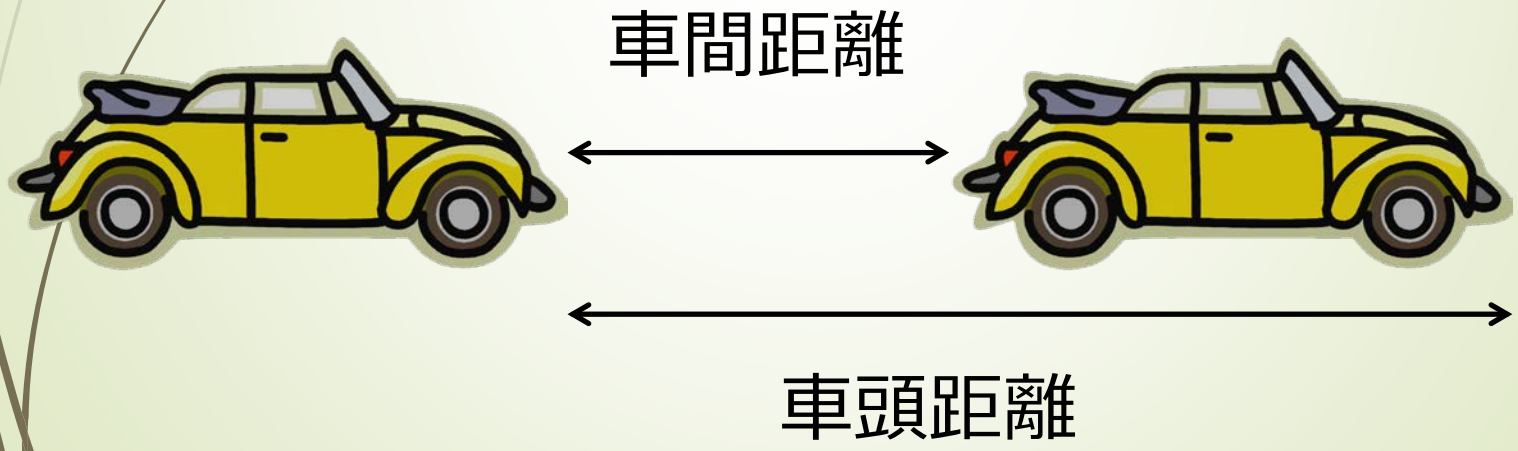
シミュレーションの例
最適速度交通流模型

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

最適速度交通流模型

- 交通流のモデル
- 車頭距離に応じた最適速度がある
- 最適速度に調整するように加減速する



モデルの定義

- 時刻tにおける、ある車両iの位置$x(t)$と速度$v(t)$
- 車両iとその先行車両$i-1$との車間距離

$$\Delta x(t) = x_{i-1}(t) - x_i(t)$$

- 車両iの加速度

$$a(t) = \alpha\left[V_{\text{optimal}}\left(\Delta x(t)\right) - v(t)\right]$$

- 加減速とは最適速度への調整
  - 速度調整の強度を示す定数α
  - 時間の逆数の次元：調整に要する時間の逆数に相当

- 長さLのサーキットにN台の車両
- $2N$個の変数の1階連立微分方程式
  - Runge-Kutta法で積分できる

$$\frac{\mathrm{d}v_i}{\mathrm{d}t} = \alpha\left[V_{\mathrm{optimal}}\left(x_{i-1} - x_i\right) - v_i\right]$$

