Travail demandé :

Compléter et tester les différents exemples et faires les exercices**.**

Sommaire

Activité 1 : alpha, scale, rotate, translate

Alpha :
Pour faire un dessin en transparence, on modifie la propriété *globalAlpha* du contexte
context.globalAlpha =0.5;

A faire : Exemple :

```
<!DOCTYPE html>
<html>
<body>
    <canvas id="myCanvas" width="300" height="150" style="border:1px solid #d3d3d3;">
    Votre navigateur ne supporte pas les canvas HTML5</canvas>
```

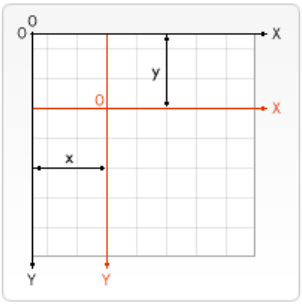
```
        <script>
        var c = document.getElementById("myCanvas");
        var ctx = c.getContext("2d");
        ctx.fillStyle = "red";
        ctx.fillRect(20, 20, 75, 50);
        //Turn transparency on
        // Appliquer globalAlpha de 0.2
        ctx.fillStyle = "blue";
        ctx.fillRect(50, 50, 75, 50);
        ctx.fillStyle = "green";
        ctx.fillRect(80, 80, 75, 50);
        </script>
</body>

</html>
```

Activité 2: La translation

Pour effectuer une translation, on utilise la méthode :

context.translate(translateX, translateY);

Où translateX est le déplacement sur l'axe des x (en pixels) et translateY la même chose mais sur l'axe des y.



.

A faire : Exemple :

```
<!DOCTYPE html>
<html>
<head>
<title>HTML5 Canvas - translate</title>
</head>
<body>
  <canvas id="DemoCanvas" width="300" height="400"></canvas>
  <script>
        var canvas = document.getElementById("DemoCanvas");
        if (canvas.getContext)
        {
```

```
        var ctx = canvas.getContext('2d');
        ctx.beginPath();
        ctx.lineWidth = "3";
        ctx.strokeStyle = "blue";
        ctx.strokeRect(60, 60, 160, 160);
        //déplacement de  (60,60)
        ctx.strokeStyle = "red";
        ctx.strokeRect(60, 60, 160, 160);
        ctx.stroke();
        }
</script>
</body>
</html>
```

Le scale :

Activité 3 : Changement d'échelle avec scale

La méthode scale Met à l'échelle les unités du canevas avec x horizontalement et y verticalement. Les deux paramètres sont des nombres réels. Les valeurs inférieures à 1,0 réduisent la taille de l'unité et les valeurs supérieures à 1,0 augmentent la taille de l'unité. Les valeurs 1.0 laissent les unités à la même taille.

```
context.scale( scaleX, scaleY);
```

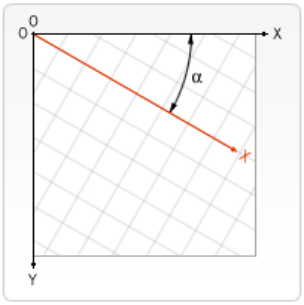
Où *scaleX* est l'échelle sur l'axe des x que vous souhaitez obtenir et *scaleY* la même chose mais sur l'axe des y.

A faire : Exemple :

```
<!DOCTYPE html>
<html>
   <body>
        <canvas id="myCanvas" width="300" height="150" style="border:1px solid #d3d3d3;">
        Votre navigateur ne supporte pas les canvas HTML5.</canvas>

        <script>
        var c = document.getElementById("myCanvas");
        var ctx = c.getContext("2d");
        ctx.strokeRect(5, 5, 25, 15);
        // Appliquer un changement d'échelle de (2, 2);
        ctx.strokeRect(5, 5, 25, 15);
        </script>

   </body>
</html>
```

Activité 4 : La rotation :

La méthode ***rotate*** Fait pivoter le canevas, dans le sens des aiguilles d'une montre autour de l'origine actuelle, par le nombre de radians de l'angle ***context.rotate(angle_radian);***



A faire : **Exemple :**

```
<!DOCTYPE html>
<html>
   <body>
        <canvas id="myCanvas" width="300" height="150" style="border:1px solid
        #d3d3d3;">
        Votre navigateur ne  supporte pas les canvas the HTML5.</canvas>

        <script>
        var c = document.getElementById("myCanvas");
        var ctx = c.getContext("2d");
        // Rotation de 20 * Math.PI / 180 puis de Math.PI / 4
        ctx.fillRect(50, 20, 100, 50);
        </script>

   </body>
</html>
```

Activité 5 : Cumul des transformations, sauvegarde et restauration du contexte

L'objet context utilise une **matrice** pour représenter et stocker le résultat de toutes les transformations qu'on lui applique.

