

# Canvas Second Step

小林 統 \*

2022 年 4 月 27 日

## 概要

HTML5 の Canvas の機能を利用して、物理的な表現に取り組んでみましょう。

## 目次

1	はじめに	2
1.1	読み間違えないでね . . . . .	2
1.2	注意 . . . . .	2
1.3	コンピュータの 2D の座標について . . . . .	2
2	Canvas 応用	3
2.1	物理的な動き . . . . .	3
2.1.1	04-101.html 円を描こう . . . . .	3
2.1.2	04-102.html 一定時間ごとに円を描くけど... 動かない . . . . .	4
2.1.3	04-103.html 等速度運動 . . . . .	5
2.1.4	04-104.html 摩擦のシミュレーション . . . . .	7
2.2	Particle というクラスを用いた生成方法 . . . . .	8
2.2.1	04-201.html オブジェクトを利用した等加速度運動 . . . . .	9
2.2.2	04-202.html 配列に Particle を入れてアニメーション . . . . .	10
2.2.3	04-203.html 色を変更 . . . . .	12
2.2.4	04-204.html 大きさを変更 . . . . .	14
2.3	より高度な表現 . . . . .	17
2.3.1	04-301.html 一定の方向に等加速度運動 . . . . .	17
2.3.2	04-302.html 重力のシミュレーション . . . . .	19
2.3.3	04-303.html 床の跳ね返りをシミュレーション . . . . .	21
2.3.4	04-304.html とある点からの吹き出し . . . . .	24
2.3.5	04-305.html 線の追加 . . . . .	26
2.3.6	04-306.html 三角形の追加 . . . . .	30

---

\* 帝京平成大学人文社会学部人間文化学科メディア文化コース

# 1 はじめに

## 1.1 読み間違えないでね

ソースコード 1 読み間違えないでね

---

```
1 数字: 0123456789
2 小文字:abcdefghijklmnopqrstuvwxyz
3 大文字:ABCDEFGHIJKLMNOPQRSTUVWXYZ
4
5 1:イチ
6 l:小文字のエル
7 i:小文字のアイ
8 |:ビックリマーク
9 |:バーティカルバー。Shift と ¥ を押したものの。
10
11 0:ゼロ
12 o:小文字のオー
13 O:大文字のオー
14
15 .:ピリオド
16 ,:コンマ
```

---

## 1.2 注意

- これから出てくるソースコードには、左に「行番号」と呼ばれる番号が出てくるけど、入力する必要ないからね。
- script タグの中で「//」で始まる文は、コメントで、プログラムは読み飛ばすよ。
- コピーできるところはコピーして効率よく入力して行こう
- 徐々に追加されていくから、量が多く見えるけど、平気だよ！
- 改行されていても、行番号が書かれていないところは、1 行だからね。表示上改行されて見えてるだけ

## 1.3 コンピュータの 2D の座標について

数学では、右に  $x$ , 上に  $y$  だったけど、  
コンピュータでは、左上が原点、右に  $x$ , 下に  $y$  と考えるので、気をつけよう。

## 2 Canvas 応用

入門を受けて、いろんなアニメーション作って行こう

### 2.1 物理的な動き

#### 2.1.1 04-101.html 円を描こう

CanvasFirstStep のおさらい 1。これ間違えると動かないよ。先週のコピッてきても許す。

##### ソースコード 2 円を描こう

---

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Canvas Test</title>
5         <script type="text/javascript">
6             //canvas の中身を指し示すもの
7             let ctx;
8             //canvas の width 幅
9             let cw;
10            //canvas の height 高さ
11            let ch;
12
13            //初期化处理
14            function init(){
15                //ID が mycanvas のものを canvas という変数にいれておく
16                let canvas = document.getElementById('mycanvas');
17                //canvas が存在しないか、canvas の中身がなければ処理終了
18                if(!canvas || !canvas.getContext){
19                    return false;
20                }
21
22                //canvas の中身を指し示すものを代入
23                ctx = canvas.getContext('2d');
24                //canvas の幅・高さを代入しておく
25                cw = canvas.width;
26                ch = canvas.height;
27
28                //実際の描画処理
29                draw();
30            }
31
32            function draw(){
33                //円の描画
34                ctx.beginPath();
35                ctx.arc(100, 100, 20, 0, 2*Math.PI, false);
36                ctx.fillStyle = '#ff0000';
37                ctx.fill();
```

```

38         }
39     </script>
40 </head>
41 <body onload="init();">
42     <H1>Canvas Test</H1>
43     <canvas id="mycanvas" width="1000" height="800"></canvas>
44 </body>
45 </html>

```

---

### 2.1.2 04-102.html 一定時間ごとに円を描くけど... 動かない

CanvasFirstStep のおさらい 2。円を描くのに 4 行使うから、それを drawCircle という関数にまとめるよ。

41 行目、44-50 行目あたり。

一定時間で描写してるけど、同じところに同じもの書いてるだけだから動作は変わらないよ。

32 行目あたりが増えてるね。

ソースコード 3 一定時間ごとに円を描くけど... 動かない

---

```

1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Canvas Test</title>
5         <script type="text/javascript">
6             //canvas の中身を指し示すもの
7             let ctx;
8             //canvas の width 幅
9             let cw;
10            //canvas の height 高さ
11            let ch;
12
13            //初期化处理
14            function init(){
15                //ID が mycanvas のものを canvas という変数にいれておく
16                let canvas = document.getElementById('mycanvas');
17                //canvas が存在しないか、canvas の中身がなければ処理終了
18                if(!canvas || !canvas.getContext){
19                    return false;
20                }
21
22                //canvas の中身を指し示すものを代入
23                ctx = canvas.getContext('2d');
24                //canvas の幅・高さを代入しておく
25                cw = canvas.width;
26                ch = canvas.height;
27
28                //実際の描画処理
29                draw();
30

```

