1- LES DESSINS 2D– PARTIE 1

Présenté par Ronald Jean-Julien

420-P46: Programmation 3D

Hiver 2020

LES DESSINS

- En ***HTML5***, la balise <canvas></canvas> a été ajoutée. Cette balise est utilisée pour dessiner.
- Au point de départ, c'est un rectangle vide.
- On doit habituellement donner une taille au canevas pour délimiter la surface du dessin. Ici, la surface de dessin est de 350X350 pixels.
- Si le navigateur ne supporte pas cette balise, le texte situé à l'intérieur de la balise va être affiché sur la page Web.

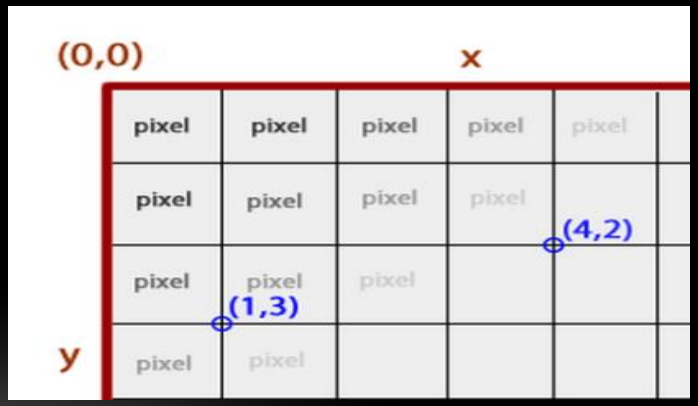
```
<canvas id="monCanvas" width="350" height="350">
        Votre navigateur ne supporte pas la balise canvas
</canvas>
```

LE CONTEXTE 2D (1)

- Pour dessiner, on doit le faire en *Javascript*.
- L'objet le plus important est le contexte. En fait, le contexte représente la surface du dessin ainsi que la librairie qu'on va utiliser pour dessiner.
- Ici, nous allons utiliser le contexte 2D (**objC2D**).

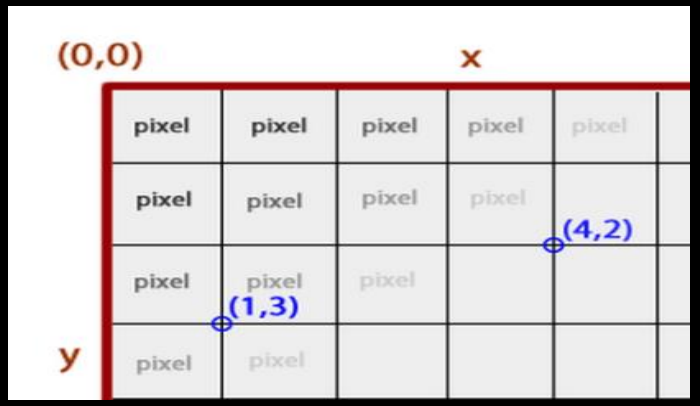
```
var objCanvas = document.getElementById('monCanvas');
var objC2D = objCanvas.getContext('2d');
```

- Dans le contexte 2D, la surface de dessin est exprimée en pixels
et le point de référence (0,0)
est situé en haut à gauche de la surface du dessin.



LE CONTEXTE 2D (2)

- Il n'est pas nécessaire que les coordonnées soient des nombres entiers. Étant donné que chaque point représente un pixel alors chaque coordonnée sera arrondie ou interpolée à l'affichage.
- On peut dessiner à l'extérieur de la surface du dessin mais l'utilisateur ne peut pas voir ce qui a été dessiné.
  Par exemple, les coordonnées (-1,1) sont situées à l'extérieur de la surface du dessin. Il est possible de dessiner à cet endroit mais l'utilisateur ne voit rien.



LES TRACÉS (1)

- La base du contexte 2D, ce sont les tracés. Voici quelques méthodes et propriétés.
- .beginPath() : Pour définir un nouveau tracé.
- .moveTo(intX,intY) : Pour déplacer « le crayon à main levée » au début du tracé.
- .lineTo(intX,intY) : Pour tracer un segment en ligne droite à l'intérieur du tracé.
- .closePath() : Lorsqu'on désire que le début et la fin du tracé se rejoignent
- .lineWidth : La largeur d'un segment de droite (par défaut 1 pixel)
- .setLineDash : Le style de segment de droite (par défaut plein)
- .lineCap : Le style de fin du segment de droite (par défaut plat)
- .lineJoin : Le style de rencontre entre 2 segments de droite (par défaut angle droit)
- .strokeStyle : Le style de couleur pour le contour du tracé (par défaut noir)
- .fillStyle : Le style de couleur pour le remplissage du tracé (par défaut noir)

