パフォーマンスチューニング

技術研修第10回

佐藤彩理

目的

- パフォーマンスチューニングの基礎知識を身につける💡
- 今日のゴール⚽

  「パフォーマンスチューニングに関しての基礎知識を身につけて、

  必要な時に調べて対応できるようになること」

  → もう出来てる人もいるかもしれない……

目次

1. どうしてパフォーマンスチューニングが必要なの？
2. パフォチューの基礎知識
3. 高速化へのTIPS
4. 実例紹介

Unity®

PERFORMANCE TUNING BIBLE

CyberAgent®

どうしてパフォーマンスチューニングが必要なの？

ゲームの面白さを
最大限に表現するため

どうしてパフォーマンスチューニングが必要なの？

- いくら面白いゲームを作ってもカックカクだったら面白くない
- いくらインゲームが快適に動いても画面遷移に1分かかったら？
- 全部めっちゃ面白くてもスマホがめちゃくちゃに発熱して30分で充電が切れちゃうくらい消耗するとしたら？

100%を目指す必要はないが必要不可欠

パフォーマンスチューニングのタイミング

- ゲーム作りはスクラップ＆ビルドを繰り返す
- まずは早く作って面白いかを見るのが大切なので、
  初期からパフォーマンスを意識しすぎる必要はない
  ※ ただし、FPS安定しないと想像できないゲーム
  　リズムアクション、シューティング、格闘などは必要がある
- ゲームとしてのコアが決まってからはじめて、
  パフォーマンスも意識したゲーム作りが始まる

パフォチューの基礎知識

パフォチューの指標

量産フェーズに入る前に
必ず決めておくこと！

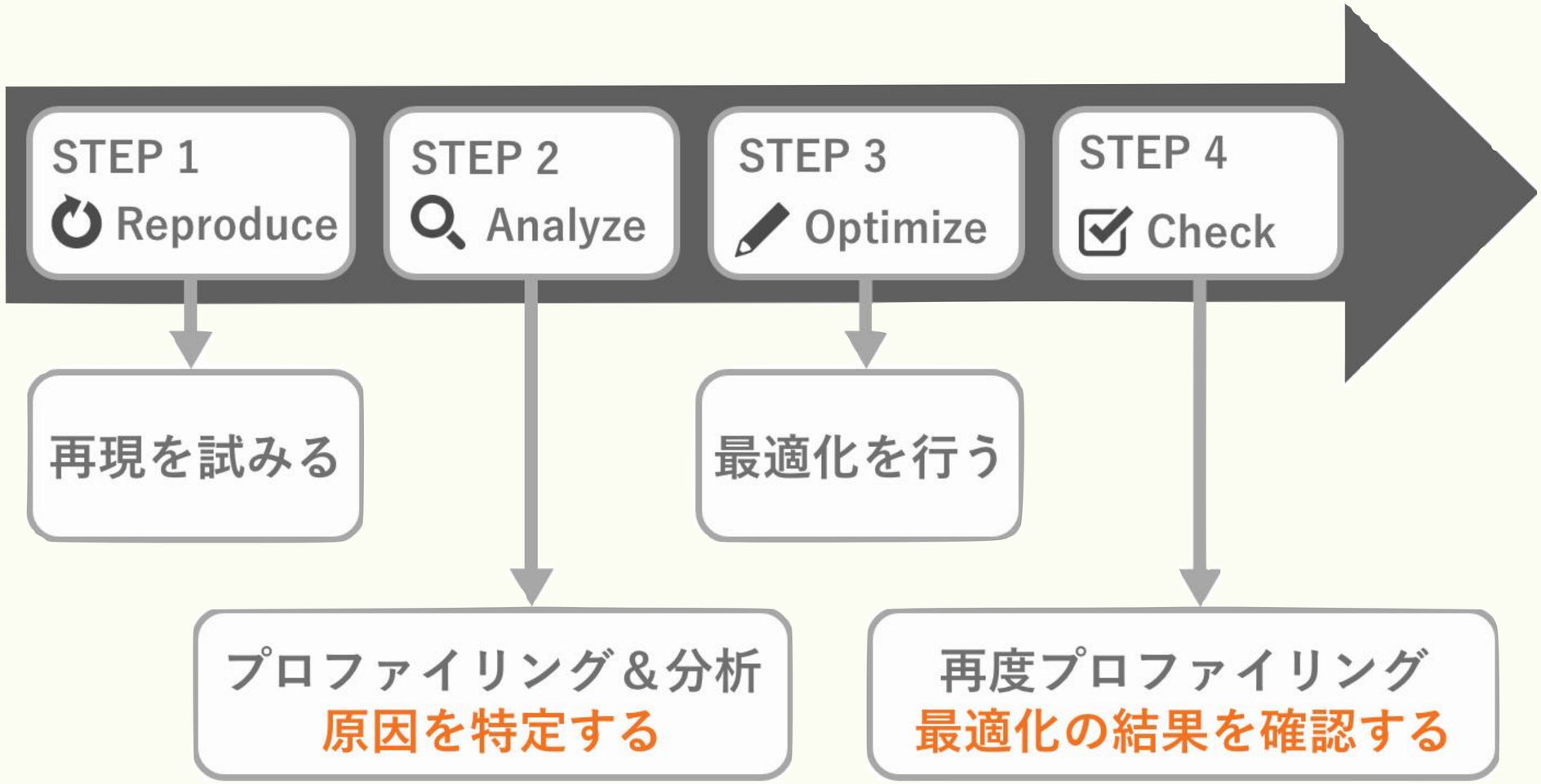
項目	要素
フレームレート	常時どのぐらいのフレームレートを目指すか
メモリ	どの画面でメモリが最大になるか試算し、限界値を決める
遷移時間	遷移時間待ちはどれぐらいが適切か
熱	連続プレイ X 時間で、どのぐらいの熱さまで許容できるか
バッテリー	連続プレイ X 時間で、どれぐらいのバッテリー消費が許容できるか