```

31          //一定時間ごとに書き換える 30fps にするために 1000msec/30
           = 33msec
32          setInterval("draw()",33);
33      }
34
35      function draw(){
36          //画面をリセットする
37          ctx.fillStyle = "rgba(255,255,255,1)";
38          ctx.fillRect(0,0,cw,ch);
39
40          //円の描画
41          drawCircle(100, 100, 20, '#FF0000');
42      }
43
44      function drawCircle(x,y,scale,color){
45          //円の描画
46          ctx.beginPath();
47          ctx.arc(x, y, scale, 0, 2*Math.PI, false);
48          ctx.fillStyle = color;
49          ctx.fill();
50      }
51      </script>
52  </head>
53  <body onload="init();">
54      <H1>Canvas Test</H1>
55      <canvas id="mycanvas" width="1000" height="800"></canvas>
56  </body>
57 </html>

```

---

### 2.1.3 04-103.html 等速度運動

一定のスピードで動くのを等速度運動って言ったよね????

14,15 行目あたり 45,46 行目あたりかな。

---

#### ソースコード 4 等速度運動

---

```

1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Canvas Test</title>
5         <script type="text/javascript">
6             //canvas の中身を指し示すもの
7             let ctx;
8             //canvas の width 幅
9             let cw;
10            //canvas の height 高さ
11            let ch;
12
13            //移動速度

```

```

14         const speed = 2;
15         let x = 0;
16
17         //初期化处理
18         function init(){
19             //ID が mycanvas のものを canvas という変数にいておく
20             let canvas = document.getElementById('mycanvas');
21             //canvas が存在しないか、canvas の中身がなければ処理終了
22             if(!canvas || !canvas.getContext){
23                 return false;
24             }
25
26             //canvas の中身を指し示すものを代入
27             ctx = canvas.getContext('2d');
28             //canvas の幅・高さを代入しておく
29             cw = canvas.width;
30             ch = canvas.height;
31
32             //実際の描画処理
33             draw();
34
35             //一定時間ごとに書き換える
36             setInterval("draw()",33);
37         }
38
39         function draw(){
40             //画面をリセットする
41             ctx.fillStyle = "rgba(255,255,255,1)";
42             ctx.fillRect(0,0,cw,ch);
43
44             //円の描画
45             x += speed;
46             drawCircle(x, 100, 20, '#FF0000');
47         }
48
49         function drawCircle(x,y,scale,color){
50             //円の描画
51             ctx.beginPath();
52             ctx.arc(x, y, scale, 0, 2*Math.PI, false);
53             ctx.fillStyle = color;
54             ctx.fill();
55         }
56     </script>
57 </head>
58 <body onload="init();">
59     <H1>Canvas Test</H1>
60     <canvas id="mycanvas" width="1000" height="800"></canvas>
61 </body>

```

#### 2.1.4 04-104.html 摩擦のシミュレーション

目的地を設定して、そこに向かって動いて止まる動きをするよ。

16,46 行目あたり

##### ソースコード 5 摩擦のシミュレーション

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Canvas Test</title>
5         <script type="text/javascript">
6             //canvas の中身を指し示すもの
7             let ctx;
8             //canvas の width 幅
9             let cw;
10            //canvas の height 高さ
11            let ch;
12
13            //移動速度
14            const speed = 20;
15            let x = 0;
16            const target_x = 400;
17
18            //初期化处理
19            function init(){
20                //ID が mycanvas のものを canvas という変数にいれておく
21                let canvas = document.getElementById('mycanvas');
22                //canvas が存在しないか、canvas の中身がなければ処理終了
23                if(!canvas || !canvas.getContext){
24                    return false;
25                }
26
27                //canvas の中身を指し示すものを代入
28                ctx = canvas.getContext('2d');
29                //canvas の幅・高さを代入しておく
30                cw = canvas.width;
31                ch = canvas.height;
32
33                //実際の描画処理
34                draw();
35
36                //一定時間ごとに書き換える
37                setInterval("draw()",33);
38            }
39
40            function draw(){
```

```

41          //画面をリセットする
42          ctx.fillStyle = "rgba(255,255,255,1)";
43          ctx.fillRect(0,0,cw,ch);
44
45          //円の描画
46          x += (target_x - x) / speed;
47          drawCircle(x, 100, 20, '#FF0000');
48      }
49
50      function drawCircle(x,y,scale,color){
51          //円の描画
52          ctx.beginPath();
53          ctx.arc(x, y, scale, 0, 2*Math.PI, false);
54          ctx.fillStyle = color;
55          ctx.fill();
56      }
57      </script>
58  </head>
59  <body onload="init();">
60      <H1>Canvas Test</H1>
61      <canvas id="mycanvas" width="1000" height="800"></canvas>
62  </body>
63 </html>

```

---

## 2.2 Particle というクラスを用いた生成方法

これまでは、Canvas に対して円を書いて！と指示してきたよね。

オブジェクト指向という考え方があって、例えば、車を例にしてみよう。

車に「走れ」「止まれ」「右に曲がって」「左に曲がって」と指示をアクセル・ブレーキ・ハンドルを使って指示をすると、勝手にそう動くよね。別に、エンジンの仕組みを知らなくても人は操作できるわけだ。

この様に、機能を持ったモノを定義して、そこに指示を出していくプログラミング方法を「オブジェクト指向」っていうよ。

また、車というクラスがあったときに、A さんの車、B さんの車、という風に色々な車があるよね。オブジェクトは抽象化された概念なのに対して、A さんの車とかは実体化されたもの(インスタンス)と捉えるよ。