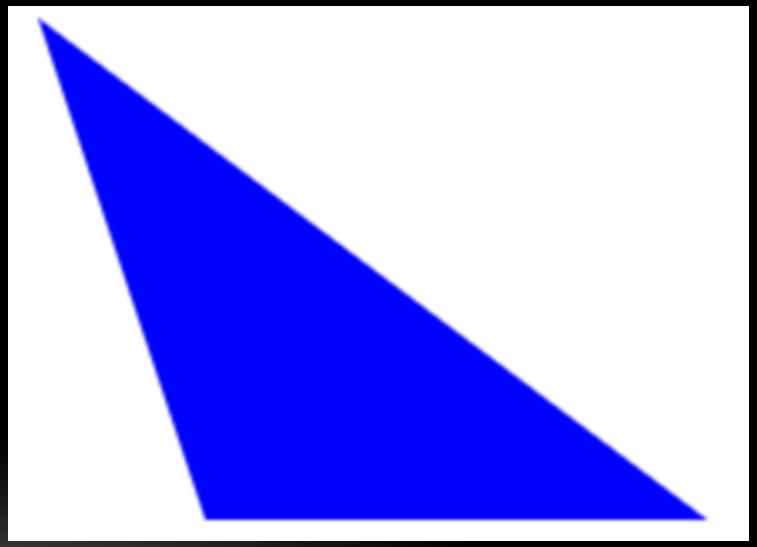
LES TRACÉS (2)

- Lorsque le tracé a complètement été défini, on peut dessiner le tracé
- .stroke() : Pour dessiner le contour du tracé
- .fill() : Pour dessiner le remplissage du tracé (pour remplir le tracé)
- **ATTENTION**: On doit toujours définir le tracé, avant de dessiner son contour et/ou son remplissage

EXEMPLE DE TRACÉ TRIANGULAIRE

```
// Le triangle plein sans contour
objC2D.beginPath();          //  Définir un nouveau tracé
objC2D.moveTo(50,80);        // Le crayon au point (50,80)
objC2D.lineTo(100,230);      // Un segment de droite vers le point (100,230)
objC2D.lineTo(250,230);      // Un segment de droite vers le point (250,230)
objC2D.closePath();          //  Fermer le tracé
objC2D.fillStyle = 'blue';   // Style de couleur du remplissage
objC2D.fill();               // Remplir le tracé
```

Attention: **C'est la méthode .fill() qui dessine réellement. Les autres méthodes et propriétés ne sont utilisées que pour définir le tracé.**



EXEMPLE DE TRACÉ RECTANGULAIRE

```
// Le rectangle plein avec contour
objC2D.beginPath();          //  Définir un nouveau tracé
objC2D.moveTo(50,250);  objC2D.lineTo(50,300);
objC2D.lineTo(250,300);  objC2D.lineTo(250,250); // Le tracé du rectangle
objC2D.closePath();           //  Fermer le tracé
objC2D.fillStyle = 'yellow';  // Style de couleur du remplissage
objC2D.strokeStyle = 'black';    // Style de couleur du contour
 objC2D.lineWidth = 5;        // Largeur de la ligne du contour
objC2D.fill();                // Remplir le tracé
objC2D.stroke();              // Dessiner le contour du tracé
```

Attention: Ce sont les méthodes **.fill()** et **.stroke()** qui dessinent réellement. Les autres méthodes et propriétés ne sont utilisées que pour définir le tracé.

EXEMPLE DE SEGMENT AVEC BOUTS ARRONDIS

```
// Un segment de droite avec bouts arrondis
objC2D.beginPath();           //  Définir un nouveau tracé
objC2D.moveTo(50,50);  objC2D.lineTo(250, 50);
objC2D.lineCap = 'round'; // Bout arrondi
objC2D.strokeStyle = 'black';    // Couleur du segment
objC2D.lineWidth = 20;         // Largeur du segment
objC2D.stroke();               // Dessiner le segment
```

EXEMPLE DE RECTANGLE VIDE AVEC COINS ARRONDIS

```
// Le rectangle plein avec contour
objC2D.beginPath();           //  Définir un nouveau tracé
objC2D.moveTo(50,250);  objC2D.lineTo(50,300);
objC2D.lineTo(250,300);  objC2D.lineTo(250,250); // Le tracé du rectangle
objC2D.closePath();            //  Fermer le tracé
objC2D.strokeStyle = 'black';    // Style de couleur des segments
objC2D.lineWidth = 20;         // Largeur des segments
objC2D.lineJoin = 'round'      // Rencontre de 2 segments arrondis
objC2D.stroke();               // Dessiner le contour
```

EXEMPLES DE STYLES DE SEGMENT (1)

```
// Style: 5 pleins, 5 vides, 5 pleins, 5 vides, ...
objC2D.beginPath();          //  Définir un nouveau tracé
objC2D.moveTo(50,50);  objC2D.lineTo(250, 50);
objC2D.setLineDash([5]); // Style de segment
objC2D.stroke();             // Dessiner le segment

// Style: 10 pleins, 5 vides, 10 pleins, 5 vides, ...
objC2D.beginPath();          //  Définir un nouveau tracé
objC2D.moveTo(50,100);  objC2D.lineTo(250, 100);
objC2D.setLineDash([10,5]); // Style de segment
objC2D.stroke();             // Dessiner le segment
```

La méthode setLineDash définit le modèle à utiliser pour les pointillés lors du dessin de la ligne, en utilisant un tableau de valeurs qui spécifie les longueurs des alternances entre pleins et creux.

