

Space Wandering

宇宙に浮かぶミニゲームを巡りメダルを集めるゲーム

つまみ 淵野アタリ ちくわぶ

分担作業



Model担当
使用する部品の親クラスの作成
ミニゲームの制作



View担当
ロケットや星などの部品の制作
ミニゲームの制作



Controller担当
操作のチューニング
操作に使用するデータ構造の整備

分担作業



Model担当
使用する部品の親クラスの作成
ミニゲームの制作



View担当
ロケットや星などの部品の制作
ミニゲームの制作



Controller担当
操作のチューニング
操作に使用するデータ構造の整備

基本はこのように分かれていたが、GitHubのIssues機能で手が空いた人にどんどん仕事を割り振った

グループ制作での工夫点

GitHubの使用

The screenshot shows a GitHub repository page for 'medipro-game'. The repository is private. The main navigation bar includes links for Pull requests, Issues (22), Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation, there are buttons for 'main' (selected), '5 branches', '0 tags', 'Go to file', 'Add file', and 'Code'. The commit history is displayed in a table:

Commit	Message	Date
Merge pull request #258 from GalaxyExpress ...	4d8e233 5 hours ago	385 commits
docs/javadoc	docs: Javadocを更新	2 months ago
resource	feat:GalaxyExpressをとりあえず遊べるようにしたよ	6 hours ago
src/net/trpfrog/medipro_game	feat:MovableSymbolが書き換えられている	5 hours ago
LICENSE	Initial commit	2 months ago
README.md	docs: README.mdの更新	last month

At the bottom of the commit list, there is a single row for 'README.md' with an edit icon.

コードをオンラインで共有できる

GitHubの使用

The screenshot shows the GitHub Issues interface. On the left, a list of 22 open issues is displayed, each with a title, a small icon, and labels. On the right, a specific issue is expanded, showing its details and a comment thread.

Issues List:

- ① 口ケットの無敵状態から通常状態に遷移するのをクロスフェードにしてほしい (easy issue improvement) #260 opened 5 hours ago by Alpha release
- ① SpaceGameでクリックしたところにキラキラが残るときれいそう (easy issue new feature) #259 opened 5 hours ago by Alpha release
- ① 画面サイズの調整 (improvement opinion wanted) #255 opened 6 hours ago by Alpha release
- ① ヘルプ画面を作る (Priority: high easy issue new feature) #254 opened 6 hours ago by Alpha release
- ① ヘルプの星が欲しい (easy issue new feature) #252 opened 6 hours ago by Alpha release
- ① EventStarの画像を作る (easy issue) #251 opened 6 hours ago by Alpha release
- ① メダル獲得画面を作る (new feature) #250 opened 6 hours ago by Alpha release
- ① 包含しているかという厳しめの当たり判定が欲しい (easy issue new feature) #241 opened 19 hours ago by
- ① FieldMap上のオブジェクトを移動できるようにしたい (improvement) #219 opened 3 days ago by
- ① rangeSymbolStreamのアルゴリズムの変更 (improvement opinion wanted) #206 opened 3 days ago by
- ① ミニゲームのリザルト画面をもう少し詳細に作り込めるようにしたい (easy issue new feature) #205 opened 4 days ago by Alpha release
- ① ミニゲームにクリア条件を持たせたい (Priority: high) #201 opened 4 days ago by Alpha release

Expanded Issue #83:

Title: SpaceViewが呼ばれたときのPaintComponentの挙動がおかしい #83
Status: Closed
Comments:

- commented 11 days ago: これだけだと原因がよく分からないのでViewのコードを貼ってみてもらえますか？
- commented 11 days ago: repaintのタイマーを
- private Timer timer = new Timer(10, e -> repaint());
とした上で、

```
@Override  
public void suspend() {  
    timer.stop();  
}  
  
@Override  
public void resume() {  
    timer.start();  
}
```
- commented 9 days ago: をsuspend/resumeに追加すると直ると思います。
- commented 9 days ago: 解決しました！

Action Buttons: (closed this 9 days ago)

Issues機能でタスク管理と意見交換ができる

GitHubの使用

The screenshot shows a GitHub interface for managing a milestone titled "Alpha release". At the top, there are two tabs: "Labels" and "Milestones", with "Milestones" being the active tab. Below the title, it says "Due by January 21, 2021 78% complete". A note in Japanese below the due date reads "発表できる段階に持ち込みます". The main area lists issues under the milestone:

- ⓘ 12 Open ✓ 43 Closed
- ⓘ どんな操作感にするか opinion wanted
#114 opened 5 days ago by [redacted]
- ⓘ 頃石オブジェクトがSpaceGameに欲しい new feature
#154 opened 5 days ago by [redacted]
- ⓘ ミニゲームにクリア条件を持たせたい Priority: high
#201 opened 4 days ago by [redacted]
- ⓘ メダル獲得画面を作る new feature
#250 opened 6 hours ago by [redacted]
- ⓘ EventStarの画像を作る
#251 opened 6 hours ago by [redacted]
- ⓘ ミニゲームのリザルト画面をもう少し詳細に作り込めるようにしたい easy issue new feature
#205 opened 4 days ago by [redacted]

Milestones機能で締切の管理ができる

GitHubの使用

Update README.md #188

Closed wants to merge 1 commit into `main` from `[REDACTED]-patch-1`

Conversation 1 Commits 1 Checks 0 Files changes

commented 4 days ago マッチョです

commented 4 days ago 本件はこれでクローズとします.

closed this 4 days ago

deleted the `[REDACTED]-patch-1` branch 4 days ago

Merged fix: マウスでの旋回がヌルヌルになった #189
merged 2 commits into `main` from `issue#187` 4 days ago

requested changes 4 days ago

`src/net//medipro_game/space/SpaceController.java` Outdated

```
... ... @@ -21,16 +23,17 @@
21 23     private double acceleration;
22 24     private int fps, spf;
23 25     private boolean isUpperAngle;
26 +     private Point p;
```

4 days ago `pointerLocation`とか`rocketDestination`とかそんな感じの名前だとたすかります.....

4 days ago Author すまねえ.....

Reply... Resolve conversation

fix: Pointの変数名変えた c7014f4

requested a review from 4 days ago

approved these changes 4 days ago

left a comment

merged commit `db3dddb` into `main` 4 days ago

Revert

PullRequest機能で事前にコードをチェックできる
→ 重大なミスでコードが消し飛ぶことがない！

Discordの使用

meditのプロ

2021/01/19 あ、明日ぐらいに頑張ってデジタル数字のフォント突っ込もうと考えています
できるなら自分でやってもいいよ

2021/01/19 いいですね

2021/01/19 座して待つ...

2021/01/19 資料のプロに上がってるやつフリーだから入れちはい、メインの部分ができないとフォントも活用

2021/01/19 やべ～、htmlの埋め込み忘れとる
これ道中に変数あってもいけんだっけ

2021/01/19 あ、drawStringはhtml使えなかったはず

2021/01/19 クソ～

2021/01/19 うみやー

IntelliJ IDEAをプレイ中

IntelliJ IDEAをプレイ中

画面共有しています

```
if(!mouseState.isWheeled()) {  
    if(keyState.isPressed(KeyEvent.VK_Z) {  
        // .....  
    }  
    if(keyState.isPressed(KeyEvent.VK_X) {  
        // .....  
    }  
} else {  
    // .....  
}
```

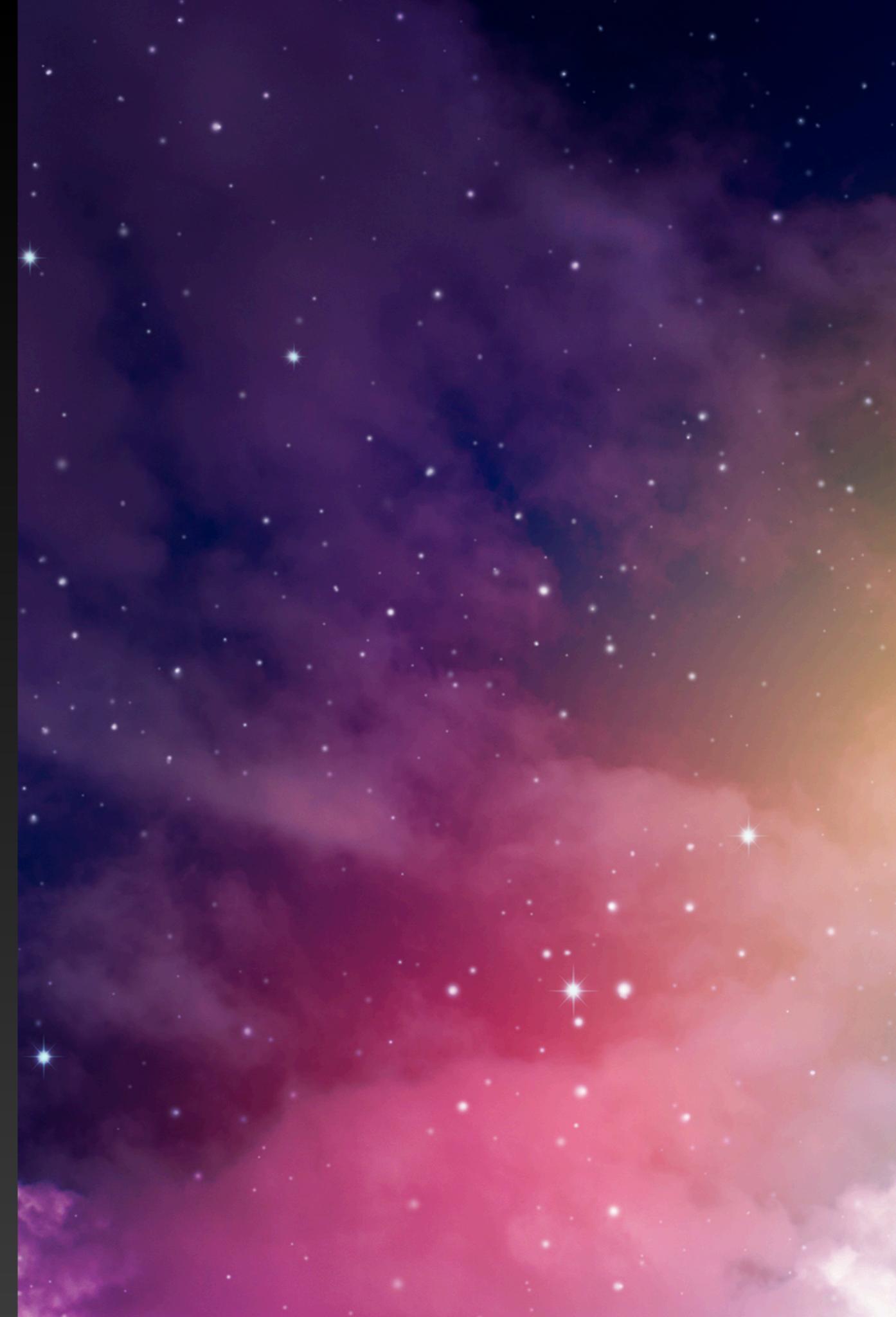
あーそういうことか、あのコードの意味をようやく理解した
確かにああ書きたくなるな