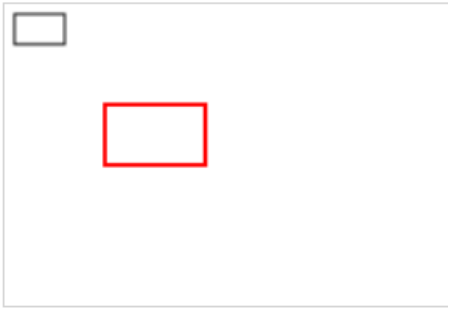
Remarque 1 : Les transformations que l'on applique à une matrice se cumulent et l'ordre dans lequel on les exécute influe sur le résultat obtenu.

A faire : Exemple 1 :

```
<!DOCTYPE html>
<html>
   <body>
```

```
        <canvas id="myCanvas" width="300" height="150" style="border:1px solid
        #d3d3d3;">
        Your browser does not support the HTML5 canvas tag.</canvas>

        <script>
          var c = document.getElementById("myCanvas");
          var ctx = c.getContext("2d");
          ctx.strokeRect(5, 5, 25, 15);
          //scale de (2,2)  puis translate de (20,20)
        // Dessiner un rectangle creux de bordure rouge avec les données (5, 5, 25, 15);
        </script>
    </body>
</html>
```

Résultat à l'écran :



Exemple 2

```
<!DOCTYPE html>
<html>
    <body>

        <canvas id="myCanvas" width="300" height="150" style="border:1px solid
        #d3d3d3;">
        Your browser does not support the HTML5 canvas tag.</canvas>

        <script>
                var c = document.getElementById("myCanvas");
                var ctx = c.getContext("2d");
                ctx.strokeRect(5, 5, 25, 15);
                //translate puis scale
                // Dessiner un rectangle creux de bordure rouge avec les données (5, 5, 25, 15);
        </script>

    </body>
</html>
```

Résultat à l'écran :

On peut voir que le résultat obtenu à l'écran est différent suivant que l'on applique scale **avant** ou **après** la translation

Remarque 2 : Si je veux définir une échelle à 0,5 après avoir l'avoir définie à 2, le code suivant ne fonctionne pas :

context.scale(2, 2); // l'échelle est à 2

context.scale(0.5, 0.5);

// ici l'échelle ne vaut pas 0.5 MAIS 1 car on a multiplié la valeur

// de l'échelle courante par 0.5, donc le résultat est 1, pour avoir une valeur d'échelle à 2, il aurait fallu appliquer un scale de 0,25

Activité 6 : save() et restore()

Il est possible de stocker en mémoire l'état du contexte et de le restaurer par la suite. Cela fonctionne avec les méthodes suivantes :

context.save()

context.restore()

La méthode ***context.save()*** permet de sauvegarder l'état actuel du contexte, la méthode ***context.restore()*** permet de restituer l'état du dernier contexte sauvegardé, c'est-à-dire que les données de transformations de la matrice ainsi que les données de dessins etc. seront exactement les mêmes que lors du dernier appel à context.save().

Ces méthodes fonctionnent un peu à la manière d'une pile d'assiettes, c'est-à-dire que le dernier contexte sauvegardé ira « au-dessus » de la pile et donc, lors du prochain appel à context.restore() ce sera cette dernière « assiette » qui sera restituée.

