Rapport de stage

Uni-Dufour

Microscope Virtuel

Service NTICE

Etudiant : Tomas Serpa

Maitre de stage : M. Pierre Lehmann (Ingénieur système Web, e-learning)

Responsable de stage : M. Pierre Yves Burgi (Responsable NTICE)

Conseiller pédagogique: M. Eric Batard (Enseignant ESIG)

Période de stage : 1 février 2011 au 30 juin 2011

Remerciements

Avant tout, je tiens à remercier mes maîtres de stage Monsieur Lehmann et Monsieur Burgi pour m'avoir accepté en tant que stagiaire au sein du service des NTICE.

Je remercie de même, mon responsable de stage Monsieur Batard pour son encadrement pendant celui-ci.

Je remercie également Madame Panchard, Monsieur Diallo et toute l'équipe du studio 360 pour leur accueil et leur sympathie.

Sommaire

Cadre de travail

Dans le cadre de ma deuxième année de formation à l'ESIG, j'ai effectué un stage d'une durée de cinq mois à mi-temps au service des Nouvelles Technologies de l'Information, de la Communication, et de l'Enseignement (NTICE).

Présentation du service NTICE

Le NTICE est un service de la division informatique de l'université de Genève. Il se trouve à l'uni Dufour et a pour mission de mettre en place les technologies nouvelles au service de la communauté universitaire.

Domaines principaux d'activités du NTICE

L'accès à l'information

- Data Mangement
    - Permettre l'accès aux données scientifiques, éducatives, et administratives.

- Portail
    - Dispositif global fournissant à un usager un point d'accès à travers les réseaux à l'ensemble des ressources et des services numériques en rapport avec son activité.

Les outils de communication permettant le travail collaboratif

- Environnement Web
    - Gestion du serveur institutionnel qui héberge le site web de l'université de Genève (www.unige.ch) ainsi que les applications attachées au Web.

L'enseignement et l'apprentissage

- e-learning
    - Les technologies du domaine de l'e-learning englobent les plates-formes d'enseignement, la numérisation des cours et leur diffusion sur Internet, et les outils de simulation à des fins didactiques.

Description du stage

Objectifs du stagiaire

Réaliser un microscope virtuel

Le but de ce microscope est de pouvoir zoomer de manière très approfondie sur une image afin d'analyser les contenus des différentes couches.

Pour faire une image il faut rassembler toutes les tuiles qui l'a compose, comme un puzzle avec ses pièces. Une image est donc composée de plusieurs tuiles et cela pour chaque niveau de zoom, plus le niveau de zoom est bas et moins l'image à de tuiles.

Une image sera initialement affichée, on pourra zoomer dessus jusqu'au moment ou la qualité de l'image devient mauvaise. Une fois cette limite atteinte, c'est une autre image qui est affichée.

Le microscope doit donc permettre de montrer les différentes couches du sujet traité en passant d'une image à une autre. Il doit aussi être capable de le faire en sens inverse lorsque l'on dé-zoom, en gardant les mêmes points d'ancrages que lors du zoom.

Logiciel utilisé : Adobe Flash Creative Suite 5

Langage de programmation utilisé : Action Script 3

Fonctions de l'application

- Charger des tuiles depuis un fichier XML
- Classer les tuiles
- Assembler les tuiles par niveau de zoom (visualisation de l'image)
- Permettre de zoomer et de se déplacer
- Afficher les tuiles du niveau suivant lorsque le nombre de pixels de celles-ci sont trop bas pour le niveau de zoom
- Lors du zoom, ne charger et afficher que les tuiles qui sont concernées par cet évènement à défaut de charger toutes les tuiles y compris celles qui ne seront pas sur la scène
- Retenir la position x et y avant de passer au niveau de zoom suivant pour pouvoir revenir de la où l'on est parti

Les étapes du stage :

Apprendre à connaitre l'Action Script 3

Découverte du langage à travers des tutoriels, forums et exercices

Apprendre à connaitre Flash

Réalisation de petits exercices pour apprendre à utiliser Flash CS5

Modéliser la future application

Schématiser le mieux possible les futures procédures qui vont faire tourner notre application afin de créer une sorte de marche à suivre pendant le développement

Cette partie a aussi été faite en étudiant le fonctionnement du programme Zoomify

Développer l'application

A partir de ce qu'on a appris en s'exerçant et de ce qu'on a modélisé

Répartition dans le temps :

Février	Apprentissage de Flash et de l'Action Script 3
Mars	Modélisation de la future application
Avril, Mai, Juin	Développement

Modélisation

Pour comprendre comment les images sont découpées puis reformées grâce aux tuiles, j'ai pris l'exemple d'une photo qui après avoir été découpée a 4 niveaux de zoom.

