イベント通知とラムダ式

目次

１．イベント通知とは

今回は、イベント通知という仕組みについて話をしていきたいと思います。
この話はゲームプログラミングというより、一般的なプログラミングでも使用される
「ソフトウェア工学」の中の話になります。

例えば、コンビニのレジを想像してください。店員がバーコードを読み取った時、

- 金額が表示される(画面)
- 商品情報が表示される(画面)
- 音が鳴る(音)

ある1つの出来事(商品をスキャン)が、複数の処理を発生させます。
上記のプログラムを作る際に、例えば将来的に商品をスキャンではなくて
出入り口のゲートを通った時に実施したい場合、使いまわしをしたいですよね！

１．イベント通知とは

以下のように全部まとめてレジクラスとして作ってしまうと、メソッド1をスキャンから出入り口に変更が必要となる他、クラス内の処理を変えているのでメソッド2やメソッド3の再検証も必要となってしまいます。
仕事で言うと、工数がより多くかかってしまうということになります。

[買い物]クラス

＜メソッド1＞
スキャン
　⇨スキャン成功でメソッド2と3を呼び出し

＜メソッド2＞
画面表示

＜メソッド3＞
音を鳴らす

[買い物]クラス

＜メソッド1＞
出入り口で自動判定
　⇨自動判定成功でメソッド2と3を呼び出し

＜メソッド2＞
画面表示

＜メソッド3＞
音を鳴らす

１．イベント通知とは

先ほどの例のことを頭に置きつつ、まずは以下の実装をしてみてください。

・Player：画面上にSquareを表示し、WASDで動かせる。
・Enemy：ひたすらDebug.Logを出力する。
・5秒経過した時点でPlayerを動かせなくしてください。
・5秒経過した時点でEnemyのDebug.Logを止めてください。
・1クラス1責務のルールを守り、以下の3つのクラスを作成してください。
　(1) PlayerTestクラス
　(2) EnemyTestクラス
　(3) GameManagerTestクラス(5秒をカウント)

１．Action/delegateとは

以下のような実装になっていませんか？
この場合、クラス間の繋がりが高い(依存している)状態となっています。
結合後が高い、という表現ですね！

[GameManager]クラス
＜変数＞
5秒経過フラグ
カウント処理用の変数

＜メソッド＞
5秒のカウント処理

[Player]クラス
＜変数＞
[SerializeField]GameManagerクラス

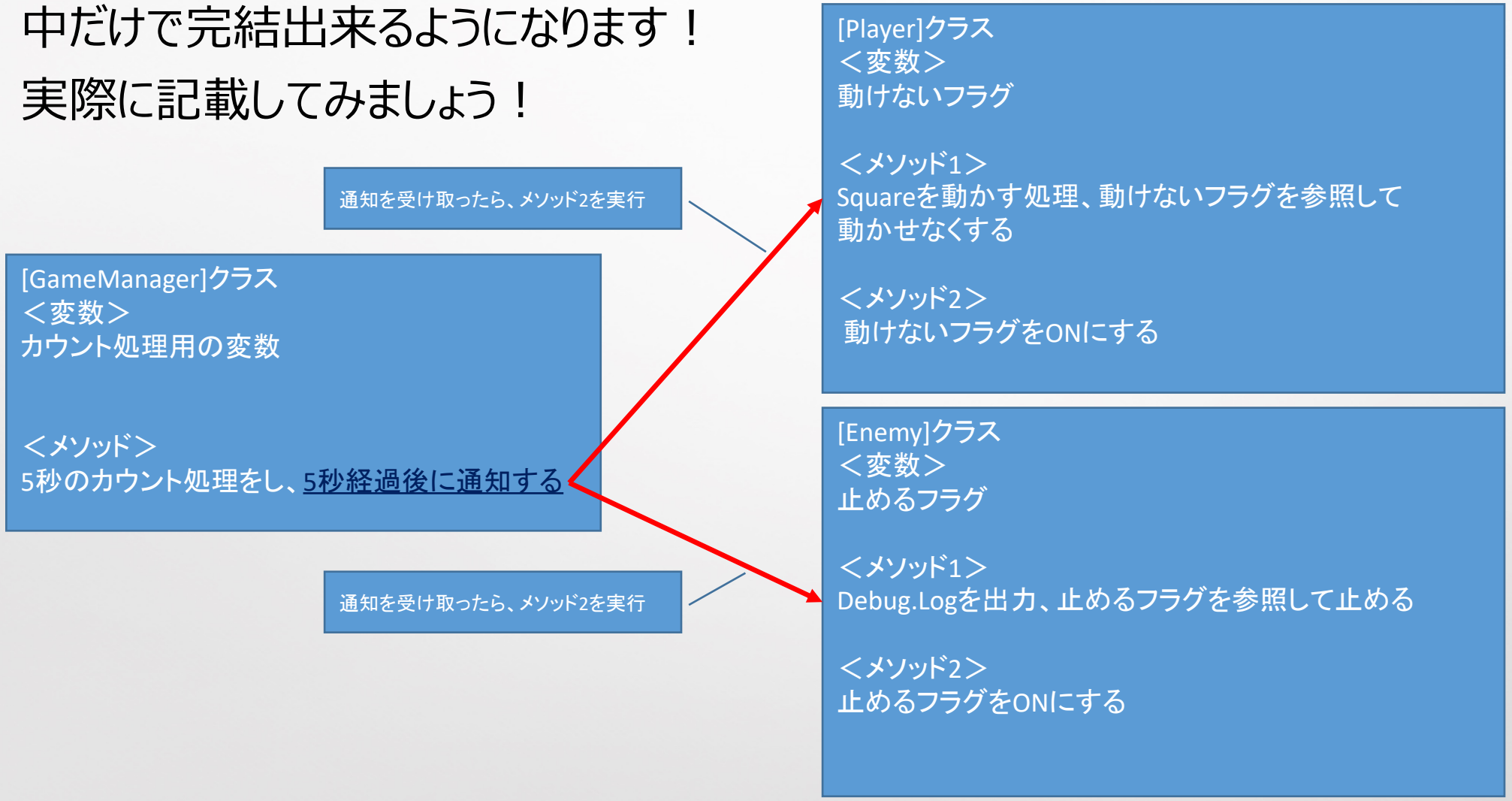
＜メソッド＞
Squareを動かす処理、GameManagerの
5秒経過のフラグを参照して動かせなくする