EXEMPLES DE STYLES DE SEGMENT (2)

```
// Style: 10 pleins, 5 vides, 5 pleins, 10 vides, ...
objC2D.beginPath();          //  Définir un nouveau tracé
objC2D.moveTo(50,150);  objC2D.lineTo(250,150);
objC2D.setLineDash([10,5,5,10]); // Style de segment
objC2D.stroke();             // Dessiner le segment

// Segment plein
objC2D.beginPath();          //  Définir un nouveau tracé
objC2D.moveTo(50,200);  objC2D.lineTo(250, 200);
objC2D.setLineDash([]); // Style de segment
objC2D.stroke();             // Dessiner le segment
```

LES TRACÉS RECTANGULAIRES

- En 2D, il arrive fréquemment que l'on ait des rectangles à tracer. Plusieurs fonctions ont été définies pour les tracés rectangulaires.
- .rect(intX,intY, intLargeur, intHauteur) **:** Pour définir un tracé rectangulaire
- .fillRect(intX,intY, intLargeur, intHauteur) : Pour définir et remplir un tracé rectangulaire
- .strokeRect(intX,intY, intLargeur, intHauteur) : Pour définir et dessiner les contours
- .clearRect(intX,intY, intLargeur, intHauteur) : Pour effacer un rectangle sur le canevas

EXEMPLES DE TRACÉS RECTANGULAIRES

```
objC2D.beginPath();              // Définir un tracé
objC2D.rect(50,50, 200,100);     // Un  tracé rectangulaire
objC2D.fillStyle = 'purple';     // Couleur de remplissage
objC2D.fill();                   // Remplissage du tracé
```

```
objC2D.fillStyle = 'purple';        // Couleur de remplissage
objC2D.fillRect(50,50, 200,100);    // Définition et remplissage du tracé rectangulaire
```

LES TRACÉS CURVILIGNES (LES COURBES)

- En 2D, il existe quatre types de tracés curvilignes (de courbes)
  - Les arcs de cercle
  - Les ellipses
  - Les courbes de Bézier quadratiques
  - Les courbes de Bézier cubiques

LES ARCS DE CERCLE (1)

- .arc(intXCentre, intYCentre, intRayon, fltAngle1, fltAngle2, binSensAntiHoraire) :

  Pour définir un arc de cercle.

  (intXCentre, intYCentre) : Le centre du cercle

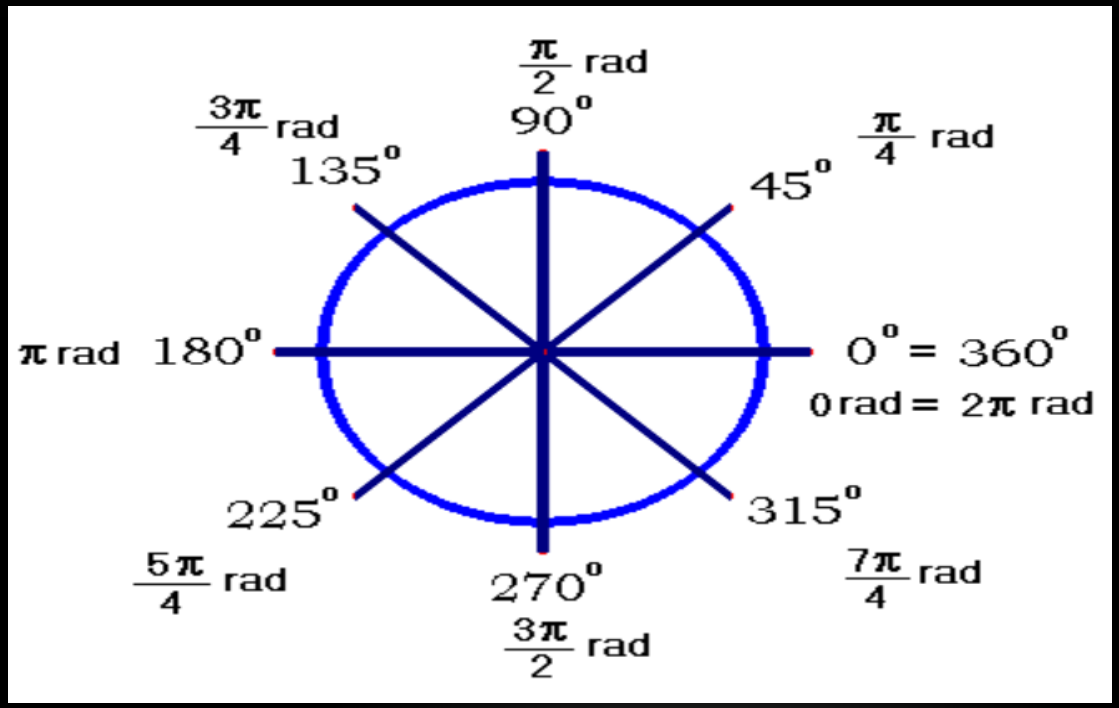
  intRayon : Le rayon du cercle

  (fltAngle1, fltAngle2) : L'angle de début et l'angle de fin de l'arc par rapport au cercle (mesurés en radians)