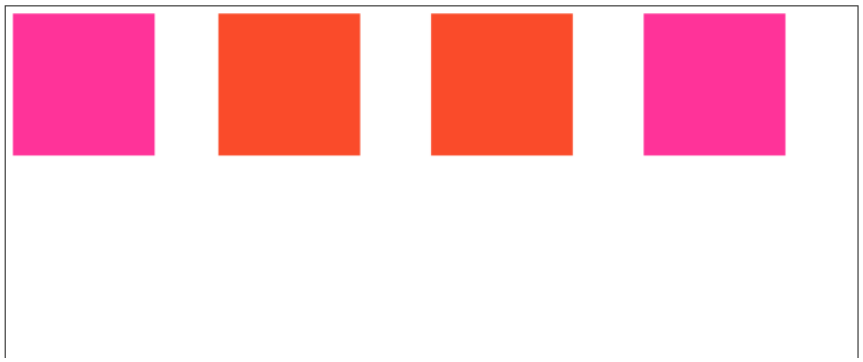
A faire : Exemple :

```
<!DOCTYPE HTML>
<html>
```

```
<head>

<style>
        #test {
          width: 100px;
          height:100px;
          margin: 0px auto;
        }
</style>
</head>
<body>
<canvas id="mycanvas">Votre navigateur ne supporte ps les cnvas HTML5</canvas>
<script type="text/javascript">
          var canvas = document.getElementById('mycanvas');
           var ctx = canvas.getContext('2d');
            ctx.fillStyle = 'red';
           ctx.fillRect(0,0,150,150);
                      //  Save the  state

                        // Make changes to the settings
            ctx.fillStyle = 'blue';
            ctx.fillRect( 15,15,120,120);

                        // Save the current state
                        // Make the new changes to the settings
            ctx.fillStyle = 'yellow';
            ctx.fillRect(30,30,90,90);
                        // Restore previous state
            ctx.restore();
                        // Draw a rectangle with restored settings
            ctx.fillRect(45,45,60,60);
            // Restore original state
                        // Draw a rectangle with restored settings
             ctx.fillRect(50,50,45,45);
</script>
</body>

</html>
```

Exercice : A faire :

Créer l'application ci-dessous en utilisant save et restore.



Ajouter une translation de 100 par rapport à Y pour obtenir

Activité 7 :ombrage, composition, masque

Ombrage

Des effets d'ombrage sont prévus par Canvas, ce qui est une bonne nouvelle en soi, car les calculer pixel par pixel n'est pas une mince affaire. Quatre propriétés contrôlent le rendu de l'ombrage à partir de sa source.

Propriété	Rôle	Valeurs
shadowOffsetX	Étendue de l'ombrage sur l'axe horizontal (X)	Nombre entier, positif, négatif ou nul
shadowOffsetY	Étendue de l'ombrage sur l'axe vertical (Y)	Nombre entier, positif, négatif ou nul
shadowBlur	Valeur du flou	Nombre entier positif ou nul
shadowColor	Couleur de l'ombrage	Code couleur ou RGBA

A faire : Exemple

```
<!DOCTYPE HTML>
<html>
   <head>
      <style>
         canvas {
           border:1px solid #2C2C2B; /* 1px border around canvas */
         }
```

```
        </style>
    </head>
    <body>
        <canvas id="Canvas" width="600" height="250"></canvas>

        <script>
         var canvas = document.getElementById('Canvas');
        var ctx = canvas.getContext('2d');
        // Configuration des ombrages pour le rectangle
        //Etendue sur l'axe X = 0; Etendue sur Y ;Valeur du flou = 15; couleur ombre =
        //'rgba(204,69,228,1)';
        // Essayez par la suite offsetX et offsetY 10 et valueur du flou 30
        // Tracé d'un rectangle
        ctx.fillStyle = '#fff';
        ctx.fillRect(10,10,150,150);
        // Tracé d'un autre rectangle
        ctx.fillStyle = '#9f0c9d';
        ctx.fillRect(50,50,70,70);
        </script>

    </body>
</html>
```

Activité 8 : Composition

La propriété ***globalCompositeOperation*** régit la façon dont sont menées les opérations