Chaque case représente une tuile avec son nom, ses dimensions et sa taille

Le nom d'une tuile est composé de trois nombres, le premier indique le niveau de zoom, le second la position x et le troisième la position y.

New York City.JPG				
2560*1600				
1.82 Mo				
Niveau	0	0.0.0	20.9 Ko	
		160*100		
Niveau	1	1.0.0		"1.1.0
		256*200		64*200
		61.7 Ko		14.2 Ko

Niveau	2	2.0.0	"2.1.0	"2.2.0
		256*256	256*256	128*256
		75.2 Ko	75.4 Ko	35 K0
		2.0.1	"2.1.1	"2.2.1
		256*144	256*144	128*144
		40.9 Ko	40.7 Ko	17.9 Ko

Niveau	3					
		3.0.0	"3.1.0	"3.2.0	"3.3.0	"3.4.0
		256*256	256*256	256*256	256*256	256*256
		3.0.1	"3.1.1	"3.2.1	"3.3.1	"3.4.1
		256*256	256*256	256*256	256*256	256*256
		3.0.2	"3.1.2	"3.2.2	"3.3.2	"3.4.2
		256*256	256*256	256*256	256*256	256*256
		3.0.3	"3.1.3	"3.2.3	"3.3.3	"3.4.3
		256*32	256*32	256*32	256*32	256*32

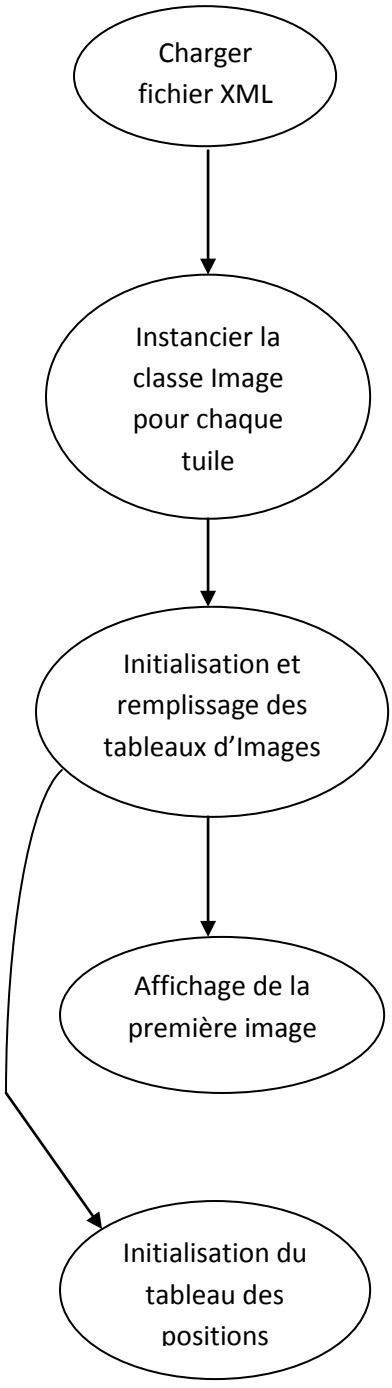
Niveau	4	"4.0.0	"4.1.0	"4.2.0	"4.3.0	"4.4.0	"4.5.0	"4.6.0	"4.7.0	"4.8.0	"4.9.0
		"4.0.1	411	421	431	441	451	461	471	481	491
		"4.0.2	412	422	432	442	452	462	472	482	492
256*256		"4.0.3	413	423	433	443	453	463	473	483	493
		"4.0.4	414	424	434	444	454	464	474	484	494
		"4.0.5	415	425	435	445	455	465	475	485	495
256*64		"4.0.6	416	426	436	446	456	466	476	486	496

Pour chacune des tuiles chargées, appeler le constructeur Image. On aurait alors pour chaque tuile un conteneur du type Sprite qui contient l'image(tuile), son niveau de zoom pour pouvoir les classer correctement, la largeur et la hauteur.

