視覚表現のためのプログラミング

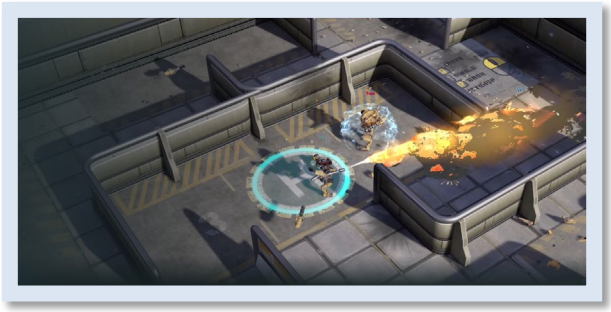
HTML5 描画　canvas

今回は、HTML5 の新しい描画機能 canvas の基本的な使い方を学びます。

HTML5 から実装された描画タグ**<canvas>**

HTML5 の 3DCG 機能を使えば、ブラウザ単体で動作する 2D・3D ゲームの作成が可能です。

現在、多くのゲームが HTML 上で開発されています



上の画像をクリックしてください
YouTube が立ち上がり、デモがスタートします

◆HTML5 で 2D・3D を描く利点

・標準的なブラウザだけでいい・・・共通化、デバイス・OS 不問

・アプリケーションやプラグインなど不要

・HTML または JavaScript のソースだけなのでダウンロード時間が短い

■2D・3D 制作

・ライブラリ系

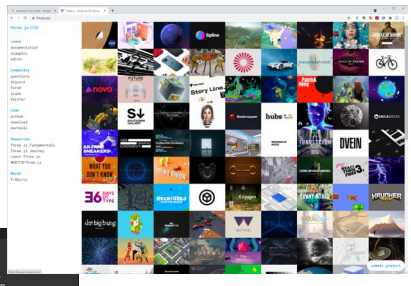
jQuerry

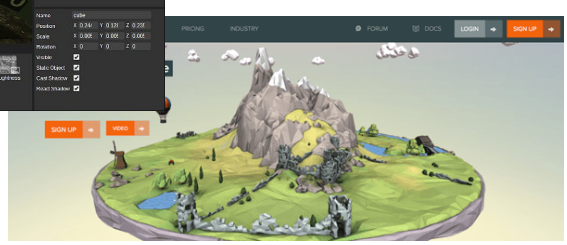
three.js

・ツール系

Unity

nunustudio





■canvas のための準備

canvas を使うための html の枠を準備します。

```
<!doctype html>
<html lang="ja">
<head>
<meta charset="UTF-8" />
<title>canvas study</title>
</head>

<canvas id="myCanvas" width="300" height="300"> </canvas>

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

◆canvas id をセット

次に、<canvas>タグに canvas id をセットします。

canvas id ="myCanvas"

id 名の myCanvas の名前は何でもいいです。多くのソースではこの myCanvas を使っています。

そして、canvas を描く範囲（縦 height、横 width）を指定します。絵を描くためには画用紙が必要なのと同じです。

ここでは、横：300px、縦：300px の大きさの場所を用意しています。

描画をするソース本体は<script>内に書きます。

■HTML canvas と JavaScript の橋渡し

<canvas>は HTML タグで、実際の描画のための指示は JavaScript 内に記述されます。

そこで、JavaScript 内に記述された指示を HTML 側に渡す必要があります。

```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
```

最初に、canvas id と script 内の Id 名とを関連図づける（一致させる）必要があります。

```
canvas = document.getElementById("myCanvas");
```

次に、実際に描く 2D の指定です。
本来は変数名「context」なのですが、長いのでよく省略されて「ctx」になっています。
この 2 行は canvas で 2D を描く場合には必ず必要ですので、空の html 入るに一緒に入れておくと良いでしょう。

■四角形を描く