車という概念を定義しておいて、必要に応じてインスタンスを適宜生成することによって、いろいろなことができるようになるよ。

今回は、Particle(粒子) クラスを作成して、そのインスタンスをたくさん作ることで面白い表現をしてみよう。

35-52 あたり追加。それを受けて、54 あたりが追加

37 行目から、Particle の初期化の話

46 行目からは、Particle に draw()... 描け、と言った時の挙動が定義されています。

54 行目で、Particle というクラスから実態としての particle を生成しています。



### 2.2.1 04-201.html オブジェクトを利用した等加速度運動

ソースコード 6 オブジェクトを利用した等加速度運動

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Canvas Test</title>
5         <script type="text/javascript">
6             //canvas の中身を指し示すもの
7             let ctx;
8             //canvas の width 幅
9             let cw;
10            //canvas の height 高さ
11            let ch;
12
13            //初期化处理
14            function init(){
15                //ID が mycanvas のものを canvas という変数にいれておく
16                let canvas = document.getElementById('mycanvas');
17                //canvas が存在しないか、canvas の中身がなければ処理終了
18                if(!canvas || !canvas.getContext){
19                    return false;
20                }
21
22                //canvas の中身を指し示すものを代入
23                ctx = canvas.getContext('2d');
24                //canvas の幅・高さを代入しておく
25                cw = canvas.width;
26                ch = canvas.height;
27
28                //実際の描画処理
29                draw();
30
31                //一定時間ごとに書き換える
32                setInterval("draw()",33);
33            }
34
35            //Particle クラス
36            class Particle {
37                constructor(scale, color,speed){
38                    this.scale = scale;
39                    this.color = color;
40                    this.speed = speed;
41                    this.position = {
42                        x: 100,
43                        y: 100
44                    };
45                }
46            }
```

```

45         }
46         draw(){
47             ctx.beginPath();
48             ctx.arc(this.position.x, this.position.y, this
                    .scale, 0, 2*Math.PI, false);
49             ctx.fillStyle = this.color;
50             ctx.fill();
51         }
52     }
53     //パーティクルの準備
54     let particle = new Particle(20, "#ff0000", 2);
55
56     function draw(){
57         //画面をリセットする
58         ctx.fillStyle = "rgba(255,255,255,1)";
59         ctx.fillRect(0,0,cw,ch);
60
61         //円の描画
62         particle.position.x += particle.speed;
63         particle.draw();
64     }
65
66     </script>
67 </head>
68 <body onload="init();">
69     <H1>Canvas Test</H1>
70     <canvas id="mycanvas" width="1000" height="800"></canvas>
71 </body>
72 </html>

```

---

### 2.2.2 04-202.html 配列に Particle を入れてアニメーション

一つだけじゃ、メリットわかりませんね。複数 particle を生成してみましょう。

15 行目、33 行目、72 行目あたり

---

#### ソースコード 7 配列に Particle を入れてアニメーション

---

```

1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Canvas Test</title>
5         <script type="text/javascript">
6             //canvas の中身を指し示すもの
7             let ctx;
8             //canvas の width 幅
9             let cw;
10            //canvas の height 高さ
11            let ch;
12

```

```

13      //パーティクルの準備
14      const density = 100; //パーティクルの密度
15      let particles = [];
16
17      //初期化処理
18      function init(){
19          //ID が mycanvas のものを canvas という変数にいれておく
20          let canvas = document.getElementById('mycanvas');
21          //canvas が存在しないか、canvas の中身がなければ処理終了
22          if(!canvas || !canvas.getContext){
23              return false;
24          }
25
26          //canvas の中身を指し示すものを代入
27          ctx = canvas.getContext('2d');
28          //canvas の幅・高さを代入しておく
29          cw = canvas.width;
30          ch = canvas.height;
31
32          //円の初期化
33          for(let i=0;i < density; i++){
34              particles[i] = new Particle(6, "#FF0000",
35                  Math.random()*(4-2)+2);
36              particles[i].position.x = Math.random()*ch;
37              particles[i].position.y = Math.random()*cw;
38              particles[i].draw();
39          }
40
41          //実際の描画処理
42          draw();
43
44          //一定時間ごとに書き換える
45          setInterval("draw()",33);
46      }
47
48      //Particle クラス
49      class Particle {
50          constructor(scale, color,speed){
51              this.scale = scale;
52              this.color = color;
53              this.speed = speed;
54              this.position = {
55                  x: 100,
56                  y: 100
57              };
58          }
59          draw(){
60              ctx.beginPath();

```

```

60             ctx.arc(this.position.x, this.position.y, this
61                     .scale, 0, 2*Math.PI, false);
62             ctx.fillStyle = this.color;
63             ctx.fill();
64         }
65     }
66     function draw(){
67         //画面をリセットする
68         ctx.fillStyle = "rgba(255,255,255,1)";
69         ctx.fillRect(0,0,cw,ch);
70
71         //円の描画
72         for(let i=0;i < density; i++){
73             particles[i].position.x += particles[i].speed;
74             particles[i].draw();
75
76             if(particles[i].position.x > cw) particles[i].
77                 position.x -= cw;
78         }
79     }
80 </script>
81 </head>
82 <body onload="init();">
83     <H1>Canvas Test</H1>
84     <canvas id="mycanvas" width="1000" height="800"></canvas>
85 </body>
86 </html>

```

---

### 2.2.3 04-203.html 色を変更

円を生成する時に色を適当に指定しましょう。

81 行目あたり getRandomColor(),getRandomAlpha() という関数を追加

35 行目あたり

---

#### ソースコード 8 色を変更

---

```

1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Canvas Test</title>
5         <script type="text/javascript">
6             //canvas の中身を指し示すもの
7             let ctx;
8             //canvas の width 幅
9             let cw;
10            //canvas の height 高さ
11            let ch;
12

```

```

13
14 //パーティクルの準備
15 const density = 100; //パーティクルの密度
16 let particles = [];
17
18 //初期化处理
19 function init(){
20     //ID が mycanvas のものを canvas という変数にしておく
21     let canvas = document.getElementById('mycanvas');
22     //canvas が存在しないか、canvas の中身がなければ処理終了
23     if(!canvas || !canvas.getContext){
24         return false;
25     }
26
27     //canvas の中身を指し示すものを代入
28     ctx = canvas.getContext('2d');
29     //canvas の幅・高さを代入しておく
30     cw = canvas.width;
31     ch = canvas.height;
32
33     //円の初期化
34     for(let i=0;i < density; i++){
35         particles[i] = new Particle(6, "rgba("+
36             getRandomColor()+","+getRandomColor
37             (+","+getRandomColor()+","+
38             getRandomAlpha()+")", Math.random
39             ()*(4-2)+2);
40
41         particles[i].position.x = Math.random()*ch;
42         particles[i].position.y = Math.random()*cw;
43         particles[i].draw();
44     }
45
46     //実際の描画処理
47     draw();
48
49     //一定時間ごとに書き換える
50     setInterval("draw()",33);
51 }
52
53 //Particle クラス
54 class Particle {
55     constructor(scale, color,speed){
56         this.scale = scale;
57         this.color = color;
58         this.speed = speed;
59         this.position = {
60             x: 100,
61             y: 100

```