返し縫いみたいな順のコードになるの嫌だなってこうしたけど2回同じ条件式置くのもたいがいだな

あれ、そうなるとこのコードだとまずい？
マウスぐるぐるしてるとキーが優先にならない

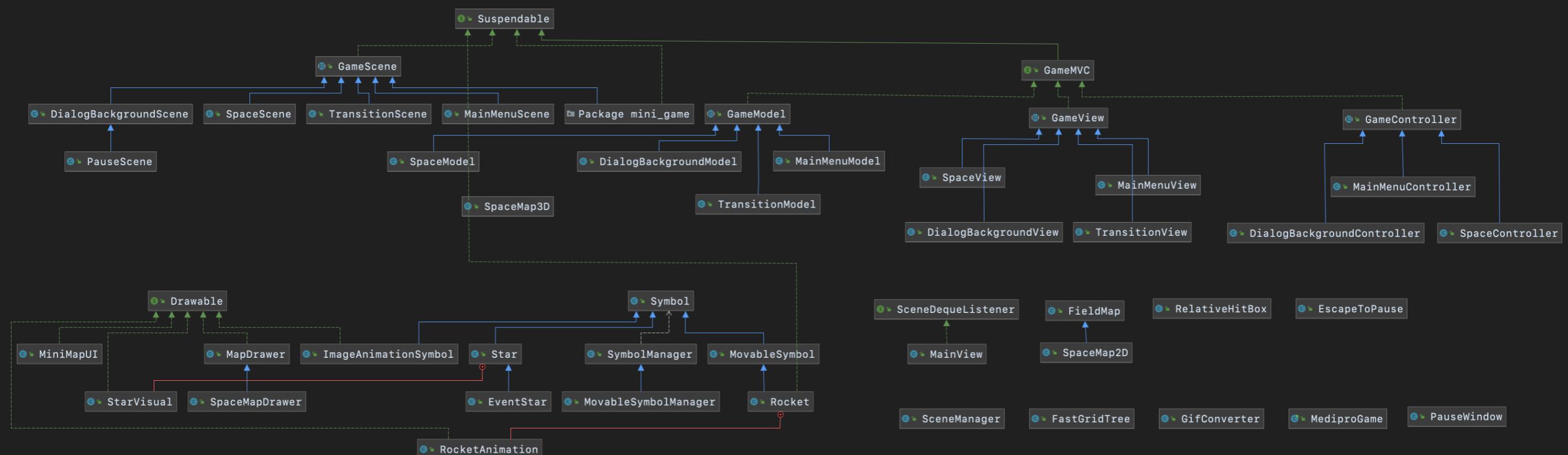
会話にコードを埋めめたり、通話ができたり、
画面共有ができたり、いろいろ揃っていて便利

プログラム 構成

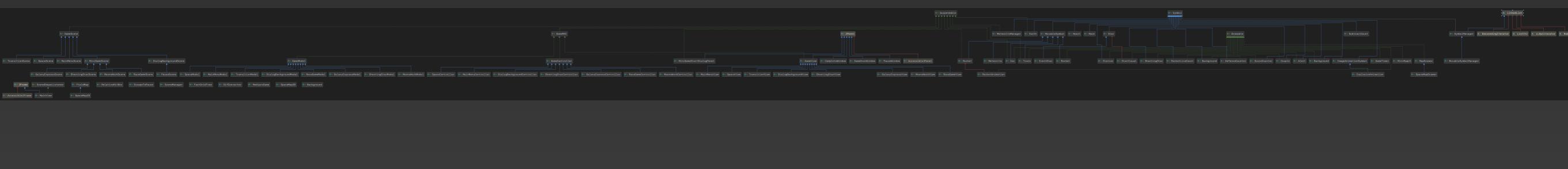


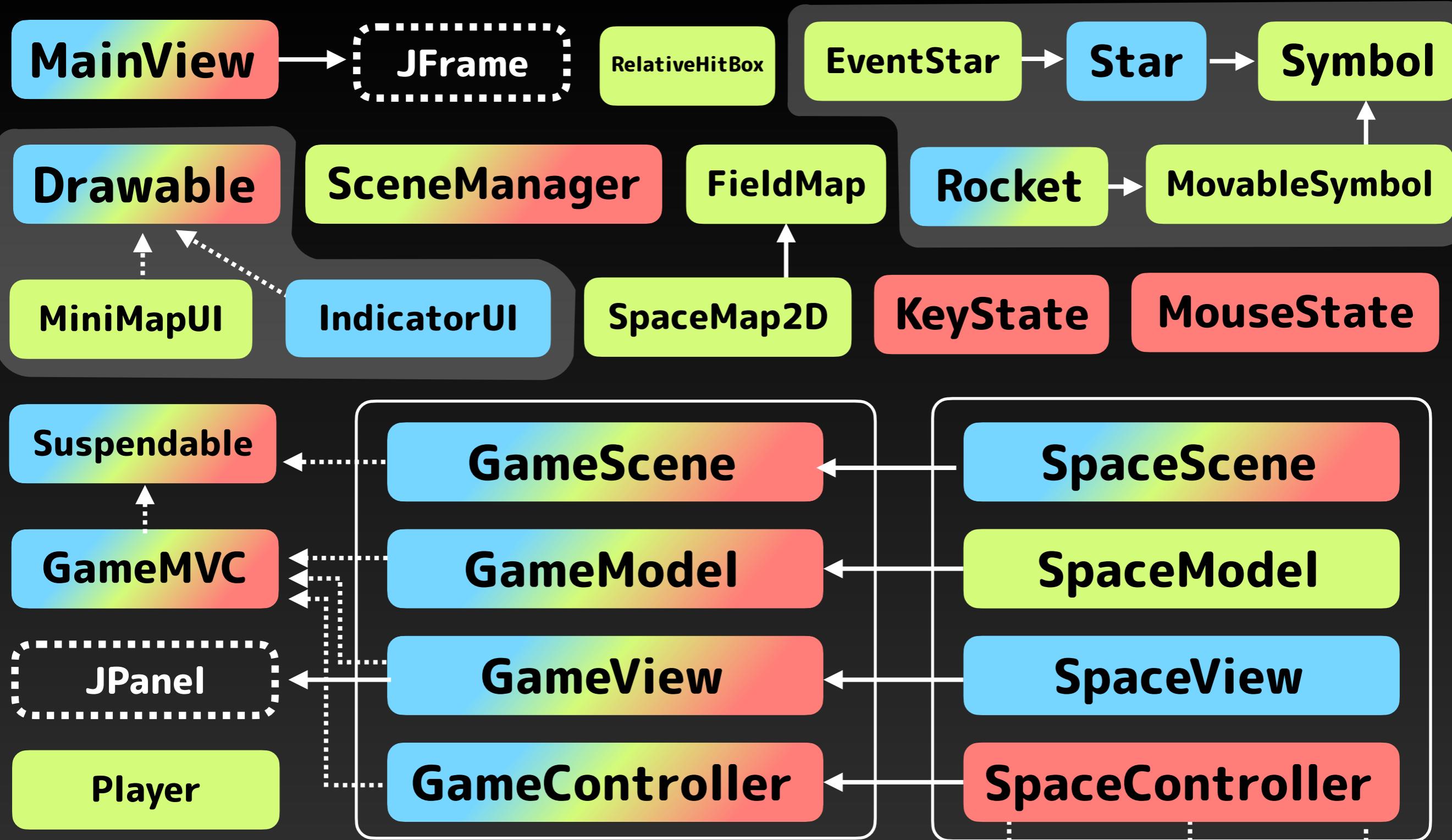
プログラム構成

ミニゲームを除いたクラス図



全体





プログラム構成
(メイン部分)

