Introduction à l'animation avec canvas en HTML5

Dessin et animation

dans votre navigateur

Rappels sur la balise `<canvas>`

- C'est une paire de balises ouvrante/fermante, similaire à `<div>`

```
<div> </div>
<canvas> </canvas>
```

- Cependent, la zone délimitée par les balises `<canvas>` peut être utilisée pour dessiner ou réaliser des animations

Les attributs de la balise <canvas>

```
<canvas id="myCanvas" width="600"
height="400"> </canvas>
```

- Avant de pouvoir utiliser le canvas pour dessiner ou faire une animation il **faut** définir les attributs ID, width, et height.
- Ils doivent soit directement apparaitre dans le HTML ou être crées avec JavaScript/jQuery.

Contenu par défaut pour <canvas>

```
<canvas id="myCanvas" width="600"
height="400">
<p>Some default content can appear
here.</p>
</canvas>
```

- Dans des navigateurs ne supportant pas HTML5, le canevas n'apparaît pas.
- En mettant un contenu par défaut entre les balises canvas, seuls des utilisateurs sans support pour *HTML5* le verront.

Faire des choses avec <canvas>

```
<canvas id="myCanvas" width="600"
height="400">
<p>Some default content can appear
here.</p>
</canvas>
```

- C’est la seule chose qui figurera dans votre html, tout le reste se fait en JavaScript!

Qu'y a-t-il dans le fichier **.js**?

- Le code JavaScript ne doit pas commencer à s'executer avant que le HTML ne soit chargé.
- Nous devons donc utiliser l'évènement `window.onload` dans le fichier **.js**
- Tout le code opérant sur le canevas doit être appelé depuis une fonction se déclanchant lors de `window.onload`

Une fonction pour appeler votre code canvas (1)

```
window.onload = draw;
// appelle une fonction "draw" – voir ci-dessous

function draw() {
        // mettre tout le code de dessin ici

} // on ferme la fonction draw()
```

C'est l'un des moyens d'appeler du code de dessin.

Une fonction pour appeler votre code canvas (2)

```
window.onload = function () {
// appelle une fonction annonyme


        // mettez tout votre code de dessin ici

} // fermez la fonction anonyme
```

C'est un autre moyen d'appeler du code de dessin.

Cibler le canvas par son attribut ID (défini dans le html)

```
window.onload = draw;
// Appel de la fonction "draw"

function draw() {
    var canvas = document.getElementById('myCanvas');
    // un canvas avec id="myCanvas"

        // mettez tout votre code de dessin ici


}
```