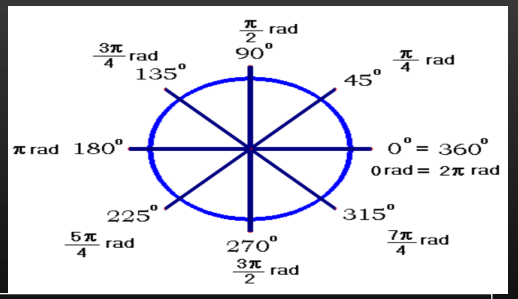
  binSensAntiHoraire : true si l'arc doit se dessiner dans le sens antihoraire.

LES ARCS DE CERCLE (2)

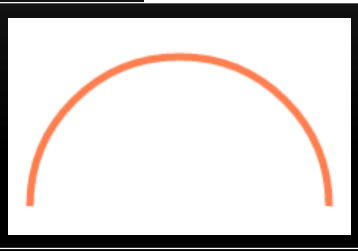
- π radians = 180 degrés ou 1 radian =180/π ou 1 degré = π /180



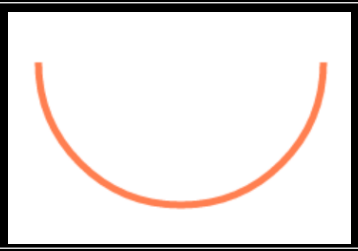
EXEMPLES D'ARCS DE CERCLE



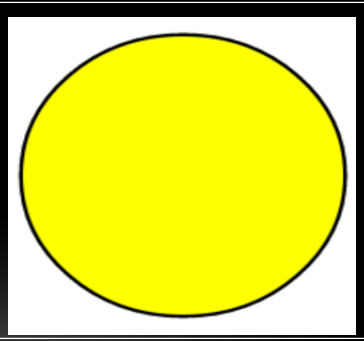

```
objC2D.strokeStyle = 'coral';
objC2D.lineWidth = 5;
objC2D.beginPath();
objC2D.arc(150,150,100, 0,Math.PI, true);
objC2D.stroke();
```



```
objC2D.strokeStyle = 'coral';
objC2D.lineWidth = 5;
objC2D.beginPath();
objC2D.arc(150,150,100, 0,Math.PI, false);
objC2D.stroke();
```



```
objC2D.strokeStyle = 'black';
objC2D.fillStyle = 'yellow';
objC2D.lineWidth = 5;
objC2D.beginPath();
objC2D.arc(150,150,100, 0,2*Math.PI, false);
objC2D.stroke();
objC2D.fill();
```



LES ELLIPSES

- .ellipse(intXCentre, intYCentre, intRayonX, intRayonY, fltRotation, fltAngle1, fltAngle2, binSensAntiHoraire) :

  Pour définir une ellipse (ou un arc d'ellipse)

  (intXCentre, intYCentre) : Le centre de l'ellipse

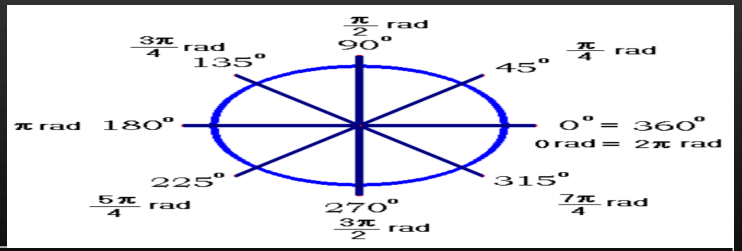
  (intRayonX, intRayonY) : Les rayons de l'ellipse (horizontal et vertical)

  fltRotation : La rotation appliquée à l'ellipse dans le sens horaire avant de la dessiner

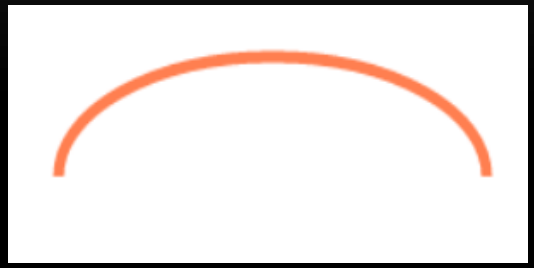
  (fltAngle1, fltAngle2) : L'angle de début et l'angle de fin de l'ellipse dessinée (mesurés en radians)

  binSensAntiHoraire : true si l'ellipse doit se dessiner dans le sens antihoraire.

