講義メモ
・「ジャンケン」その3の補足をしてから、p.098「繰り返し文」に進みます

提出：追加演習「ジャンケン」その3 janken3

・判定を表示しよう
・例「あなたの勝ち」「わたしの勝ち」「あいこ」
・ヒント：先に「あいこ」を判断すると楽。判定はORで連結した条件ですると良い
　※ 計算式で判定することもできる：(3 + hand - cpuh) % 3 == 2 ならばhandの勝ち

作成例1

```
using UnityEngine;
public class janken3 : MonoBehaviour {
    [Header("手をどうぞ(0=グー,1=チョキ,2=パー)")]
    public int hand;
    void Start() {
        System.Random r = new System.Random(); //乱数クラスのオブジェクトを生成
        string s = "あなたの手＝";
        Debug.Log(s + ((hand == 0) ? "グー" : (hand == 1) ? "チョキ" : "パー
"));
        string c = "わたしの手＝";
        int cpuh = r.Next(3); //乱数で0,1,2を得る
        Debug.Log(c + ((cpuh == 0) ? "グー" : (cpuh == 1) ? "チョキ" : "パー
"));
        if(hand == cpuh) {
            Debug.Log("あいこ");
        } else if (hand == 0 && cpuh == 1 ||
                   hand == 1 && cpuh == 2 ||
                   hand == 2 && cpuh == 0) { //条件で判定
            Debug.Log("あなたの勝ち");
        } else {
            Debug.Log("わたしの勝ち");
        }
    }
    void Update() { }
}
```

作成例2

```
using UnityEngine;
public class janken3 : MonoBehaviour {
    [Header("手をどうぞ(0=グー,1=チョキ,2=パー)")]
    public int hand;
    void Start() {
        System.Random r = new System.Random(); //乱数クラスのオブジェクトを生成
        string s = "あなたの手＝";
        Debug.Log(s + ((hand == 0) ? "グー" : (hand == 1) ? "チョキ" : "パー
"));
```

```
        string c = "わたしの手＝";
        int cpuh = r.Next(3); //乱数で0,1,2を得る
        Debug.Log(c + ((cpuh == 0) ? "グー" : (cpuh == 1) ? "チョキ" : "パー
"));
        if(hand == cpuh) {
            Debug.Log("あいこ");
        } else if ((3 + hand - cpuh) % 3 == 2) { //式で判定
            Debug.Log("あなたの勝ち");
        } else {
            Debug.Log("わたしの勝ち");
        }
    }
    void Update() { }
}
```

Chapter 3 繰返し文

p.100 while文

・前判定の単純な繰返しに向く構文
・前判定：繰返しの1回目の前に条件をチェックすること。1度も繰り返さないことがある場合に便利
・書式： while(継続条件) { 繰返し内容;… }
・継続条件はbool型(p.56)の値または式で、主に、比較演算子の式を用いる
・例： while(a < 3) { a = a + 1; } //aが0だったら3回繰り返す
・この例のように式がtrueである間＝条件式が成立する間だけ繰返すというパターンが多い
・例： while(a > 0) { a = a - 1; } //aが3だったら3回繰り返す

p.101 chap3_2_1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class chap3_2_1 : MonoBehaviour {
    void Start() {
        int shikin = 30000;
        while(shikin >= 0) { //shikinが0以上であれば繰返す
            Debug.Log(shikin);
            shikin = shikin - 5080; //shikinから5080を差し引く
        } //whileブロック(繰返し内容)の終わり
    }
    void Update() {

    }
}
```

p.103 計算もできる代入演算子

・正式には複合代入演算子といい、左辺の変数を用いた式を代入の右辺に書く場合に、誤読しやす

いことから、誤読を防ぎ、かつ、冗長さを省く仕掛け
・+=演算子: ○ = ○ + ◆ を ○ += ◆ と書ける。意味は「足し込む」
 例: a = a + 5; ⇒ a += 5;
・-=演算子: ○ = ○ - ◆ を ○ -= ◆ と書ける。意味は「差し引く」
 例: a = a - 5; ⇒ a -= 5;
・他に「*=(N倍する、掛けた積にする)」「/=(割った商にする)」「%=(割った余りにする)」などがある

アレンジ演習:p.101 chap3_2_1①

・-=演算子で書き直そう

作成例

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class chap3_2_1 : MonoBehaviour {
    void Start() {
        int shikin = 30000;
        while(shikin >= 0) { //shikinが0以上であれば繰返す
            Debug.Log(shikin);
            shikin -= 5080; //【変更】shikinから5080を差し引く
        } //whileブロック(繰返し内容)の終わり
    }
    void Update() {

    }
}
```

