



2020/11/11

**Start the announcement**

**2020/11/07**

**本城佳樹**

# 目次

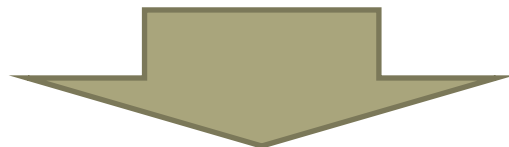
- (再掲) 中間発表の内容
  - テーマ
  - ゲームの説明
- 独自性
- こだわりのプログラム
- 開発スケジュール
- 使用素材・アセット
- まとめ

# 目次

- (再掲) 中間発表の内容
  - テーマ
  - ゲームの説明
- 独自性
- こだわりのプログラム
- 開発スケジュール
- 使用素材・アセット
- まとめ

# (再掲)何を作ろう？(1/2)

- 「あつめる」「大きくなる」は楽しい
- ある程度いい加減なゲーム
  - 「詰み」を許す
  - 何でもかんでも無茶苦茶を許す

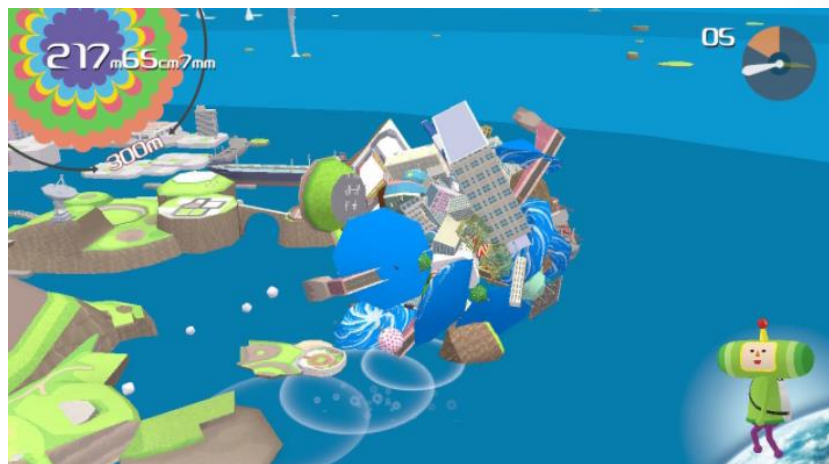


作りやすい2Dで  
「技術ドリブン」ではなく「作りたいもののドリブン」  
これでゲームを作ろう

# (再掲)何を作ろう？(2/2)

- 「あつめる」や「大きくなる」  
→ 塊魂・Hole.io
- できなかったことができるように
- 予想外の結果

楽しい・おもしろい



# 目次

- （再掲） 中間発表の内容
  - テーマ
  - **ゲームの説明**
- 独自性
- こだわりのプログラム
- 開発スケジュール
- 使用素材・アセット
- まとめ

# (再掲)作ったゲーム

## EverythingEater

ゲームシーン上のもの「**すべて**」食べられる  
食べることで**大きく**なっていく  
時間内に目標の大きさになることを目指す  
最終的にはゲームの**マップ**や**壁**まで……



# ゲームサンプル





# (再掲)苦勞したこと

- プレイヤの挙動
  - コライダ方向に入力を続けると反転を繰り返す
- 分裂・合体機能
- すべてを「たべる」ということ
  - 分裂したプレイヤーも食べたい
- カメラやUIが「どの」プレイヤを追うか？
  - プレイヤを食べられるようになった副作用
- 機能別にスクリプトを分けすぎた……
- 上手に設計しようとして未使用のスクリプトが生まれた

# 目次

- (再掲) 中間発表の内容
  - テーマ
  - ゲームの説明
- **独自性**
- こだわりのプログラム
- 開発スケジュール
- 使用素材・アセット
- まとめ

# 独自性 (1/3)

## 分裂

- スペースキーで分裂
  - 自機が増えることで「食べる」効率あっぷ！
  - いっぱい増えて楽しい



# 独自性 (2/3)

## 合体

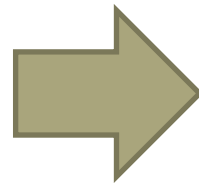
- L-SHIFTで合体
  - 増えた自機を一瞬で**吸収**
  - 一気に大きくなり**楽しい**