```

57         };
58     }
59     draw(){
60         ctx.beginPath();
61         ctx.arc(this.position.x, this.position.y, this
            .scale, 0, 2*Math.PI, false);
62         ctx.fillStyle = this.color;
63         ctx.fill();
64     }
65 }
66
67 function draw(){
68     //画面をリセットする
69     ctx.fillStyle = "rgba(255,255,255,1)";
70     ctx.fillRect(0,0,cw,ch);
71
72     //円の描画
73     for(let i=0;i < density; i++){
74         particles[i].position.x += particles[i].speed;
75         particles[i].draw();
76
77         if(particles[i].position.x > cw) particles[i].
            position.x -= cw;
78     }
79 }
80
81 function getRandomColor(){
82     return Math.floor(Math.random()*255);
83 }
84 function getRandomAlpha(){
85     return Math.random();
86 }
87
88 </script>
89 </head>
90 <body onload="init();">
91     <H1>Canvas Test</H1>
92     <canvas id="mycanvas" width="1000" height="800"></canvas>
93 </body>
94 </html>

```

---

#### 2.2.4 04-204.html 大きさを変更

86 行目あたり、getRandomScale()っていう関数を追加

34 行目あたり

---

ソースコード 9 大きさを変更

```

1 <!DOCTYPE html>

```

```

2 <html>
3     <head>
4         <title>Canvas Test</title>
5         <script type="text/javascript">
6             //canvas の中身を指し示すもの
7             let ctx;
8             //canvas の width 幅
9             let cw;
10            //canvas の height 高さ
11            let ch;
12
13            //パーティクルの準備
14            const density = 100; //パーティクルの密度
15            let particles = [];
16
17            //初期化处理
18            function init(){
19                //ID が mycanvas のものを canvas という変数にしておく
20                let canvas = document.getElementById('mycanvas');
21                //canvas が存在しないか、canvas の中身がなければ処理終了
22                if(!canvas || !canvas.getContext){
23                    return false;
24                }
25
26                //canvas の中身を指し示すものを代入
27                ctx = canvas.getContext('2d');
28                //canvas の幅・高さを代入しておく
29                cw = canvas.width;
30                ch = canvas.height;
31
32                //円の初期化
33                for(let i=0;i < density; i++){
34                    particles[i] = new Particle(getRandomScale(),
35                                                "rgba("+getRandomColor()+","+
36                                                getRandomColor()+","+getRandomColor
37                                                (+","+getRandomAlpha()+")", Math.random
38                                                ()*(4-2)+2);
39
35                    particles[i].position.x = Math.random()*ch;
36                    particles[i].position.y = Math.random()*cw;
37                    particles[i].draw();
38                }
39
40                //実際の描画処理
41                draw();
42
43                //一定時間ごとに書き換える
44                setInterval("draw()",33);
45            }

```

```

46
47 //Particle クラス
48 class Particle {
49     constructor(scale, color,speed){
50         this.scale = scale;
51         this.color = color;
52         this.speed = speed;
53         this.position = {
54             x: 100,
55             y: 100
56         };
57     }
58     draw(){
59         ctx.beginPath();
60         ctx.arc(this.position.x, this.position.y, this
            .scale, 0, 2*Math.PI, false);
61         ctx.fillStyle = this.color;
62         ctx.fill();
63     }
64 }
65
66 function draw(){
67     //画面をリセットする
68     ctx.fillStyle = "rgba(255,255,255,1)";
69     ctx.fillRect(0,0,cw,ch);
70
71     //円の描画
72     for(let i=0;i < density; i++){
73         particles[i].position.x += particles[i].speed;
74         particles[i].draw();
75
76         if(particles[i].position.x > cw) particles[i].
            position.x -= cw;
77     }
78 }
79
80 function getRandomColor(){
81     return Math.floor(Math.random()*255);
82 }
83 function getRandomAlpha(){
84     return Math.random();
85 }
86 function getRandomScale(){
87     return (Math.random()*(8-3))+3;
88 }
89 </script>
90 </head>
91 <body onload="init();">

```



```

92             <H1>Canvas Test</H1>
93             <canvas id="mycanvas" width="1000" height="800"></canvas>
94         </body>
95 </html>

```

---

## 2.3 より高度な表現

### 2.3.1 04-301.html 一定の方向に等加速度運動

49,52,53 行目で speed の代わりに vx,vy が増えているね。2 次元の速度を持たせるよ。

これを受けて、34 行目も変更

update っていう処理を 65 行目で追加して、79 行目あたりでそれを呼び出している。

ソースコード 10 一定の方向に等加速度運動

---

```

1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Canvas Test</title>
5         <script type="text/javascript">
6             //canvas の中身を指し示すもの
7             let ctx;
8             //canvas の width 幅
9             let cw;
10            //canvas の height 高さ
11            let ch;
12
13            //パーティクルの準備
14            const density = 100; //パーティクルの密度
15            let particles = [];
16
17            //初期化処理
18            function init(){
19                //ID が mycanvas のものを canvas という変数にしておく
20                let canvas = document.getElementById('mycanvas');
21                //canvas が存在しないか、canvas の中身がなければ処理終了
22                if(!canvas || !canvas.getContext){
23                    return false;
24                }
25
26                //canvas の中身を指し示すものを代入
27                ctx = canvas.getContext('2d');
28                //canvas の幅・高さを代入しておく
29                cw = canvas.width;
30                ch = canvas.height;
31
32                //円の初期化
33                for(let i=0;i < density; i++){
34                    particles[i] = new Particle(getRandomScale(),

```

```

        "rgba("+getRandomColor()+","+getRandomColor()+","+getRandomColor()+","+getRandomAlpha()+")", 5, 1);
        particles[i].position.x = Math.random()*ch;
        particles[i].position.y = Math.random()*cw;
        particles[i].draw();
    }

    //実際の描画処理
    draw();

