

Compte rendu

TP : SIR_WEB 2016

**Baron Aziliz
Mellali Ismail**

Introduction

Ce tp consiste à réaliser une application Web pour faire du dessin avec canvas (HTML5 et JavaScript) en utilisant le patron de conception MVC (Modèle-Vue-Contrôleur) pour dessiner des rectangles, des lignes, ainsi que définir leur couleur et leur épaisseur de trait.

Interaction : (Drag-n-Drop)

Créer la classe DnD contenant les coordonnées de la position initial du DnD et celles de la position finale (x1,x2,y1,y2) et les 3 fonctions correspondant aux 3 événements (pression, déplacement, relachement).

Voir fichier **interaction.js**

Puis on a associé les 3 fonctions aux évènements du canvas :

```
canvas.addEventListener("mousedown", this.pression, false);  
canvas.addEventListener("mousemove", this.deplacement, false);  
canvas.addEventListener("mouseup", this.relachement, false);
```

Le modèle

Le modele contient 4 classes : classe Drawing() représente un dessin qui contient un tableau de formes, une forme étant soit un rectangle(classe Rectangle), soit une ligne (classe Ligne).

fichier **model.js**

```
function Drawing() {  
  this.formes= [];  
  this.addSh = function(form){  
    this.formes.push(form);//ajouter une forme dans formes[]  
  };  
  this.deleteSh= function(id){  
    this.formes.splice(id,1);  
    //alert(JSON.stringify(this.formes));//afficher les shapes de formes  
  };  
}  
  
function Shape(color,size,type){  
  this.color=color;  
  this.size=size;  
  this.type=type;  
}
```

```
function Ligne(x1,y1,x2,y2,size,color){  
  Shape.call(this,size,color,"Ligne");  
  
  this.x1=x1;  
  this.y1=y1;  
  this.x2=x2;  
  this.y2=y2;  
}  
  
function Rectangle(width,height,x1,y1,size,color){  
  Shape.call(this,size,color,"Rectangle");  
  
  this.width=width;  
  this.height=height;  
  this.x1=x1;  
  this.y1=y1;  
}
```

La vue

Dans le fichier **view.js** on a ajouté les fonctions d’affichage dans les classes (Drawing, Rectangle, Ligne) créées dans le modèle.

```

Rectangle.prototype.paint = function(ctx) {
  //TODO Manager color
  ctx.strokeStyle = this.color;
  ctx.lineWidth = this.size;
  ctx.rect(this.width, this.height, this.x1, this.y1);
  ctx.stroke();
};

Ligne.prototype.paint = function(ctx) {
  //TODO Manager color
  ctx.strokeStyle = this.color;
  ctx.lineWidth = this.size;
  ctx.beginPath();
  ctx.moveTo(this.x1, this.y1);
  ctx.lineTo(this.x2, this.y2);
  ctx.stroke();
};

Drawing.prototype.paint = function(ctx) {
  console.log(this.getForms());
  ctx.fillStyle = '#F0F0F0'; // set canvas' background color
  ctx.fillRect(0, 0, canvas.width, canvas.height);
  this.getForms().forEach(function(eltDuTableau) {
    // now fill the canvas
    eltDuTableau.paint(ctx);
  });
};
};

```

Le contrôleur

Lié une interaction DnD à Pencil (un crayon pour dessiner) dans le fichier `controller.js`.

Dans chacune des 3 fonction de DnD on va ajouter un appel aux fonctions :

`interactor.onInteractionStart(this)/interactor.onInteractionUpdate(this)/interactor.onInteractionEnd(this)`. Pour créer une forme(Ligne ou Rectangle) et la mettre à jour lorsque l'utilisateur bouge la souris et l'ajouter au dessin lors du relâchement.

fichier `controller.js`

```

var editingMode = { rect: 0, line: 1 };

function Pencil(ctx, drawing, canvas) {
  this.currEditingMode = editingMode.line;
  this.currLineWidth = 5;
  this.currColour = '#000000';
  this.currentShape = 0;
  this.currenttype=""; // variable pour tester sur le type de la forme (Ligne ou Recangle)

  // Liez ici les widgets à la classe pour modifier les attributs présents ci-dessus.

  new DnD(canvas, this);

  // Implémentez ici les 3 fonctions onInteractionStart, onInteractionUpdate et onInteractionEnd

  this.onInteractionStart=function(DnD){
    //initialiser la couleur et l'épaisseur
    this.currLineWidth=document.getElementById("spinnerWidth").value;
    this.currColour =document.getElementById("colour").value;
    console.log(this.currLineWidth,this.currColour);

    //tester la forme choisi (ligne ou rectangle)
    if(document.getElementById("butLine").checked){
      this.currentShape = new Ligne(DnD.x1,DnD.y1,DnD.x2,DnD.y2,this.currColour,this.currLineWidth);
      this.currEditingMode = editingMode.line
    }else if (document.getElementById("butRect").checked){
      this.currentShape = new Rectangle(0,0,DnD.x1,DnD.y1,this.currLineWidth,this.currColour);
      this.currEditingMode = editingMode.Rectangle
    }
  }

  }.bind(this);