# 独自性 (3/3)

## すべてを「食べられる」

- ステージのアイテムだけでなくテキストも
- 分裂した自分自身さえも



「**食べられる**」



# 目次

- (再掲) 中間発表の内容
  - テーマ
  - ゲームの説明
- 独自性
- **こだわりのプログラム**
- 開発スケジュール
- 使用素材・アセット
- まとめ

# こだわりのプログラム(1/3)

## Playerクラス

シーン上の「一番大きいプレイヤー」を見つけるクラス

- Biggestプロパティ
- BiggestSizeプロパティ
- IsMainPlayerメソッド

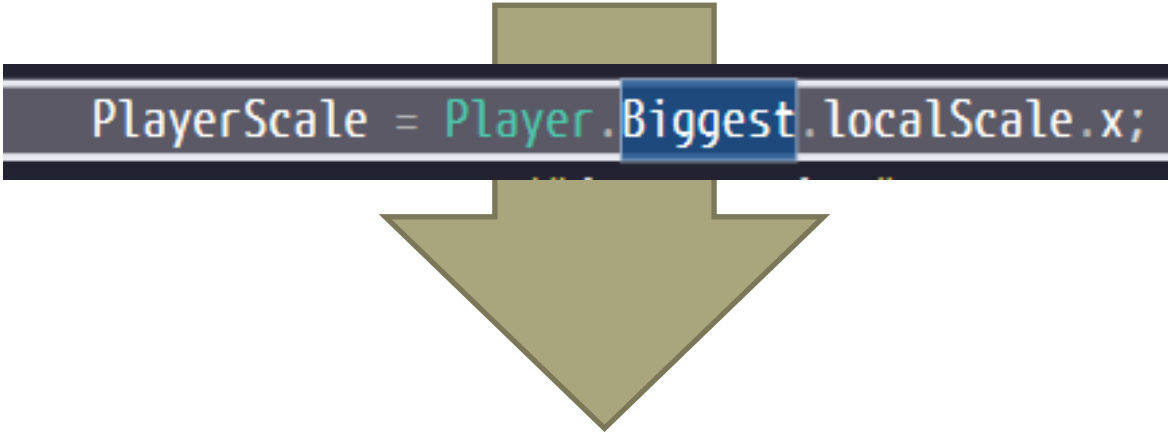
```
public static Transform Biggest
{
    get
    {
        var players = new List<GameObject>();
        players = GameObject.FindGameObjectsWithTag("Player").ToList();

        return players.OrderByDescending(p => p.transform.localScale.x).First().transform;
    }
}
```

# こだわりのプログラム(2/3)

## Biggestプロパティ

「一番大きいプレイヤー」のトランスフォームを返す



```
PlayerScale = Player.Biggest.localScale.x;
```

UI等で扱いやすくなる

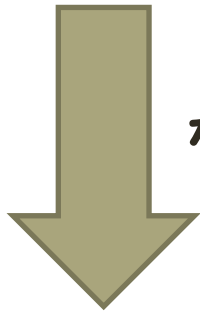


# こだわりのプログラム(2/3)

## GameDirector クラス

シングルトンパターンで実装

- ステージの情報（制限時間・目標サイズ）を返す
- **DontDestroyOnLoad()** シーンチェンジで消えない



なぜ??

ゲームオーバーシーン  
に  
情報を渡したいから

```
public void Awake()  
{  
    if (this != Instance)  
    {  
        Destroy(this.gameObject);  
        return;  
    }  
  
    DontDestroyOnLoad(this.gameObject);  
}
```

# 目次

- (再掲) 中間発表の内容
  - テーマ
  - ゲームの説明
- 独自性
- こだわりのプログラム
- **開発スケジュール**
- 使用素材・アセット
- まとめ

# 開発スケジュール

10月26日～11月12日 計18日 バージョン管理には「**Github**」利用

日時	行ったこと
26, 27日	ゲーム企画
28日～2日	ゲームの基本機能実装
3, 4日	「分裂」「合体」実装
5日～9日	シーン作成（タイトル、ゲームオーバー）
10日	チュートリアルステージ作成
11日	ゲームバランス調整
12日	発表用資料作成