Ensuite on range ces conteneur dans leur tableau, les tableaux étant construits en fonction du nombre de niveaux (pour savoir combien il y aura de tableaux) et leur taille en fonction du nombre de tuiles correspondant à chaque niveau.

```
public class Image extends Sprite
{
    var imNum:Number;
    var imWidht : Number;
    var imHeight : Number;
    var imTuile:Bitmap;

    public function Image(p_num : Number, p_width : Number, p_height : Number, p_tuile : Bitmap)
    {
        imNum = p_num;
        imWidht = p_width;
        imHeight =  p_height;
        imTuile = p_im;
    }
```




```
var chargeur : URLLoader = new URLLoader ();
var adresse : URLRequest = new URLRequest ("source.xml");
chargeur.load(adresse);
chargeur.addEventListener(Event.COMPLETE, Image);
```

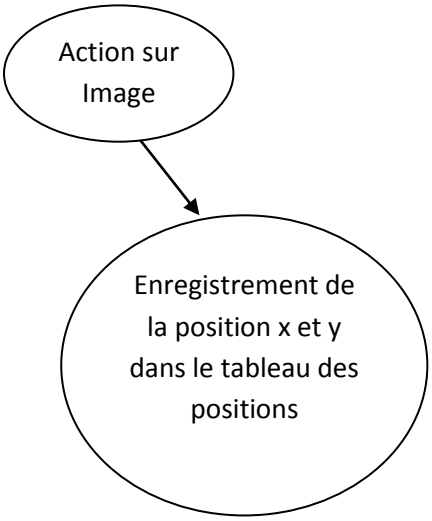
Pour chaque tour de boucle, faire une nouvelle instance de la classe Image qui prend en paramètres le numéro du niveau de zoom récupérer sur le premier caractère du nom de la tuile, la largeur, la hauteur ainsi que l'image elle-même.

Faire un tableau d'Images par niveau de zoom

Afficher le contenu du premier tableau

Dernière position retenue sur chaque niveau

	Niveau 0	Niveau 1	Niveau 2	Niveau 3
x				
y				



Lorsqu' on se déplace sur une image, les tuiles visibles doivent être affichées, les anciennes positions x et y sont effacées et remplacées lorsque l'on s'arrête sur un autre point de l'image.

Adobe Photoshop et Zoomify

Ce microscope virtuel ne s'occupera pas du découpage des images par niveau de zoom. Pour obtenir les tuiles, il faudra utiliser le plug-in Zoomify qui se trouve dans Photoshop

Description de Zoomify

Adobe Photoshop permet aux utilisateurs d'ouvrir une image et de l'exporter sur Zoomify, ceci pour pouvoir zoomer et se déplacer sur une image. Mais la différence qu'il y a entre Zoomify et la majorité des programmes de visualisation d'images (Paint, Photoshop, visionneuse de photos Windows, etc.) c'est que Zoomify instaure lui-même la limite jusqu'à laquelle on peut zoomer sur une image. C'est pour que l'on puisse zoomer tant que la qualité de l'image est bonne (supérieur ou égale à 72 dpi). Il n'est pas possible de zoomer jusqu'à ce que les pixels deviennent gros et que l'image ne soit plus nette (inférieur à 72 dpi).

Obtention des tuiles

Lorsqu'on ouvre une image dans Photoshop et qu'on l'exporte avec le Plug-in Zoomify, Zoomify va concevoir les différents niveaux de zoom possible et chacun de ces niveaux est constitué des tuiles qui composent l'image. Il va donc découper l'image en tuiles pour chaque niveau de zoom. Par exemple le premier niveau de zoom aura moins de tuiles que les niveaux suivants car plus le niveau est bas plus cela signifie que les tuiles sont grosses et plus le niveau de zoom est élevé plus il y aura de tuiles car celles-ci montreront de minuscule partie de l'image.

Après avoir exporté une image depuis Photoshop (avec le plug-in Zoomify) dans le répertoire choisi, on récupère :

