Leçon 6.2 – HTML5 : animations

s1 ----------

s2 ----------

Dans cette partie, nous allons voir comment réaliser une animation simple avec HTML5 et la balise canvas, qui constitue de ce point de vue une vraie alternative au logiciel Flash. Ce procédé peut s'appuyer notamment sur la notion de sauvegarde et de restauration du contexte.

s3 ---------

Comment animer ? Une première façon élémentaire est la suivante.

En ce qui concerne HTML, on s'appuie sur la balise Canvas.

Grâce à JavaScript, on associe au canvas un contexte (au sein duquel on crée des objets).

On se rappelle que le contexte par défaut considère que l'on dessine avec pour origine le coin supérieur gauche du canvas. On se rappelle que le contexte peut subir des rotations, des translations et des changements d'échelle et que les objets qui vont être dessinés dans ce contexte modifié seront donc affectés par ces modifications.

Toujours grâce à JavaScript et à la méthode setInterval(), on lance à intervalles réguliers l'exécution d'une fonction qui va faire deux choses :

1. cette fonction va d'abord modifier ce contexte, par exemple le faire tourner, translater ou changer d'échelle, avec à chaque fois à une valeur de rotation de translation ou d'échelle différente ; on pourra aussi effacer tout ou partie de ce qui avait été dessiné au préalable ;
2. la seconde chose que fait cette fonction c'est de redessiner le ou les objets ; leur affichage va donc être affecté par ces modifications de contexte.

Voilà pour le principe.

s4 ---------

Voyons ce que ça donne sur un exemple simple à l'occasion duquel nous allons mettre un rectangle en rotation.

La première étape

- définit le contexte
- le style de remplissage
- translate le contexte au centre du canvas (c'est-à-dire que l'on commencera à dessiner à partir de ce point)

- dessine un premier rectangle

Si vous refermez la balise script à ce niveau et si vous testez, vous obtiendrez un simple rectangle bleu.

Seconde étape : l'introduction du timer et de la fonction qu'il va lancer à intervalles réguliers.

Que fait cette fonction :

- une rotation du contexte
- un nouveau dessin
- un test sur le nombre de fois que l'on est passé dans cette fonction afin le cas échéant d'arrêter l'animation

On notera qu'il est préférable de déclarer la fonction avant le timer qui l'utilise.

On peut à ce stade faire deux remarques :

- dessiner des formes complexes peut prendre beaucoup de temps
- comme le principe est que l'on redessine à chaque instant une forme, les performances d'une animation en terme de rendu et de fluidité dépendront fortement de la vitesse de l'ordinateur sur lequel on se trouve.

s5 ---------

Comment faire à présent si l'idée est de donner l'impression que le rectangle bouge, par exemple en diagonale, de haut en bas et vers la droite de l'écran ?

C'est simple, on reprend les étapes précédentes :

- définition du contexte
- introduction du timer qui modifie le contexte et redessine

Sauf que la zone de dessin doit être effacée à chaque étape. Pour cela, nous allons utiliser la méthode clearRect() qui permet d'effacer une partie rectangulaire du contexte

Voyons ce que ça donne sur un exemple simple à l'occasion duquel nous allons mettre un rectangle en translation.

La première étape

- définit le contexte
- le style de remplissage
- translate le contexte au centre du canvas (c'est-à-dire que l'on commencera à dessiner à partir de ce point)
- dessine un premier rectangle

Si vous refermez la balise script, vous obtiendrez un simple rectangle bleu.

Seconde étape : l'introduction du timer et de la fonction qu'il va lancer à intervalles réguliers.

Que fait cette fonction :

- l'effacement de la totalité du contenu du contexte (c'est-à-dire depuis le point 0,0 et sur une largeur et une hauteur qui correspondent aux dimensions du canvas
- une translation du contexte
- un nouveau dessin
- un test sur le nombre de fois que l'on est passé dans cette fonction afin le cas échéant d'arrêter l'animation

s6 ---------

On se pose à présent la question de savoir comment, dans une animation un peu plus complexe, faire en sorte d'animer séparément plusieurs objets.

Cherchons par exemple à animer un petit carré de bas en haut avec un cercle qui tourne autour de ce carré, comme sur cette figure.

On reprend les étapes précédentes

- : définition du contexte
- introduction d'une variable i sur laquelle nous allons itérer
- introduction du timer ; ici, nous introduisons une petite modification par rapport à l'exercice précédent : pour définir le moment auquel l'animation va s'arrêter, nous n'allons pas utiliser la variable i comme précédemment, mais un timeout qui appellera une fonction d'arrêt après une période de temps définie, ici 20 secondes.

s7 ---------

Détaillons à présent la fonction que le timer va lancer à intervalles réguliers. Que fait cette fonction ?

