



酔歩のシミュレーション

オブジェクト指向プログラミング特論

只木進一:工学系研究科

シミュレーションの例題

- モデルと、その観測及びGUIを分離する
- モデル
 - モデルの動作を正確に記述する
- シミュレーション
 - モデルを動作させて状態を観測する
 - 結果をファイルに書く
- GUI
 - 動作をチェックする
 - デモンストレーションに使う

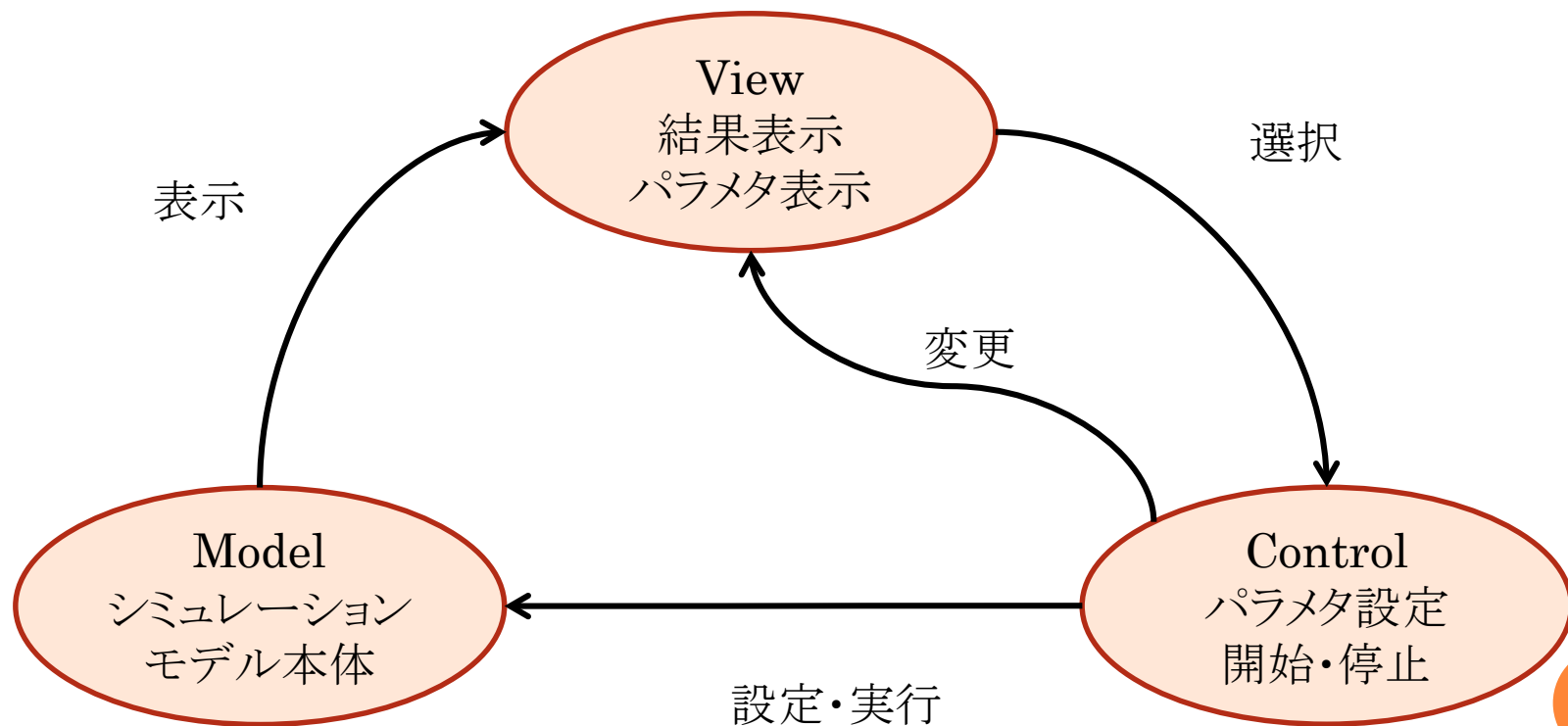


MVC (MODEL-VIEW-CONTROL)

- OOPプログラミングの指針
 - MVCを分離することで、開発効率を上げる
- Model
 - シミュレーションモデル、ビジネスロジックなど
- View
 - ユーザーインターフェイス
- Control
 - UIからの操作の反応