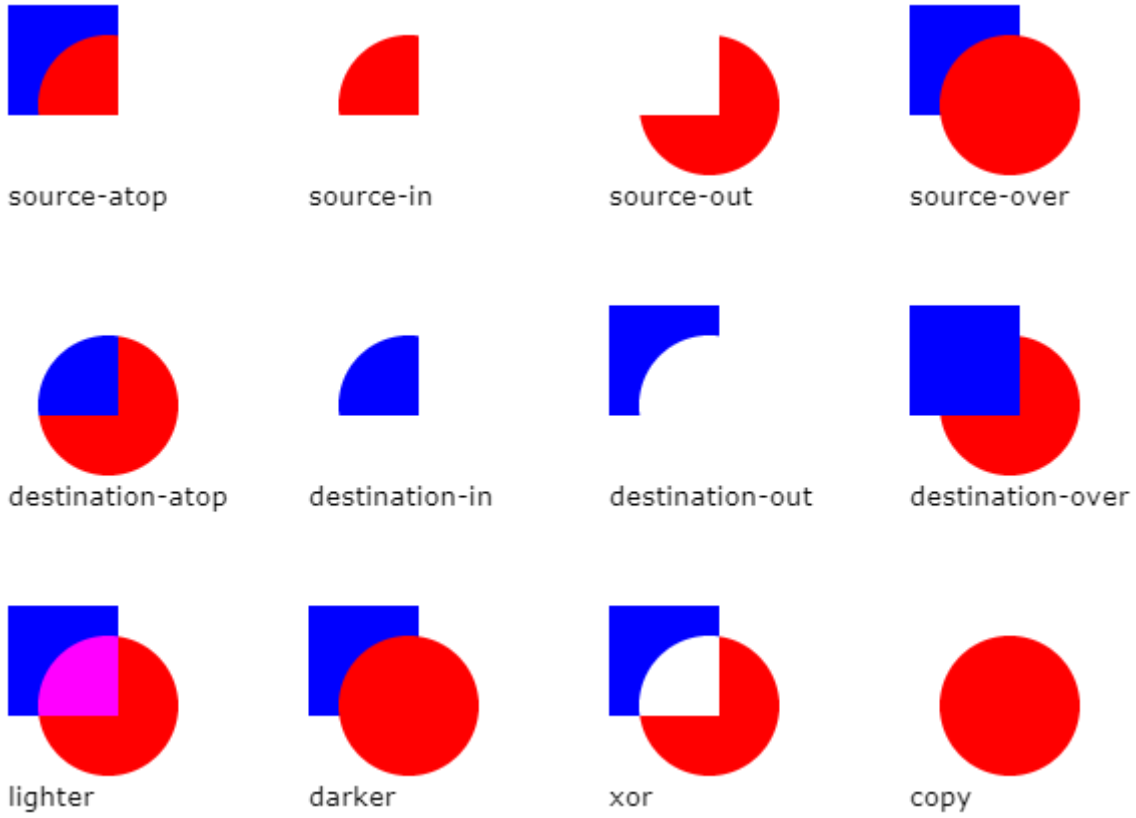
de composition sur le canvas (définit ou renvoie comment une nouvelle image est dessinée sur une image existante), pour les tracés, les formes et les images, en plus de la transparence.

Par défaut, la valeur ***source-over*** produit un effet de superposition, c'est-à-dire que

la dernière forme tracée recouvre de façon opaque (hors modification de ***globalAlpha***) le contenu déjà présent. C'est l'effet que l'on retrouve de façon logique dans la plupart des programmes de dessin.

Cette propriété peut prendre plusieurs valeurs dont les résultats sont illustrés à l'aide du graphique ci-dessous (source:**http://www.html5canvastutorials.com/advanced/html5-canvas-global-composite-operations-tutorial/**) :

https://www.w3schools.com/tags/playcanvas.asp?filename=playcanvas_globalcompop&preval=source-over



A faire : Exemple composition

```
<!DOCTYPE html>
<html>
    <body>
        <canvas id="myCanvas" width="300" height="150" style="border:1px solid
        #d3d3d3;">
        Your browser does not support the HTML5 canvas tag.</canvas>

        <script>
          var c = document.getElementById("myCanvas");
          var ctx = c.getContext("2d");
          ctx.fillStyle = "red";
          ctx.fillRect(20, 20, 75, 50);
          ctx.fillStyle = "blue";
          // globalCompositeOperation vaut  "source-over";
          ctx.fillRect(50, 50, 75, 50);
          ctx.fillStyle = "red";
          ctx.fillRect(150, 20, 75, 50);
          ctx.fillStyle = "blue";
          // globalCompositeOperation vaut  "destination-over";
          ctx.fillRect(180, 50, 75, 50);
```

```
        </script>
    </body>
</html>
```

Activité 9 : Masque

Un masque dissimule une région de la zone d'affichage, tout en révélant l'autre région.

La méthode utilisée est : ***clip()*** (Utilise le chemin courant comme masque.)