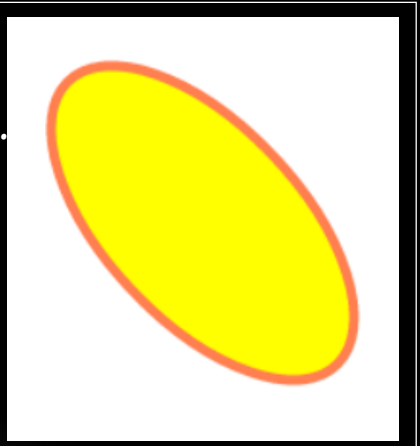
EXEMPLES D'ELLIPSE




```
// Portion d'ellipse de rayon (100,50) dessinée
// dans le sens antihoraire.
// Aucune rotation n'a été appliquée à l'ellipse
objC2D.strokeStyle = 'coral';
objC2D.lineWidth = 5;
objC2D.beginPath();
objC2D. ellipse(150,150, 100,50, 0, 0,Math.PI, true);
objC2D.stroke();
```

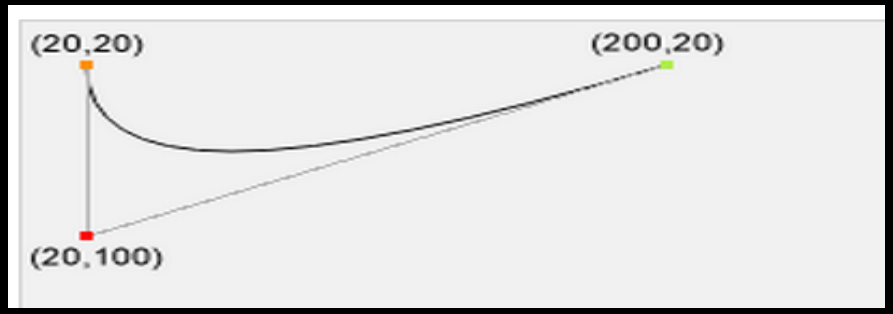


```
// Ellipse complète de rayon (100,50) dessinée
// dans le sens antihoraire.
// A subi une rotation de 45 degrés dans le sens horaire (π/4 radians).
objC2D.strokeStyle = 'coral';
objC2D.strokeStyle = 'yellow';
objC2D.lineWidth = 5;
objC2D.beginPath();
objC2D. ellipse(150,150, 100,50, Math.PI/4, 0,2*Math.PI, true);
objC2D.fill();
objC2D.stroke();
```



LES COURBES DE BÉZIER QUADRATIQUES (1)

- Une courbe de Bézier quadratique est une courbe dotée de 2 points d'attache et d'un point de contrôle.



- Ici, les points d'attache sont (20,20) et (200,20). Le point de contrôle est (20,100).
- On doit imaginer une courbe de Bézier quadratique comme étant un segment de droite dont les points d'attache sont reliés à une ficelle. En tirant sur la ficelle, le segment de droite se déforme ce qui donne une courbe.
- Les points d'attache sont faciles à déterminer mais pas le point de contrôle.

LES COURBES DE BÉZIER QUADRATIQUES (2)

- .quadraticCurveTo(intXControl, intYControl, intXFin, intYFin)

  Pour tracer une courbe de Bézier quadratique à partir d'un point d'attache

  (intXControl, intYControl) : Le point de contrôle

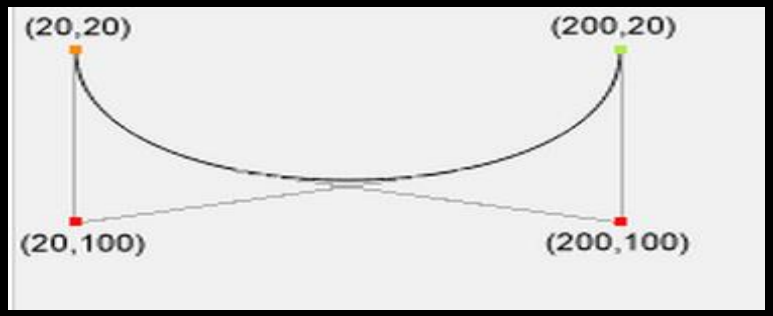
  (intXFin, intYFin) : Le 2ième point d'attache

EXEMPLE DE COURBE DE BÉZIER QUADRATIQUE

```
objC2D.strokeStyle = 'black';
objC2D.lineWidth = 5;
objC2D.beginPath();
objC2D.moveTo(20,20);  // Point d'attache #1
objC2D.quadraticCurveTo(20,100,200,20); // Point de contrôle et point d'attache #2
objC2D.stroke();
```

LES COURBES DE BÉZIER CUBIQUES (1)

- Une courbe de Bézier cubique est une courbe dotée de 2 points d'attache et de 2 points de contrôle.