[Enemy]クラス
＜変数＞
[SerializeField]GameManagerクラス

＜メソッド＞
Debug.Logを出力、GameManagerの
5秒経過のフラグを参照して止める

１．Action/delegateとは

イベント通知を使うことで、以下のようにPlayer/Enemyクラスともに、自分の中だけで完結出来るようになります！
実際に記載してみましょう！



１．イベント通知とは

＜ちょっと補足＞

C#のアクション(Action)とは、メソッドを変数のように扱うための機能です。

機能としてはCやC++の「関数ポインタ」と同様の機能となります！

(C#は基本的にはポインタが使えないので、その代替え手段となるものです)

趣味でゲームを作る際にはあまり使用しないかもしれませんが、仕事でゲームを作る際は必須となる機能です。

ざっくり言えば、「柔軟で再利用性の高いコードが書ける」ことが特徴です！

再利用が高いと異なるプロジェクト間で使い回しが出来る

⇨ 設計や評価の省略が出来る ⇨ 工数が減る ⇨ 早く帰れる！！

１．イベント通知とは

その他、Actionは引数を指定することが出来ます、以下のように記述します。

```
public Action<int> OnEventNotification;
```

ちなみに、Actionは戻り値を使うことが出来ません。
どうしても使いたい場合はDelegateというものもあるので、戻り値を使う場合と使わない場合で記述が違うことを知っておいてください。

＜練習問題＞

先ほどの5秒経過したら通知、をXのキーボードを押した際に通知に変更してみてください。

また、引数で今の時間を渡してPlayer側でDebug.Logで表示してください。

１．イベント通知とは

最後に、Unityのボタンについてもこのイベント通知を使ってスクリプトから登録する事が出来ます。
今までみたいに直接Actionを指定しなくても、以下のような記述で対応可能です。

```
[SerializeField]private Button btnAction;

<Startの中で>
btnAction.onClick.AddListener(ActionButton);

private void ActionButton()  ～
```

２．ラムダ式とは

イベント通知の際によく使う機能がもう1つあるので紹介します。
別名、匿名関数と呼ばれている名前が無い関数を記載することが出来る
ラムダ式という書き方です。(C#、C++、Java、Pythonなどで採用)

例えば先ほど書いてもらったPlayerやEnemyですが、たった数行の通知を
受け取るだけなのにメソッドを作る必要があるのか、、、と思った人もいるかと
思います。
そういう場合に、ちょっと楽して記述することが出来るのがラムダ式というものに
なるのでまずはEnemy側のスクリプトで試してみましょう！

２．ラムダ式とは

＜Enemy.csのStart内の記述＞

```
//通知を受ける設定をする
//gameManager.OnEventNotification += StopMove;
gameManager.OnEventNotification += dt =>
{
    //出力を止める
    Debug.Log("Enemy側が通知を受け取りました");
    isLogStop = true;
};
```

出来た人は、Player側もやってみてください！！