Android ゲームプログラミング

○　第４－２回　ScheduledExecutorServiceによるメインループ　○

ScheduledExecutorService

・Java SE 1.5のから導入されたマルチスレッドをより安全・効率的に制御するクラス
・Runnableインタフェースのrunメソッドを定期的もしくは定間隔に実行できる
・アプリケーションは常に実行状態になる(Threadクラスを用いた場合はSleepメソッドで休眠状態)

概要

ScheduledExecutorServiceクラスは、Java SE 1.5から導入された、マルチスレッドを安全・効率的に制御することができるクラスです。このクラスは、定期的もしくは一定間隔でRunnableインタフェースのrunメソッドを実行することができます。従来はThreadクラスとSleepメソッドを組み合わせたり、Timerクラスを用いることで実装していましたが、それらの欠点を補ったものです。

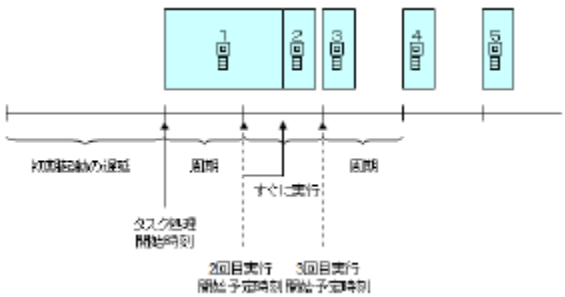
スレッドは、生成時にOSやJava仮想マシンに負荷がかかります。また、生成していくたびに、CPUやメモリなどの資源を消費しますが、スレッドの数がCPUの能力を超えてしまうと、スレッドの実行が停滞してしまいます。これらの問題点を簡単に解決できるのがExecutorServiceクラスです。このクラスは、スレッドの生成数に制限をもたせることができます。CPUにあったスレッド数に制限することで、資源を無理なく最大限に利用できます。また、スレッドの終了を安全にできる機能もあります。

ScheduledExecutorServiceクラスは、ExecutorServiceクラスを継承し、定期的もしくは一定間隔にスレッドを実行する機能を追加したクラスです。ゲームは一定間隔で画面を更新する必要があります。また、処理落ちした場合は、描画処理などを省略し、本来の時間まで追いかけるアルゴリズムがありますが、そういった機能を簡単に実装できる機能が備わっています。

ScheduledExecutorServiceクラスのスケジュール機能

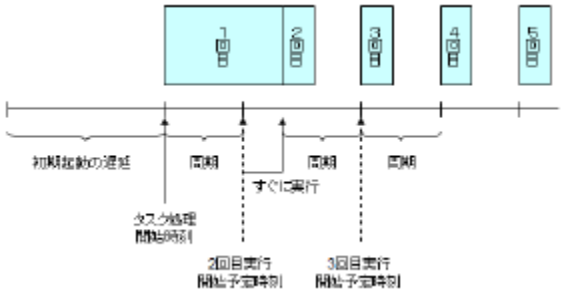
ScheduledExecutorServiceクラスのスケジュール方法は２つあり、「処理を開始する間隔」もしくは「処理の終了から次の処理の開始までの間隔」を指定することができます。

「処理を開始する間隔」とは、処理が開始される時期が一定であるということです。たとえば「毎週月曜日に処理が実行される」とすると、処理が火曜日に終わったとしても、日曜日までかかったとしても、次の実行は月曜日になるというものです。処理に時間がかかりすぎて、次の週の火曜日までかかった場合は、さらに次の月曜日まで待つのではなく、処理が終わり次第、すぐに次の処理にとりかかります。月曜日に実行できるようになるまで、待つことなく「次の処理」を実行し続けます。このようなスケジュール方法を行っているのがscheduleAtFixedRateメソッドです。



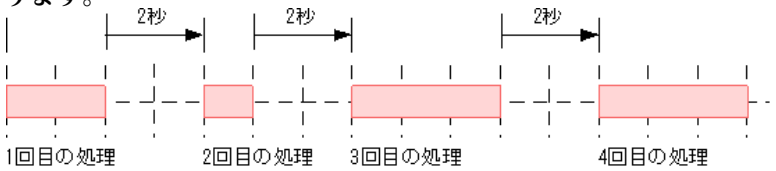
▲scheduleAtFixedRateメソッドのスケジュール方法

同じようなスケジュールを行っているのがscheduleメソッドです。これは「毎週同じ曜日を目指す」のではなく「処理の開始」と「次の処理の開始」の間隔が一定というものです。間隔が７日間だとすると、最初の処理の開始が月曜日であれば、次の処理の開始は基本的には月曜日になります。しかし、ある処理で滞り、次の週の火曜日までかかってしまった場合は、その次の処理はすぐに開始されますが、さらにその次の処理の開始は次の火曜日になる、というものです。第４回で実装したメインループの間隔も、このスケジュール方法と同じです。



▲scheduleメソッドのスケジュール方法

これらのメソッドのほかに、間隔そのもの(処理と処理の間の休止時間)が一定のscheduleWithFixedDelayメソッドもあります。



▲scheduleWithFixedDelayメソッドのスケジュール方法

コーディング例

```
// 定数
public static final long            FPS      = 60;
public static final long            INTERVAL = (1000 * 1000) / FPS;

private ScheduledExecutorService    mExecutor;
private Runnable                    mRunnable = new Runnable()
{
    public void run() {
        // 定期的に実行したい処理を記述
    }
};

        ⋮

// runメソッドの定期実行開始
mExecutor = Executors.newSingleThreadScheduledExecutor();
mExecutor.scheduleAtFixedRate(mRunnable, 0, INTERVAL, TimeUnit.MICROSECONDS);

        ⋮

// スレッド終了
mExecutor.shutdownNow();
try {
    // スレッド終了まで待つ
    mExecutor.awaitTermination(Long.MAX_VALUE, TimeUnit.MILLISECONDS);
} catch (Exception e) {
}
```

課　　題

メインループをScheduledExecutorServiceクラスのscheduleAtFixedRateメソッドを用いたものに変更しましょう。