



# SIR TP 4 – Mini editeur web

## Compte rendu

Pauline Flambard  
Sandra Villafuerte

## Interaction : le glisser-déposer (Drag-n-Drop)

Créer une classe DnD contenant les 4 attributs suivants initialisé à 0 : les coordonnées de la position initial du DnD ; celles de la position finale.

```
function DnD(canvas, interactor) {  
    this.initialX = 0;  
    this.initialY = 0;  
    this.finalX = 0;  
    this.finalY = 0;  
    ...  
}
```

Déclarer 3 fonctions à cette classes correspondant aux 3 types d'événements à gérer.

```
this.onPress = function (evt){  
    if(!this.isPressed){  
        this.isPressed = true;  
        var pos = getMousePosition(canvas, evt);  
        this.initialX = pos.x;  
        this.initialY = pos.y;  
        console.log("initialX : " + this.initialX + " // initialY : " + this.initialY);  
    }  
}.bind(this);
```

```
this.onMove = function(evt){  
    if(this.isPressed){  
        var pos = getMousePosition(canvas, evt);  
        this.finalX = pos.x;  
        this.finalY = pos.y;  
        console.log("Move");  
        console.log("finalX : " + this.finalX + " // finalY : " + this.finalY);  
    }  
}.bind(this);
```

```
this.onUp = function (evt){  
    if(this.isPressed){  
        this.isPressed = false;  
        var pos = getMousePosition(canvas, evt);  
        this.finalX = pos.x;  
        this.finalY = pos.y;  
        console.log("finalX : " + this.finalX + " // finalY : " + this.finalY);  
    }  
}.bind(this);
```

## Implémenter ces fonctions.

```
function getMousePosition(canvas, evt) {  
  var rect = canvas.getBoundingClientRect();  
  return {  
    x: evt.clientX - rect.left,  
    y: evt.clientY - rect.top  
  };  
};
```

## Enregistrer chaque fonction auprès du canvas (addEventListener).

```
canvas.addEventListener('mousedown', this.onPress, false);  
canvas.addEventListener('mousemove', this.onMove, false);  
canvas.addEventListener('mouseup', this.onUp, false);
```

## Le modèle

### model.js

```
function Form(couleur, epaisseur ){  
  this.couleur=couleur;  
  this.epaisseur=epaisseur;  
}
```

```
function Line( startX, startY, endX, endY, epaisseur, couleur){  
  Form.call(this, couleur, epaisseur);  
  this.startX = startX;  
  this.startY = startY;  
  this.endX = endX;  
  this.endY = endY;  
  
}
```

```
function Rectangle(startX, startY, longueur, largeur, epaisseur, couleur){  
  Form.call(this, couleur, epaisseur);  
  this.startX = startX;  
  this.startY = startY;  
  this.endX = longueur;  
  this.endY = largeur;  
}
```

```
function Drawing() {  
  this.forms = [];  
  this.addShape = function(shape){  
    this.forms.push(shape);  
  };  
  this.removeShape = function(id){  
    this.forms.splice(id,1);  
  };  
};
```

## La vue

### View.js

```
Rectangle.prototype.paint = function(ctx) {  
  ctx.beginPath();  
  ctx.rect(this.startX, this.startY, this.endX, this.endY);  
  ctx.lineWidth = this.epaisseur;  
  ctx.strokeStyle = this.couleur;  
  ctx.stroke();  
};
```

```
Line.prototype.paint = function(ctx) {  
  ctx.beginPath();  
  ctx.moveTo(this.startX, this.startY);  
  ctx.lineTo(this.endX, this.endY);  
  ctx.lineWidth = this.epaisseur;  
  ctx.strokeStyle = this.couleur;  
  ctx.stroke();  
};
```

```
Drawing.prototype.paint = function(ctx) {  
  ctx.fillStyle = '#F0F0F0';  
  ctx.fillRect(0, 0, canvas.width, canvas.height);  
  this.forms.forEach(function(eltDuTableau) {  
    eltDuTableau.paint(ctx);  
  });  
};
```

### Tester votre code (main.js)

```
// Code temporaire pour tester l'affiche de la vue  
var rec = new Rectangle(50, 30, 76, 188, 5, '#ff0000');  
rec.paint(ctx);  
var ligne = new Line(180, 130, 300, 160, 5, '#ff0000');  
ligne.paint(ctx);
```



## Le contrôleur

Dans chacune des 3 fonctions de DnD, ajouter un appel aux fonctions `interactor.onInteractionStart(this);`, `interactor.onInteractionUpdate(this);`, `interactor.onInteractionEnd(this);`.

```
this.onPress = function (evt){
    if(!this.isPressed){
        ...
        pencil.onInteractionStart(this);
    }
}.bind(this);

this.onMove = function (evt){
    if(this.isPressed){
        ...
        pencil.onInteractionUpdate(this);
    }
}.bind(this);

this.onUp = function (evt){
    if(this.isPressed){
        ...
        pencil.onInteractionEnd(this);
    }
}.bind(this);
```

