Steve Visinand

INF3DLM\_B

he-arc

Projet p3

Présenté à : Mr. Grunenwald

29 September 2015

Editeur Bitmap Online

# Abstract

Table des matières

[Le résumé 2](#_Toc431288677)

[En bref: 2](#_Toc431288678)

[Introduction 4](#_Toc431288679)

[L’analyse 4](#_Toc431288680)

[L’implémentation 4](#_Toc431288681)

[Conclusion 4](#_Toc431288682)

# Introduction

Ni trop longue ni trop courte (un peu moins d’un dixième de l’ensemble), l’introduction ne doit pas empiéter sur le développement qu’elle ne fait qu’annoncer. Il n’y a pas de recette pour une introduction mais on peut, pour lancer le sujet, commencer par une citation ou par une question ou encore par une idée paradoxale qui attirera l’attention du lecteur. On doit ensuite

exposer le sujet (ou le thème, la matière prise en considération) en faisant valoir son importance et son originalité

articuler la problématique soulevée

annoncer le plan

Cela revient à annoncer les différentes parties du document dans l’ordre selon lequel elles seront abordées (à travers la problématique soulevée et ses solutions).

<http://www.iue.it/ecrire/m4_plan_intro.htm>

# Analyse

# Conception

## Objectifs

## Liste des tâches

## Planning et suivi

# Développement

## Itération 0

### Fonctionnalité

#### Problèmes rencontrés, analyse

#### Solutions retenues, implémentation

tests

## Itération 1

### Objectifs

**Voici les objectifs de cette itération :**

* Interface minimaliste du logiciel en HTML5 :
  + zone de dessin adaptable
  + panneaux des outils
  + barre de menu
* Outils de dessin standards
  + Pinceau (avec système de brush)
  + Gomme (avec système de brush)
  + Système de brosse (forme ronde uniquement)
    - Opacité de la brosse
    - Sélection de la couleur (color picker)
    - Taille de la brosse

### Mise en place des controlleurs

#### Problèmes rencontrés, analyse

#### Solutions retenues, implémentation

Tests

### Pencil et Brushs

#### Problèmes rencontrés, analyse

Pencil est l’outils implémenté lors de cette itération, avec ce dernier on peu dessiner à main levée sur le canvas.

Les spécifications mentionnaient la possibilité de dessiner sur le canvas à l’aide de « brosses » (brush). L’intérêt pour l’utilisateur n’est pas discutable, on peu dans la théorie, avec ce système, dessiner avec des pinceaux de formes/couleur et tailles différentes.

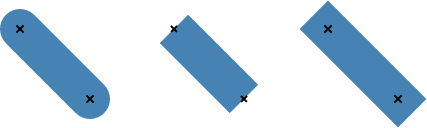
Ainsi la classe « Pencil » (qui dérive de « Tool ») utilise une « Brush ». Ce cas représente clairement l’utilisation du Design pattern « State », la méthode « drawBrush(x,y) » est implémentée dans chaque « Brush » et c’est le « Pencil » qui dessine cette dernière aux endroits voulus.

Le premier problème rencontré avec ce système à été la vitesse de capture de la position de la souris sur le canvas. Le navigateur ne détecte pas assez fréquemment la position du curseur pour obtenir un trait ininterrompu lors du dessin.

La possibilité d’utiliser les « Path » de canvas à été envisagée mais il n’était alors plus possible d’utiliser un système de brush comme désiré.

Canvas prévoit uniquement la possibilité de personnalisé les « bouts » des traits en plus de leur couleur et opacité et de leur ajouter une ombre.

Configurations des bouts des traits :



(<http://www.alsacreations.com/tuto/lire/1511-introduction-canvas.html)>

#### Solutions retenues, implémentation

Pour palier au problème de trait discontinu, un algorithme de complétion du trait à été créé. Ce dernier est implémenté dans la méthode « AddPoint » de « Pencil », à chaque point qu’on veux ajouter, la méthode se charge de vérifier qu’il n’y a pas d’espace plus grand qu’une certaine « densité » depuis le dernier point dessiné, si c’est le cas elle « remplira » d’une manière linéaire la zone entre les deux points.

Ainsi on ajoute une notion de « densité » du trait. Concrètement elle caractérise l’espacement entre deux « impression » de la brosse.

Cette première notion basique était fonctionnelle dans le cas d’une densité d’un pixel, mais lorsque l’on réglait une densité supérieure la vitesse de détection de la position de la souris impliquait que le trait n’avait pas une densité continue.