- Ici, les points d'attache sont (20,20) et (200,20). Les points de contrôle sont (20,100) et (200, 100).
- Ici, on a un seul segment de droite mais 2 ficelles.

LES COURBES DE BÉZIER CUBIQUES (2)

- .bezierCurveTo(intXControl1, intYControl1, intXControl2, intYControl2, intXFin, intYFin)

  Pour tracer une courbe de Bézier cubique à partir d'un point d'attache

  (intXControl1, intYControl1) : Le point de contrôle 1

  (intXControl2, intYControl2) : Le point de contrôle 2

  (intXFin, intYFin) : Le 2ième point d'attache

EXEMPLE DE COURBE DE BÉZIER CUBIQUE

```
objC2D.strokeStyle = 'black';
objC2D.lineWidth = 5;
objC2D.beginPath();
objC2D.moveTo(20,20);  // Point d'attache #1
objC2D.bezierCurveTo(20,100,200,100,200,20); // Points de contrôle et point d'attache #2
objC2D.stroke();
```

EXEMPLE DE COURBES DE BÉZIER CUBIQUES

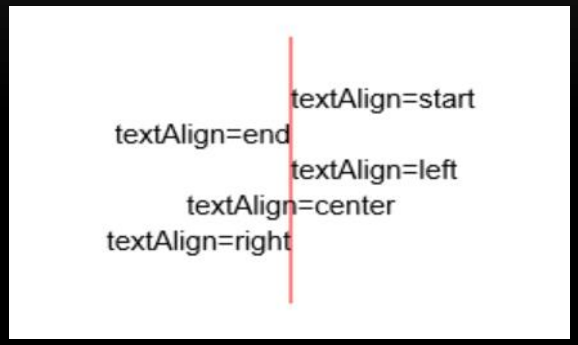
Nuage.htm

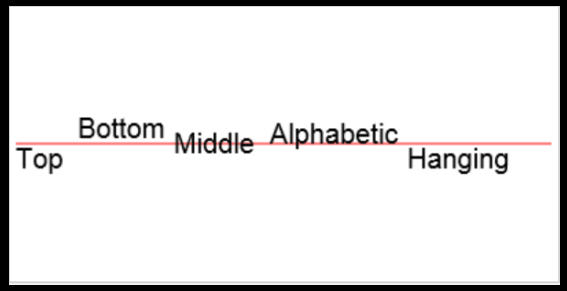
```
var objC2D = objCanvas.getContext('2d');

objC2D.beginPath();
objC2D.moveTo(70, 80);
objC2D.bezierCurveTo(30, 100, 30, 150, 130, 150);
objC2D.bezierCurveTo(150, 180, 220, 180, 240, 150);
objC2D.bezierCurveTo(320, 150, 320, 120, 290, 100);
objC2D.bezierCurveTo(330, 40, 270, 30, 240, 50);
objC2D.bezierCurveTo(220, 5, 150, 20, 150, 50);
objC2D.bezierCurveTo(100, 5, 50, 20, 70, 80);
objC2D.closePath();
objC2D.lineWidth = 5;
objC2D.fillStyle = '#8ED6FF';
objC2D.fill();
objC2D.strokeStyle = 'blue';
objC2D.stroke();
```

LE DESSIN D'UN TEXTE

- .font = La police de caractères
- .textAlign = À partir d'où le texte va se dessiner horizontalement ('start' par défaut)
- .textBaseLine = À partir d'où le texte va se dessiner verticalement ('alphabetic' par défaut)
- .strokeText(srTexte, intX, intY) : Pour dessiner le contour du texte
- .fillText(srTexte, intX, intY) : Pour remplir le texte
- .measureText(srTexte, intX, intY) : Pour mesurer (en pixels) le texte qui a été ou qui sera dessiné





EXEMPLE DU DESSIN D'UN TEXTE

```
objC2D.strokeStyle = 'black';
objC2D.fillStyle = 'yellow';
objC2D.lineWidth = 1;
objC2D.font = '30pt Verdana'; // Police de caractères
objC2D.textAlign = 'right'; // De la droite vers la gauche (à partir de la fin)
objC2D.fillText('Allo',objCanvas.width, 120); // Dessiner Allo
objC2D.strokeText('Allo',objCanvas.width,120);
```

LE DESSIN D'UNE IMAGE

- .drawImage(objImage, intX, intY, intLargeur, intHauteur) : Pour dessiner une image

  objImage: L'image qu'il faut dessiner

  (intX, intY) : La position de l'image (coin supérieur gauche) sur le canevas

  (intLargeur, intHauteur) : La taille de l'image; si on omet la taille, l'image va se dessiner dans sa taille originale

- **Attention**: Avant de dessiner une image, il faut la créer puis attendre qu'elle soit complètement chargée.