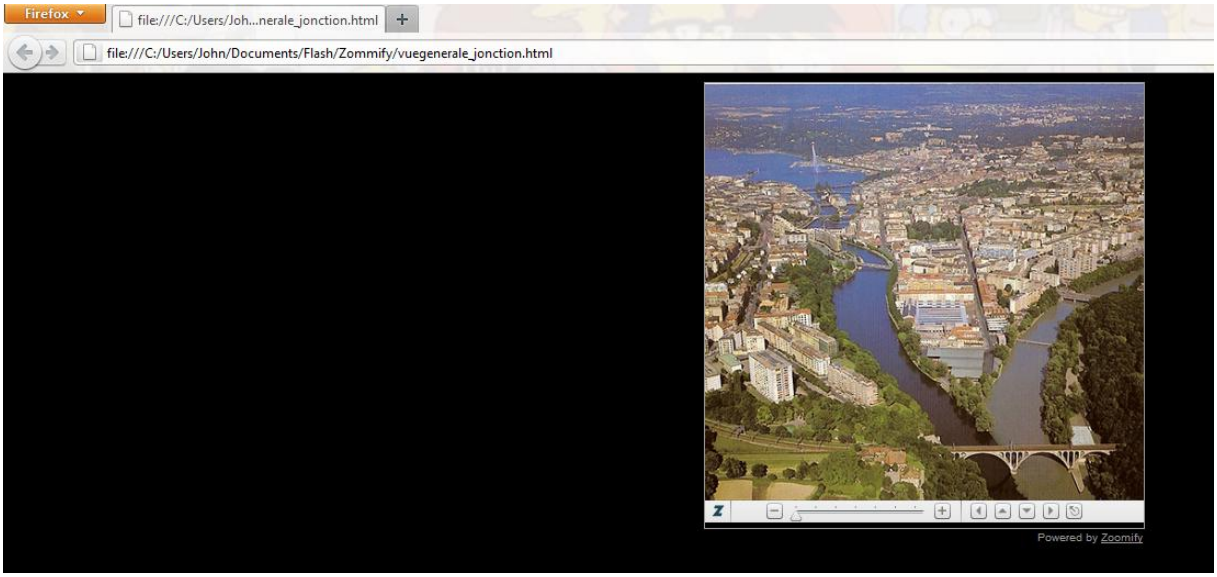
1. **L'image**. Le nom de l'image reste le même mais avec une extension html, ce qui permet de visualiser l'image directement depuis l'interface de Zoomify.

2. **Un répertoire nommé « TileGroup0 »**, c'est là que se trouvent toutes les tuiles. Les tuiles sont classées de manière très précise. Le nom de chaque tuile est une série de trois nombres. Le premier nombre indique le niveau du zoom, le second indique la position de la tuile sur l'image sur l'axe des x (horizontale) et le troisième indique la position sur l'axe des y.

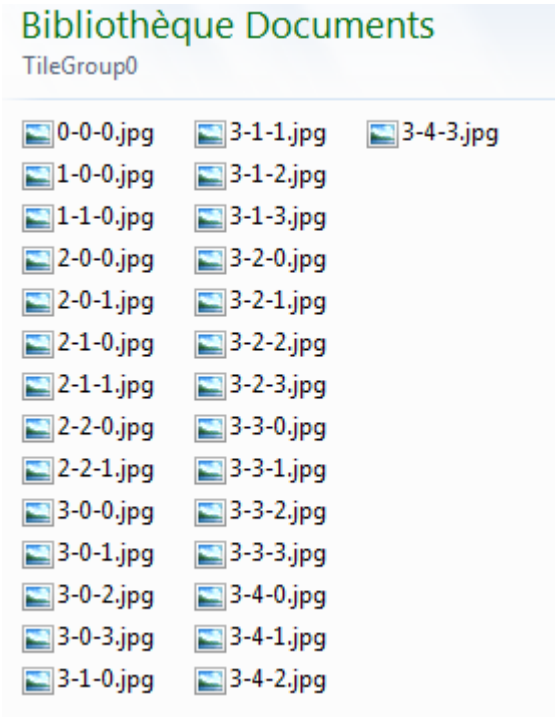
3. **Un fichier XML nommé « ImageProperties »,** qui indique les propriétés de l'image comme ses dimensions, le nombre de tuiles total ou la taille des tuiles.

En voici des illustrations :