Première chose, elle va devoir effacer ce qui a été dessiné à l'itération précédente.

Ensuite, on translate le contexte de la valeur i suivant x et suivant y et dessin du rectangle, comme indiqué sur la figure.

Ensuite, on fait une rotation (en radian, par exemple i/50) du contexte (le rectangle n'est pas modifié, car la modification du contexte est effectuée postérieurement au dessin du rectangle).

Enfin, on fait une translation du contexte selon le nouvel axe x pour dessiner le cercle.

Si l'on teste, ce programme, on n'obtient pas tout à fait ce qui est recherché comme vous pouvez le voir en lançant la vidéo.

La raison est simple, lorsque l'on entre à nouveau dans la fonction Animer(), on part du contexte courant, c'est-à-dire celui que l'on a laissé à la fin de l'itération précédente. Pour s'en convaincre, il suffit d'imaginer la suite des imagettes de gauche lorsque l'on entre à nouveau dans la fonction Animer(). Il faudrait donc commencer à chaque fois par revenir au contexte initial (mettre une flèche) avant d'appliquer de nouvelles translations et rotations, en d'autres termes, à chaque itération, après avoir dessiné le cercle, il faudrait appliquer au contexte dans l'ordre inverse toutes les translations et rotations qu'on lui a appliquées. Ce qui semble bien compliqué pour une simple animation !

s8 ---------

Une façon simple de réaliser une telle animation cela est de sauver le contexte au tout début de la fonction de dessin grâce à la fonction save() et de le recharger avant de la quitter (fonction restore()).

Cependant, si l'on revient en quelque sorte à l'état initial du contexte, il faut que les valeurs de translation et de rotation qui lui sont appliquées soient différentes d'une itération à l'autre pour garantir l'illusion du mouvement. C'est pourquoi on introduit une variable i qui va être incrémentée à chaque itération et servir comme valeur de translation et de rotation.

Une remarque : dans l'exemple proposé, nous aurions pu restaurer non pas le contexte initial, mais à un contexte intermédiaire : le contexte après la première translation. Je vous propose d'essayer cette option : dans ce cas, la valeur de la première translation peut rester à la valeur deux d'une itération à l'autre (puisqu'on part d'un contexte modifié) ; la valeur de la rotation en revanche doit rester une fonction de la variable i.

Seconde remarque : nous aurions pu également nous passer du save() et du restore() en dessinant le cercle à chaque fois avec un centre de coordonnées x et y tels que ce centre soit sur un cercle. Cependant, il était important de présenter ces deux fonctions sur un exemple simple.

Je vous conseille de bien comprendre ce mécanisme. Plusieurs états du contexte peuvent être ainsi sauvés par une succession de save(), comme si on rangeait ces différents états dans une pile (figure).

Une succession de restore() permet de se déplacer dans la pile pour restaurer un de ces états.

s9 ---------

La suite de cet exercice va nous permettre de découvrir comment deux formes peuvent interagir.

On reprend l'exercice précédent en agrandissant le carré de telle façon que le cercle intersecte avec ses coins.

Dans les exemples précédents, nous avons dessiné des formes les unes au-dessus des autres dans l'ordre dans lequel elles sont dessinées. C'est le comportement par défaut. Cependant, il est non seulement possible de dessiner les nouvelles formes derrière des formes existantes, mais également de définir la façon dont elles interagissent grâce à la propriété globalCompositeOperation

Testons par exemple le comportement destination-over : les nouvelles formes sont dessinées derrière le contenu existant du canvas. Essayez aussi lighter, darker ou xor.

Vous trouverez à la fin de ce chapitre un lien vous permettant d'essayer d'autres comportements.

s10 ---------

Pour finir ce chapitre, nous vous proposons de réaliser une animation dans laquelle trois cercles descendent verticalement à une vitesse différente.

Petit indice, commencez par définir une fonction générique qui sauve le contexte, dessine un cercle (dont la vitesse est passée en argument de cette fonction) et enfin restaure le contexte.

Si vous voulez pousser plus loin, vous pouvez assez facilement - en enrichissant cette fonction générique - réaliser une animation avec des cercles de tailles, de couleur et de vitesses verticales différentes et - pourquoi pas - changeantes aléatoirement au cours du temps.

Bon travail !

s11 ---------

Nous venons de découvrir comment réaliser une animation grâce à HTML5. Dans la leçon suivante, je vous propose de voir comment on peut introduire de l'interactivité dans une animation. Nous allons reprendre le concept d'événement JavaScript et l'adapter au contexte du canvas.

À bientôt.