Projet de Web Avancé : SpriteSheetEngine

Rapport de développement

I.Début du développement

Pour nous familliariser avec l'utilisation des canvas et des images dans Javascript, nous avons d'abord codé un spritesheet fixe avec une image prédéfinie, avec toutes les données en dur dans une fonction toolBox de création d'élément (du type de la méthode newElement vu en cours). Nous avons utilisé cette image.

Etant donné que nous n'avions pas de serveur a gérer dans notre projet, nous utiliserons un fichier view.js et controller.js. Les spritesheet à manipuler étant situé dans un dossier image à la racine du projet.

Les fonctions utililisées dans cette première étape étaient :
La méthode newElement, qui fabriquait le canvas et l'image.

```
/**
 * @method newElement : create new element to put in the window page
 * @param element : HTML tag
 * @param parent : parent which receives the element
 * @param width : element's width
 * @param height : element's height
 * @param x : coordinate x
 * @param y : coordinate y
 * @param identifier :·
 * @param style : CSS property, write in this form {"cssProperty1":"value1", "cssProperty2":"value2", "etc..."}
 *
 * Write "" in the parameters you don't need
 */
```

La fonction drawImage permettant d'afficher l'image dans le canvas, nous avons utilisé la troisième variante de celle-ci qui permet de redimentionner l'image dans le canvas.
Les différents paramètres de nos fonctions et leurs utilités peuvent être connu dans les commentaires du code source.

II.Milieu du développement

Après nous être entendu avec l'utilisation du canvas, nous avions comme objectif de généraliser nos fonctions pour permettre une utilisation plus libre des images. Nous avons donc créés la classe SpriteSheet permettant de prendre n'importe quel image (sous forme de spriteSheet), de la traiter et de l'afficher dans le canvas.

```
/**
 * @construtor SpriteSheet : the object SpriteSheet which contains all the necessary informations of a spritesheet
 * @param source : the link of the image source
 * @param numberRow : the number of row of sprite in the spritesheet
 * @param numberColumn : the number of column of sprite in the spritesheet
 * @param sourceWidth : the width of 1 sprite of the spritesheet (Watch the commentary of the function getSizeAfterLoad() for more information)
 * @param sourceHeight : the height of 1 sprite of the spritesheet (Watch the commentary of the function getSizeAfterLoad() for more information
)
 * @param destinationX : the coordinate X of the origin of the spritesheet in the canvas
 * @param destinationY : the coordinate Y of the origin of the spritesheet in the canvas
 * @param destinationWidth : the width of the spritesheet in the canvas
 * @param destinationHeight : the height of the spritesheet in the canvas
 */
```

Dans le soucis de généralisation, nous avons codés deux méthodes permettant de récupérer la taille de l'image d'origine et celle de chaque case de celle-ci.

```
/**
 * @method getSizeAfterLoad :
 * @param imageObject : the object SpriteSheet
 *
 * The sourceWidth and sourceHeight property can only be obtained after the image have been loaded,
 * otherwise, the width and height will be always 0.
 * So, firstly, sourceWidth and sourceHeight will temporarily take the size of the image source then,
 * those properties will take the size of 1 sprite of the image after the function spriteSize() have been called.
 */
```

```
/**
 * @method spriteSize : calculate the size of 1 sprite from a spritesheet
 * @param image : the object SpriteSheet
 * @param numberRow : the number of row of sprite in the spritesheet
 * @param numberColumn : the number of column of sprite in the spritesheet
 *
 * When the function spriteSize() is called by the function getSizeAfterLoad(),
 * the variables image.sourceWidth and image.sourceHeight have the size of the image source.
 */
```

Le moteur a proprement parler est la méthode update qui parcourt l'image 'case par case'. Pareillement modifier pour plus de généralité.

```
/**
 * @method update : function that will update the animation
 * @param imageObject : the object SpriteSheet
 *
 * Change the value of the variable moveSpeed to change the speed of the movement perform by the keyboard event
 */
```

Nous avons rencontrés plusieurs problèmes lors de l'écriture de cette classe. Le premier étant l'attente du chargement de l'image avant de l'afficher dans le canvas. Problème résolu par la récupération des dimensions de l'image après le chargement de celle-ci. Les paramètres utilisés et leurs utilités peuvent être connu dans le code source. Un autre problème rencontré dans cette phase de développement est le changement de type de l'objet image qui lors du frameLoop après un passage perd sont type et devient un number quand on l'utilise en tant que paramètre de frameloop. Pour régler le problème, qui n'est pas vraiment une solution, nous avons dû enregister l'image en dur à quatre endroits différents et y faire appel avec la fonction update dans le frameloop.

III.Fin du developpement

L’objectif de fin de développment étais de rajouter des boutons pour permettre le chargement d’image utilisateur, de début de script, de pause du script et d’effacement d’image.
Les trois derniers sont fonctionnel mais nous n’avons pas réussi à permettre à l’utilisateur de charger sa propre image. Une autre tentative est l’utilisation d’un tableau d’image mais le problème du chargement des images a resurgi lors de celles-ci. Suite a cela nous avons trouvés trop confus le view.js et le controller.js, nous avons donc créés un nouveau dossier view pour pouvoir laisser la place aux calculs des images.

Le fichier buttonBar.js contenant les méthodes pour les bouttons et le fichier canvas.js pour le canvas et les images. A ce stade du développement il reste encore un problème étant la téléportation de l’image au sein du canvas lors d’un évènement clavier.

IV.Future du développement

Ils restent, malgré le fin du développement, des points du projet à ajouter ou à améliorer. Le premier consisterais à régler le problème de chargement des images utilisateur. En effet le problème de changement de type du tableau d’images lors du frameloop bloque cette fonctionalité. La seconde grosse amélioration serais la capacité à animer plusieurs SpriteSheets sur le même canvas.