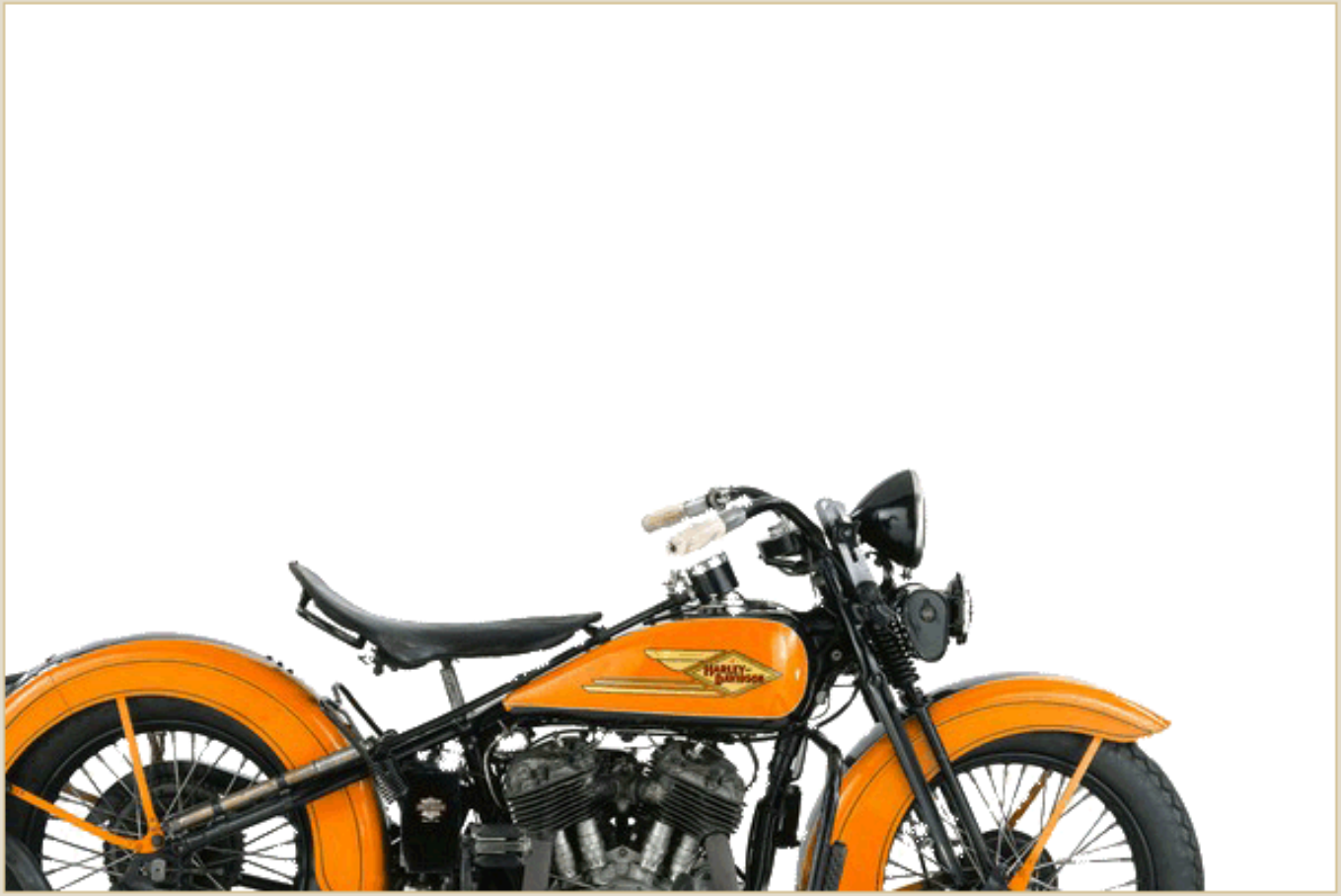
Ajout d'un contexte, encapsuler avec une condition 'if'

```
window.onload = draw;

function draw() {
   var canvas = document.getElementById('myCanvas');
   // canvas avec pour id="myCanvas"
      if (canvas.getContext) {
         var ctx = canvas.getContext('2d');

         // mettez tout votre code de dessin ici


      } // fermer if
}
```

Le "if" empêche JavaScript de lever une exception si le canvas n'est pas présent ou ne fonctionne pas.

Un petit exemple statique …

Canvas and Image Files



Above this, there is a canvas tag.

L’image de la moto fait 600 x 300.comment *la faire bouger pour qu’elle soit enièrement visible*?

Canvas and Image Files

Above this, there is a canvas tag.

Il faut diviser sa taille par deux en gardant les proportions…

Rétraissir la moto

```
// le code qui a été donné dans les diapos d'avant

   var img = new Image();
   img.onload = function() {
         ctx.drawImage( img, 300, 50, 300, 150 );
   }

   img.src = 'images/motorcycle.png';
```

Une petite animation simple…

Canvas and Animation

Press Return/Enter to see the animation.

```
window.onload = init; // Appel de la fonction "init"

//ici on définit une variable timer
var newInterval;


// initialisation des images et appel à "draw"
var bgImage   = new Image();
var motoImage = new Image();
function init() {
    bgImage.src   = "images/sketch.jpg";
    motoImage.src = "images/motorcycle.png";
    draw();
}
```

```
function draw() {
    var ctx =
document.getElementById('motoCanvas').getContext('2d');

    ctx.drawImage(bgImage, 0, 0, 600, 450); // afficher le fond

    // création d'un objet avec des valeurs
    // pour garder l'état de la moto à animer
    var moto = {
        factor: 0.991,
        x: -600, // on commence en la mettant en dehors du canvas
        y: 400,
        w: motoImage.width,
        h: motoImage.height
    }
```

Le début de la fonction “draw”…

```
var render = function () {// on définit une fonction d'animation
                          // qui sera appelée à chaque frame
    if (moto.x < 650) {
        ctx.drawImage(bgImage, 0, 0);
        // il faut redessiner le fond à chaque fois
        ctx.drawImage(motoImage, moto.x, moto.y, moto.w, moto.h);
    // ici nous allons changer les valeurs de la position de la
    // moto et de sa taille. C'est ici que se fait l'animation!
        moto.x += 10; // déplacer 10 px à droite
        moto.y -= 2.5;  // move 3 px closer to top
        moto.w = moto.w * moto.factor; // decrease size
        moto.h = moto.h * moto.factor; // decrease size
    } else {//Quand la moto est sortie de l'écran à l'opposé
        clearInterval(newInterval); // arrêt du timer
        // on remet à zéro pour rejouer:
        moto.x = -600;
        moto.y = 400;
        moto.w = motoImage.width;
        moto.h = motoImage.height;
    }
}
```