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = v_i$$

最適速度

- 一般的にはシグモイド型
- 例：階段関数

$$V_{\text{optimal}}(\Delta x) = v_{\max}\theta(\Delta x - d)$$

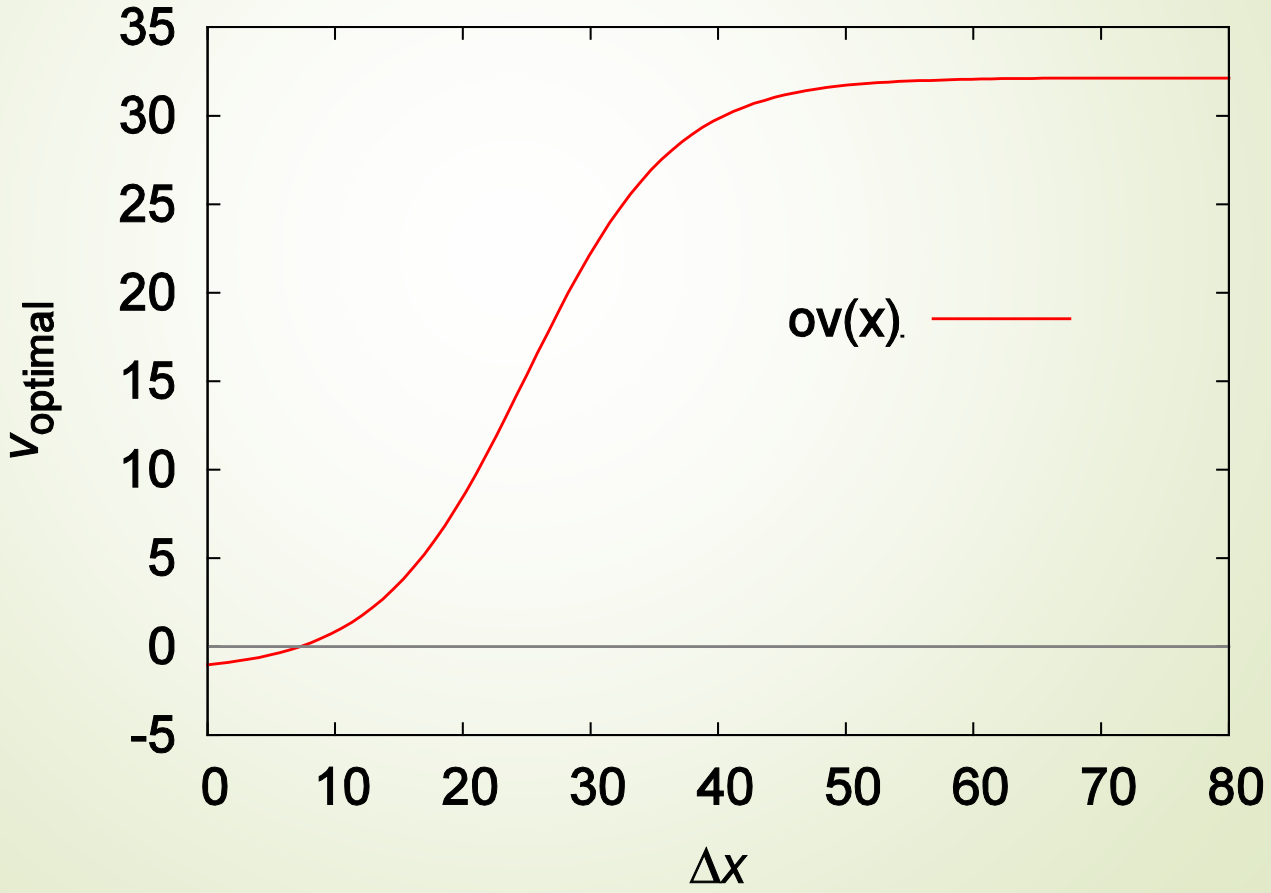
- 例：区分線形

$$V_{\text{optimal}}(\Delta x) = \begin{cases} 0 & \Delta x \le \Delta x_{\min} \\ v_{\max}\dfrac{\Delta x - \Delta x_{\min}}{\Delta x_{\max} - \Delta x_{\min}} & \Delta x_{\min} \le \Delta x \le \Delta x_{\max} \\ v_{\max} & \Delta x_{\max} \le \Delta x \end{cases}$$

- 例：双曲線正接

$$V_{\text{optimal}}(\Delta x) = \frac{v_{\max}}{2}\left[\tanh\left(2\frac{\Delta x - d}{w}\right) + c\right]$$