# 目次

- (再掲) 中間発表の内容
  - テーマ
  - ゲームの説明
- 独自性
- こだわりのプログラム
- 開発スケジュール
- **使用アセット**
- まとめ

# 使用アセット

## Cinemachine

- プレイヤーがどれだけ分裂しても一つのカメラに収める
- 自然に追従する

## Post Processing Stack v2

- タイトルを「それっぽく」するために
- VignetteとGrainを使用

### Vignette

中心よりも周囲のほうが暗く

### Grain

ノイズ



# 目次

- (再掲) 中間発表の内容
  - テーマ
  - ゲームの説明
- 独自性
- こだわりのプログラム
- 開発スケジュール
- 使用アセット
- まとめ

# まとめ

- 「モチベーション」の話
  - 趣味ならばやりたいこと重視でよい
- プログラムの「設計」の難しさ・大切さ
  - 拡張性の高さ
  - 感覚的な話
  - SOLID原則
- 「完成させること」の大切さ
- 「企画」の大切さ

# まとめ

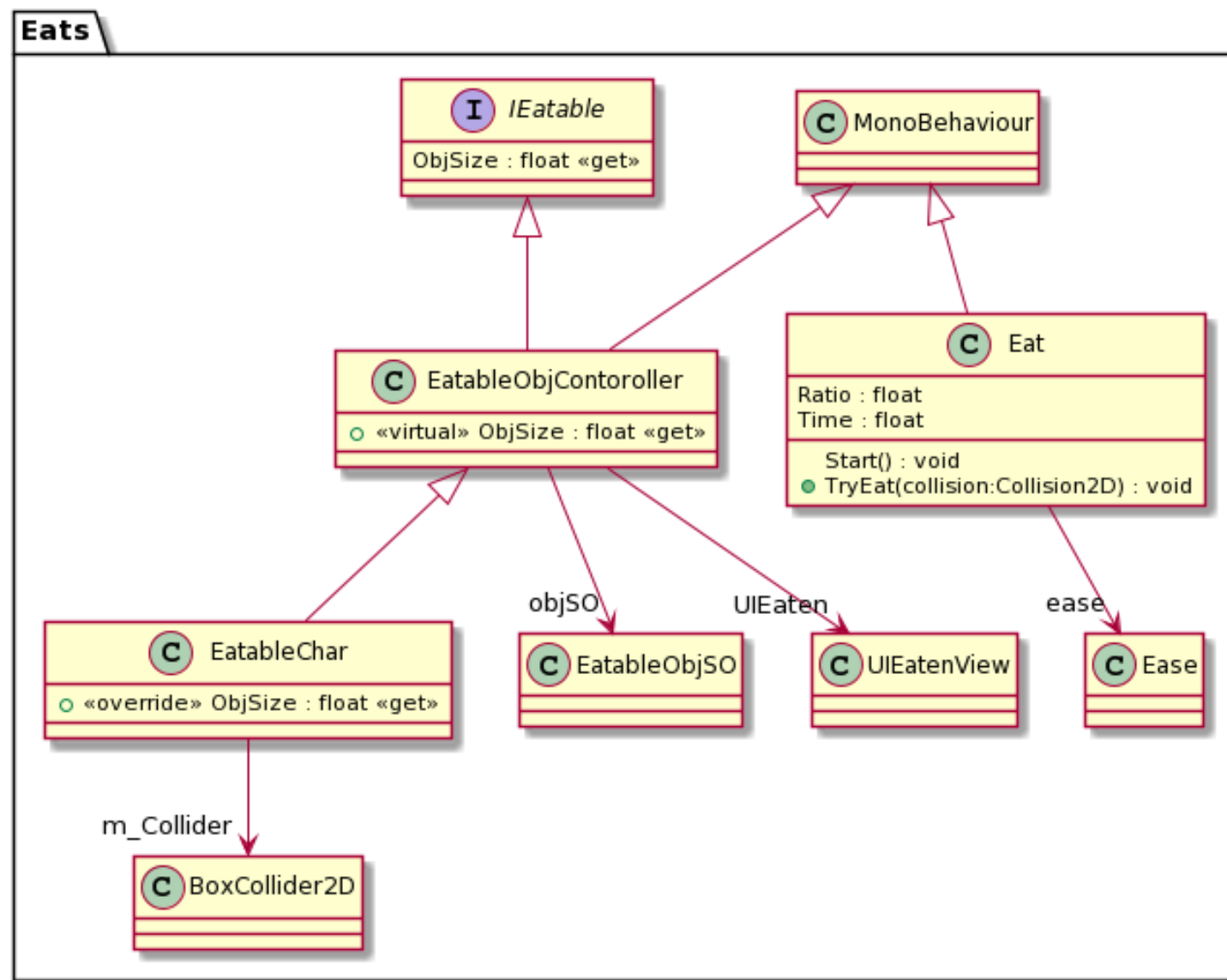
- **得意不得意・向き不向き**
  - 「やってみる」ことでわかる
- 世に出ている「**ゲームバランス**」の妙
  - パラメータ調整
  - レベルデザイン
  - 演出（SE・モーション）