```

```

  this.onInteractionUpdate = function(DnD) {
    //récupérer les coordonnées
    if(document.getElementById("butLine").checked){

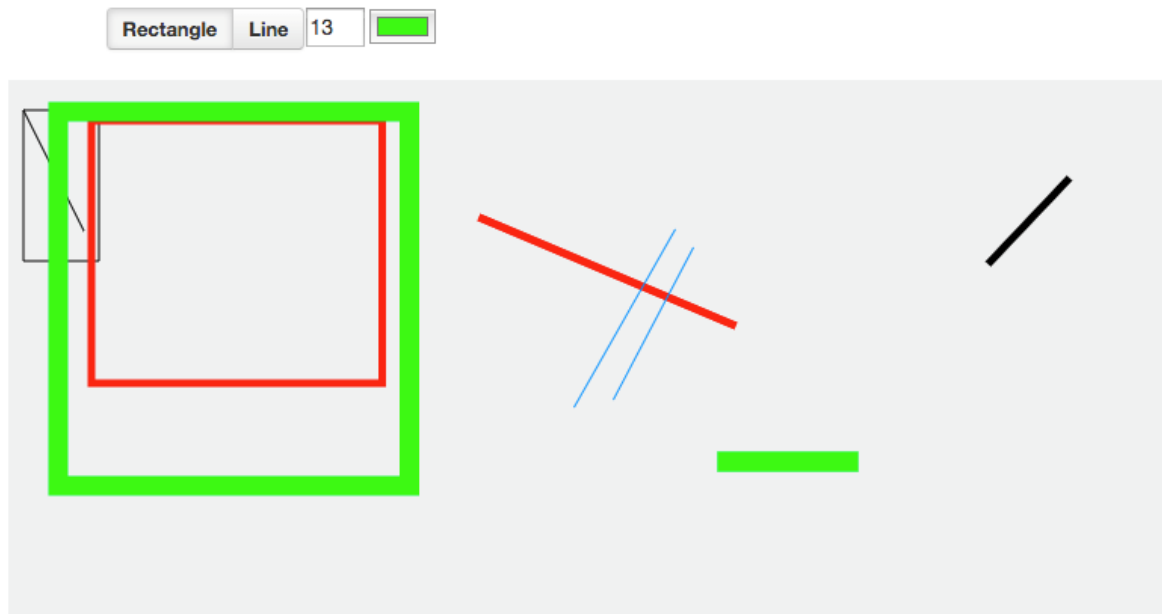
      this.currentShape.x2=DnD.x2;
      this.currentShape.y2=DnD.y2;
      this.currenttype="Ligne";
    }else if (document.getElementById("butRect").checked){

      this.currentShape.width=DnD.x2 - DnD.x1;
      this.currentShape.height=DnD.y2 - DnD.y1;
      this.currenttype="Rectangle";
    }
  }.bind(this);

  this.onInteractionEnd = function() {
    //dessiner la forme
    this.currentShape.paint(ctx);
    drawing.addSh(this.currentShape);
    updateShapeListe(drawing);
  }.bind(this);
};

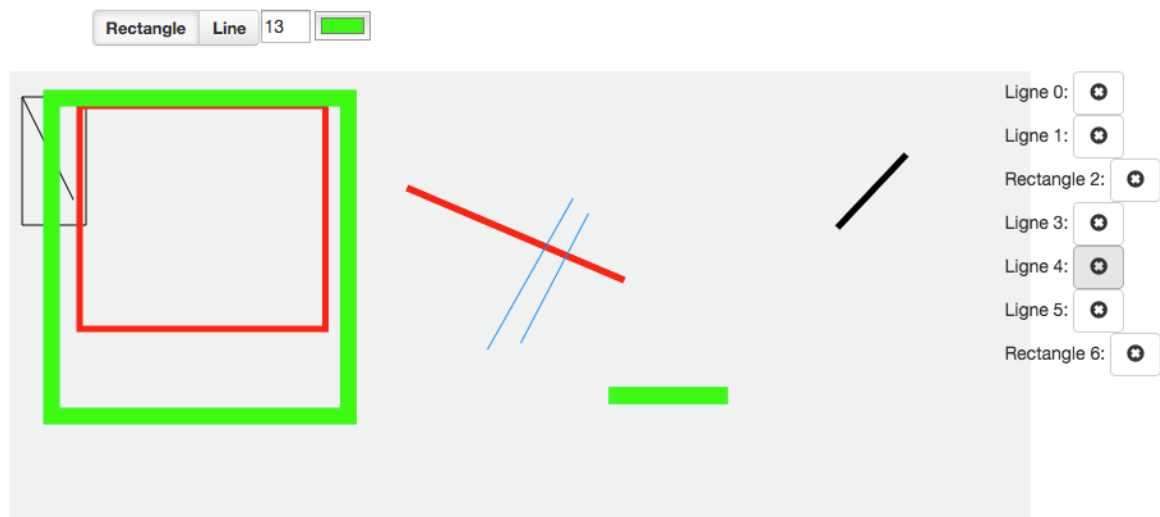
```

Resultat



Liste des modifications

Afficher une liste de formes (ListeShape) pour chaque forme dessinée et pouvoir la supprimer de canvas avec un button .



Remarque: La suppression ne fonctionne pas parfaitement. En effet la forme est bien supprimée mais toutes les autres aussi.