    //一定時間ごとに書き換える
    setInterval("draw()",33);
}

//Particle クラス
class Particle {
    constructor(scale, color, vx, vy){
        this.scale = scale;
        this.color = color;
        this.vx = vx;
        this.vy = vy;
        this.position = {
            x: 100,
            y: 100
        };
    }
    draw(){
        ctx.beginPath();
        ctx.arc(this.position.x, this.position.y, this
            .scale, 0, 2*Math.PI, false);
        ctx.fillStyle = this.color;
        ctx.fill();
    }
    update(){
        this.position.x += this.vx;
        this.position.y += this.vy;
        this.draw();
    }
}

function draw(){
    //画面をリセットする
    ctx.fillStyle = "rgba(255,255,255,1)";
    ctx.fillRect(0,0,cw,ch);

    //円の描画
    for(let i=0;i < density; i++){

```

```

79             particles[i].update();
80         }
81     }
82
83     function getRandomColor(){
84         return Math.floor(Math.random()*255);
85     }
86     function getRandomAlpha(){
87         return Math.random();
88     }
89     function getRandomScale(){
90         return (Math.random()*(8-3))+3;
91     }
92
93     </script>
94 </head>
95 <body onload="init();">
96     <H1>Canvas Test</H1>
97     <canvas id="mycanvas" width="1000" height="800"></canvas>
98 </body>
99 </html>

```

---

### 2.3.2 04-302.html 重力のシミュレーション

49,54 行目に gv:gravity が増えているね…67 行目あたりで速度に重力加速度が増えている。これを受けて、34,67 行目あたりも増えてるよ。

#### ソースコード 11 重力のシミュレーション

```

1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Canvas Test</title>
5         <script type="text/javascript">
6             //canvas の中身を指し示すもの
7             let ctx;
8             //canvas の width 幅
9             let cw;
10            //canvas の height 高さ
11            let ch;
12
13            //パーティクルの準備
14            const density = 100; //パーティクルの密度
15            let particles = [];
16
17            //初期化处理
18            function init(){
19                //ID が mycanvas のものを canvas という変数にしておく
20                let canvas = document.getElementById('mycanvas');

```

```

21 //canvas が存在しないか、canvas の中身がなければ処理終了
22 if(!canvas || !canvas.getContext){
23     return false;
24 }
25
26 //canvas の中身を指し示すものを代入
27 ctx = canvas.getContext('2d');
28 //canvas の幅・高さを代入しておく
29 cw = canvas.width;
30 ch = canvas.height;
31
32 //円の初期化
33 for(let i=0;i < density; i++){
34     particles[i] = new Particle(getRandomScale(),
35     "rgba("+getRandomColor()+","+getRandomColor()
36     +","+getRandomColor()+","+getRandomAlpha()+")", 5, 1, 0.4);
37     particles[i].position.x = Math.random()*cw;
38     particles[i].position.y = Math.random()*ch;
39     particles[i].draw();
40 }
41
42 //実際の描画処理
43 draw();
44
45 //一定時間ごとに書き換える
46 setInterval("draw()",33);
47
48 }
49
50 //Particle クラス
51 class Particle {
52     constructor(scale, color, vx, vy, gv){
53         this.scale = scale;
54         this.color = color;
55         this.vx = vx;
56         this.vy = vy;
57         this.gv = gv;
58         this.position = {
59             x: 100,
60             y: 100
61         };
62     }
63     draw(){
64         ctx.beginPath();
65         ctx.arc(this.position.x, this.position.y, this
66             .scale, 0, 2*Math.PI, false);
67         ctx.fillStyle = this.color;
68         ctx.fill();

```

```

65         }
66         update(){
67             this.vy += this.gv;
68             this.position.x += this.vx;
69             this.position.y += this.vy;
70             this.draw();
71         }
72     }
73
74     function draw(){
75         //画面をリセットする
76         ctx.fillStyle = "rgba(255,255,255,1)";
77         ctx.fillRect(0,0,cw,ch);
78
79         //円の描画
80         for(let i=0;i < density; i++){
81             particles[i].update();
82         }
83     }
84
85     function getRandomColor(){
86         return Math.floor(Math.random()*255);
87     }
88     function getRandomAlpha(){
89         return Math.random();
90     }
91     function getRandomScale(){
92         return (Math.random()*(8-3))+3;
93     }
94     </script>
95 </head>
96 <body onload="init();">
97     <H1>Canvas Test</H1>
98     <canvas id="mycanvas" width="1000" height="800"></canvas>
99 </body>
100 </html>

```

---

### 2.3.3 04-303.html 床の跳ね返りをシミュレーション

71,72 行目あたりで、跳ね返る様に設定している。

ソースコード 12 床の跳ね返りをシミュレーション

---

```

1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Canvas Test</title>
5         <script type="text/javascript">
6             //canvas の中身を指し示すもの

```

```

7         let ctx;
8         //canvas の width 幅
9         let cw;
10        //canvas の height 高さ
11        let ch;
12
13        //パーティクルの準備
14        const density = 100; //パーティクルの密度
15        let particles = [];
16
17        //初期化处理
18        function init(){
19            //ID が mycanvas のものを canvas という変数にいれておく
20            let canvas = document.getElementById('mycanvas');
21            //canvas が存在しないか、canvas の中身がなければ処理終了
22            if(!canvas || !canvas.getContext){
23                return false;
24            }
25
26            //canvas の中身を指し示すものを代入
27            ctx = canvas.getContext('2d');
28            //canvas の幅・高さを代入しておく
29            cw = canvas.width;
30            ch = canvas.height;
31
32            //円の初期化
33            for(let i=0;i < density; i++){
34                particles[i] = new Particle(getRandomScale(),
35                    "rgba("+getRandomColor()+","+getRandomColor()
36                    "+","+getRandomAlpha()+")", 5, 1, 1);
37                particles[i].position.x = Math.random()*cw;
38                particles[i].position.y = Math.random()*ch;
39                particles[i].draw();
40            }
41
42            //実際の描画処理
43            draw();
44
45            //一定時間ごとに書き換える
46            setInterval("draw()",33);
47        }
48
49        //Particle クラス
50        class Particle {
51            constructor(scale, color, vx, vy, gv){
52                this.scale = scale;
53                this.color = color;

```