## Implementation de fonctions dans la classe Pencil

```
this.onInteractionStart = function(DnD){
    currColour = $('#colour').val();
    currLineWidth = $('#spinnerWidth').val();
    if ($('#butRect')[0].checked){
        var rec = new Rectangle(DnD.initialX, DnD.initialY, DnD.finalX-
            DnD.initialX, DnD.finalY- DnD.initialY, currLineWidth, currColour);
    }else{
        var line = new Line(DnD.initialX, DnD.initialY, DnD.finalX, DnD.finalY,
            currLineWidth, currColour);
    }
}.bind(this);

this.onInteractionUpdate = function(DnD){
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    drawing.paint(ctx);
    currColour = $('#colour').val();
    currLineWidth = $('#spinnerWidth').val();
    if ($('#butRect')[0].checked){
        var rec = new Rectangle(DnD.initialX, DnD.initialY, DnD.finalX-
            DnD.initialX, DnD.finalY- DnD.initialY, currLineWidth, currColour);
    }
```

```

        rec.paint(ctx);
    }else{
        var line = new Line(DnD.initialX, DnD.initialY, DnD.finalX, DnD.finalY,
            currLineWidth, currColour);
        line.paint(ctx);
    }
}.bind(this);

this.onInteractionEnd = function(DnD){
    currColour = $('#colour').val();
    currLineWidth = $('#spinnerWidth').val();
    if ($('#butRect')[0].checked){
        var rec = new Rectangle(DnD.initialX, DnD.initialY, DnD.finalX-
            DnD.initialX, DnD.finalY- DnD.initialY, currLineWidth, currColour);
        rec.paint(ctx);
        console.log(rec);
    }else {
        var line = new Line(DnD.initialX, DnD.initialY, DnD.finalX, DnD.finalY,
            currLineWidth, currColour);
        line.paint(ctx);
        console.log(line);
    }
    console.log(drawing.forms);
}.bind(this);

```

## Liste des modifications

### Fonction *updateShapeList*

Drawing.prototype.**updateShapeList** = function(shape){

```

    /****Variables****/
    //Obtenir la liste a modifier a l'aide de son id
    var list = document.getElementById('shapeList');
    //Creation de l'element li (item de la liste)
    var li = document.createElement('li');
    //Creation d'un id pour les li et boutons
    var id = list.childNodes.length;
    //Creation de l'element bouton qui permettra de supprimer les formes
    var bouton = document.createElement('button');
    //Creation element span
    var span = document.createElement('span');
    //Position debut X
    var sx = shape.startX;
    //Position debut Y
    var sy = shape.startY;
    //Position final X
    var ex = shape.endX;
    //Position final Y
    var ey = shape.endY;

```

```

/****Bouton effacer****/
//On change l'id du bouton
bouton.setAttribute('id', id);
//Changement de l'attribut class
bouton.setAttribute('class', 'btn btn-default')
//On ajoute l'icone "croix" au span
span.setAttribute('class', 'glyphicon glyphicon-remove-sign');
//On ajoute le span au bouton
bouton.appendChild(span);
//Function onclick
bouton.setAttribute('onClick', 'drawing.deleteShape('+id+')');

/****Elements li****/
//On ajoute le bouton au item li
li.appendChild(bouton);

if (shape instanceof Rectangle){
    //texte pour un rectangle
    li.appendChild(document.createTextNode('Rectangle'+(''+sx+'',''+sy+'',''+ex+'',''+ey+'')));
} else {
    //texte pour une ligne
    li.appendChild(document.createTextNode('Line'+(''+sx+'',''+sy+'',''+ex+'',''+ey+'')));
}

//On change l'attribut id du li
li.setAttribute('id', 'li'+id);
//On change l'attribut class du li
li.setAttribute('class', 'list-group-item');

/****La liste de formes****/
//On ajoute l'element li au liste 'shapeList'
list.appendChild(li);
};

```

### **Fonction pour supprimer la forme correspondante dans le dessin**

```

Drawing.prototype.deleteShape = function(id){
    //Obtenir l'element li qui contient le bouton que l'utilisateur a appuyé pour
    //effacer une forme
    var li = document.getElementById('li'+id);
    //Obtenir l'index de li choisis
    var index = $(li).index();
    //Effacer l'element li de la liste
    li.remove();
    //Effacer la forme qui a le meme index que la ligne (li) effacée. La forme est
    //effacé du array 'forms'
    this.removeShape(index);
}

```

```

//Effacer tout le contenu du canvas
ctx.clearRect(0, 0, canvas.width, canvas.height);
//Redessiner le contenu du array 'forms'
drawing.paint(ctx);
};

```

## Dessiner un cercle

Nous avons ajouté les fonctionnalités nécessaires pour dessiner un cercle.

## Resultat final