つ ち ア
作業分担

MouseWheel
Listener
Key
Listener
Mouse
Listener

Model



シーン追加の仕組み

GameScene

GameModel

GameView

GameController

MVCをまとめて管理するGameSceneクラスを用意

ゲーム本編

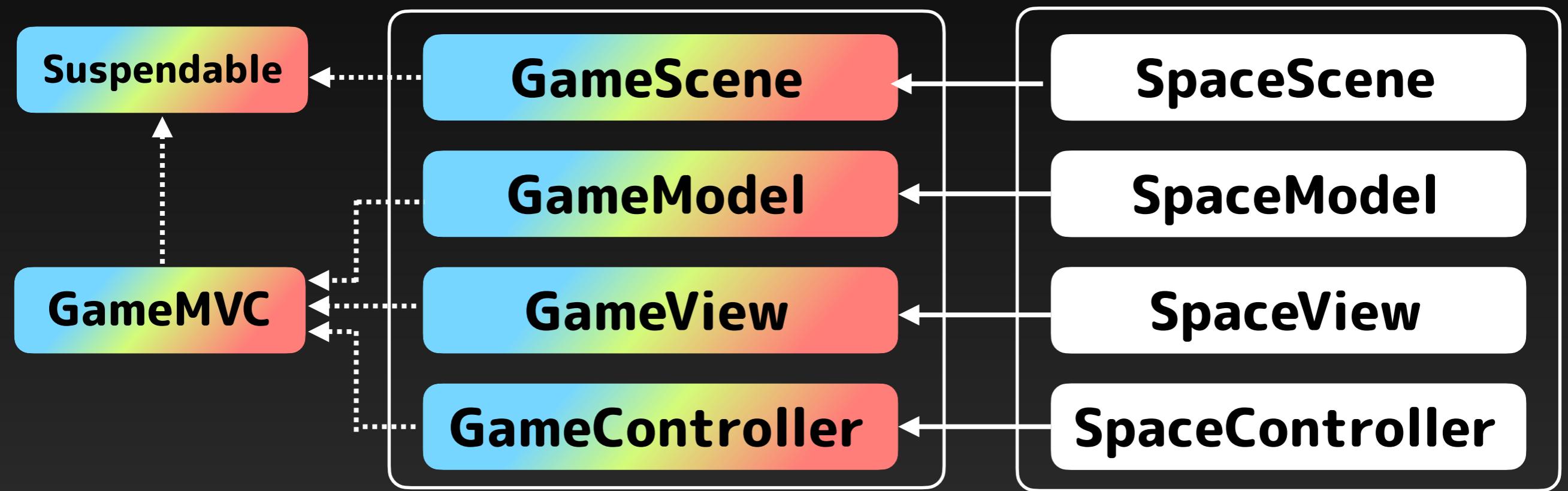
メインメニュー

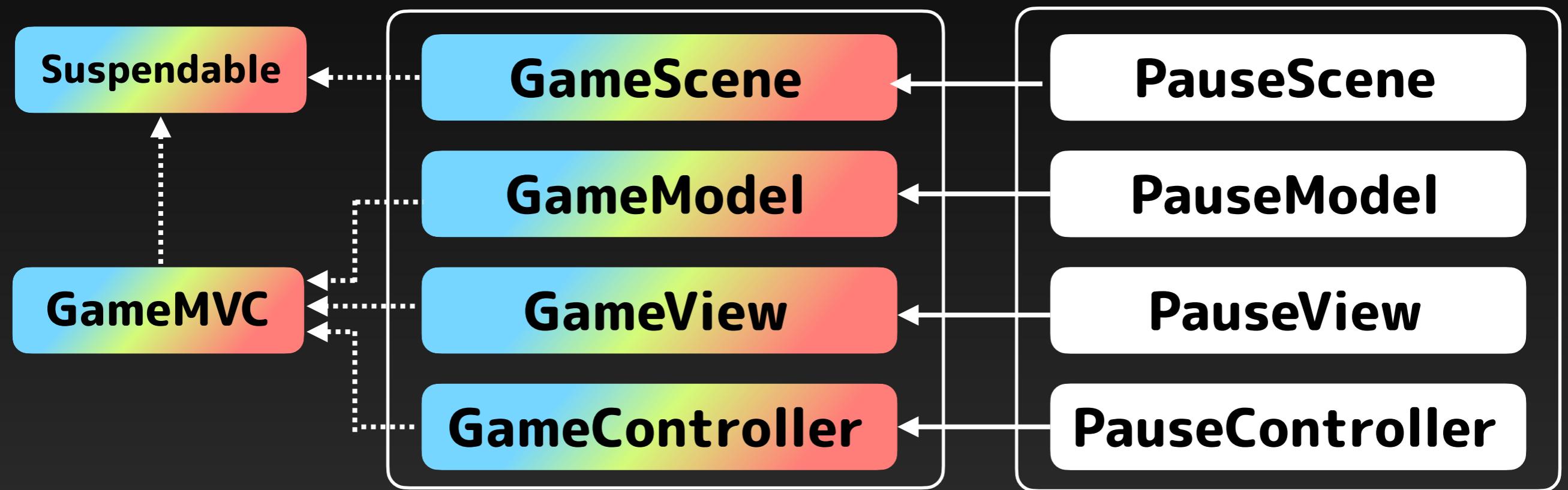
ポーズ画面

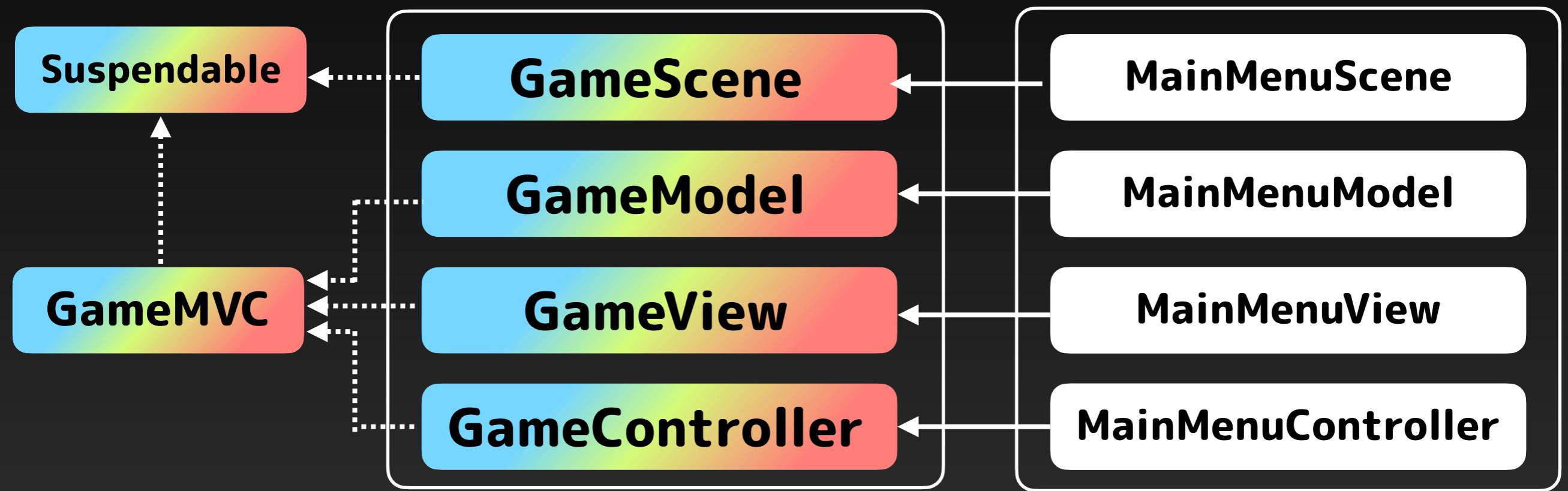
シーン遷移
アニメーション

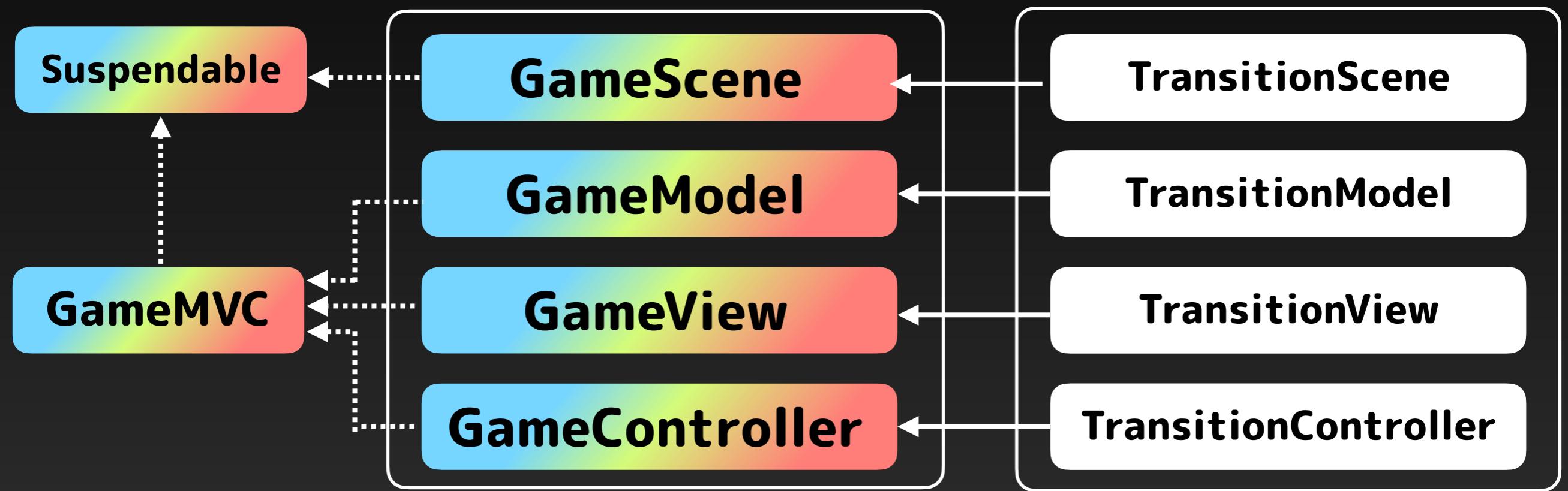
ミニゲーム

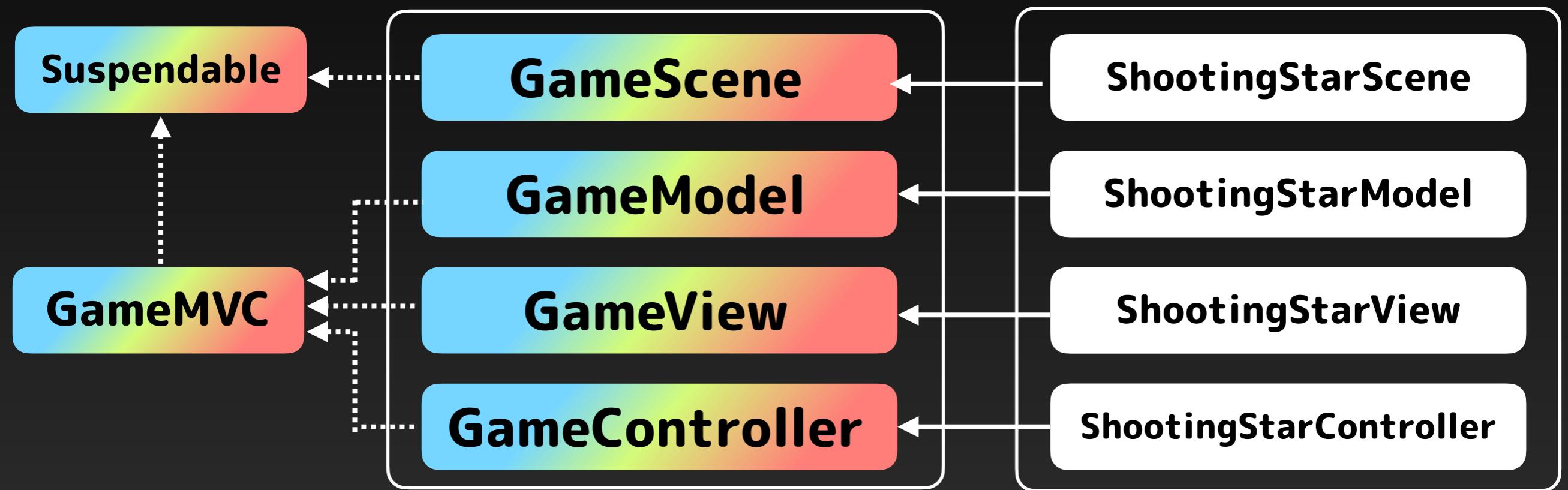
リザルト画面

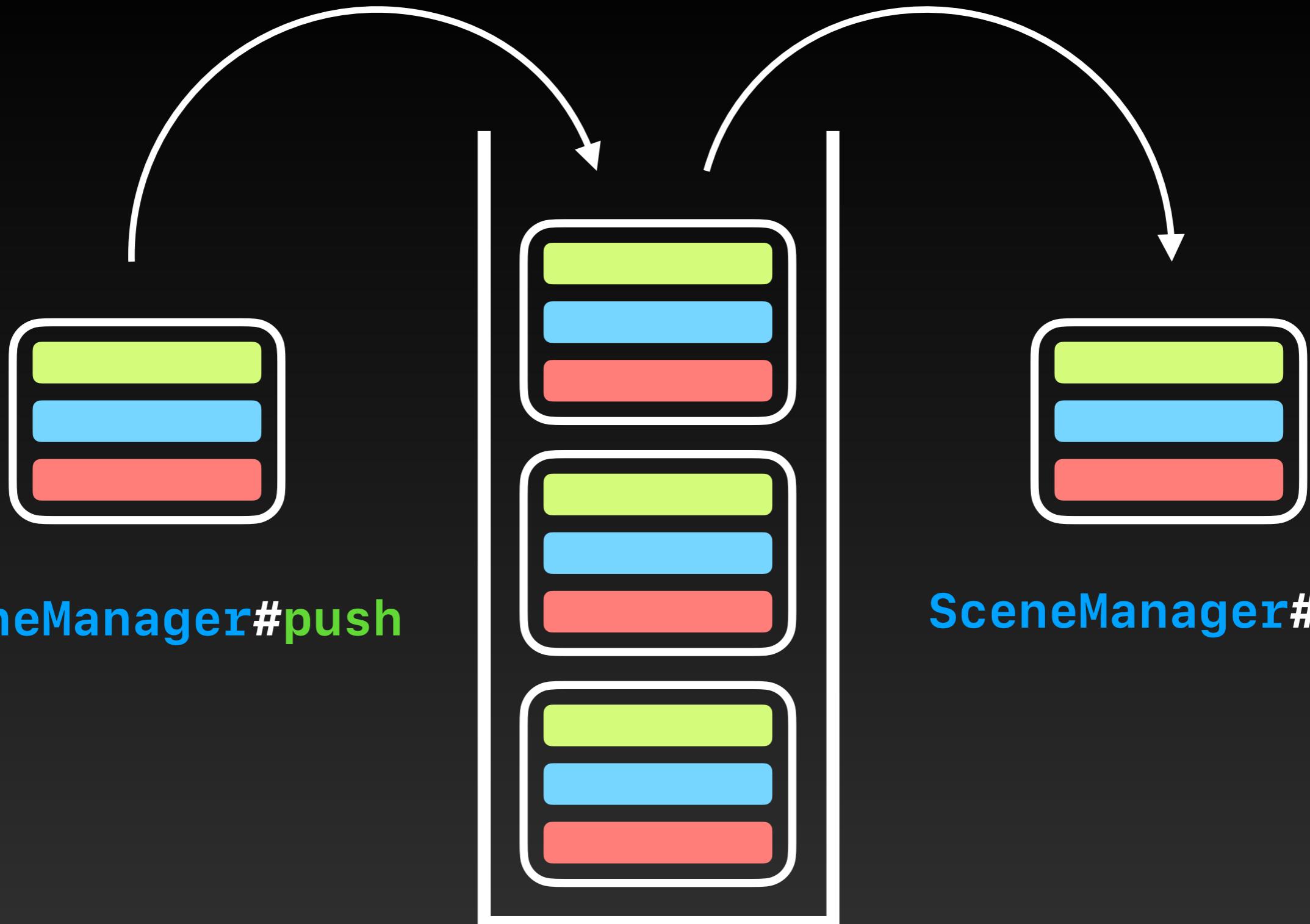




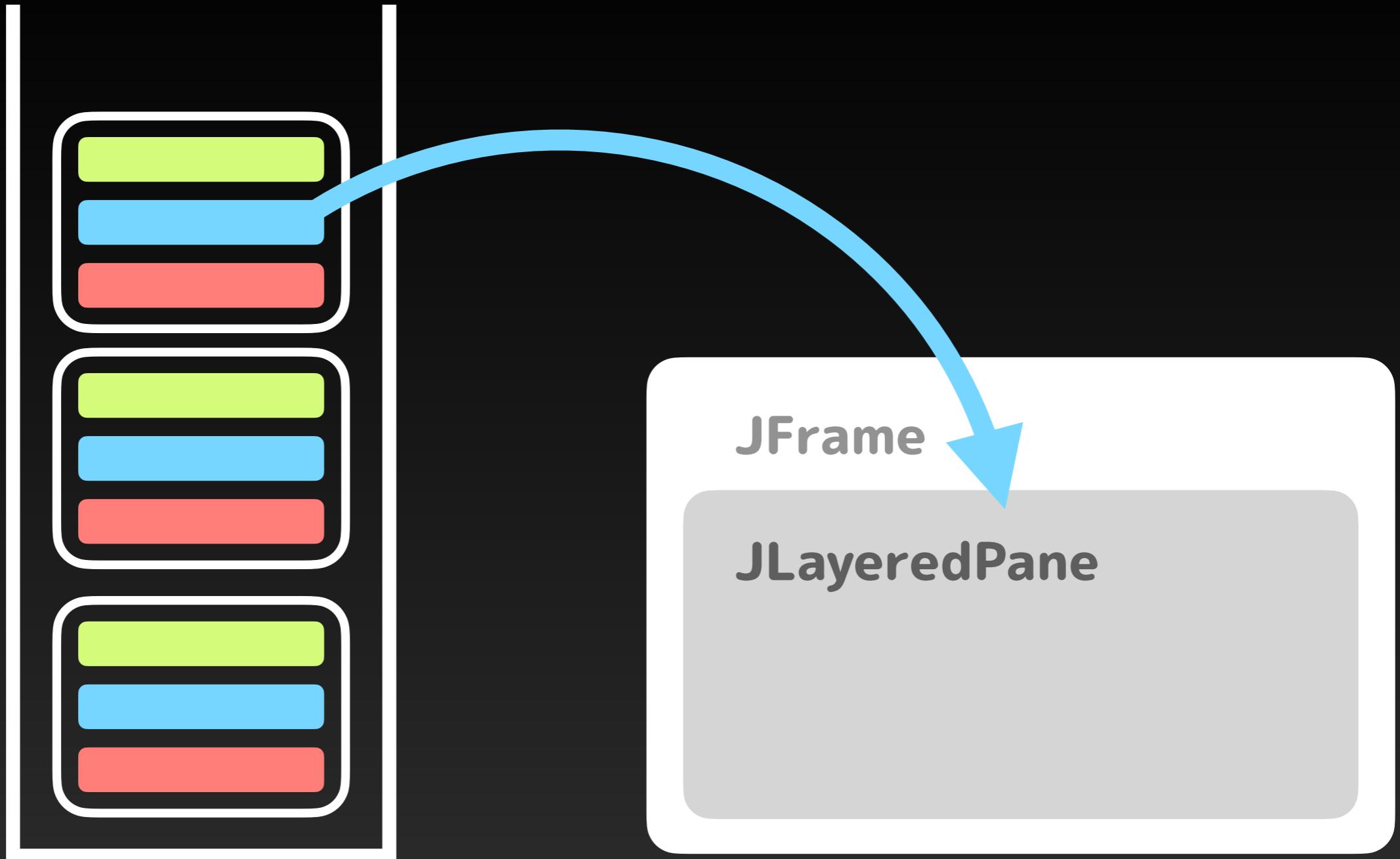








さらにGameSceneをスタックで管理する
SceneManagerクラスを用意



SceneManager

MainViewはSceneManagerの上部から得られる
GameSceneのViewを最上部に追加

MainView

画面に現れる
オブジェクトのクラス

画面に現れるオブジェクト

```
public class Symbol {  
    // オブジェクトの描画クラス  
    private Drawable drawer;  
  
    // 座標  
    private Point2D point;  
  
    // 角度  
    private double angleDegrees;  
  
    // 当たり判定  
    private RelativeHitBox hitBox;
```

画面に現れるオブジェクト

```
public class Symbol {  
    // オブジェクトの描画クラス  
    private Drawable drawer;  
  
    // 座標  
    private Point2D point;  
  
    // 角度  
    private double angleDegrees;  
  
    // 当たり判定  
    private RelativeHitBox hitBox;
```

描画用クラス

```
public interface Drawable {  
    public void draw(Graphics2D g);  
}
```

描画用メソッドを1つ持つインターフェース

描画用クラス（使用例）

```
private Symbol rocket;