```

52         this.vx = vx;
53         this.vy = vy;
54         this.gv = gv;
55         this.position = {
56             x: 100,
57             y: 100
58         };
59     }
60     draw(){
61         ctx.beginPath();
62         ctx.arc(this.position.x, this.position.y, this
            .scale, 0, 2*Math.PI, false);
63         ctx.fillStyle = this.color;
64         ctx.fill();
65     }
66     update(){
67         this.vy += this.gv;
68         this.position.x += this.vx;
69         this.position.y += this.vy;
70         this.draw();
71         if(this.position.x > cw) this.position.x -= cw
            ;
72         if(this.position.y > ch) this.vy = -this.vy;
73     }
74 }
75
76 function draw(){
77     //画面をリセットする
78     ctx.fillStyle = "rgba(255,255,255,1)";
79     ctx.fillRect(0,0,cw,ch);
80
81     //円の描画
82     for(let i=0;i < density; i++){
83         particles[i].update();
84     }
85 }
86
87 function getRandomColor(){
88     return Math.floor(Math.random()*255);
89 }
90 function getRandomAlpha(){
91     return Math.random();
92 }
93 function getRandomScale(){
94     return (Math.random()*(8-3))+3;
95 }
96 </script>
97 </head>

```

```

98     <body onload="init();">
99         <H1>Canvas Test</H1>
100         <canvas id="mycanvas" width="1000" height="800"></canvas>
101     </body>
102 </html>

```

---

### 2.3.4 04-304.html とある点からの吹き出し

getRandomVelocity() を 96 行目あたりで定義。34 行目で利用。

35,36 行目あたりで左右の真ん中、上から 1/4 の位置からスタートする様に設定してる。

---

#### ソースコード 13 とある点からの吹き出し

---

```

1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Canvas Test</title>
5         <script type="text/javascript">
6             //canvas の中身を指し示すもの
7             let ctx;
8             //canvas の width 幅
9             let cw;
10            //canvas の height 高さ
11            let ch;
12
13            //パーティクルの準備
14            const density = 100; //パーティクルの密度
15            let particles = [];
16
17            //初期化处理
18            function init(){
19                //ID が mycanvas のものを canvas という変数にいれておく
20                let canvas = document.getElementById('mycanvas');
21                //canvas が存在しないか、canvas の中身がなければ処理終了
22                if(!canvas || !canvas.getContext){
23                    return false;
24                }
25
26                //canvas の中身を指し示すものを代入
27                ctx = canvas.getContext('2d');
28                //canvas の幅・高さを代入しておく
29                cw = canvas.width;
30                ch = canvas.height;
31
32                //円の初期化
33                for(let i=0;i < density; i++){
34                    particles[i] = new Particle(getRandomScale(),

```



```

        ()+" "+getRandomAlpha()+"",
        getRandomVelocity(), getRandomVelocity(),
        1);
35         particles[i].position.x = cw/2;
36         particles[i].position.y = ch/4;
37         particles[i].draw();
38     }
39
40     //実際の描画処理
41     draw();
42
43     //一定時間ごとに書き換える
44     setInterval("draw()",33);
45 }
46
47 //Particle クラス
48 class Particle {
49     constructor(scale, color, vx, vy, gv){
50         this.scale = scale;
51         this.color = color;
52         this.vx = vx;
53         this.vy = vy;
54         this.gv = gv;
55         this.position = {
56             x: 100,
57             y: 100
58         };
59     }
60     draw(){
61         ctx.beginPath();
62         ctx.arc(this.position.x, this.position.y, this
            .scale, 0, 2*Math.PI, false);
63         ctx.fillStyle = this.color;
64         ctx.fill();
65     }
66     update(){
67         this.vy += this.gv;
68         this.position.x += this.vx;
69         this.position.y += this.vy;
70         this.draw();
71         if(this.position.x > cw) this.position.x -= cw
            ;
72         if(this.position.y > ch) this.vy = -this.vy;
73     }
74 }
75
76 function draw(){
77     //画面をリセットする

```

```

78         ctx.fillStyle = "rgba(255,255,255,1)";
79         ctx.fillRect(0,0,cw,ch);
80
81         //円の描画
82         for(let i=0;i < density; i++){
83             particles[i].update();
84         }
85     }
86
87     function getRandomColor(){
88         return Math.floor(Math.random()*255);
89     }
90     function getRandomAlpha(){
91         return Math.random();
92     }
93     function getRandomScale(){
94         return (Math.random()*(8-3))+3;
95     }
96     function getRandomVelocity(){
97         return Math.random()*20-10;
98     }
99     </script>
100 </head>
101 <body onload="init();">
102     <H1>Canvas Test</H1>
103     <canvas id="mycanvas" width="1000" height="800"></canvas>
104 </body>
105 </html>

```

---

### 2.3.5 04-305.html 線の追加

17,18 行目で線の密度や、配列を追加。

84 行目から Line というクラスを追加

141 行目あたりで getRandomX(),getRandomY(),getRandomWidth() を追加

42 行目で線を初期化、126 行目で更新をしている。

ソースコード 14 線の追加

---

```

1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Canvas Test</title>
5         <script type="text/javascript">
6             //canvas の中身を指し示すもの
7             let ctx;
8             //canvas の width 幅
9             let cw;
10            //canvas の height 高さ
11            let ch;

```