```
<html>
<head></head>
<canvas id="canvas" width="640"
height="480"></canvas>
<script>
  var canvas = document.getElementById("canvas");
  var ctx = canvas.getContext("2d");
//四角形
  ctx.beginPath();
  ctx.rect(20, 40, 50, 50);
  ctx.fillStyle = "#FF0000";
  ctx.fill();
  ctx.closePath();
</script>
<body></body>
</html>
```

描画の指示はすべて　ctx.○○　になっています。自分で名前を付けた ctx に命令語が紐付いています。

この描画は、四角形を「線画」（Path）として捉えています。
したがって、Path のはじまりから Path の終わりまでを明確にします。

```
ctx.beginPath();
  ・・・・・
ctx.closePath();
```

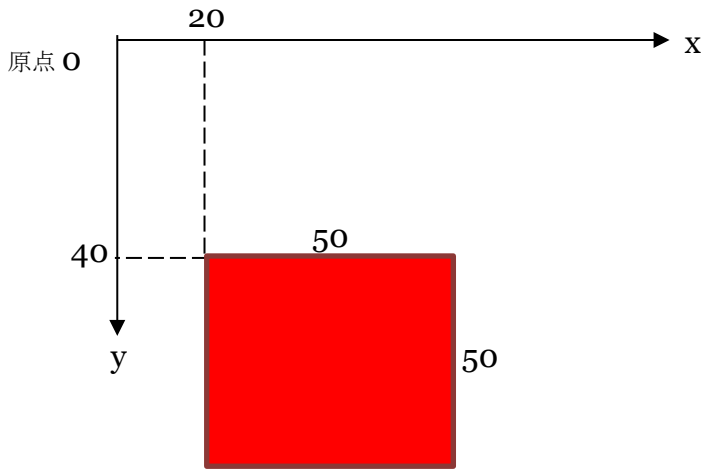
この中に描画が内容を書きます。

ctx.rect(20, 40, 50, 50);は、rect つまり rectangle（矩形、長方形）です。

最初の数字 20,40 は、書き出す**左上の頂点の座標**が x=20、y=40 の座標からはじまることを示しています。

プログラムの場合、ディスプレイの描画に合わせて左上が原点になります。そこから右方向に x 軸。下方向に y 軸がとられます。数学とは y 軸の方向が逆です。

プログラムの（20,40）の点が矩形の左上の点になります。



つぎの 2 つの数字 50,50 は、この点から幅 50px（x 軸方向）,高さ 50px（y 軸方向）に矩形を描くことを示しています。

合わせて書けば、ctx.rect(**左上の座標 x,y、矩形の幅と高さ**)となります。

次の 2 行は「**塗りつぶし**」の命令です。

```
ctx.fillStyle = "#FF0000";
ctx.fill();
```

Style の方で**色**などを指定します。ここでは RGB の指定で「R」が ff なので、「赤」です。

そして、この「**塗りつぶし**」(fill) に対しての実行命令を書きます。もし、Style 指定がない場合には「黒」で塗りつぶされます。

また、もし fill の命令がない場合には、何も描かれません。それは、「矩形を書きなさい」との命令があっても、どのように描くかの指示がないためです。

■線の描画

それでは、塗りつぶしではなく枠だけの四角形を描くときにはどうすればいいのでしょうか。
その場合は、「線」で描きます。
ではまず、線を描く基本を見ていきましょう。

線も矩形と同じく beginPath( )からはじめます。

```
ctx.beginPath();
```

次に始点と次の点を指示します。

始点：　ctx.moveTo(50,50);
次の点：　ctx.lineTo(250,250);

これで、始点（50,50）から（250,250）までの線を意味します。
実際には、それを実行する　ctx.stroke();　で実行されます。

```
ctx.beginPath();
ctx.moveTo(50,50);
ctx.lineTo(250,250);
ctx.stroke();
```

この lineTo を繰り返していけば、次々と線が描かれていきます。
では実際に、lineTo を繰り返して三角形を描いてみます。

■三角形の描画