双曲線正接型の最適速度関数



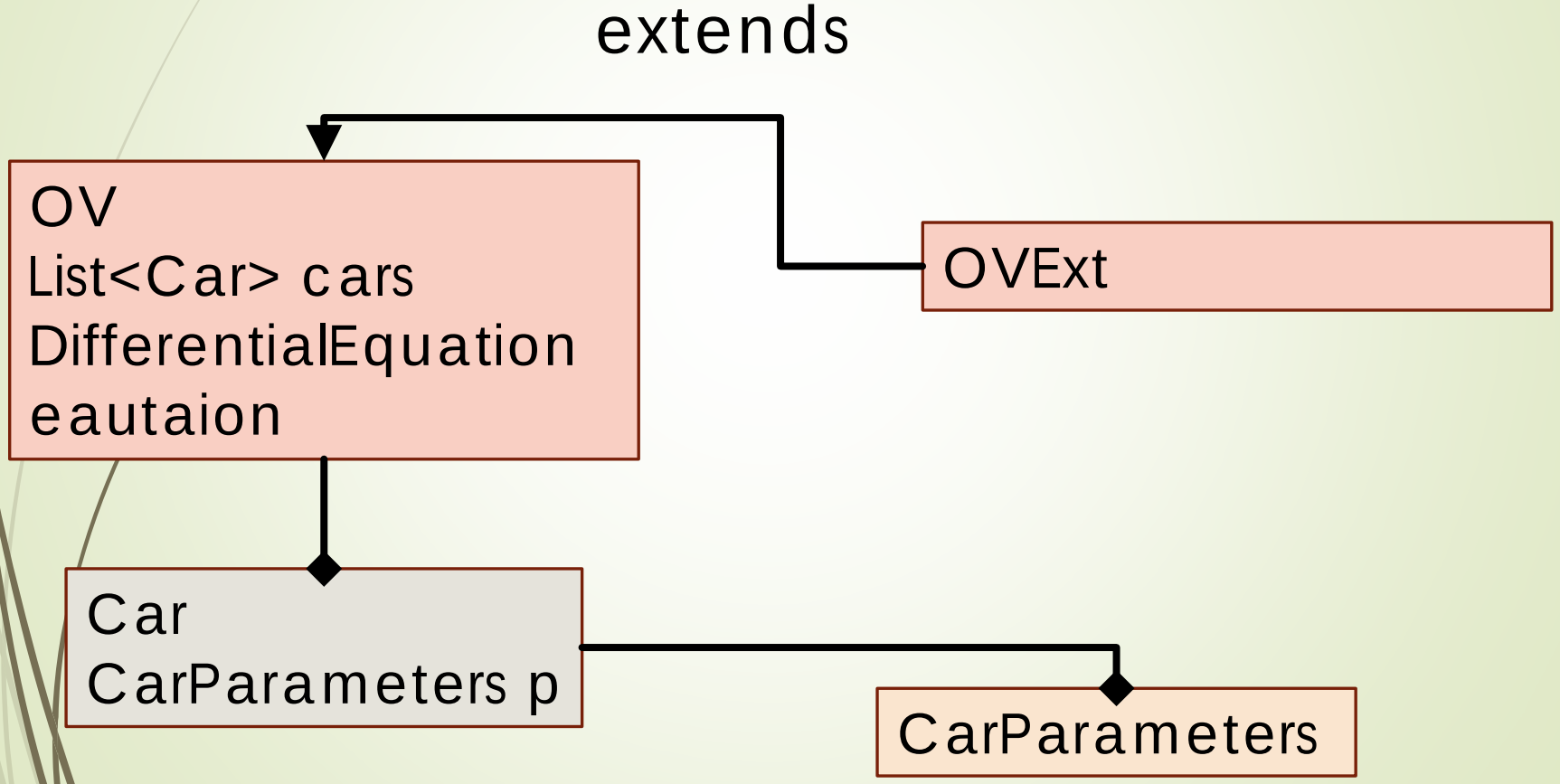
クラスの全体構成：model パッケージ

- BaseCar ：一台の車両を表すクラス
  - 最適速度が指定されてえいない
- Car：一台の車両を表すクラス
  - 双曲線正接型最適速度
  - パラメタ、位置、速度を保持
- CarParameters：OV関数のパラメタを保持