シミュレーションの場合のMVC



酔歩 (RANDOM WALK)

- 確率過程 (Stochastic Process)
 - 系の時間発展が非決定的なもの
- 酔歩
 - 確率過程の標準モデル
 - 一次元格子
 - 各時刻で、確率 p で右に、 $1-p$ で左に移動



クラス設計

- 歩行者のクラス
 - 一歩移動するメソッド
 - 現在位置を返すメソッド
- シミュレーションのクラス
 - 多数の歩行者を動かす
 - 初期化メソッド: 全歩行者を原点に
 - 全歩行者を一歩動かすメソッド
 - 全歩行者のリストを返すメソッド
- シミュレーション実行のクラス
 - 歩行者数を定めてシミュレーションを実行
 - 一定時間経過後に歩行者位置の分布を生成する



クラス構成:MODELパッケージ

○ Walker

- 一つの酔歩を行うWalker
- walk()メソッドで確率的に位置を ± 1 変更

○ Simulation

- oneStep()メソッドで多数のWalkerを「同時に」動かす

○ CLIMain

- コマンドラインのメイン
- Simulationクラスを駆動
- PositionHistogramクラスを使ってヒストグラムを生成

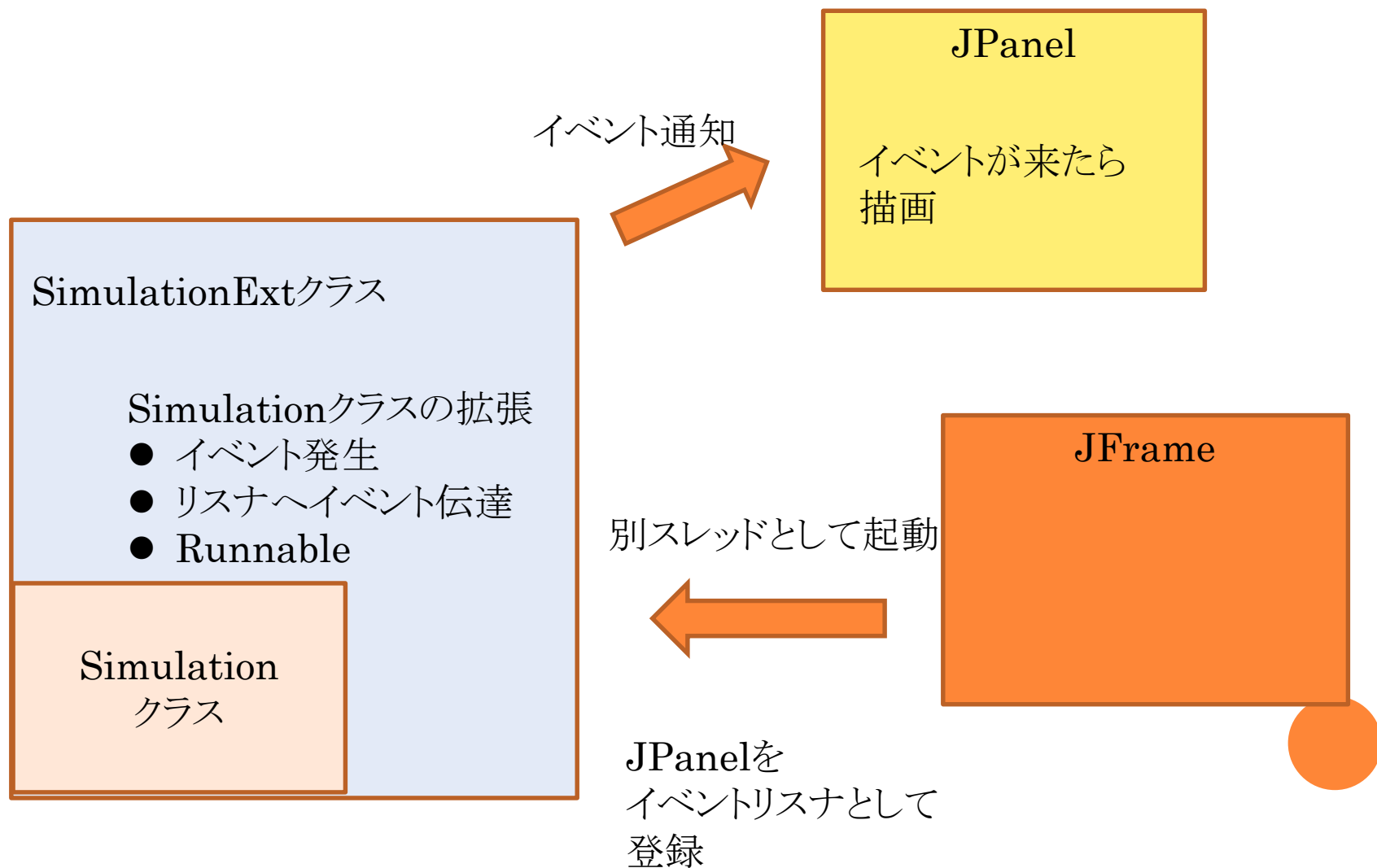


モデルからGUIへ

- GUIをつけるための必要な事
 - コントロールに対応するクラス(Jframeなど)からモデルを起動する
 - モデルの状態変化があった時、それを描画する
- イベントを用いる方法
 - モデルで状態変化があったらイベントを発生する
 - 描画側にイベントを伝える
 - 描画側はイベントが伝えられたら、モデルから状態を取得して描画する
- できるだけ、モデルを変更しないように



構成概要



EVENTとEVENTLISTENER

○ SimulationEventクラス

- EventObjectクラスの拡張として定義する
- イベントの種別を定義する
 - 例: 初期化と状態更新

○ SimulationEventListenerインターフェイス

- EventListenerインターフェイスの拡張
- stateChanged()メソッド

```
public interface SimulationEventListener
    extends EventListener{
    public void stateChanged(
        SimulationEvent event);
}
```



```
public class SimulationEvent extends EventObject {  
  
    public static enum EventType {  
        INITIALIZED, UPDATED;  
    }  
  
    private final EventType eventType;  
  
    public SimulationEvent(Object source,  
                           EventType eventType) {  
        super(source);  
        this.eventType = eventType;  
    }  
  
    public EventType getEventType() {  
        return eventType;  
    }  
}
```



SIMULATIONEXTクラス

- Simulationクラスの拡張クラス
 - 初期化および更新時にeventをfire
 - eventのリスナを登録

```
public class SimulationExt extends Simulation implements Runnable
```



すべてのイベントリスナに状態変更を通知する

```
protected void fireStateChanged(EventType eventType) {  