**ゲームを作るのは楽しい！！**



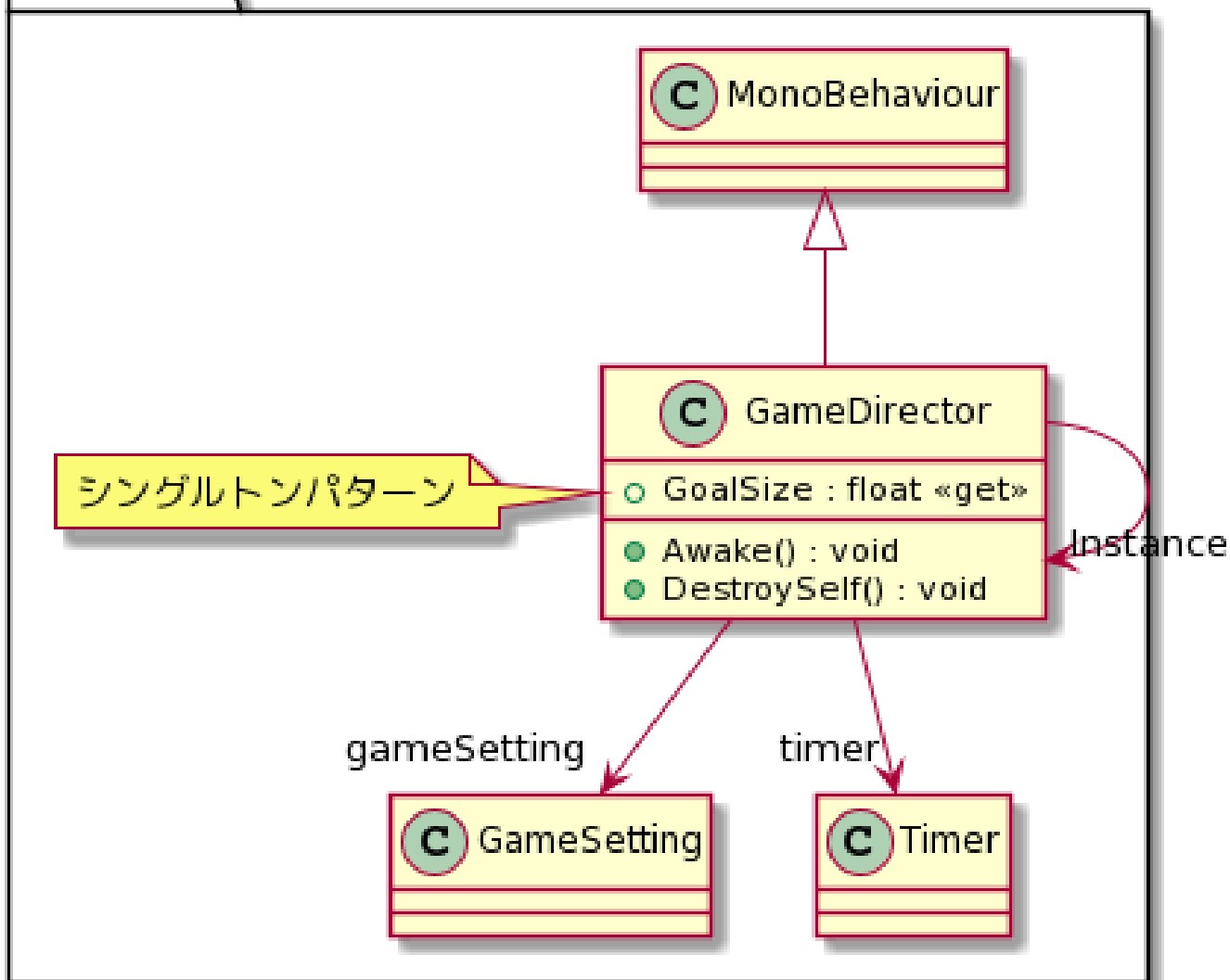
以上です

# クラス図