@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D) g;
    rocket.getDrawer().draw(g2);
}
```

drawを呼び出すことでオブジェクトを描画できる

画面に現れるオブジェクト

```
public class Symbol {  
    // オブジェクトの描画クラス  
    private Drawable drawer;  
  
    // 座標  
    private Point2D point;  
  
    // 角度  
    private double angleDegrees;  
  
    // 当たり判定  
    private RelativeHitBox hitBox;
```

画面に現れるオブジェクト

```
public class Symbol {  
    // オブジェクトの描画クラス  
    private Drawable drawer;  
  
    // 座標  
    private Point2D point;  
  
    // 角度  
    private double angleDegrees;  
  
    // 当たり判定  
    private RelativeHitBox hitBox;
```

画面に現れるオブジェクト

```
public class Symbol {  
    // オブジェクトの描画クラス  
    private Drawable drawer;  
  
    // 座標  
    private Point2D point;  
  
    // 角度  
    private double angleDegrees;  
  
    //当たり判定  
    private RelativeHitBox hitBox;
```

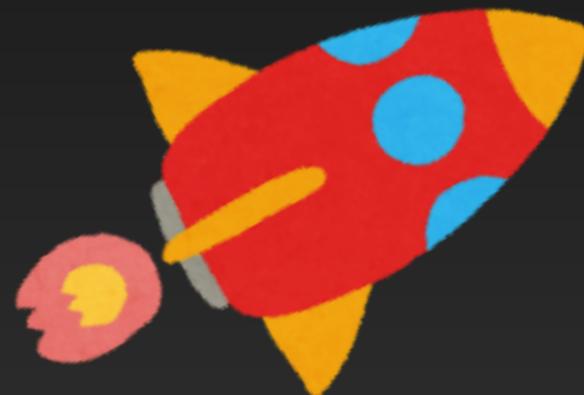
当たり判定クラス

```
public class RelativeHitBox extends Area {  
    public Area createAbsoluteHitBoxArea(Symbol symbol) {  
        double x = symbol.getX();  
        double y = symbol.getY();  
        double r = symbol.getAngleRadian();  
        var af = new AffineTransform();  
        af.translate(x, y);  
        af.rotate(r);  
        return this.createTransformedArea(af);  
    }  
}
```