```
ctx.beginPath();
ctx.moveTo(150,50);
ctx.lineTo(250,200);
ctx.lineTo(50,200);
ctx.closePath();
ctx.stroke();
```

描くためには、事前に座標を計算しておく必要があります。

この「線」に色を付けるためには strokeStyle を使い、色を RGB で指定します。

```
ctx.strokeStyle = "#1874CD";
```

この指定は、stroke の前に入れます。

また、線の幅は　ctx.lineWidth = 7;　で、「px 単位」の指定ができます。

■円の描画

円は「円弧」arc を応用して描きます。

```
ctx.beginPath();
ctx.arc(150,150,100,0, Math.PI*2 ,true);
ctx.closePath();
ctx.strokeStyle = "#1874CD";
ctx.lineWidth = 3;
ctx.fillStyle = "#C6E2FF";
ctx.fill();
ctx.stroke();
```

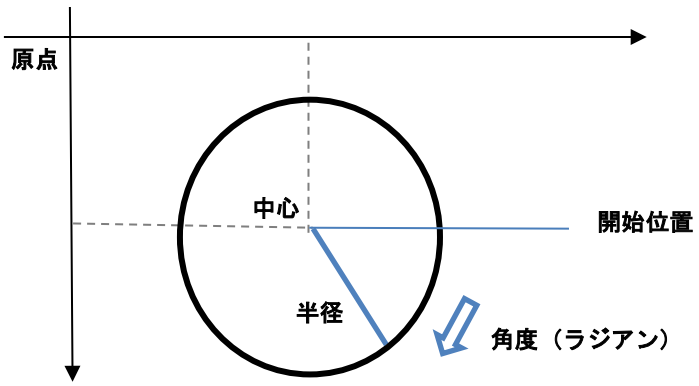
arc の括弧内は

arc（円の中心座標 x,y、円の半径、開始角度、終了角度、描く方向）;

開始角度と終了の角度はラジアンで表記します。

方向は、true を指定すると「反時計回り」になる。（このパラメータは省略可能）

いずれの図形も、事前に、各点の座標を紙の上で計画しておく必要があります。



■ ベジェと二次曲線

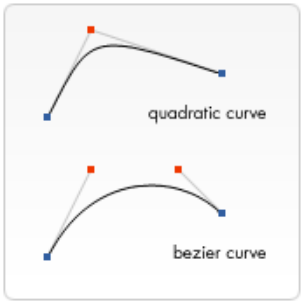
quadraticCurveTo(cp1x, cp1y, x, y)

現在のペンの位置から x および y で指定した終端へ、 cp1x および cp1y で指定した制御点を使用して二次ベジェ曲線を描きます。

bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)

現在のペンの位置から x および y で指定した終端へ、(cp1x, cp1y) および (cp2x, cp2y) で指定した制御点を使用して三次ベジェ曲線を描きます。

これらの違いは右の画像を使うことで説明することができます。二次ベジェ曲線は始点と終点 (青い点) と 1 つの制御点 (赤い点で示したもの) を持つのに対して、三次ベジェ曲線は 2 つの制御点を持ちます。



それらのメソッドの両方の x と y パラメータは終点の座標です。cp1x と cp1y は最初の制御点、cp2x と cp2y は 2 番目の制御点の座標です。

二次ベジェ曲線

この例では、吹き出しをレンダリングするために複数の二次ベジェ曲線を使用しています。

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

    // 二次曲線の例
    ctx.beginPath();
    ctx.moveTo(75, 25);
    ctx.quadraticCurveTo(25, 25, 25, 62.5);
```

```
        ctx.quadraticCurveTo(25, 100, 50, 100);
        ctx.quadraticCurveTo(50, 120, 30, 125);
        ctx.quadraticCurveTo(60, 120, 65, 100);
        ctx.quadraticCurveTo(125, 100, 125, 62.5);
        ctx.quadraticCurveTo(125, 25, 75, 25);
        ctx.stroke();
    }
}
```