EXEMPLE DU DESSIN D'UNE IMAGE (1)

```
var objImage = new Image();   // Créer l'objet qui va contenir l'image
objImage.src = 'prof.jpg';   // Le fichier qui contient l'image
objImage.onload = function()
{   // L'image est dessinée lorsque le chargement de l'image est terminé
    objC2D.drawImage(objImage, 0, 0);
}
```

<u>Problème</u>: L'image n'est pas complètement dessinée car l'image est plus grande que le canevas.

EXEMPLE DU DESSIN D'UNE IMAGE (2)

```
var objImage = new Image();   // Créer l'objet qui va contenir l'image
objImage.src = 'prof.jpg';   // Le fichier qui contient l'image
objImage.onload = function()
{   // L'image est dessinée lorsque le chargement de l'image est terminé
    objC2D.drawImage(objImage, 0, 0, 300, 300);
}
```

Ici, nous avons affecté la taille 300X300 à l'image.
Elle est complètement située dans le canevas.

<u>Problème</u>: L'image n'a pas conservé ses proportions originales; elle semble déformée dans le sens horizontal.

EXEMPLE DU DESSIN D'UNE IMAGE (3)

```
var objImage = new Image();   // Créer l'objet qui va contenir l'image
objImage.src = 'prof.jpg';   // Le fichier qui contient l'image
objImage.onload = function()
{   // L'image est dessinée lorsque le chargement de l'image est terminé
    objC2D.drawImage(objImage, 0, 0, objImage.width * 0.50, objImage.height * 0.50);
}
```

Ici, nous avons appliqué une proportion (50% de la taille originale de l'image).

LES STYLES DE COULEURS

- En 2D, il existe trois (3) styles de couleurs :
  - Les couleurs pleines
  - Les dégradés (linéaires et radiaux)
  - Les motifs
- Dans les exemples précédents, nous avons utilisé des couleurs pleines.

LES DÉGRADÉS

- .createLinearGradient(intX1, intY1, intX2, intY2) : Pour créer un dégradé linéaire

  (intX1, intY1) : Le point de départ du dégradé (sur le canevas)

  (intX2, intY2) : Le point d'arrivée du dégradé (sur le canevas)

- .createRadialGradient(intX1, intY1, intR1, intX2, intY2, intR2) : Pour créer un dégradé radial

  (intX1, intY1, intR1) : Le cercle de départ du dégradé (sur le canevas)

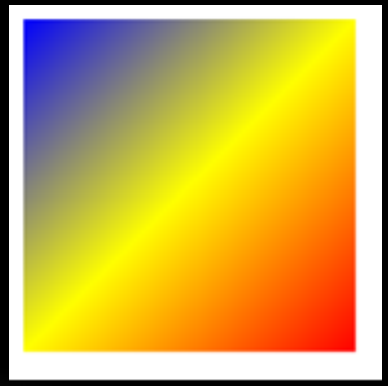
  (intX2, intY2, intR2) : Le cercle d'arrivée du dégradé (sur le canevas)

- .addColorStop(fltPcCouleur, strCouleur) : Pour ajouter une couleur à une certaine position (en pourcentage) sur le dégradé.

EXEMPLE D'UN DÉGRADÉ LINÉAIRE

```
// Création du dégradé linéaire
var objDegrade=objC2D.createLinearGradient(0,0, 170,170);
// Ajout de couleurs au dégradé
objDegrade.addColorStop(0, 'blue');
objDegrade.addColorStop(0.5, 'yellow');
objDegrade.addColorStop(1, 'red');
objC2D.fillStyle=objDegrade; // Appliquer ce dégradé au remplissage
objC2D.fillRect(0,0,170,170);
```

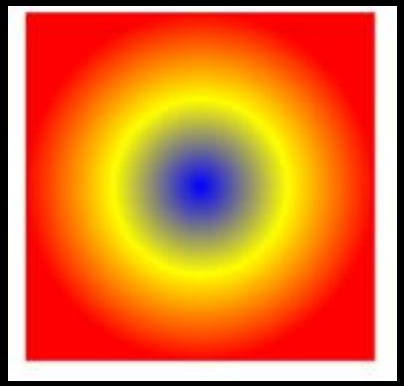
Le point de départ est (0,0)
Le point d'arrivée est (170,170)



EXEMPLE D'UN DÉGRADÉ RADIAL (1)

```
// Création du dégradé radial
var objDegrade=objC2D.createRadialGradient(85,85,0, 85,85,85);
// Ajout de couleurs au dégradé
objDegrade.addColorStop(0, 'blue');
objDegrade.addColorStop(0.5, 'yellow');
objDegrade.addColorStop(1, 'red');
objC2D.fillStyle=objDegrade; // Appliquer ce dégradé au remplissage
objC2D.fillRect(0,0,170,170);
```