Area に平行移動, 回転を同時に使うメソッドを
追加したクラス

当たり判定クラス

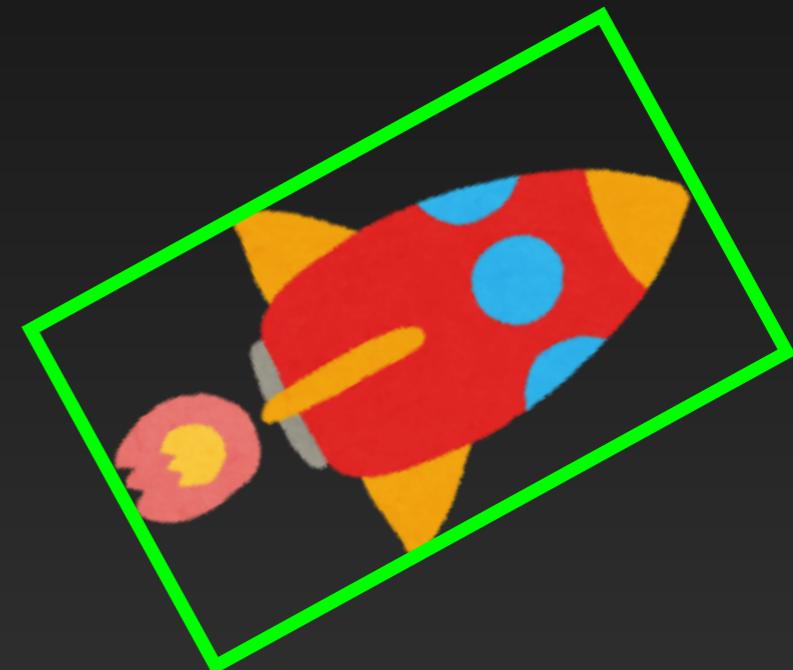
`createAbsoluteHitBoxArea(Symbol)`



図形は相対的な位置で保存しているので
実際の位置にずらしてあげる

当たり判定クラス

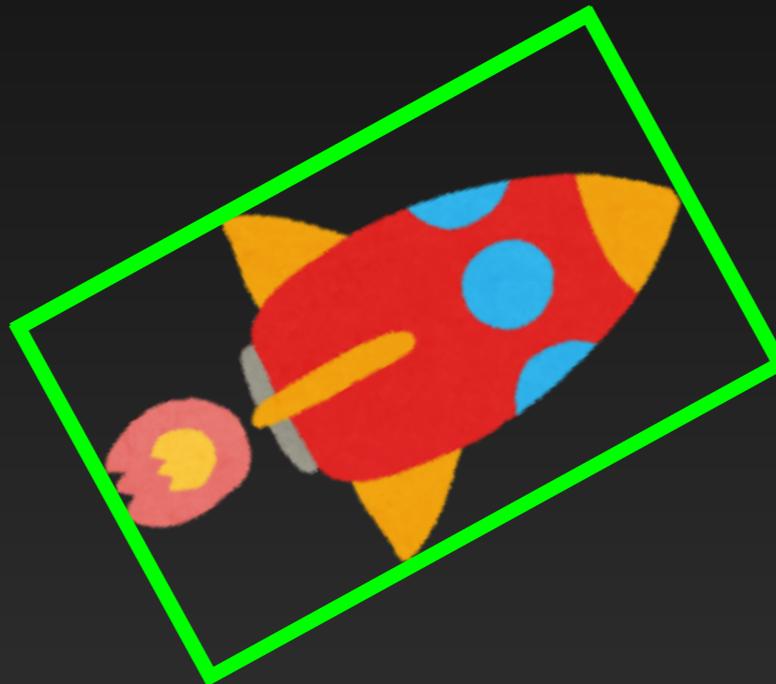
`createAbsoluteHitBoxArea(Symbol)`



図形は相対的な位置で保存しているので
実際の位置にずらしてあげる

当たり判定クラス

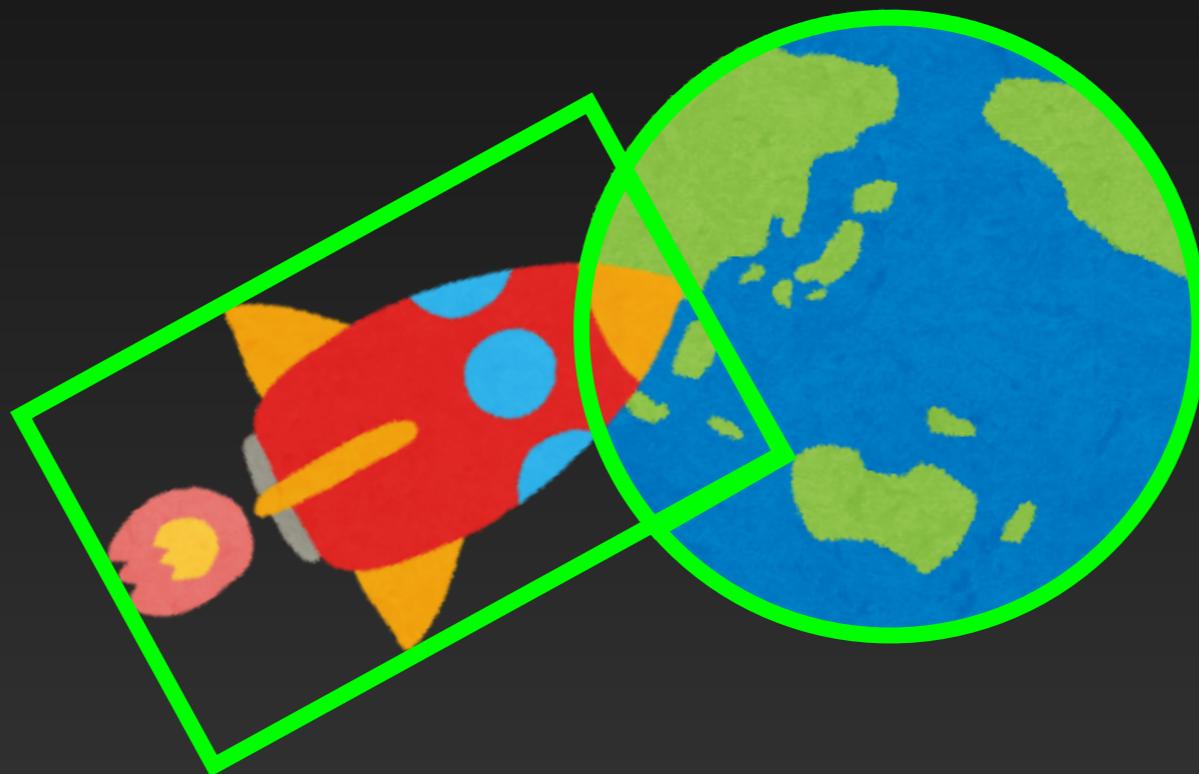
`createAbsoluteHitBoxArea(Symbol)`



図形は相対的な位置で保存しているので
実際の位置にずらしてあげる

当たり判定クラス

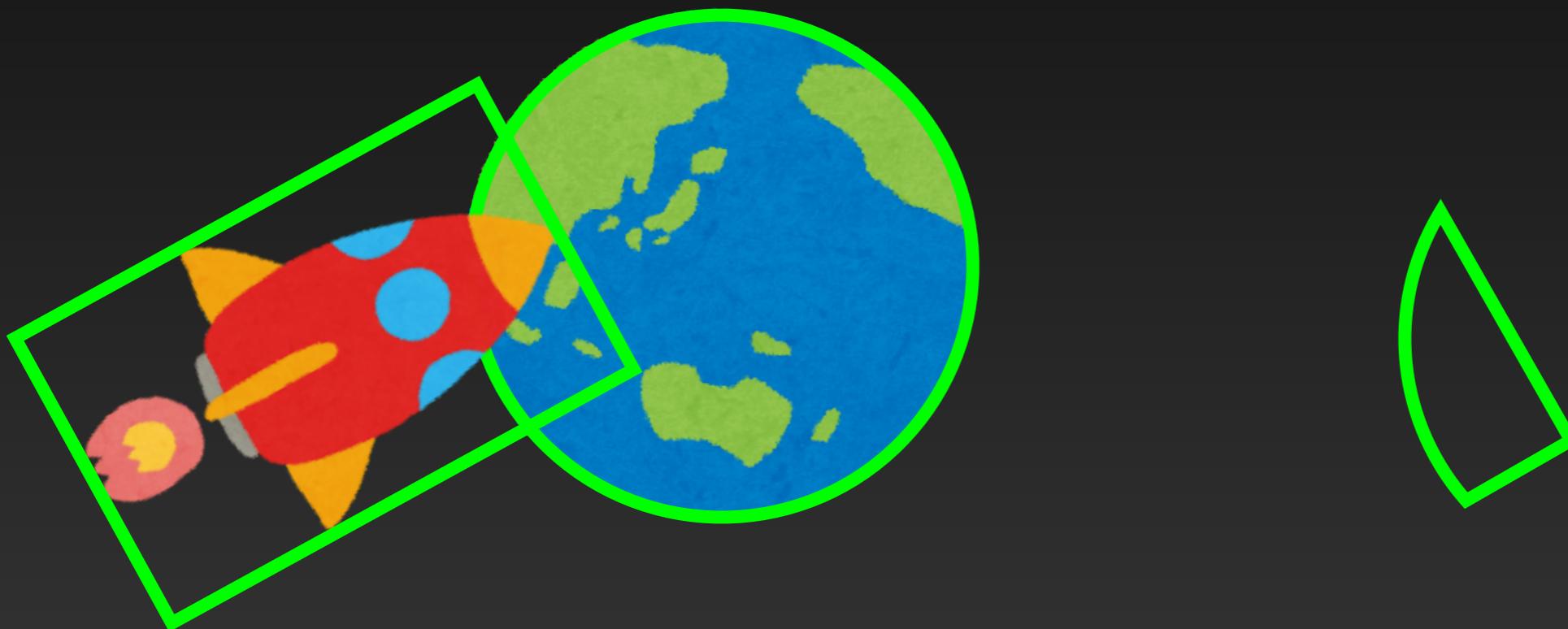
`Area#intersect(Area)`



Areaには共通部分を抜き出すメソッドが存在

当たり判定クラス

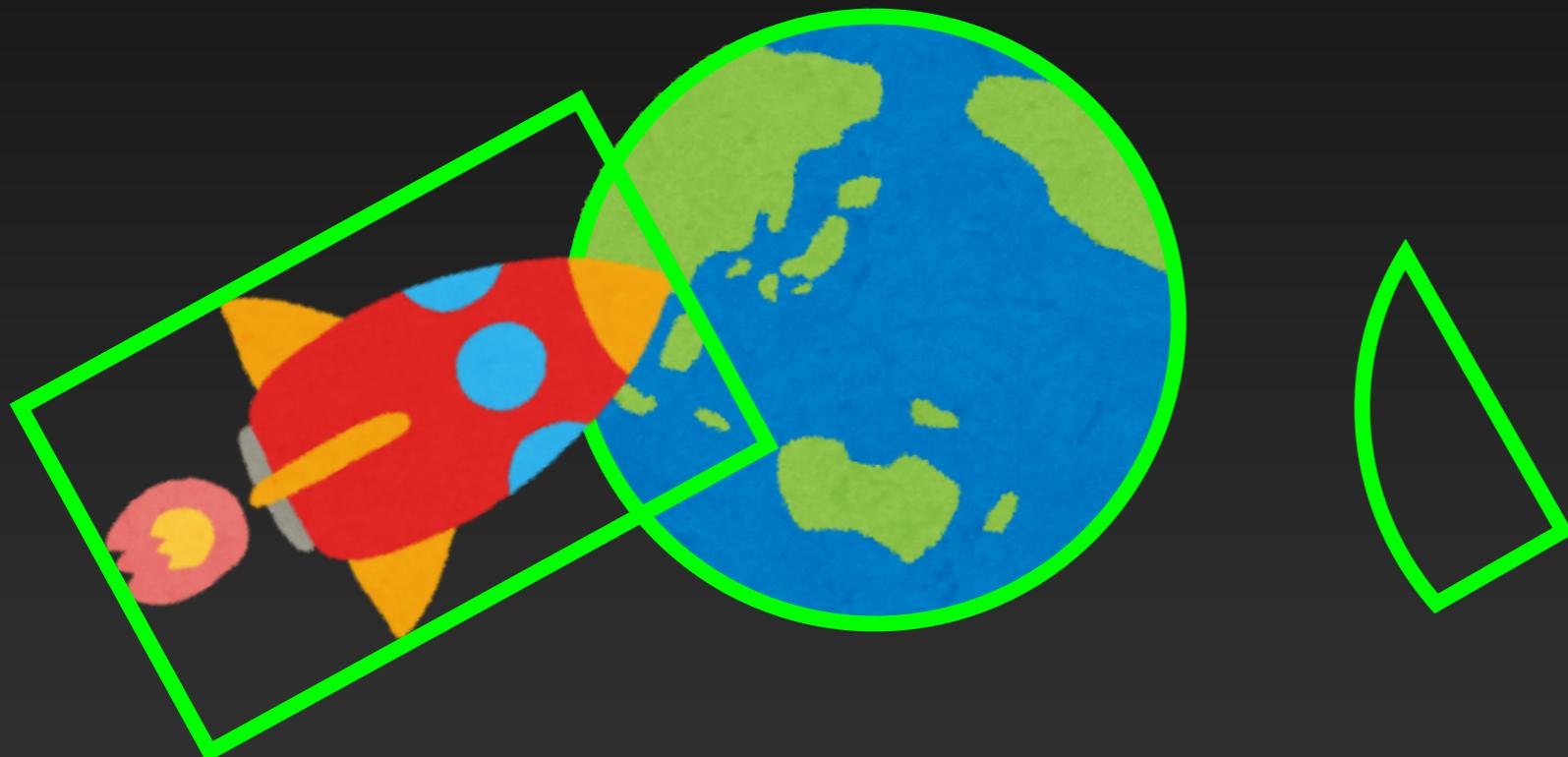
`Area#intersect(Area)`



Areaには共通部分を抜き出すメソッドが存在

当たり判定クラス

`Symbol#touches(Symbol)`



`!= null`

抜き出した図形がnullでなければ当たったと判定

Symbolクラスの使用例

```
public class Earth extends Symbol implements Drawable {  
    private int r = 10;  
  
    public Earth() {  
        // 座標の設定  
        this.setLocation(10, 10);  
        // 当たり判定の範囲設定  
        this.setRelativeHitBox(RelativeHitBox.makeCircle(r));  
        // 描画内容の設定  
        this.setDrawer(this);  
    }  
  
    @Override  
    public void draw(Graphics2D g) {  
        g.setColor(Color.BLUE);  
        g.fillOval(getX() - r, getY() - r, 2 * r, 2 * r);  
    }  
}
```