1. L'image depuis l'interface de Zoomify

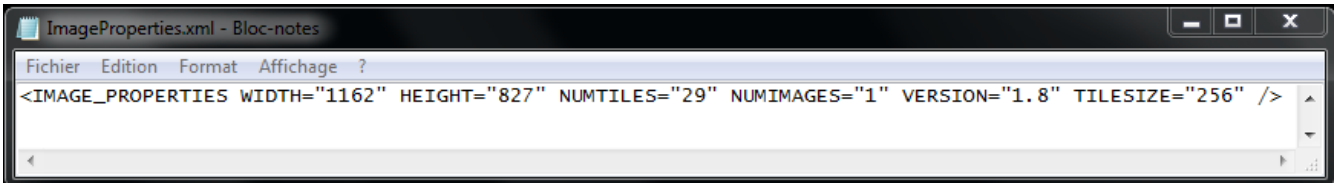


2. Le répertoire TileGroup0 contenant toutes les tuiles



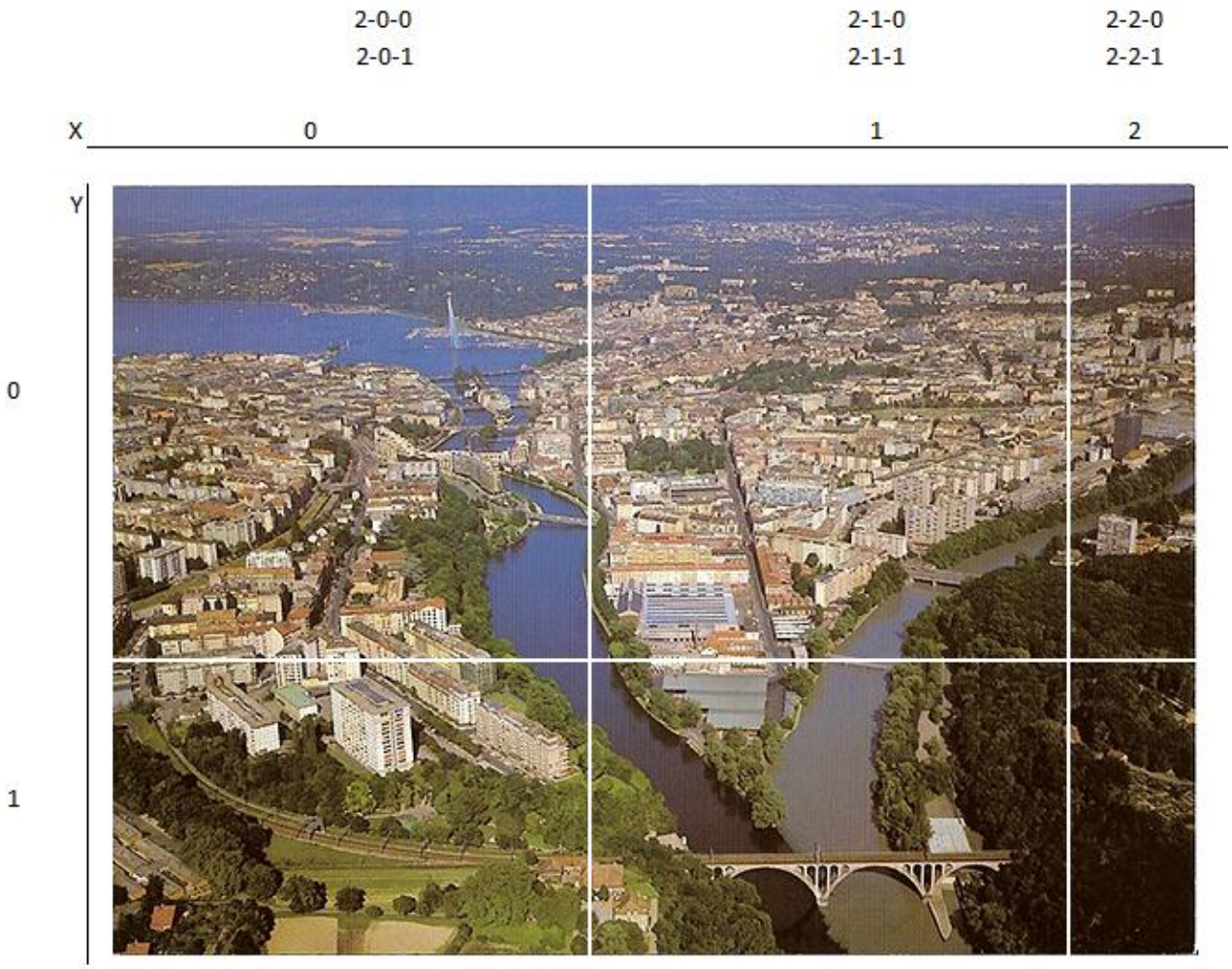
Ici, on voit bien que l'image à 3 niveau de zoom et que le 3ème niveau est celui qui contient le plus de tuiles.

3. Le fichier XML « ImageProperties »



Exemple

J'ai pris les 6 tuiles qui composent le second niveau de zoom de l'image présentée dans la page précédente.



Parties de l'application

J'ai pu constater qu'en action script il y avait beaucoup de classes avec pour toutes une liste de propriétés et de méthodes. Il y a donc plusieurs chemins possibles pour arriver à un seul but.

Le problème essentiel que j'ai eu lors de ce stage est d'avoir perdu trop de temps sur des chemins qui ne menaient nulle part.

Vu que je ne connaissais pas au préalable une ligne à suivre, j'ai essayé différentes manières.

Donc les scripts sont différents pour chaque manière. Certaines choses sont réutilisé comme l'interface les Sprites et la les fonctions de zoom.