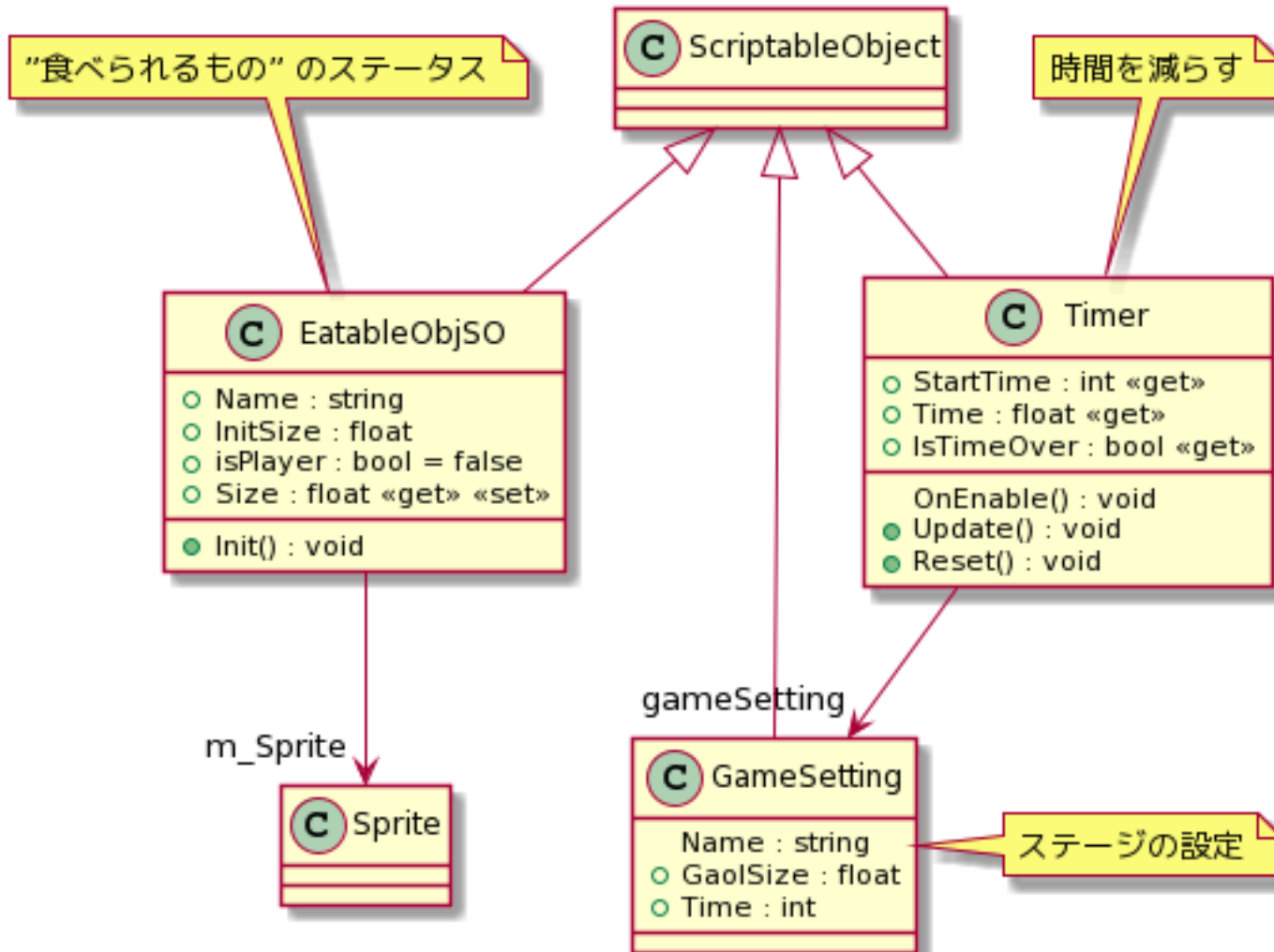
# クラス図

## Director



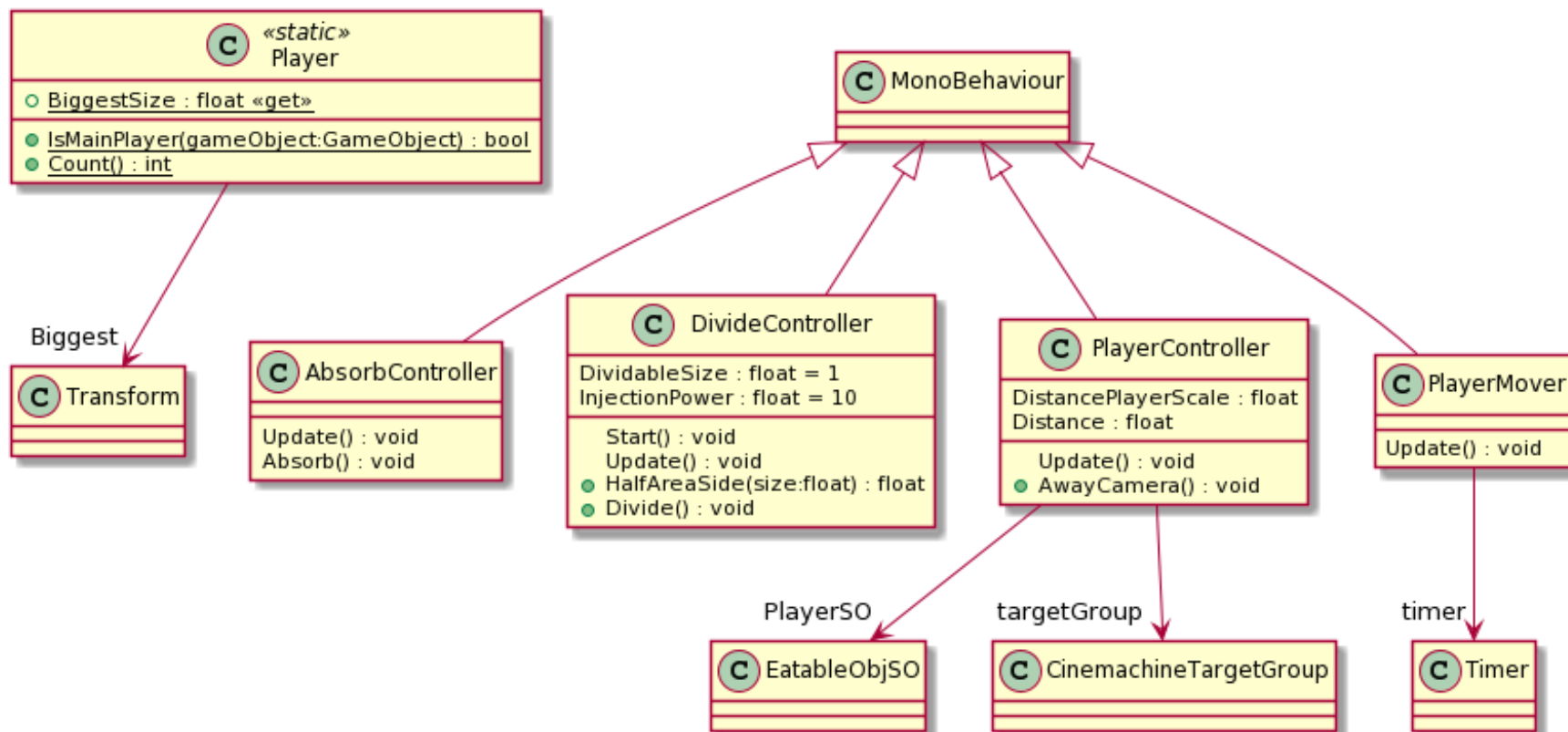