クラスの全体構成：model パッケージ

- OV：車両群を動かすクラス
  - RungeKutta法で一時間ステップ動かす
  - Carクラスインスタンスを更新
- OVExt
  - OVモデルで状態更新毎にイベントを投げるように拡張

Modelのクラス関連



クラスCar

- 一台の車輌のクラス
- 基本となる変数は位置xと速度v
- 一つのクラスCarVariablesにまとめる
- 現在、過去の値を保持
- 現在の値を過去の値として保存：saveValues();
  - 車輌の軌跡を描くために必要

クラスOV

- シミュレーションを実行する
- クラスCarのインスタンスのリスト
  - private java.util.Vector<Car> cars;
- 状態更新手順
  - 現在の値を過去の値として保存
  - 次の時刻の量を計算
  - 現在の量へ更新
  - Carクラスインスタンスへ保存

```
public void updatestate(int tstep) {//状態更新
        double y[] = new double[2 * numCar];
        for (int i = 0; i < numCar; i++) {
            //i 番の車両の位置
            y[2 * i] = cars.get(i).readposition() % length;
            //i 番の車両の速度
           y[2 * i + 1] = cars.get(i).readspeed();
        }
        double yy[][] = RungeKutta.rkdumb(y, 0, dt, tstep, equation);

        //位置及び速度の保存
        cars.stream().forEach(c -> c.savevalue());
```

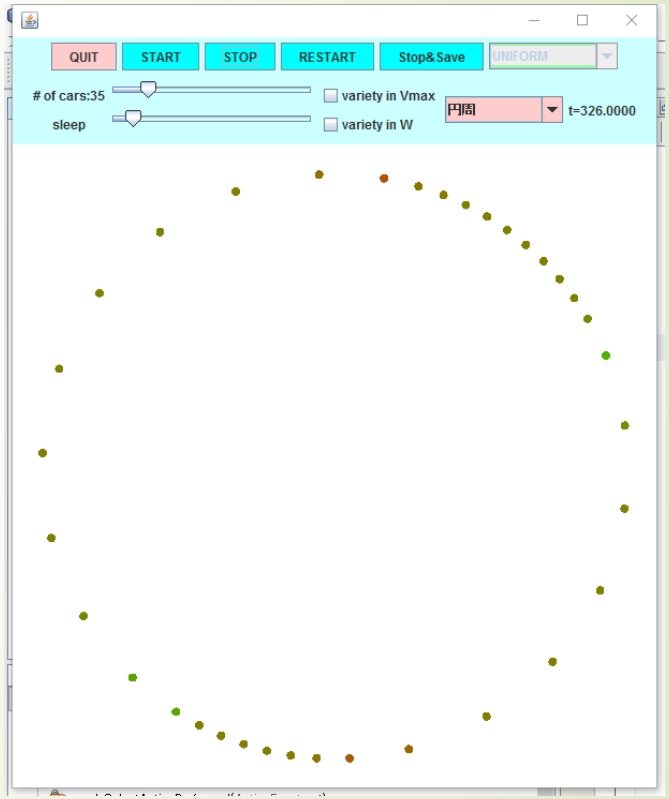
```
        for (int i = 0; i < numCar; i++) {
            cars.get(i).setX(yy[2 * i][tstep - 1] % length);
            cars.get(i).setV(yy[2 * i + 1][tstep - 1]);
            int j = (i + 1) % numCar;
            double headway = yy[2 * j][tstep - 1] - yy[2 * i][tstep - 1];
            headway = (headway + length) % length;
            cars.get(i).setDx(headway);
        }

        t += dt;
        for (int i = 0; i < numCar; i++) {
            Car c = cars.get(i);
            histories[i].append(new
                Data(t, c.readposition(), c.readspeed()));
        }
    }
```

