ゲームのタスクシステム　導入編

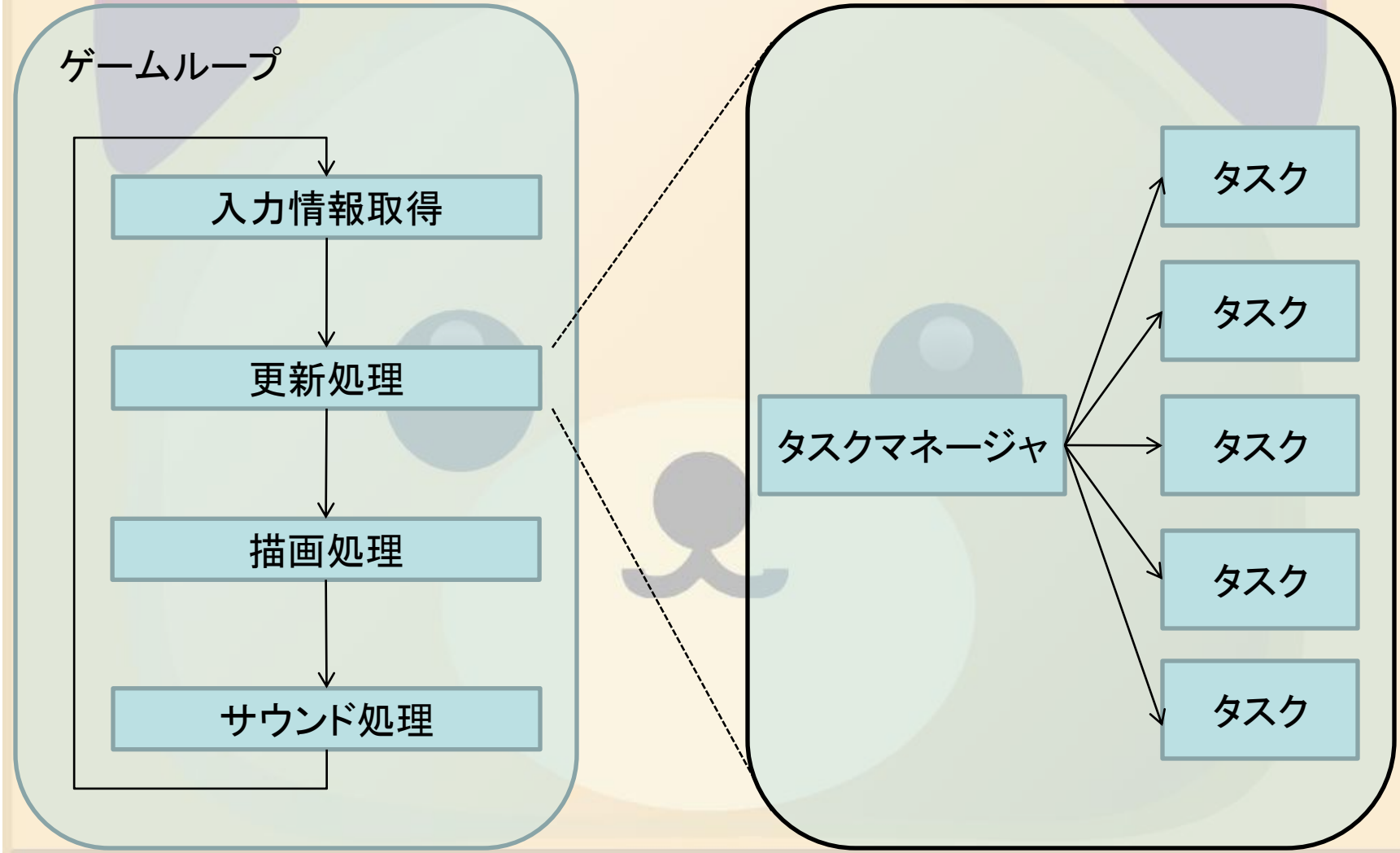
レベル2くまー

By keychan

アジェンダ

- ゲームタスクについて
- タスクオブジェクトの設計
- タスクマネージャの設計
- タスクオブジェクトの実装
- 最後に

ゲームタスクについて



ゲームタスクについて

- 当セッションにおけるゲームタスクの位置付け
  - ゲーム上で変化する/させる最も単純な処理
- ゲームタスクの基本的な粒度はオブジェクト単位
  - キャラクタ
  - BG
  - エフェクト
  - などなど…