L’algorithme à été repris, actuellement la brosse n’est pas imprimée avant d’avoir atteint la distance de la densité. L’ajout de cette fonctionnalité nous permet d’utiliser des brosses composées d’image, comme des étoiles par exemple, et de les voires apparaitre avec une certaine densité au déplacement de la souris.

### Dessin

#### Problèmes rencontrés, analyse

##### CTRL-Z

Ce programme à pour but de dessiner en bitmap.

La question de quand procéder à la pixelisation des éléments du dessin s’est posée au début de l’implémentation pour anticiper le problème à venir du « ctrl-z ». La décision à été de pixéliser le dessin après avoir fini le trait actuel.

Le « memento » devra analyser les changements effectués sur le canvas après chaque pixellisation et les enregistrer pour pouvoir les annuler.

#### Solutions retenues, implémentation

##### Taille de la Zone de dessin

Une réflexion préalable sur la zone de dessin à été faite au début du projet. Le canvas devrait-il prendre la taille de l’image souhaitée par l’utilisateur ou devrait-il prendre toute la place à disposition ?

C’est la seconde solution qui à été retenue, avec la première l’utilisation d’un outil « zoom » aurait été frustrant, l’utilisateur en agrandissant une partie du dessin se limiterait à une vision tronquée par la taille initiale de son dessin car le zoom s’effectuera dans le canvas et non en agrandissant l’élément canvas de la page HTML.

###### Taille du canvas : ResizeUtils, GlobalViewCtrl

Le canvas prend alors toute la place à disposition, son redimensionnement est effectué par une « factory » AngularJS. La Factory à l’avantage de pouvoir être utilisée par tout les contrôleurs, elle représente une collection de méthodes que nos contrôleurs pourront utiliser.

La factory peut mettre à disposition certaine méthodes, variable et peu en conserver certaines en privé.

Cette factory se nomme "ResizeUtils" et est implémentée dans « resize\_module.js »

Le contrôleur "GlobalViewCtrl" gère le redimensionnement du canvas et utilise "ResizeUtils", il est assigné au control de l’ensemble du « body » du document HTML.

Concrètement il appel la méthode « resizeCanvas(w,h) » lorsque la taille de la fenètre change.

###### Zone de dessin

Comme le canvas prend la taille en fonction de la place à disposition le dessin à une taille différente de ce dernier.

La classe « draw » à été crée pour représenter le dessin affiché dans le canvas, son constructeur initialise un fond blanc de la taille spécifié afin de représenter la zone de dessin.

La méthode « save() » est appelée après chaque trait dessiné pour repixeliser la zone de dessin et ainsi « nettoyer » les pixels dessinés en dehors de la zone. Elle utilise la méthode "getImageData" du context 2D du canvas.

Enfin à chaque fois que l’on redessine le canvas (redimentionnement, etc..) on appel la méthode « DrawMe() » qui redessinera le dessin sur le canvas.

###### Controlleur du dessin

Le contrôleur du dessin gère les listeners sur la position de la souris sur le canvas, écoute si un bouton un pressé, etc…

Il contient les outils (tools) disponibles et les couleurs dessins.

Ainsi c’est lui qui dessine sur le canvas. Il doit être accessible par les autres contrôleurs pour changer d’outil et redessiner la zone de dessin

Ce contrôleur à donc été implémenté en temps que factory sous le nom de « drawUtils ». Il est dans « draw\_moule.js ».

###### Sélection des outils

Un controlleur à été créé pour gérer la selection des outils, il est implémenté dans « selectTool\_module.js » sous la forme d’une directive (drawtoolspannel) qui à la particularité de créer un élément HTML avec une vue HTML fournie. Le controlleur aura donc la main sur cette portion de code.

L’avantage à cela est d’avoir un code très épuré sur la page principale, la maintenance est facilitée car on sait exactement à quoi ce limite un controlleur.

Ce controlleur travail étroitement avec « drawUtils », il passe les outils à la vue pour le dataBinding (mise en page automatique des outils sur le panneau avec la directive ng-repeate).

La vue ayant accès uniquement aux fonction de ce controlleur, il sert alors souvent d’intermédiaire entre drawUtils et la vue.

###### Paramètre des outils

toppenpannel

# Rétrospective sur le planning

# Conclusion

# Bibliographie