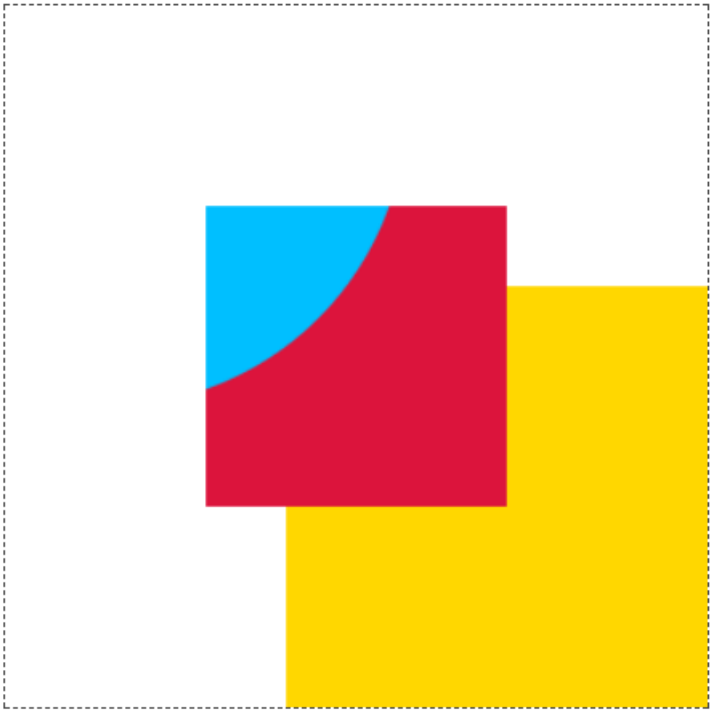
Remarque : la propriété ***globalCompositeOperation*** de l'objet context permet également de créer des masques

A faire : **Exemple :**

```
<!DOCTYPE html>
<html>
    <body>
        <canvas id="dessin" width="333" height="500"></canvas>
        <script>
        var moncanvas = document.getElementById('dessin');
        var ctx = moncanvas.getContext('2d');
        var img = new Image();
        img.src = 'photo.jpg';
        // Cercle de coordonnées 100, 100, 150, 0, Math.PI * 2, false
        // Masque sur le chemin courant
        // Cercle de coordonnées 200,200, 150, 0, Math.PI * 2, false
        // Masque sur le chemin courant

        img.onload = function() {
        // Dessin de l'image
        …..}
    </script>
</body>
</html>
```

A faire Exercice :

Créer le canvas ci-dessous :



.

A faire : Exercice :

Créer un masque circulaire sur une image (utiliser l'image fleurs.jpg)

Activité 10 : Contrôle clavier et souris

En tant qu'élément HTML à part entière, le canvas bénéficie de tous les événements DOM (clavier, souris, navigateur) pouvant survenir et de toutes les fonctions JavaScript afférentes.3

Souris

Le contrôle à la souris est disponible avec les événements ***click***, ***dblclick***,***mousedown***, ***mouseup***, ***mousemove***, ***mouseenter***et ***mouseleave***. Déclarer des gestionnairespour chacun d'entre eux est un premier pas, compléter la fonction en est unautre. Celle-ci regroupe les instructions nécessaires à l'ajout de tracés, d'images, deformes ou au rafraîchissement global de la zone de dessin.

A faire : exemple :

Le code source suivant crée une ardoise dans le navigateur, équipée d'unepalette pour les changements de couleur.

Question 1 : Tester le code et le rectifier afin de permettre de tracer des formes pleines.