    SimulationEvent e = new SimulationEvent(this, eventType);  
    listeners.stream().forEach(p -> p.stateChanged(e));  
}
```


```
@Override  
public List<Integer> oneStep() {  
    List<Integer> list = super.oneStep();  
    fireStateChanged(EventType.UPDATED);  
    return list;  
}
```

```
@Override  
public void initialize() {  
    super.initialize();  
    fireStateChanged(EventType.INITIALIZED);  
}
```



○ リスナ登録

```
public SimulationExt(int n) {  
    super(n);  
    listeners = Collections.synchronizedList(new ArrayList<>());  
}  
  
public void addSimulationEventListener(SimulationEventListener o) {  
    listeners.add(o);  
}  
  
public void removeSimulationEventListener(SimulationEventListener o) {  
    listeners.remove(o);  
}  
  
public void clearSimulationEventListener() {  
    listeners.clear();  
}
```



○ Runnableインターフェイスの実装

```
public void start() { running = true;  t = 0;  }

public void stop() { running = false;  }

@Override
public void run() {
    while (running) {
        update(updateTiming);
        t++;
        if (t > tmax) {running = false;          }
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
        }
    }
}
```



クラス構成: GUIパッケージ

○ DrawPanel

- SimulationExtクラスの状態変化を捕まえる
- ヒストグラムの時間変化を表示

○ RandomWalkFrame

- JFrameの継承クラス
- DrawPanelをSimulationExtクラスのイベントリスナとして登録
- DrawPanelに対してシミュレーション開始を指示
- Walker数を変更する機能




```
public class DrawPanel extends javax.swing.JPanel  
    implements SimulationEventListener
```

```
@Override  
    public void stateChanged(SimulationEvent event) {  
        List<Point2D.Double> hist  
            = PositionHistogram.getHist(simulation.getWalkers());  
        initializeImage();  
        createImage(hist);  
        repaint();  
    }
```

状態変化が起こったら、図を再構成して描画