アレンジ演習:p.101 chap3_2_1②

・資金をパブリック変数で与えるようにしよう
・そして、資金が負の数の時には何も表示されないことも確認しよう

作成例

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class chap3_2_1 : MonoBehaviour {
    public int shikin = 30000; //【移動】
    void Start() {
        while(shikin >= 0) { //shikinが0以上であれば繰返す
            Debug.Log(shikin);
            shikin -= 5080; //shikinから5080を差し引く
        } //whileブロック(繰返し内容)の終わり
    }
```

```
    void Update() {

    }
}
```

アレンジ演習：p.101 chap3_2_1③

・whileブロックを抜けた後で、資金の額はどうなっているか確認しよう

作成例

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class chap3_2_1 : MonoBehaviour {
    public int shikin = 30000; //【移動】
    void Start() {
        while(shikin >= 0) { //shikinが0以上であれば繰返す
            Debug.Log(shikin);
            shikin -= 5080; //shikinから5080を差し引く
        } //whileブロック(繰返し内容)の終わり
        Debug.Log("繰返し後の資金 = " + shikin); //【追加】
    }
    void Update() {

    }
}
```

アレンジ演習：p.101 chap3_2_1④

・if文を加えて、資金が赤字(負の数)にならないようにしよう
・ヒント： 資金が5080以上あれば差し引くようにすれば良い
・しかし、こうすると、繰返しが終わらなくなるので、繰返し条件を「資金 > 0」に変更しよう
・そして、資金が5080未満であれば資金を0にすれば良い（あるだけ払うイメージ）

作成例

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class chap3_2_1 : MonoBehaviour {
    public int shikin = 30000;
    void Start() {
        while(shikin > 0) { //【変更】shikinがある間繰返す
            Debug.Log(shikin);
            if (shikin >= 5080) { //【追加】shikinが5080以上ある？
                shikin -= 5080; //shikinから5080を差し引く
```

```
            } else { //【追加】shikinが5080未満？
                shikin = 0; //【追加】shikinを使い尽くす
            }
        } //whileブロック(繰返し内容)の終わり
        Debug.Log("繰返し後の資金 = " + shikin);
    }
    void Update() {

    }
}
```

ミニ演習 mini103

・3から0までカウントダウンし、続けて3までカウントアップしよう
・whileを続けて2回行えば良い

作成例

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class mini103 : MonoBehaviour {
    void Start() {
        int i = 3;
        while (i > 0) { //0超である間繰返す＝0になるまで
            Debug.Log(i);
            i = i - 1; //1を差し引く
        } //この繰返しを抜けた時点ではiは0になっている
        while (i <= 3) { //3以下ある間繰返す＝4になるまで
            Debug.Log(i);
            i = i + 1; //1を足し込む
        }
    }
    void Update() {

    }
}
```

補足：後判定繰返し文

・繰返し内容を1度行ってから繰返すかどうか判断する場合に用いる繰返し構文
・書式： do { 繰返し内容 } while(継続条件);
・例：do { 100円で遊ぶ } while(まだ金がある);
・主に、入力や受信などで得たものをチェックし、正しいものが得られるまで先に進まない場合に用いることが多い
・例：do { 画面などからyかnを入力 } while(yでもnでもない？);
・例：do { データを受信 } while(正しくない？);

ミニ演習 mini103b

・chap3_2_1を後判定繰返しにしてみよう
①パブリック変数でint zandaka = 30000 を受け取る
②5080円払う
③残高を表示する
④残高がある間、②③を繰り返す
※ 最初の残高が5080円未満でも（赤字になっても）払うことになる