Symbol関連のクラス

Symbol



ImageAnimationSymbol

List<Image> を使いアニメーションを
再生・停止する機能をついたSymbol

Symbol関連のクラス

Symbol



MovableSymbol

速度の概念を足したSymbol
向きからX軸, Y軸方向の速度も計算する

Symbol関連のクラス

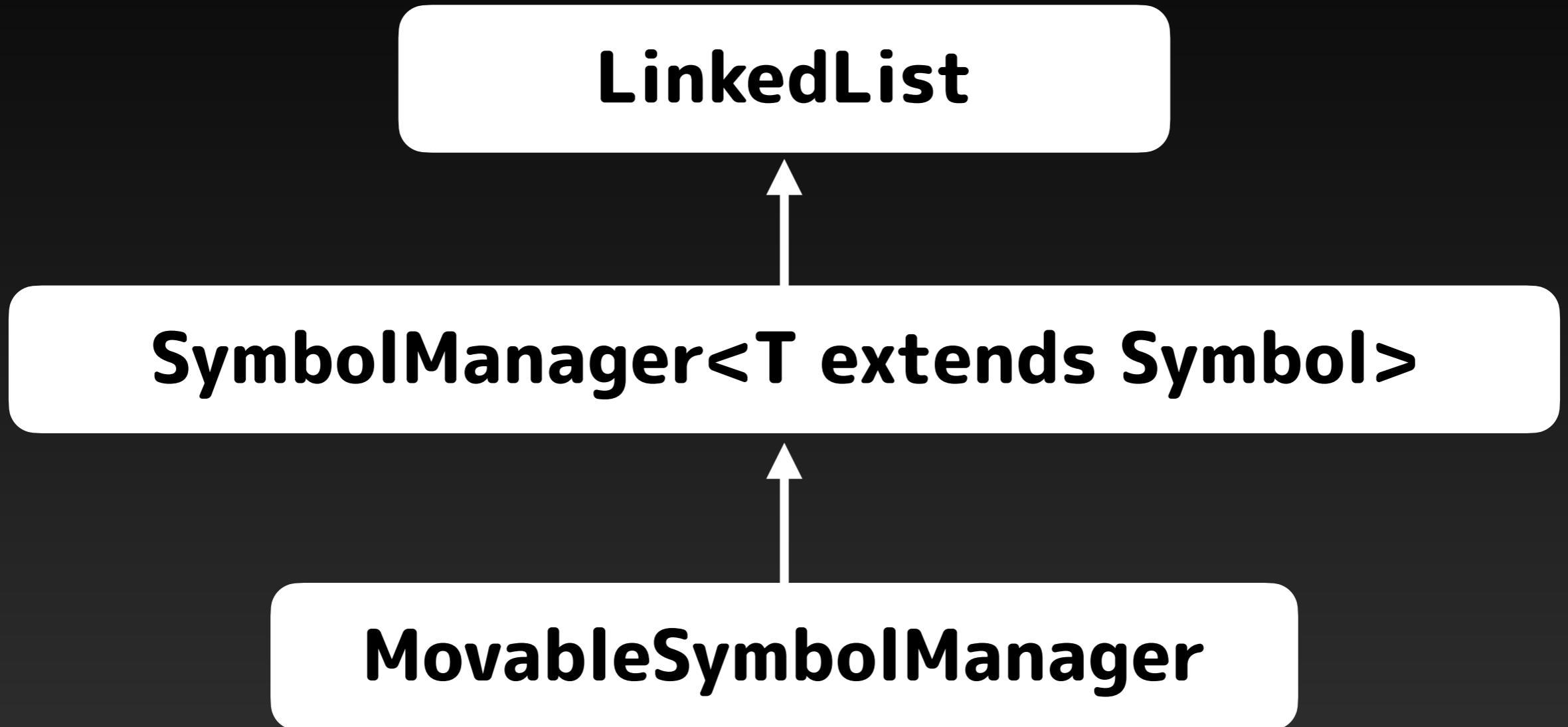
LinkedList



SymbolManager<T extends Symbol>

Symbolを継承した型を管理するクラス
条件に合うものを一括削除したりできる

Symbol関連のクラス



MovableSymbolを管理するクラス

Timerを用いて登録されたSymbolを同時に動かせる



[REDACTED] のAPI後世の為に公開すべき
メディプロの公式資料にするべき



...

[REDACTED] のAPIが優秀すぎてjava何も分からんくてもコード書
ける、これもうペタペタプログラミングだろ



View



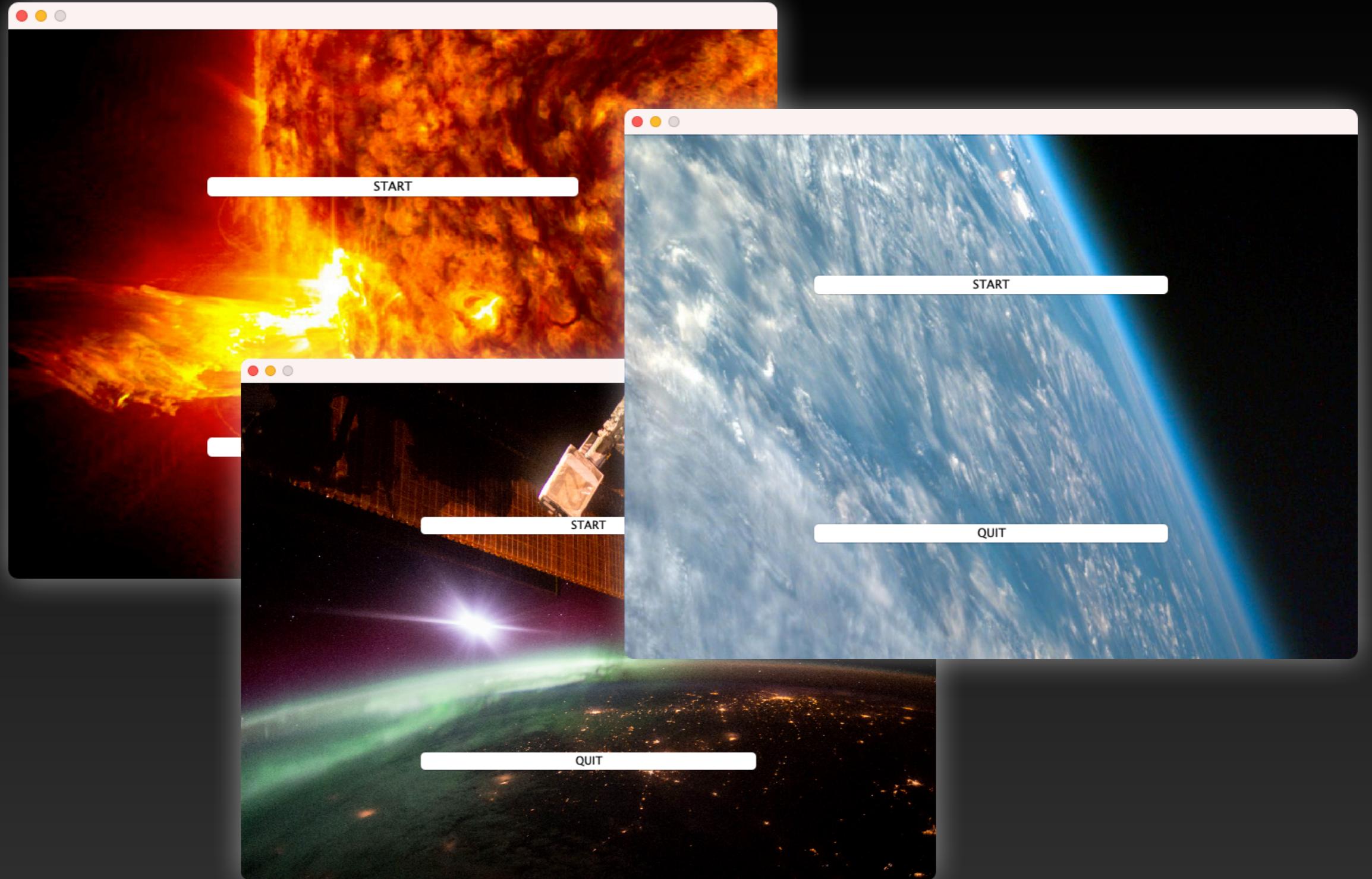
34

setSize(width: 800, height: 600),

START

QUIT

動くメインメニュー



ランダムな背景画像



[20:48] MapNo.0 CLEARED:0 (x,y) = (1536,1536)