タスクオブジェクトの設計

- 必要なメソッド
  - 更新処理
  - 描画処理

上記2点があれば、最低限の動作を行うことが可能

タスクオブジェクトの設計

```
abstract class Task
{
    // タスクの実行
    public abstract void execute();
    // タスクの描画
    public abstract void draw();
}
```

タスクオブジェクトの設計

- しかし、execute と draw のみのメソッドではいろいろ不便
  - タスクの処理順番（優先度）の決定
  - タスクの消滅タイミングはいつ？
  - タスクの描画順番の決定

タスクオブジェクトの設計

優先度を表すプロパティを追加

```
abstract class Task
{
    // タスクの優先度
    protected float priority;
    public float Priority { get; set; }
    // タスクの実行
    public abstract void execute();
    // タスクの描画
    public abstract void draw();
}
```

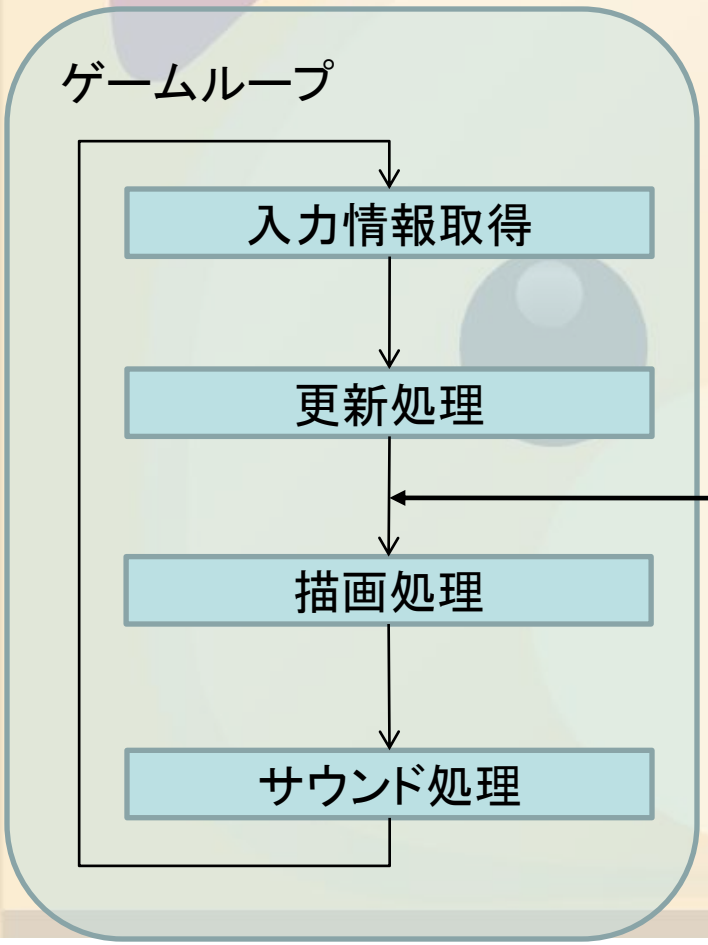
タスクオブジェクトの設計

消滅タイミングを知らせるプロパティを追加

```
abstract class Task
{
    // タスクの優先度
    protected float priority;
    public float Priority { get; set; }
    // タスクが生きているか
    protected bool exist;
    public bool Exist { get; set; }
    // タスクの実行
    public abstract void execute();
    // タスクの描画
    public abstract void draw();
}
```