Création de l'interface :

Calque 1 :

Sur ce calque se trouve l'objet Conteneur qui est de type Sprite et qui va contenir les tuiles. Il peut aussi y avoir un objet UILoader selon la manière adoptée, cet objet sert à afficher l'image.

Calque 2 :

Sur ce calque se trouve le cadre qui va servir de contour à la zone d'affichage.

Calque 3 :

Sur ce calque se trouve le masque qui a pour but de limiter la zone d'affichage.

Calque 4 :

Sur ce calque se trouve un autre objet Sprite nommé Panel sur lequel j'ai ajouté les boutons.

Affichage des images sur la scène

Pour charger et afficher les images, j'ai pu observer différentes méthodes :

1. **Sans chargement d'un fichier XML**

    1.1) Chargement de tuiles

2. **Avec chargement d'un fichier XML**

    2.1) Chargement d'images

    2.2) Chargement de tuiles

1.1) Chargement de tuiles sans fichier XML

Chargement des tuiles de manière statique

```
var adresse1:URLRequest = new URLRequest("TileGroup0/1-0-0.jpg");
var ph1:Loader = new Loader();
ph1.load(adresse1);
```

Ajout des tuiles (Loader) dans le conteneur et positionnement des tuiles

```
conteneur.addChild(cadre);
conteneur.addChild(ph1);
conteneur.addChild(ph2);
conteneur.addChild(ph3);
conteneur.addChild(ph4);
conteneur.mask = jim;


ph3.x = 256;
ph3.y = 0;
ph2.x = 0;
ph2.y = 256;
ph4.x = 256;
ph4.y = 256;
```

Ces 4 tuiles composent le premier niveau de visualisation de l'image. Le chargement de toutes les autres ce fait de la même manière. Sauf que ces 4 tuiles sont de toute façon affichées au démarrage alors que les tuiles des autres niveaux sont appelées par une fonction qui compte le nombre de zoom effectués et qui au bout d'un certain nombre enlève les tuiles affichées pour pouvoir afficher celles du niveau suivant.

```
var nMouse:Number = 0;
function zoomCurseur(e:MouseEvent):void
{

    if (e.delta < 0)
    {
        nMouse--;
        conteneur.width /=  ratio;
        conteneur.height /=  ratio;
    }
    else
    {
        nMouse++;
        conteneur.width *=  ratio;
        conteneur.height *=  ratio;
    }



   if (nMouse == 5)
   {
       FirstNext();
   }
   if (nMouse == 10)
   {
       SecondNext();
   }
   if (nMouse == 15)
   {
       ThirdNext();
   }
```

```
function FirstNext()
{
    conteneur.removeChild(ph1);
    conteneur.removeChild(ph2);
    conteneur.removeChild(ph3);
    conteneur.removeChild(ph4);


    conteneur.addChild(ph21);
    conteneur.addChild(ph22);
    conteneur.addChild(ph23);
    conteneur.addChild(ph24);
    conteneur.addChild(ph25);
    conteneur.addChild(ph26);
    conteneur.addChild(ph27);
    conteneur.addChild(ph28);
    conteneur.addChild(ph29);

    ph24.x = 256;
    ph24.y = 0;
    ph27.x = 512;
    ph27.y = 0;

    ph22.x = 0;
    ph22.y = 256;
    ph25.x = 256;
    ph25.y = 256;
    ph28.x = 512;
    ph28.y = 256;

    ph23.x = 0;
    ph23.y = 512;
    ph26.x = 256;
    ph26.y = 512;
    ph29.x = 512;
    ph29.y = 512;

}
```

Inconvénients :

- Tout est fait de manière statique
- Il ne serait pas possible de lui faire prendre en compte une image qui n'a pas les mêmes dimensions car le nombre de tuiles pour chaque niveau de zoom serait différent.
- Le zoom effectué ne prend pas en compte la qualité de l'image.
- La totalité des tuiles sont chargées et affichées au même moment