ロケットのアニメーション・UIの作成

Controller



Controllerの責務

1. プレイヤーが思った通りに操作できること
2. 環境によって差を生まないこと

KeyListener#keyPressed

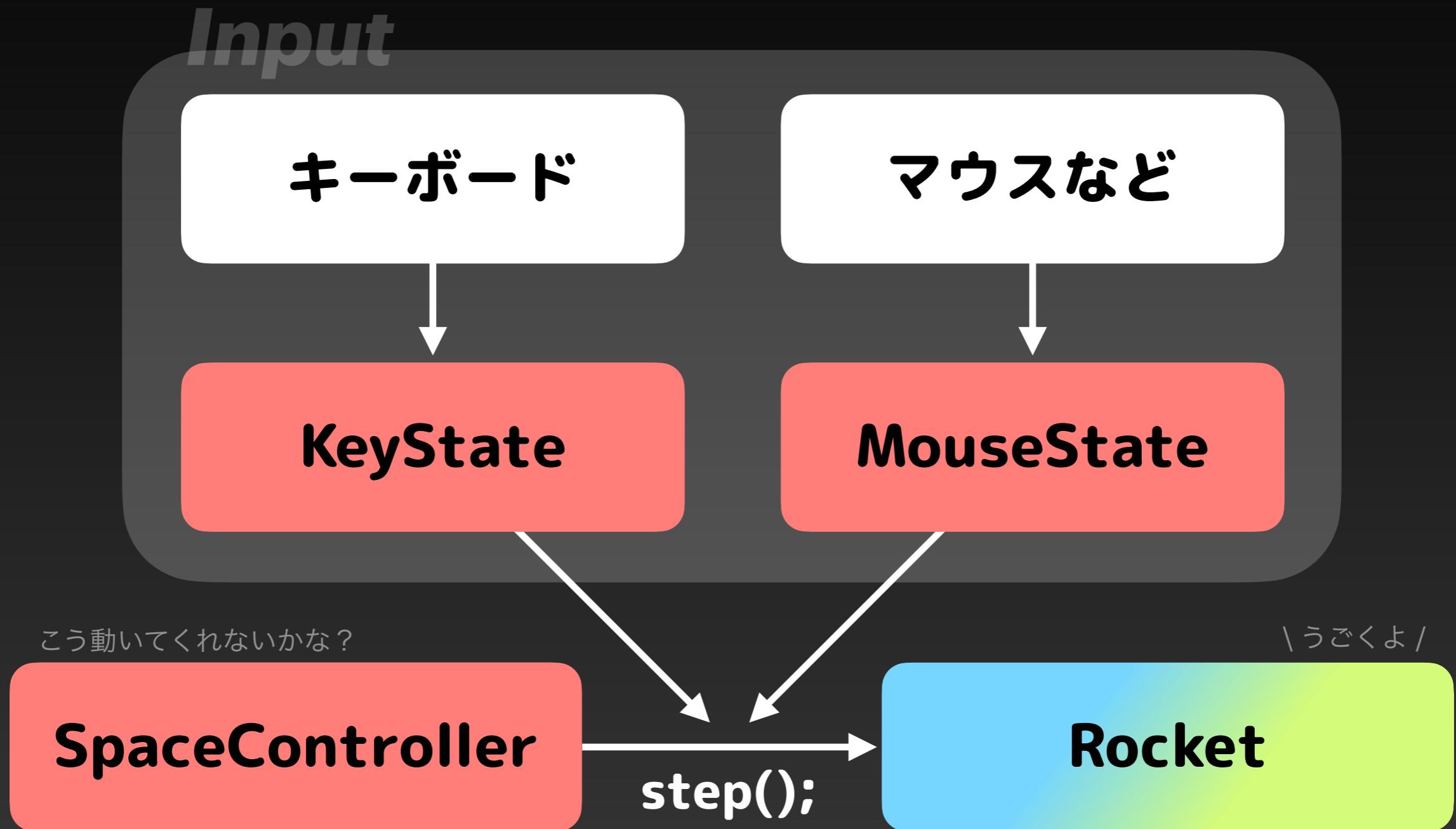
だけを使った実装の問題点

1. 一つのキーのみしか受け付けない
→ 同時押し不可能
2. PCによって発火間隔が異なる
3. 1回目と2回目以降の発火の間にラグ
→ 操作感の低下

操作の入力と処理の分離

```
public class SpaceController  
    extends GameController {  
  
    // キー入力情報の保持  
    private KeyState keyState;  
  
    // マウス入力情報の保持  
    private MouseState mouseState;  
  
    // 1フレームで行う処理  
    private void step() { ..... }  
  
    // stepを一定間隔で実行するタイマー  
    private Timer stepTimer;
```

操作の入力と処理の分離

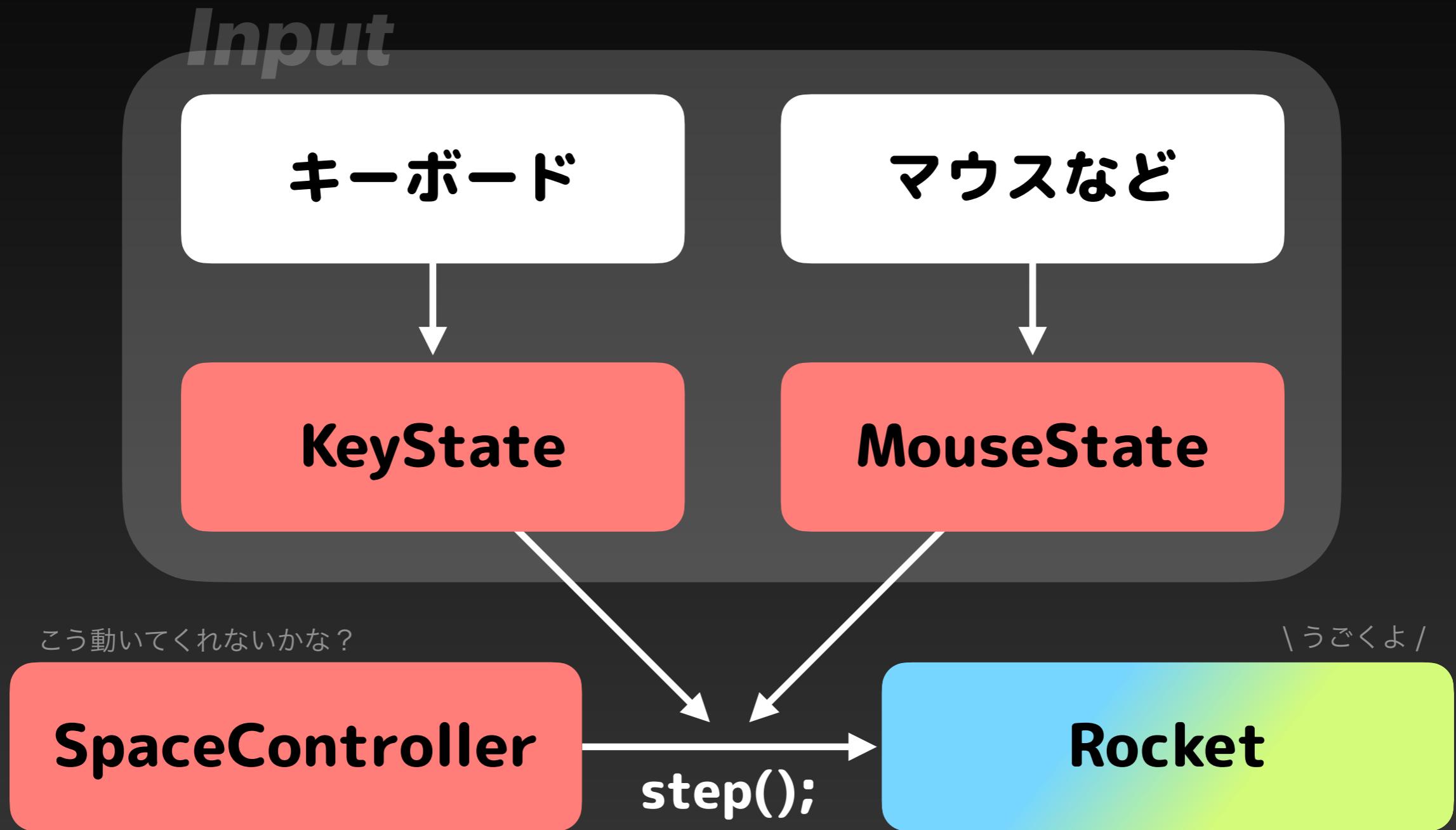


定期的に入力されたキーの情報を受け取り、Rocketに指示

```
class KeyState implements KeyListener {  
  
    // 押されているキーの集合  
    private Set<Integer> states;  
  
    // 押したら追加  
    @Override  
    public void keyPressed(KeyEvent e) {  
        states.add(e.getKeyCode());  
    }  
  
    // 離したら削除  
    @Override  
    public void keyReleased(KeyEvent e) {  
        states.remove(e.getKeyCode());  
    }  
}
```

```
class MouseState implements MouseListener {  
  
    // 押されているボタンの集合  
    private boolean clicked, wheeled, wheelUp;  
  
    // 押したら追加  
    @Override  
    public void mousePressed(MouseEvent e) {  
        // メンバ変数に変更を加える  
    }  
  
    // 離したら削除  
    @Override  
    public void mouseReleased(MouseEvent e) {  
        // メンバ変数に変更を加える  
    }  
}
```

操作の入力と処理の分離



入力されたキーの情報を受け取り、定期的にRocketに指示

```
private Timer t = new Timer(200, e -> step());  
  
private void step() {  
  
    boolean Wが押されている =  
        keyState.isPressed(KeyEvent.VK_W);  
    boolean Sが押されている =  
        keyState.isPressed(KeyEvent.VK_S);  
  
    if(Wが押されている || Sが押されている) {  
        if(Wが押されている) rocket.加速();  
        if(Sが押されている) rocket.減速();  
    } else if (mouseState.isClicked()) {  
        rocket.set速度(カーソルとの距離);  
    }  
}
```

KeyListener#keyPressed

だけを使った実装の問題点

1. 一つのキー受け付けない
→ 同時押し受け付けない
2. PCによって操作感が異なる
3. 1回目と2回目の操作感が異なる
→ 操作感の違い

デモ・質疑応答