プログラミング基礎演習 / 説明資料 / G2 / ball / 訂正

訂正 1）ボールの速度と座標をランダムにする

前回）

```
function makeBalls() {
・・・略・・・
  // ボール[i]の座標をランダムに設定
  balls[i].posX = Math.random() * (maxX - balls[i].offsetWidth) + balls[i].offsetWidth
/ 2;
  balls[i].posY = Math.random() * (maxY - balls[i].offsetHeight) + balls[i].offsetHeight
/ 2;

  // ボール[i]の速度をランダムに設定
  balls[i].vX = Math.random() * 10 - 5;
  balls[i].vY = Math.random() * 10 - 5;
}
```

↓

訂正）

```
const ballSpeed = 5; // ボールの速度

function makeBalls() {
・・・略・・・
    // ボール[i]の速度をランダムに設定
    let s = ballSpeed;
    balls[i].vX = Math.random() * s * 2 - s;
    balls[i].vY = Math.random() * s * 2 - s;

    // ボール[i]の座標をランダムに設定
    let r = balls[i].offsetWidth / 2;
    balls[i].posX = Math.random() * (maxX - r * 2 - s * 2) + r + s;
    balls[i].posY = Math.random() * (maxY - r * 2 - s * 2) + r + s;
}
```

解説：

（1）ボールの速度を変数設定する

```
const ballSpeed = 5; // ボールの速度
```

（2）ボールの速度をランダムにする

```
    let s = ballSpeed;
    balls[i].vX = Math.random() * s * 2 - s;
    balls[i].vY = Math.random() * s * 2 - s;
```

計算の仕組み

Math.random()	0〜1 未満のランダムな小数を返す 例: 0.0, 0.5, 0.9999...
* s * 2	0 以上 s 未満の範囲に値が広がる。 さらに * 2 を行うことで、0 以上 2s 未満の範囲に拡大される。 これは、最終的に -s 〜 +s のランダム値を作るための準備である。
- s	最後に s を引くことで、 値の範囲が 左に s だけシフトされる

（3）ボールの座標をランダムにする

```
    let r = balls[i].offsetWidth / 2;
    balls[i].posX = Math.random() * (maxX - r * 2 - s * 2) + r + s;
    balls[i].posY = Math.random() * (maxY - r * 2 - s * 2) + r + s;
```

計算の仕組み

コード	意味
Math.random()	0 以上 1 未満のランダムな小数を返す 例 0.0, 0.5, 0.9999...
* (maxX - r * 2 - s * 2)	画面の横幅（maxX）から ボールの直径（r * 2）とボールの最大速度分の余白（s * 2） を引いた「配置可能な範囲」を指定する
+ r + s	左端（または上端）からの 開始位置を指定する • + r → ボールの中心が画面端に来ないようにするための余白 • + s → 最初の移動で画面外に出ないようにするための余白

訂正 2）画面の左右・上下の端で跳ね返る

前回）

```
// 画面の左右の端で跳ね返る
if (balls[i].posX > maxX - balls[i].offsetWidth / 2) {
  balls[i].vX = -balls[i].vX;
}
if (balls[i].posX < balls[i].offsetWidth / 2) {
  balls[i].vX = -balls[i].vX;
}
// 画面の上下の端で跳ね返る
if (balls[i].posY > maxY - balls[i].offsetHeight / 2) {
  balls[i].vY = -balls[i].vY;
}
if (balls[i].posY < balls[i].offsetHeight / 2) {
  balls[i].vY = -balls[i].vY;
}
```

↓

訂正）黄色箇所追加

```
// 画面の左右の端で跳ね返る
if (balls[i].posX > maxX - balls[i].offsetWidth / 2) {
  balls[i].posX = maxX - balls[i].offsetWidth / 2;
  balls[i].vX = -balls[i].vX;
}
if (balls[i].posX < balls[i].offsetWidth / 2) {
  balls[i].posX = balls[i].offsetWidth / 2;
  balls[i].vX = -balls[i].vX;
}
// 画面の上下の端で跳ね返る
if (balls[i].posY > maxY - balls[i].offsetHeight / 2) {
  balls[i].posY = maxY - balls[i].offsetHeight / 2;
  balls[i].vY = -balls[i].vY;
}
if (balls[i].posY < balls[i].offsetHeight / 2) {
  balls[i].posY = balls[i].offsetHeight / 2;
  balls[i].vY = -balls[i].vY;
}
```