2.1) Chargement d'images via un fichier XML

Création du fichier XML « source » qui va contenir l'adresse de chaque image

Dans Flash

Chargement du fichier XML

Chargement des images dans un UILoader posé sur la scène

Affichage des images depuis l'UILoader

Affichage de l'image suivante sur évènements de la souris

```
chargeur.load(adresse);
chargeur.addEventListener(Event.COMPLETE, ChargeXML);

var tab:Array = new Array();
var nbImm:Number = 0;
var nbClics:Number = -1;

function ChargeXML(evt:Event):void
{
    var monXML:XML = new XML(evt.target.data);

    for (var i:String in monXML.image)
    {
        var nextfoto : UILoader = new UILoader();
        nextfoto.source = monDoss + monXML.image[i].attribute("src");
        nextfoto.width = stage.stageWidth;
        nextfoto.height = stage.stageHeight;
        trace(nextfoto.source);

        tab.push(nextfoto);
        nbImm++;
    }
    trace(nbImm);

}
```

```
function clicImage(e:Event):void
{
    nbClics++;
    btn.label = 'image n° ' + nbClics;

    if (nbClics <= nbImm + 1)
    {
        nextfoto.source = tab[nbClics];
    }
    else
    {
        trace('il n y  a plus d images ');
    }
}



conteneur.addChild(rect);
conteneur.addChild(nextfoto);

var tempX = conteneur.x;
var tempY = conteneur.y;
var tempW = conteneur.width;
var tempH = conteneur.height;
var resultW = conteneur.width;
var resultH = conteneur.height;
var ratio = 1.1;



conteneur.addEventListener(MouseEvent.MOUSE_DOWN, mouseDown);
function mouseDown(event:MouseEvent):void
{
    conteneur.startDrag();
}

conteneur.addEventListener(MouseEvent.MOUSE_UP, mouseUp);
function mouseUp(event:MouseEvent):void
{
    conteneur.stopDrag();
    tempX = conteneur.x;
    tempY = conteneur.y;
}
```

```
conteneur.addEventListener(MouseEvent.MOUSE_WHEEL, zoomCurseur);
var nbMouseEvt:Number = 0;
function zoomCurseur(e:MouseEvent):void
{
    if (e.delta < 0)
    {
        nbMouseEvt = nbMouseEvt - 1;
        conteneur.width /=  ratio;
        conteneur.height /=  ratio;
    }
    else
    {
        nbMouseEvt++;
        conteneur.width *=  ratio;
        conteneur.height *=  ratio;
    }
    if (nbMouseEvt == 5)
    {
        nextfoto.source = tab[1];
    }
    if (nbMouseEvt == 10)
    {
        nextfoto.source = tab[2];
    }
    if (nbMouseEvt == 15)
    {
        nextfoto.source = tab[3];
    }
    conteneur.x += (tempW-conteneur.width)/(tempW/(conteneur.parent.mouseX-tempX));
    conteneur.y += (tempH-conteneur.height)/(tempH/(conteneur.parent.mouseY-tempY));
    tempX = conteneur.x;
    tempY = conteneur.y;
    tempW = conteneur.width;
    tempH = conteneur.height;
}


function zoomBTN(e:MouseEvent):void
{
    conteneur.width *=  ratio;
    conteneur.height *=  ratio;
}

function dézoomBTN(e:MouseEvent):void
{
    conteneur.width /=  ratio;
    conteneur.height /=  ratio;
}
```

2.2) Chargement de tuiles via un fichier XML

Création des tuiles grâce au Plug in Zoomify de Photoshop

Création du fichier XML « source » qui va contenir l'adresse de chaque tuile

Dans Flash

Chargement du fichier XML

Récupération des données XML

Récupération du nom et du niveau de zoom de chaque tuile

Chargement de toutes les tuiles avec un Loader

Création de Bitmap prenant en paramètre le contenu du Loader

Affichage des tuiles assemblées

```
var adresse:URLRequest = new URLRequest ("donnees.xml")  ;
var chargeur:URLLoader = new URLLoader();

var donXML:XML;
var nomTuile : String;
var numImageTuile : String;
var tuile : Loader;
var tailleXML:Number = 0;


var TabNomTuiles : Array;
var TabNumImagesTuiles : Array;
var TabTuiles : Array;

chargeur.load(adresse);
chargeur.addEventListener(Event.COMPLETE, chargeXML);
```

```
function chargeXML(e:Event)
{
    donXML = new XML(e.target.data);
    tailleXML = donXML.elements("image").length();
    chargeNumImage();
    trace(tailleXML);
    chargeur.removeEventListener(Event.COMPLETE, chargeXML);
}

function chargeNumImage()
{

    TabNomTuiles = new Array();
    TabNumImagesTuiles = new Array();
    TabTuiles = new Array();


    for (var u:Number=0; u<tailleXML; u++)
    {
        //Adresse
        nomTuile = donXML.elements("image")[u].attribute("src");
        var my_str:String = new String(nomTuile);
        //trace(my_str);

        //Nom
        var mySubstrNom:String = new String();
        mySubstrNom = my_str.substr(7,5);
        nomTuile = mySubstrNom;
        TabNomTuiles.push(nomTuile);
        trace(nomTuile);

        //Num d'image
        var mySubstrNum:String = new String();
        mySubstrNum = my_str.substr(7,1);
        numImageTuile = mySubstrNum;
        TabNumImagesTuiles.push(numImageTuile);
        trace(numImageTuile);

        //Tuile
        var chargeur2:Loader = new Loader();
        var adr:URLRequest = new URLRequest(donXML.elements("image")[u].attribute("src"));
        chargeur2.contentLoaderInfo.addEventListener(Event.COMPLETE, getContent);
        chargeur2.load(adr);
    }

}


function getContent(evt:Event):void
{
    //Image chargée
    var tuileU:Bitmap = Bitmap(evt.target.content);
    addChild(tuileU);
    //trace(TabNomTuiles.length);
    //trace(TabNumImagesTuiles.length);

}
```