パフォチューの指標

- 最低限の動作 を保証する端末を決め、
  その端末で指標を満たすようにチューニングしていく
- 高フレームレートを目指すと、その分バッテリーを消費したり、
  全部を取って良い結果にすることは難しい

指標に合わせたバランスが大事

パフォチューの心構え


STEP 1
Reproduce
STEP 2
Analyze
STEP 3
Optimize
STEP 4
Check
再現を試みる
プロファイリング＆分析
原因を特定する
最適化を行う
再度プロファイリング
最適化の結果を確認する

性能低下の種類

- メモリ超過
- プログラムの実行エラー

- CPU や GPU の処理時間

メモリ超過

- メモリリーク
  - 同じ操作を繰り返した時にメモリ使用量が純増している
- メモリ使用量が単純に多い
  - リークしていない状態でメモリ使用量が多い

メモリリークの調査

- ツールを使用して原因を探っていく
  - Profiler (Memory)
  - Memory Profiler
  - Heap Explorer

メモリ削減

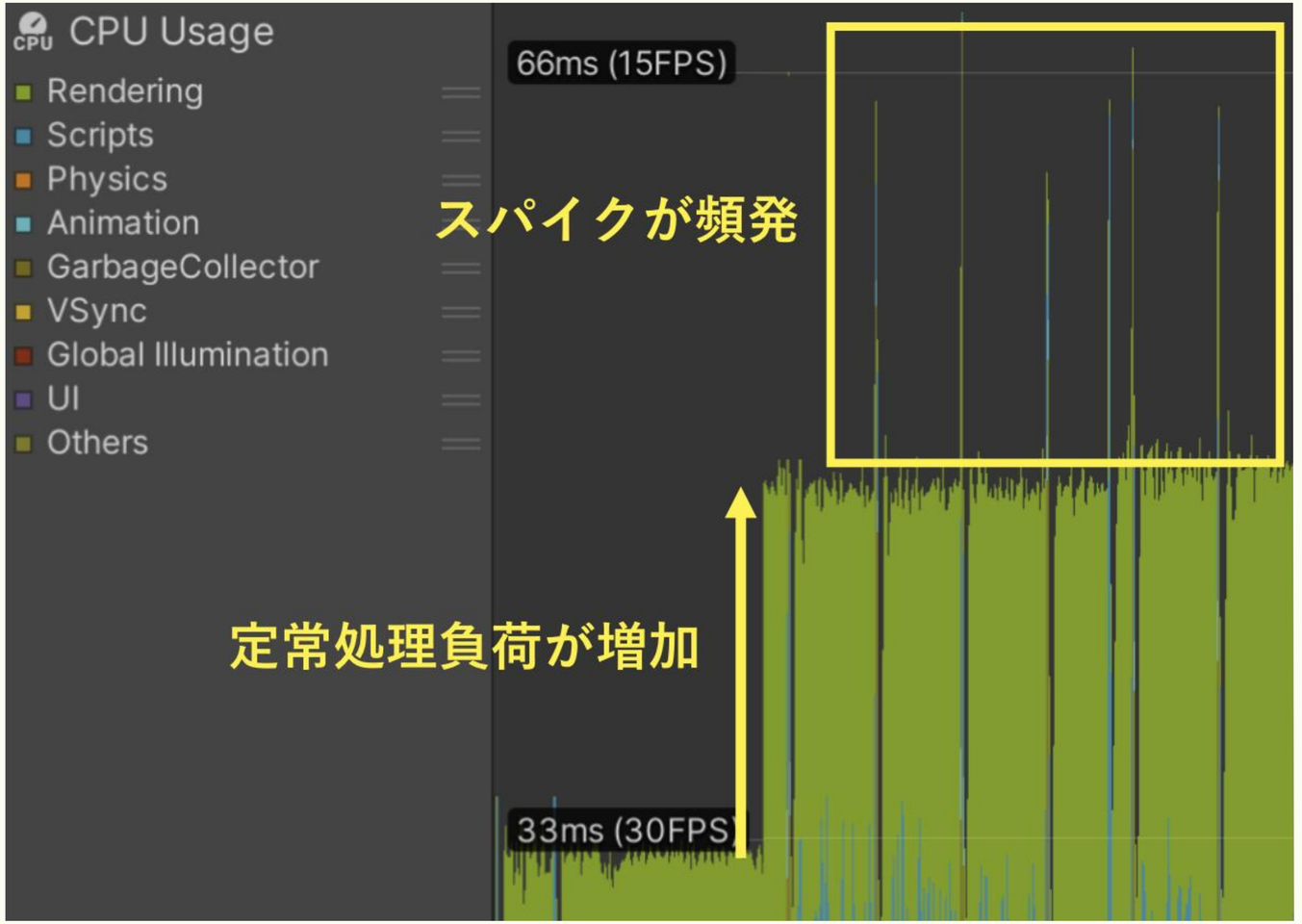
- 不要アセット調査
  - 現在のシーンに使用しないアセットがメモリにのっていないか
- 重複アセット調査
  - アセットバンドルの依存関係
- レギュレーション
  - テクスチャならサイズ・圧縮設定など