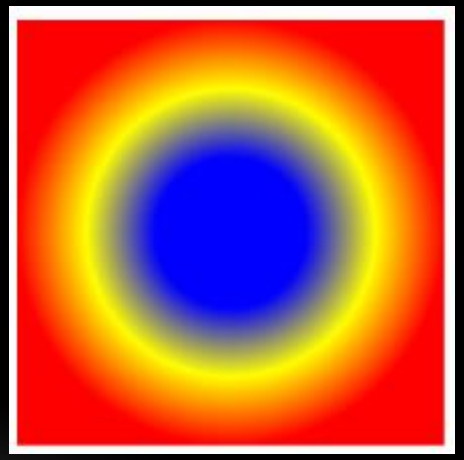
Le cercle de départ est (85,85,0)
Le cercle d'arrivée est (85,85,85)



EXEMPLE D'UN DÉGRADÉ RADIAL (2)

```
// Création du dégradé radial
var objDegrade=objC2D.createRadialGradient(85,85,30, 85,85,85);
// Ajout de couleurs au dégradé
objDegrade.addColorStop(0, 'blue');
objDegrade.addColorStop(0.5, 'yellow');
objDegrade.addColorStop(1, 'red');
objC2D.fillStyle=objDegrade; // Appliquer ce dégradé au remplissage
objC2D.fillRect(0,0,170,170);
```

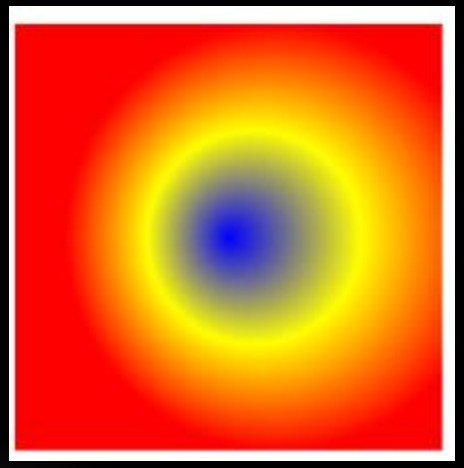
Le cercle de départ est (85,85,30)
Le cercle d'arrivée est (85,85,85)
Le cercle de départ est plus gros que le précédent



EXEMPLE D'UN DÉGRADÉ RADIAL (3)

```
// Création du dégradé radial
var objDegrade=objC2D.createRadialGradient(85,85,0, 105,85,85);
// Ajout de couleurs au dégradé
objDegrade.addColorStop(0, 'blue');
objDegrade.addColorStop(0.5, 'yellow');
objDegrade.addColorStop(1, 'red');
objC2D.fillStyle=objDegrade; // Appliquer ce dégradé au remplissage
objC2D.fillRect(0,0,170,170);
```

Le cercle de départ est (85,85,0)
Le cercle d'arrivée est (105,85,85)
Le cercle d'arrivée est décalé vers la droite



LES MOTIFS

- Un motif est une image que l'on utilise pour dessiner le contour et/ou le remplissage.
- .createPattern(objImage, strRépétition) : Pour créer un motif

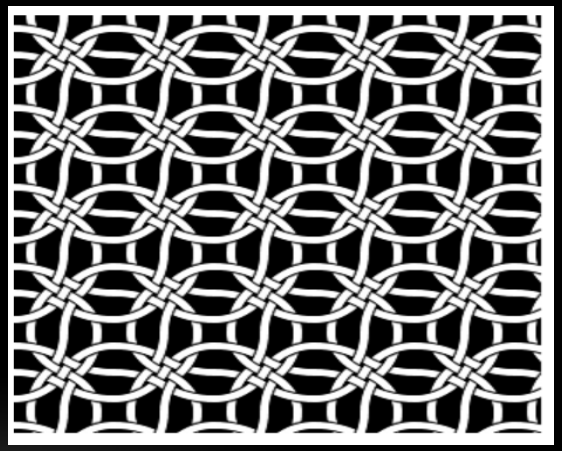
  objImage : L'image utilisée par le motif

  strRépétition : Peut avoir une des valeurs suivantes: ***repeat***, ***repeat-x***, ***repeat-y*** et ***no-repeat***
- **Attention**: Avant de mettre l'image à l'intérieur du motif, il faut la créer puis attendre qu'elle soit complètement chargée.

EXEMPLE D'UN MOTIF

```
var objImage = new Image();
objImage.src = 'motif.png'
objImage.onload = function() {
    objMotif= objC2D.createPattern(objImage, 'repeat'); // Créer le motif (répétitif)
    objC2D.fillStyle = objMotif; // Appliquer le motif au remplissage
    objC2D.fillRect(0,0,300,300);
}
```

L'image originale:



AUTRE EXEMPLE D'UN MOTIF

```
var objImage = new Image();
objImage.src = 'motif.png';
objImage.onload = function() {
    objMotif= objC2D.createPattern(objImage, 'repeat');
    objC2D.fillStyle = objMotif; // Appliquer le motif au remplissage
    var strTexte = '420-4P6';
    objC2D.font = '80px Arial';
    objC2D.textAlign = "left";
    objC2D.fillText(strTexte, 0, 80);
}
```