Problèmes rencontrés

Choix des classes à utiliser

Traitement des objets à travers un fichier XML

Je ne sais pas quelle est dans moi cas la meilleure manière de traiter les images (tuiles) une fois que le fichier XML a été chargé, sachant que l'objet qui va stocker ces images devra avoir les propriétés requises pour répondre aux attentes du microscope. Par exemple, si je veux avoir accès aux pixels d'une image, il faut que l'objet (Loader, UILoader, Bitmap, etc.) conteneur ait une propriété qui permette d'y accéder.

J'aurais voulu, une fois que le fichier XML est chargé, créer une boucle FOR qui va pour chaque tuile faire un nouvelle instance de la classe Image, puis rangé ces images dans le tableau qui leur correspond, c'est-à-dire un tableau par niveau de zoom

Problèmes de regroupements des tuiles : Elles sont toutes mises dans le même tableau.

J'ai tenté de charger les tuiles dans des UILoader puis de les rassembler mais il doit y avoir un problème lors du chargement des tuiles car il semble que les dimensions ne soient pas prises en compte. Lorsque les tuiles sont affichées sur la scène, elles ne sont pas correctement assemblées, il y a de grands espaces entre elles.

Structure de données

N'ayant pas d'exemple concret de code qui correspond à ce que je voulais construire, j'ai eu beaucoup de peine à visualiser la structure de mon code.

J'ai donc pris tous les exemples que je trouvais, étudié leur structure puis tenté de reproduire les parties dont j'avais besoin à un moment dans mon code.

Le temps

J'ai perdu pas mal de temps à essayer des techniques qui ne jouaient pas pour l'application en question. Sauf qu'au moment où j'étais dessus je ne savais pas encore que ça ne me serait d'aucune utilité et que je devrais ensuite chercher et étudier d'autres voies.

Résumé

Réaliser un microscope virtuel au sein du NTICE, service de la division informatique de l'université de Genève. Situé à l'Uni Dufour, le NTICE a pour mission de mettre en place les technologies nouvelles au service de la communauté universitaire.

Ce microscope virtuel doit permettre de zoomer et de se déplacer sur une image qui est faite de tuiles. Lorsque l'on zoom sur une partie de l'image, seules les tuiles qui se rattachent à cette partie doivent être chargées. Lorsque l'on passe sur un autre endroit de la scène on peut voir que les tuiles sont en train d'être chargée car il y a un petit moment ou les tuiles sont floues puis elles deviennent nettes.

Il faut donc charger pour une seule image tout un dossier de tuiles. Ces tuiles s'assemblent entre elles par niveau de zoom puis grâce aux coordonnées (x et y) qu'elles indiquent. L'application doit être en mesure de charger et afficher instantanément les tuiles qui correspondent à l'action entreprise par l'utilisateur (déplacement, zoom avant et zoom arrière). Le nombre de pixels détermine s'il faut charger les tuiles suivantes ou non. C'est-à-dire qu'une fois un premier zoom effectué, les pixels de l'image diminuent, ils peuvent atteindre les 72 dpi avant de devoir charger les tuiles suivantes.

Le but serait finalement de pouvoir mettre cet outil à disposition, des enseignants pour qu'ils puissent s'en servir pour donner leurs cours et des élèves pour qu'ils puissent facilement accéder au support de cours depuis chez eux ou même ailleurs.

Sources

Sites Internet

Action Script 3

www.bases-as3.fr

www.siteduzero.com

www.flashkod.com

www.snoupix.com

http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3

http://fr.wikipedia.org/wiki/ActionScript

www.flash.mediabox.fr

XML

http://fr.wikipedia.org/wiki/xml

Zoomify

www.zoomify.com

Université de Genève

www.unige.ch