Fonction draw, la suite …

```
    //On doit appuyer sur entrée pour démarrer arrêter l'animation
    document.body.onkeydown = function(e) {  // On attends
        e = event || window.event;                  // n'importe quel
                                                       évènement
        var keycode = e.charCode || e.keyCode; //ou appui de touche
        if(keycode === 13) {  // on ne veut que la touche Entrée
     // On déclenche l'appel à "render" à intervalle régulier
        (en millisecondes)
            newInterval = setInterval(render, 16);
    // Pour avoir 60 fps (idéal sur un écran 60Hz), il faut un
    // intervalle de 1000/60 = 16.333 ms,
    // il faut arrondir soit à 16, soit à 17…
        }
    }

}
```

Suite de la fonction "draw" … et fin

Comment dessiner une fonction périodique?

- Prenons **sin** qui a une période de T=2*Pi
- En entrée elle prend une valeur modulo sa période: sin(T) = sin(2T) = sin (nT) donc entre 0 et T (ici 0 et 2*Pi donc).
- En sortie elle retourne une valeur entre -1 et 1
- Si on veut la dessiner sur un canvas de 800x600 en entier, il faut l'étirer (de 0 à 2*Pi vers 0 à 800 pour les x et de -1 à 1 vers 0 à 600 pour les y)

Tracé de fonction périodique

- On va devoir changer d'échelle pour passer de l'espace fonction à l'espace canvas avec la formule suivante:

$$newVal = \frac{(newmax - newmin) * (value - min)}{(max - min) + newmin;}$$

- Pour tracer la courbe, il faut dessiner sa valeur sur le canvas tous les **pas** pixels (le **pas** sera le seuil de quantization, par exemple pas=2).

Tous les **pas** pixels en **x**…

- Il faut itérer sur l'axe des x pour **i** de 0 à largeur (0 à 800 ici)
  - Changer l'échelle de **i** de [0..800] vers [0..T]
  - Passer la valeur résultante à la fonction sinus
- Pour chaque i, on obtiens une valeur j dans [-1…1]
  - On la fait passer dans l'échelle du canvas
  - On a désormais i, la coordonnée x en échelle du canvas et j, la coordonnée y en échelle du canvas du point à tracer
- On trace le segment entre le i-1, j-1 de l'itération précédente et le i, j courant

Dans le js… quelques paramètres!

```
window.onload = draw;

var echelleMaxY = 1; // Maximum Y dans l'espace fonction (sin va de -1 à 1)
var echelleMinY = -1; // Minimum Y dans l'espace fonction
var echelleMaxX = 10; // Maximum X dans l'espace fonction (ici seront visibles (Max-Min)/2PI périodes)
var echelleMinX = -10; // Minimum X dans l'espace fonction
var pas=1; // Pas de tracé, nombre de pixels entre chaque point de la fonction dessiné. Le plus pas est grand, la moins lisse la fonction à l'air

var fonction = Math.sin; // La fonction à dessiner
var periode= Math.PI*2; // La période de la fonction à dessiner
```

Le tracé: début

```
function tracerFonctionPeriodique(phase){

        var canvas = document.getElementById('sigcanvas');
        var ctx = canvas.getContext('2d');
        var largeur = canvas.offsetWidth;
        var hauteur = canvas.offsetHeight;

        ctx.beginPath(); // Début du chemin

        //On calcule la première valeur à la position 0
        //On a besoin de convertir de l'intervalle de la fonction vers
        //l'intervalle du canvas, pour cela on utilise la fonction miseEchele
        valeurFonction = fonction(0)
        valeurY = miseEchele(valeurFonction, echelleMinY, echelleMinY, 0, hauteur);

        //On pose le premier point du chemin
        ctx.moveTo(0,hauteur-valeurY);
```