メモリ削減

- GC
  - GC.Allocによるメモリの断片化
- Other・プラグイン
  - 余分のメモリ確保している箇所がないかどうか
- 仕様
  - ロードタイミングの変更、ゲーム中のモデル数の制限など

処理落ちの原因切り分け

- 瞬間的な処理落ち
- 定常的な処理落ち



瞬間的な負荷を調査する

- GC によるスパイク
  - 毎フレームアロケートしている箇所
  - 大量のアロケートが発生している箇所
- 重い処理によるスパイク
  - Instantiate処理
  - 大量のオブジェクト、階層が深いオブジェクトのアクティブ切り替え

定常的な負荷を調査する

- 1フレーム内での処理を削減する
- CPU処理とGPU処理どちらがボトルネックなのかを切り分ける
- GPUバウンドの可能性の見つけ方
  - 画面の解像度を下げた際に処理負荷が劇的に改善する
  - Profilerで計測した際にGfx.WaitForPresentが存在する

CPUバウンド

- Profilerによる調査
- 特定のアルゴリズム（関数など）に処理負荷がかかっていないか

アルゴリズムの負荷に関しては、
アルゴリズム編の復習をぜひ

GPUバウンド

- Frame Debuggerによる調査
- 解像度設定が適切か
- 不要なオブジェクトがあるか
  - オクルージョンカリングも検討
- バッチングが適切か
  - 描画対象をまとめて描画する（ドローコールの削減）

GPUバウンド

- 描画するオブジェクトが多すぎないか
- 1オブジェクトの頂点数が多すぎないか
  - リダクション、LODを考える
- シンプルな Shaderに付け替えて処理負荷が改善するか
  - Shaderの処理を見直す

高速化へのTIPS

高速化へのTIPS

- ゲーム開発では、可読性を犠牲にしてでも性能を取る場合もある
- Allocation許容できるところでは可読性優先した方が便利なことも
- Allocationしても良いところとダメなところが正しく理解している状態でないとハンドリングしにくくなるので、プロジェクトの方針としてAllocationする書き方を最初から禁止するところもある

高速化へのTIPS

- たとえば・・・
  - LINQ
  - foreach
- https://sharplab.io/

実例紹介 - LINQ

```
public class C {
    public void M() {
        var array = new []{0, 1, 1, 2, 2};
        var targetValue = 1;
        var count = 0;
        foreach(var value in array)
        {
            if (value == targetValue)
            {
                count++;
            }
        }
    }
}
```

実例紹介 - LINQ

```
using System.Linq;

public class C {
    public void M() {
        var array = new []{0, 1, 1, 2, 2};
        var targetValue = 1;
        var count = array
                    .Count(x => x == targetValue);
    }
}
```

実例紹介 - foreach

```
public class C {
    public void M() {
        var array = new []{0, 1, 2};
        foreach(var value in array)
        {
        }
        for(var i = 0;i < array.Length; i++)
        {
            var value = array[i];
        }
    }
}
```

実例紹介 - foreach

```
using System.Collections.Generic;

public class C {
    public void M() {
        var list = new List<int>(){0, 1, 2};
        foreach(var value in list)
        {
        }
        for(var i = 0;i < list.Count; i++)
        {
            var value = list[i];
        }
    }
}
```

さいごに

さいごに

- ゲーム開発では、可読性を犠牲にしてでも性能を取る場合もある
- 行き過ぎた最適化で更新のしやすさが失われるのはダメ
  何事もバランスが大事
- 今回紹介したのは触りに過ぎないので、
  パフォーマンスチューニングバイブルを読み直してみてください！