# クラス図

## ScriptableObjects

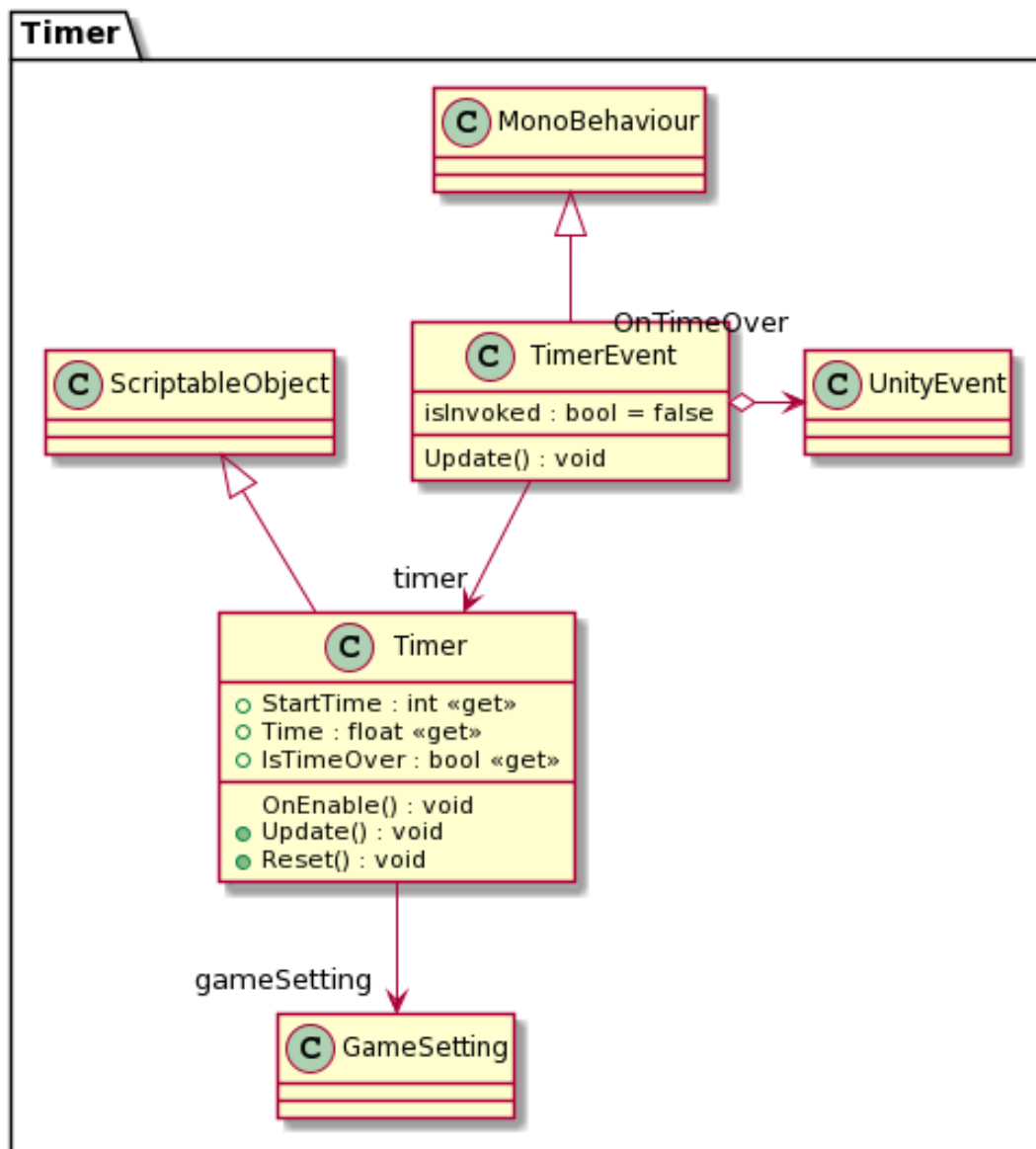


# クラス図

## Players



# クラス図



# やりたかったこと

- SEをつける
- 徹底したBUG取り
- プログラム設計しなおし
- ステージづくり
- クリア判定
- アニメーション
- エフェクト