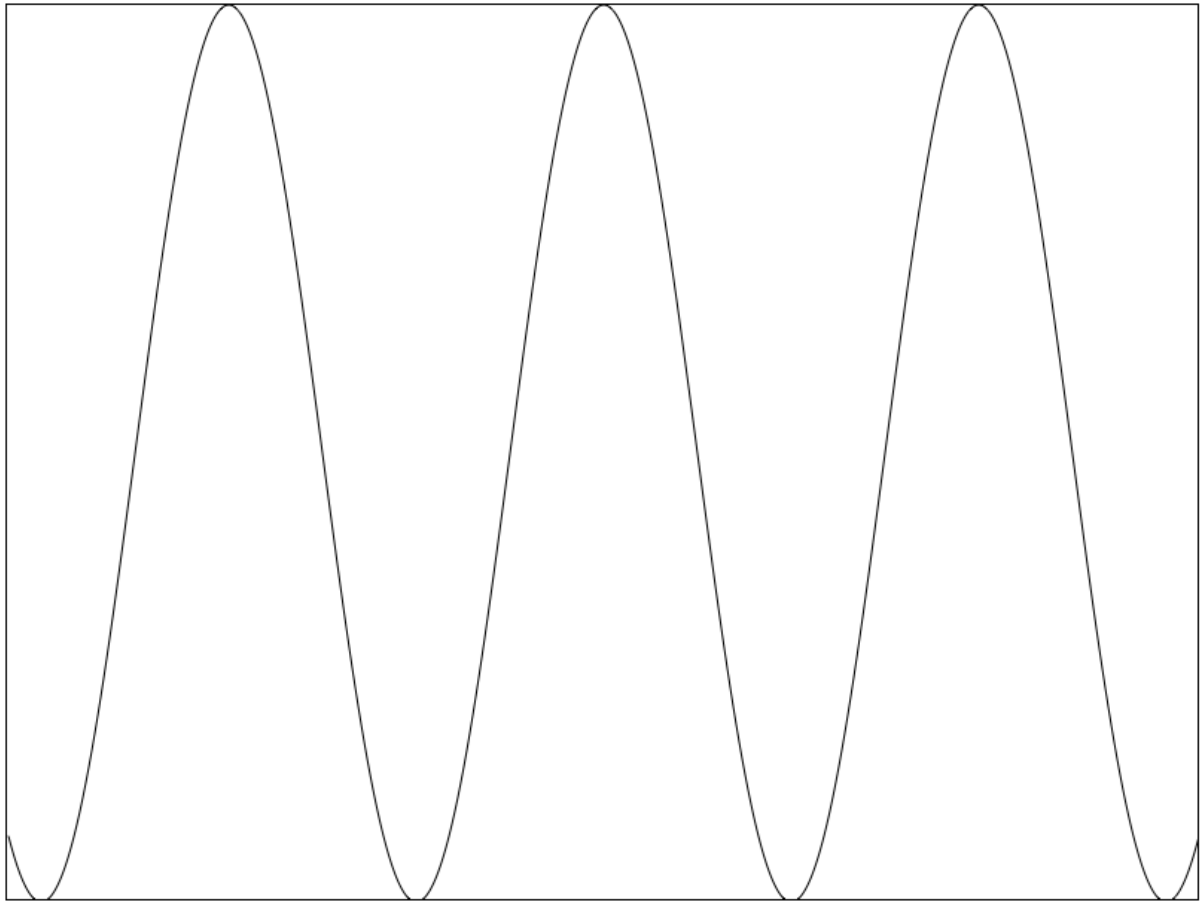
Le tracé: suite

```
        //On trace le reste tous les *pas* pixels
        for (i=pas;i<largeur;i+=pas){
            //On convertis la coordonnée X du point courant à tracer dans l'espace
            //de coordonnées de la fonction
            valeurX = miseEchele(i,0,largeur,echelleMinX,echelleMaxX);
            //On calcule la valeur de la fonction, la phase est un
            //décalage entre 0 et T (la période de la fonction)
            valeurFonction = fonction(valeurX + phase)
            //On convertir dans l'échelle de tracé
            valeurY = miseEchele(valeurFonction, echelleMinY, echelleMaxY, 0, hauteur);
            //On ajoute le segment depuis le point précédent
            ctx.lineTo(i,hauteur-valeurY);
        }
        //On peint le chemin à l'écran
        ctx.stroke();
}
```

La fonction draw

```
function draw(){
        // Tracé de la courbe avec un décalage
        // de phase de 0
        tracerFonctionPeriodique(0);
}
```