タスクオブジェクトの設計

タスクの描画順番の決定



全タスクの更新が完了しないと描画順は決定しない

更新処理と描画処理の間に入れる

更新処理優先度と描画処理優先度は同期しない為、描画用タスク配列を用意

タスクオブジェクトの設計

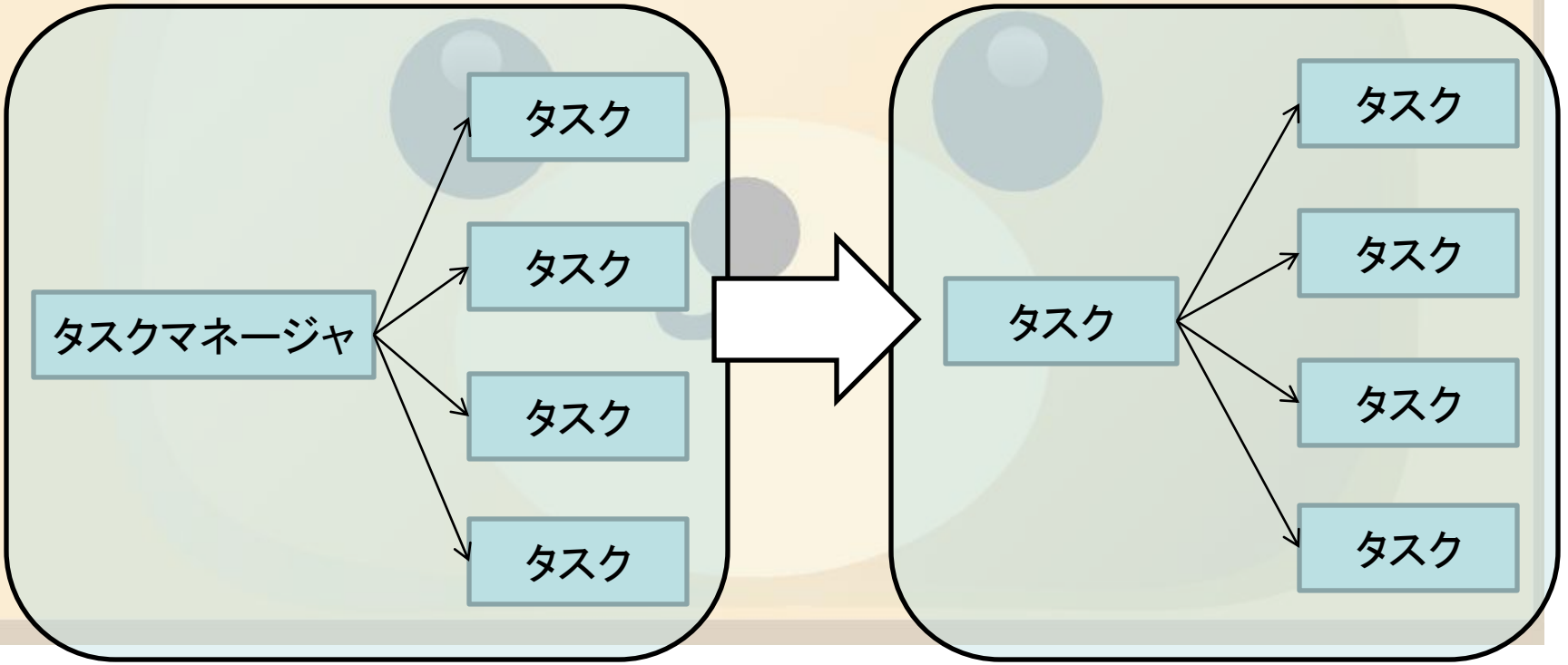
描画優先度を決定するためのプロパティを追加

```
abstract class Task
{
    // タスクの優先度
    protected float priority;
    public float Priority { get; set; }
    // タスクが生きているか
    protected bool exist;
    public bool Exist { get; set; }
    // タスクの描画オブジェクト
    protected IDrawObject drawObject;
    public IDrawObject DrawObject { get; }
    // タスクの実行
    public abstract void execute();
    // タスクの描画
    public abstract void draw();
}
```

```
Interface IDrawObject
{
    // 描画優先度を決定するため
    // に必要な情報を取得する
    // メソッドを用意
}
```

タスクマネージャの設計

- タスクを管理するタスクマネージャもタスクの一つであると考える
  - 階層構造をもつタスクを構築すればよい



タスクマネージャの設計

階層構造を持てるようにプロパティを追加

```
abstract class Task
{
    // 子タスクリスト
    protected List<Task> childTask;
    // タスクの優先度
    protected float priority;
    public float Priority { get; set; }
    // タスクが生きているか
    protected bool exist;
    public bool Exist { get; set; }
    // タスクの描画オブジェクト
    protected IDrawObject drawObject;
    public IDrawObject DrawObject { get; }

    // タスクの実行
    public abstract void execute();
    // タスクの描画
    public abstract void draw();
}
```

タスクオブジェクトの実装

- タスクオブジェクトの設計ができたので実装を軽くしてみます
  - シーンタスク
  - キャラクタタスク

最後に

- ゲームタスクはあらゆるジャンルのゲームで採用できるフレームワークです
- 当セッションではプラットフォームに依存しない極めて簡単なタスクシステムの一例をご紹介しました
- これが正解！というタスクシステムは当然ありません
- ゲームタスクの概念をご理解頂ければ幸いです