作成例

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class mini103b : MonoBehaviour {
    public int zandaka = 30000;
    void Start() {
        do {
            zandaka -= 5080; //5080円払う(たとえ赤字でも！)
            Debug.Log(zandaka);
        } while (zandaka > 0); //残高がある間繰返す
    }
    void Update() {

    }
}
```

補足：2重繰返し

・p.099の図の通り「繰返しの中で繰返し」を行うことが可能
・この場合、繰返しに関わる変数やカウンタの変数の変化に注意
例：2×3回繰り返す場合①

```
int a = 2;
while (a > 0) {
 a -= 1;
 int b = 3;
 while (b > 0)
  b -= 1;
  ここに処理を書くと2×3回繰返される
 }
}
```

例：2×3回繰り返す場合②

```
int a = 1;
while (a <= 2) {
 int b = 1;
 while (b <= 3)
  ここに処理を書くと2×3回繰返される
  b += 1;
```

```
 }
 a += 1;
}
```

ミニ演習　平面ダンジョンへのモンスターの配置作り mini099

・縦3×横2の6部屋のダンジョンがある
・乱数を用いて、1部屋に0～9匹のモンスターを配置したい
下記のように表示しよう
 1階1号室は9匹
 1階2号室は2匹
 2階1号室は9匹
 2階2号室は3匹
 3階1号室は0匹
 3階2号室は1匹

作成例

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class mini099 : MonoBehaviour {
    void Start() {
        System.Random r = new System.Random(); //乱数クラスのオブジェクトを生成
        int a = 1; //1階から
        while (a <= 3) { //3階まで繰返す
            int b = 1; //1号室から
            while (b <= 2) { //2号室まで繰返す
                int n = r.Next(10); //0～9の乱数を得る
                Debug.Log(a + "階" + b + "号室は" + n + "匹");
                b = b + 1; //次の部屋へ
            }
            a = a + 1; //次の階へ
        }
    }
    void Update() {

    }
}
```

ミニ演習　直列ダンジョンへのモンスターの配置作り mini099b

・6部屋のダンジョンがある
・乱数を用いて、1部屋に0～9匹のモンスターを配置したい
・モンスターの数を●の数で示そう
下記のように表示しよう
 1号室：●●●●●●●●●
 2号室：●●

3号室:●●●
4号室:
5号室:●●●●●
6号室:●●●●
・ヒント:内側の繰返しは"●"の連結をモンスターの数だけ行う処理になる
①a=1号室から
②a=6号室まで③から⑩を繰り返す
③文字列sを a + "号室:" にする
④モンスター数nを乱数rで決める
⑤カウンタbを0にする
⑥カウンタbがモンスター数n以下である間、⑦から⑧を繰り返す
⑦文字列sを s + "●" にする(つまり後ろに"●"を連結する)
⑧カウンタbを b + 1 にする(つまり次のモンスターへ)
⑨文字列sを表示
⑩カウンタaを a + 1 にする(つまり次の部屋へ)

作成例

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class mini099b : MonoBehaviour {
    void Start() {
        System.Random r = new System.Random(); //乱数クラスのオブジェクトを生成
        int a = 1; //1号室から
        while (a <= 6) { //6号室まで繰返す
            string s = a + "号室:"; //「■号室:」を用意
            int n = r.Next(10); //モンスター数を0～9にする
            int b = 0; //カウンタを0にする
            while (b < n) { //カウンタがモンスター数未満の間
                s += "●"; //文字列に"●"を連結
                b = b + 1; //次のモンスターへ
            }
            Debug.Log(s); //できた文字列を表示
            a = a + 1; //次の部屋へ
        }
    }
    void Update() {

    }
}
```

提出:アレンジ演習:直列ダンジョンへのモンスターの配置作り mini099b

・9匹満室の部屋には「満室」と表示しよう
例:
1号室:●●●●●●●●●満室
2号室:●●

3号室：●●●
4号室：
5号室：●●●●●●●●●満室
6号室：●●●●
ヒント：
・Debug.Logする直前に「nが9なら、sに"満室"を連結する」処理を挿入すれば良い

次回予告：p.104「for文」から