```

12
13 //パーティクルの準備
14 const density = 100; //パーティクルの密度
15 let particles = [];
16
17 const density_line = 10;
18 let lines = [];
19
20 //初期化処理
21 function init(){
22     //IDがmycanvasのものをcanvasという変数にいれておく
23     let canvas = document.getElementById('mycanvas');
24     //canvasが存在しないか、canvasの中身がなければ処理終了
25     if(!canvas || !canvas.getContext){
26         return false;
27     }
28
29     //canvasの中身を指し示すものを代入
30     ctx = canvas.getContext('2d');
31     //canvasの幅・高さを代入しておく
32     cw = canvas.width;
33     ch = canvas.height;
34
35     //円の初期化
36     for(let i=0;i < density; i++){
37         particles[i] = new Particle(getRandomScale(),
38             "rgba("+getRandomColor()+","+getRandomColor()
39             +","+getRandomColor()+","+getRandomAlpha()+")",
40             getRandomVelocity(), getRandomVelocity(),
41             1);
42         particles[i].position.x = cw/2;
43         particles[i].position.y = ch/4;
44         particles[i].draw();
45     }
46     //線の初期化
47     for(let j=0;j < density_line; j++){
48         lines[j] = new Line(getRandomX(), getRandomY
49             (), getRandomVelocity(), getRandomVelocity
50             (),getRandomX(), getRandomY(),
51             getRandomVelocity(), getRandomVelocity(),"
52             rgba("+getRandomColor()+","+getRandomColor
53             +","+getRandomColor()+","+getRandomAlpha()+")",getRandomWidth(),
54             1);
55         lines[j].draw();
56     }
57 }

```

```

48         //実際の描画処理
49         draw();
50
51         //一定時間ごとに書き換える
52         setInterval("draw()",33);
53     }
54
55     //Particle クラス
56     class Particle {
57         constructor(scale, color, vx, vy, gv){
58             this.scale = scale;
59             this.color = color;
60             this.vx = vx;
61             this.vy = vy;
62             this.gv = gv;
63             this.position = {
64                 x: 100,
65                 y: 100
66             };
67         }
68         draw(){
69             ctx.beginPath();
70             ctx.arc(this.position.x, this.position.y, this
                .scale, 0, 2*Math.PI, false);
71             ctx.fillStyle = this.color;
72             ctx.fill();
73         }
74         update(){
75             this.vy += this.gv;
76             this.position.x += this.vx;
77             this.position.y += this.vy;
78             this.draw();
79             if(this.position.x > cw) this.position.x -= cw
                ;
80             if(this.position.y > ch) this.vy = -this.vy;
81         }
82     }
83
84     //Line クラス
85     class Line {
86         constructor(x1, y1, vx1, vy1, x2, y2, vx2, vy2, color
            , width, gv){
87             this.position1 = {x:x1, y:y1};
88             this.v1 = {x:vx1, y:vy1};
89             this.position2 = {x:x2, y:y2};
90             this.v2 = {x:vx2, y:vy2};
91             this.color = color;
92             this.width = width;

```

```

93         this.gv = gv;
94     }
95     draw(){
96         ctx.strokeStyle = this.color;
97         ctx.lineWidth = this.width;
98         ctx.beginPath();
99         ctx.moveTo(this.position1.x, this.position1.y
100             );
101         ctx.lineTo(this.position2.x, this.position2.y
102             );
103         ctx.stroke();
104     }
105     update(){
106         this.v1.y += this.gv;
107         this.v2.y += this.gv;
108         this.position1.x += this.v1.x;
109         this.position1.y += this.v1.y;
110         this.position2.x += this.v2.x;
111         this.position2.y += this.v2.y;
112         this.draw();
113         if(this.position1.y > ch) this.v1.y = -this.
114             v1.y;
115         if(this.position2.y > ch) this.v2.y = -this.
116             v2.y;
117     }
118 }
119
120 function draw(){
121     //画面をリセットする
122     ctx.fillStyle = "rgba(255,255,255,1)";
123     ctx.fillRect(0,0,cw,ch);
124
125     //円の描画
126     for(let i=0;i < density; i++){
127         particles[i].update();
128     }
129
130     //線の描画
131     for(let j=0;j < density_line; j++){
132         lines[j].update();
133     }
134 }
135
136 function getRandomColor(){
137     return Math.floor(Math.random()*255);
138 }
139
140 function getRandomAlpha(){
141     return Math.random();

```

```

137         }
138         function getRandomScale(){
139             return (Math.random()*(8-3))+3;
140         }
141         function getRandomX(){
142             return Math.random()*cw;
143         }
144         function getRandomY(){
145             return Math.random()*ch;
146         }
147         function getRandomWidth(){
148             return Math.random()*10;
149         }
150         function getRandomVelocity(){
151             return Math.random()*20-10;
152         }
153     </script>
154 </head>
155 <body onload="init();">
156     <H1>Canvas Test</H1>
157     <canvas id="mycanvas" width="1000" height="800"></canvas>
158 </body>
159 </html>

```

---

### 2.3.6 04-306.html 三角形の追加

20 行目で三角形の密度や、配列を追加。

124 行目から三角形というオブジェクトを追加

50 行目で三角形を初期化、177 行目で更新をしている。

#### ソースコード 15 三角形の追加

---

```

1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Canvas Test</title>
5         <script type="text/javascript">
6             //canvas の中身を指し示すもの
7             let ctx;
8             //canvas の width 幅
9             let cw;
10            //canvas の height 高さ
11            let ch;
12
13            //パーティクルの準備
14            const density = 100; //パーティクルの密度
15            let particles = [];
16
17            const density_line = 10;

```

```

18         let lines = [];
19
20         const density_tri = 10;
21         let tris = [];
22
23         //初期化处理
24         function init(){
25             //ID が mycanvas のものを canvas という変数にいておく
26             let canvas = document.getElementById('mycanvas');
27             //canvas が存在しないか、canvas の中身がなければ処理終了
28             if(!canvas || !canvas.getContext){
29                 return false;
30             }
31
32             //canvas の中身を指し示すものを代入
33             ctx = canvas.getContext('2d');
34             //canvas の幅・高さを代入しておく
35             cw = canvas.width;
36             ch = canvas.height;
37
38             //円の初期化
39             for(let i=0;i < density; i++){
40                 particles[i] = new Particle(getRandomScale(),
41                     "rgba("+getRandomColor()+","+getRandomColor()
42                     +","+getRandomColor()+","+getRandomAlpha()+")",
43                     getRandomVelocity(), getRandomVelocity(),
44                     1);
45                 particles[i].position.x = cw/2;
46                 particles[i].position.y = ch/4;
47                 particles[i].draw();
48             }
49
50             //線の初期化
51             for(let j=0;j < density_line; j++){
52                 lines[j] = new Line(getRandomX(), getRandomY
53                     (), getRandomVelocity(), getRandomVelocity
54                     (),getRandomX(), getRandomY(),
55                     getRandomVelocity(), getRandomVelocity(),"
56                     rgba("+getRandomColor()+","+getRandomColor
57                     +","+getRandomColor()+","+getRandomAlpha()+")",getRandomWidth(),
58                     1);
59                 lines[j].draw();
60             }
61
62             //三角形の初期化
63             for(let k=0;k<density_tri;k++){
64                 tris[k] = new Tri(getRandomX(), getRandomY(),
65                     getRandomVelocity(), getRandomVelocity(),

```