Question 2 : Fournir une autre version de l'exemple en rajoutant deux boutons radio : Forme creuse et Forme pleine. Lorsque l'utilisateur sélectionne Forme creuse, on dessine uniquement le contour (stroke) sinon on dessine une forme pleine (fill)

```
<!doctype html>
<html lang="fr">
    <head>
    <title>HTML5 : Canvas</title>
    <meta charset="utf-8">
    <style>
    body {
    background:#eee;
    text-align:center;
    padding-top:10%;
    }
    .palette span {
    display:inline-block;
    width:40px;
    height:40px;
    cursor:pointer;
    border:2px solid transparent;
    border-radius:4px;
    }
    .palette span:hover {
    border-color:white;
    }
    canvas {
    cursor:crosshair;
    border:5px solid #666;
    background:white;
    border-radius:4px;
    box-shadow:0px 0px 20px #666;
    margin-top:20px;
    }
    </style>
    </head>
    <body>
        <!-- Palette de couleurs _ -->
        <div class="palette">
        <span onclick="modifierCouleur('#206BA4');" style="background:#206BA4"></span>
        <span onclick="modifierCouleur('#54A4DE');" style="background:#54A4DE"></span>
        <span                                          onclick="modifierCouleur('#BBD9EE');"
        style="background:#BBD9EE"></span>
```

```
        <span                                                   onclick="modifierCouleur('#BEDF5D');"
        style="background:#BEDF5D"></span>
        <span                                                   onclick="modifierCouleur('#D6EB9A');"
        style="background:#D6EB9A"></span>
        <span onclick="modifierCouleur('#FF9834');" style="background:#FF9834"></span>
        <span onclick="modifierCouleur('#FFBF80');" style="background:#FFBF80"></span>
        <span onclick="modifierCouleur('#F6E896');" style="background:#F6E896"></span>
        <span onclick="modifierCouleur('#b07d42');" style="background:#b07d42"></span>
        <span onclick="modifierCouleur('#FF5349');" style="background:#FF5349"></span>
        </div>

        <!-- Canvas -->
        <canvas id="dessin" width="480" height="360"></canvas>
        <script>
                var moncanvas = document.getElementById('dessin');
                var ctx = moncanvas.getContext('2d');
                var en_dessin = false;
                // Propriétés graphiques par défaut _
                ctx.strokeStyle = "black";
                ctx.lineWidth = 2;
                // Bouton de souris activé _
                moncanvas.onmousedown = function(e) {
                // Dessin activé
                en_dessin = true;
                ctx.beginPath();
                // Repositionnement du début du tracé
                ctx.moveTo(e.offsetX,e.offsetY);
                };
                // Mouvement de souris _
                moncanvas.onmousemove = function(e) {
                if(en_dessin) dessiner(e.offsetX,e.offsetY);
                };
                // Bouton de souris relâché _
                moncanvas.onmouseup = function(e) {
                // Dessin désactivé
                en_dessin = false;
                };
                // Ajoute un segment au tracé _
                function dessiner(x,y) {
                ctx.lineTo(x,y);
                ctx.stroke();
                }
                // Modification de la couleur du contexte _
                function modifierCouleur(codeCouleur) {
                if(codeCouleur) ctx.strokeStyle = codeCouleur;
                }
        </script>
    </body>
</html>
```

Gestion du Clavier

La gestion du clavier est analogue, avec l'interception de l'événement ***keypress***, ***keyup*** ou ***keydown***. La valeur de la touche enfoncée correspond à un code numérique stocké dans la propriété event.***keyCode***.

A faire : Exemple

L'ardoise de dessin peut être contrôlée uniquement au clavier pour la transformer en un Télécran.

Tester le code suivant le rectifier afin de render paramétrable la couleur du trait et le pas de déplacement. Prévoir deux input pour faire la saisie de ces deux valeurs.

```
<!doctype html>
<html lang="fr">
<head>
<title>HTML5 : Canvas</title>
<meta charset="utf-8">
<style>
body {
background:#eee;
text-align:center;
padding-top:10%;
}
canvas {
border:5px solid #666;
background:white;
border-radius:4px;
box-shadow:0px 0px 20px #666;
}
</style>
</head>
    <body>
        <canvas id="dessin" width="480" height="360"></canvas>
        <script>
                var moncanvas = document.getElementById('dessin');
                var ctx = moncanvas.getContext('2d');
                ctx.strokeStyle = "black";
                ctx.lineWidth = 5;
                ctx.lineCap = "round";
                // Calcul du centre
                var x = moncanvas.width/2;
                var y = moncanvas.height/2;
                // Position de départ (au centre)
```

```
            ctx.moveTo(x,y);
            // Gestionnaire d'événement keydown
            if(document.body.onkeydown) {
            document.body.onkeydown = dessiner;
            } else if(document) {
            document.onkeydown = dessiner;
            }
            // Déplacement du "pinceau"
            function dessiner(event) {
            switch(event.keyCode) {
            case 38: // Haut
            event.preventDefault();
            if(y>=5) y-=5;
            break;
            case 40: // Bas
            event.preventDefault();
            if(y<moncanvas.height) y+=5;
            break;
            case 39: // Droite
            event.preventDefault();
            if(x<moncanvas.width) x+=5;
            break;
            case 37: // Gauche
            event.preventDefault();
            if(x>=5) x-=5;
            break;
            }
            // Tracé d’après le décalage effectué
            ctx.lineTo(x,y);
            ctx.stroke();
            };
        </script>
    </body>
</html>
```