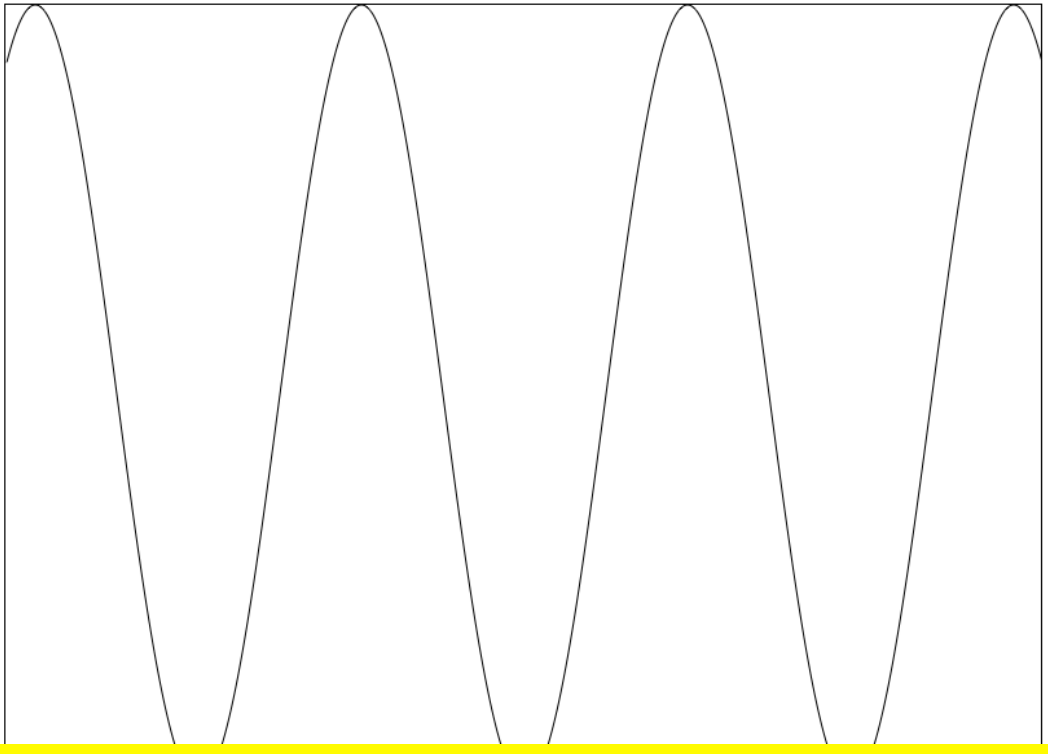
Et le résultat!



La fonction draw… tracé avec une phase de Pi?

```
function draw(){
        // Tracé de la courbe avec un décalage
        // de phase de Pi
        tracerFonctionPeriodique(Math.PI);
}
```

La courbe est décalée…



On peut donc utiliser la phase pour animer en faisant varier la valeur entre 0 et T. (ici 2PI, freqEchant nous donnera le nombre de cycles de phase de 0 à 2*PI par seconde)

Et pour animer?

```
var fps=60; // Le FPS de l'animation
var freqEchant = 2; // Le nombre de périodes par seconde
function draw(){
    var frame = 0; // Compteur de frames
    var render = function (){  // Fonction de rendu
        var canvas = document.getElementById('sigcanvas');
        var ctx = canvas.getContext('2d');
        var largeur = canvas.offsetWidth;
        var hauteur = canvas.offsetHeight;
        ctx.clearRect(0, 0, largeur, hauteur); //On efface la
frame précédente
        […]
```

Et pour animer?

```
//Pour animer, on va jouer sur la phase (décalage dans la période), on voudra en 1000 ms afficher freqEchant phases de 2*Pi
// On va donc compter le numéro de la frame courante (entre 0 et fps)
// et l'utiliser pour calculer la phase de la frame courante
        valPhase = freqEchant*periode*frame/fps;
//Tracé de la courbe
        tracerFonctionPeriodique(valPhase);
//On passe le numéro de frame à +1; si on atteint 60, on revient à 0.
        frame = (frame+1)%fps;
    }
// On appelle périodiquement la fonction de rendu toutes les 1000/fps millisecondes (17 ici)
        timer = setInterval(render,1000/fps);
}
```

Comment mieux synchroniser aux rafraichissements de l'écran?

- La fonction setInterval prends un nombre entier de millisecondes
- Or pour arriver à 60 fps, il faut un intervalle de 16.33333. Si on arrondis à 16 et 17 il y aura un décalage et ont va perdre en fluidité
- On peut alors utiliser une nouvelle fonction:

```
window.requestAnimFrame = (function(){
 return  window.requestAnimationFrame       ||
         window.webkitRequestAnimationFrame ||
         window.mozRequestAnimationFrame    ||
         function( callback ){
          window.setTimeout(callback, 1000 / 60);
         };
})();
```