evento relacionado con la actualización del grosor del pincel
- Evento relacionado con la actualización del color del trazo.
- Evento responsable de limpiar el lienzo de trazos.

Eventos relacionados con la generación de trazos

Los trazos son generados cuando un evento llama a la función `drawStroke()` (*callback function*). Aquí se genera un trazo con el valor del grosor y color del pincel seleccionado por el usuario. El grosor se ajusta al pasarle como argumento a la primitiva `strokeWeight()` el valor de la variable global `thickness` mientras que el color queda establecido al llamar a la primitiva `stroke()`, pasándo como argumento la variable global `myRGB`.

Cada trazo se define como un conjunto de líneas que tiene como punto inicial la posición del ratón en el *frame* actual del ratón y como punto final, la posición del ratón en el *frame* anterior. Este conjunto de líneas son mostrados por pantalla tras llamar a la primitiva `line()`.

```
function drawStroke()
{
    push();
    strokeWeight(thickness);
    stroke(myRGB);
    line(mouseX, mouseY, pmouseX, pmouseY);
    pop();
}
```

Los eventos `mousePressed()` y `mouseDragged()` llaman respectivamente a `drawStroke()` (función *callback*) cuando se hace clic sobre el lienzo y se arrastra el ratón sobre el mismo.

```
function mousePressed()
{
    drawStroke();
}

function mouseDragged()
{
    drawStroke();
}
```

Evento relacionado con el grosor del pincel

Tal y como se describe en la sección 2.2.2, el evento `input()` que se genera sobre el *slider* del panel que controla el grosor del pincel llama a una función *callback* denominada `updateThicknessVal()`.

Esta función ajusta el valor de la variable global `thickness` al valor que el usuario haya seleccionado en el *slider*, además de actualizar la etiqueta `<span>` asociada al mismo, el cual muestra al usuario el valor del grosor del trazo elegido.

```
function updateThicknessVal()
{
    thickness = thicknessSlider.value();
    thicknessSpan.html(thickness);
}
```

Evento relacionado con la actualización del color del trazo

En la sección 2.2.2 se describe como el botón que activa la paleta de selección de colores acciona un evento que llama al *callback* denominado `changeColor()`.

Esta función ajusta el valor de la variable global `myRGB` al valor del color elegido. Obsérvese como se llama al método `color()` del objeto de selección de color.

```
function changeColor()
{
    myRGB = colorPicker.color();
}
```

Evento responsable de limpiar el lienzo de trazos.

Cada vez que se hace clic sobre el botón `Clear` (ver sección 2.2.2) del panel de control se acciona un evento que llama al método `clear()`, el cual limpia el lienzo tras llamar a la primitiva `background()` que pintará todo el lienzo de color blanco.

```
function clear()
{
    background(255);
}
```

Resultados y Discusión

La aplicación obtenida con `p5.js` nos permite dibujar sobre un lienzo en blanco y cambiar entre los diferentes colores y grosor de pincel. Se observa la funcionalidad añadida que permite limpiar el lienzo de trazos, permitiéndonos dibujar nuevos trazos sobre el mismo de manera rápida y sencilla.

Como se puede observar, `p5.js` es una librería escrita en `JavasScript` que permite trabajar con gráficos en un soporte web de forma análoga a *Processing*.

Puede observarse lo fácil que es adaptarse de este framework y la similitud de sus diversas primitivas con las de *Processing*.

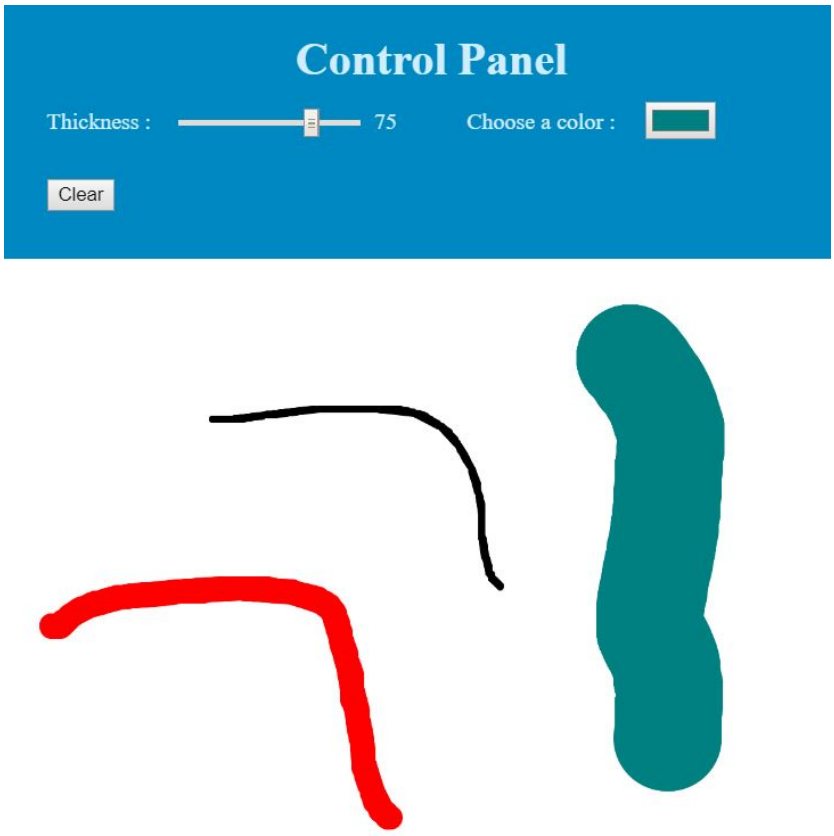

Figura 3.1: Algunos trazos de diferente grosor y color realizados con `SketchJS`

Referencias

P5.js. (s.f.-a). *P5.dom Library.* `https://p5js.org/reference/#/libraries/p5.dom/`. Autor. (Accessed: 2019-04-08)

P5.js. (s.f.-b). *P5.js Reference.* `https://p5js.org/reference/`. Autor. (Accessed: 2019-04-08)