```

53         getRandomX(), getRandomY(),
54         getRandomVelocity(), getRandomVelocity(),
55         getRandomX(), getRandomY(),
56         getRandomVelocity(), getRandomVelocity(),"
57         rgba("+getRandomColor()+","+getRandomColor
58         (")+","+getRandomColor()+","+
59         getRandomAlpha()+")",getRandomWidth(),
60         1);
61         tris[k].draw();
62     }
63
64     //実際の描画処理
65     draw();
66
67     //一定時間ごとに書き換える
68     setInterval("draw()",33);
69 }
70
71 //Particle クラス
72 class Particle {
73     constructor(scale, color, vx, vy, gv){
74         this.scale = scale;
75         this.color = color;
76         this.vx = vx;
77         this.vy = vy;
78         this.gv = gv;
79         this.position = {
80             x: 100,
81             y: 100
82         };
83     }
84     draw(){
85         ctx.beginPath();
86         ctx.arc(this.position.x, this.position.y, this
87             .scale, 0, 2*Math.PI, false);
88         ctx.fillStyle = this.color;
89         ctx.fill();
90     }
91     update(){
92         this.vy += this.gv;
93         this.position.x += this.vx;
94         this.position.y += this.vy;
95         this.draw();
96         if(this.position.x > cw) this.position.x -= cw
97             ;
98         if(this.position.y > ch) this.vy = -this.vy;
99     }
100 }

```



```

91
92 //Line クラス
93 class Line {
94     constructor(x1, y1, vx1, vy1, x2, y2, vx2, vy2, color
95         , width, gv){
96         this.position1 = {x:x1, y:y1};
97         this.v1 = {x:vx1, y:vy1};
98         this.position2 = {x:x2, y:y2};
99         this.v2 = {x:vx2, y:vy2};
100        this.color = color;
101        this.width = width;
102        this.gv = gv;
103    }
104    draw(){
105        ctx.strokeStyle = this.color;
106        ctx.lineWidth = this.width;
107        ctx.beginPath();
108        ctx.moveTo(this.position1.x, this.position1.y
109            );
110        ctx.lineTo(this.position2.x, this.position2.y
111            );
112        ctx.stroke();
113    }
114    update(){
115        this.v1.y += this.gv;
116        this.v2.y += this.gv;
117        this.position1.x += this.v1.x;
118        this.position1.y += this.v1.y;
119        this.position2.x += this.v2.x;
120        this.position2.y += this.v2.y;
121        this.draw();
122        if(this.position1.y > ch) this.v1.y = -this.
123            v1.y;
124        if(this.position2.y > ch) this.v2.y = -this.
125            v2.y;
126    }
127 }
128
129 class Tri {
130     constructor(x1, y1, vx1, vy1, x2, y2, vx2, vy2, x3,
131         y3, vx3, vy3, color, width, gv){
132         this.position1 = {x:x1, y:y1};
133         this.v1 = {x:vx1, y:vy1};
134         this.position2 = {x:x2, y:y2};
135         this.v2 = {x:vx2, y:vy2};
136         this.position3 = {x:x3, y:y3};
137         this.v3 = {x:vx3, y:vy3};
138         this.color = color;

```

```

133         this.width = width;
134         this.gv = gv;
135     }
136     draw(){
137         ctx.strokeStyle = this.color;
138         ctx.lineWidth = this.width;
139         ctx.beginPath();
140         ctx.moveTo(this.position1.x, this.position1.y
141             );
142         ctx.lineTo(this.position2.x, this.position2.y
143             );
144         ctx.lineTo(this.position3.x, this.position3.y
145             );
146         ctx.lineTo(this.position1.x, this.position1.y
147             );
148         ctx.stroke();
149     }
150     update(){
151         this.v1.y += this.gv;
152         this.v2.y += this.gv;
153         this.v3.y += this.gv;
154         this.position1.x += this.v1.x;
155         this.position1.y += this.v1.y;
156         this.position2.x += this.v2.x;
157         this.position2.y += this.v2.y;
158         this.position3.x += this.v3.x;
159         this.position3.y += this.v3.y;
160         this.draw();
161         if(this.position1.y > ch) this.v1.y = -this.
162             v1.y;
163         if(this.position2.y > ch) this.v2.y = -this.
164             v2.y;
165         if(this.position3.y > ch) this.v3.y = -this.
166             v3.y;
167     }
168 }
169
170 function draw(){
171     //画面をリセットする
172     ctx.fillStyle = "rgba(255,255,255,1)";
173     ctx.fillRect(0,0,cw,ch);
174
175     //円の描画
176     for(let i=0;i < density; i++){
177         particles[i].update();
178     }
179
180     //線の描画

```

```

174         for(let j=0;j < density_line; j++){
175             lines[j].update();
176         }
177         //三角形の描画
178         for(let k=0;k < density_tri; k++){
179             tris[k].update();
180         }
181     }
182
183     function getRandomColor(){
184         return Math.floor(Math.random()*255);
185     }
186     function getRandomAlpha(){
187         return Math.random();
188     }
189     function getRandomScale(){
190         return (Math.random()*(8-3))+3;
191     }
192     function getRandomX(){
193         return Math.random()*cw;
194     }
195     function getRandomY(){
196         return Math.random()*ch;
197     }
198     function getRandomWidth(){
199         return Math.random()*10;
200     }
201     function getRandomVelocity(){
202         return Math.random()*20-10;
203     }
204     </script>
205 </head>
206 <body onload="init();">
207     <H1>Canvas Test</H1>
208     <canvas id="mycanvas" width="1000" height="800"></canvas>
209 </body>
210 </html>

```

---

以上