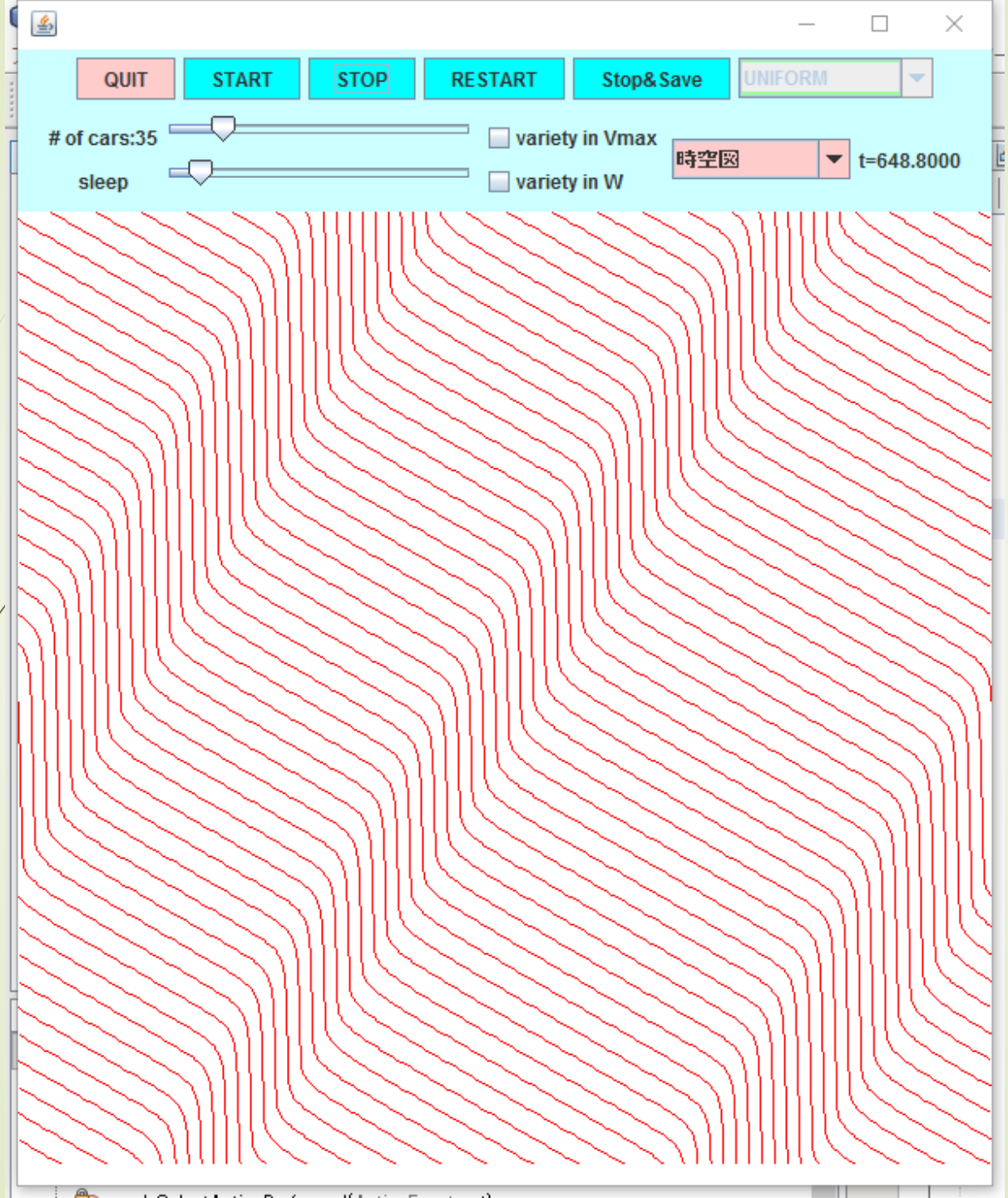
クラスの全体構成：guiパッケージ

- OVBase：軌道を図示するJPanelの拡張クラス
  - OVHV：車頭距離－速度空間内の軌道
  - OVCircle：サーキットイメージでの表示
  - OVSpacetime：時空間での軌道
- OVSimulation：シミュレーションのメインフレーム
- DistributionPanel：速度分布を表示するJPanel拡張クラス

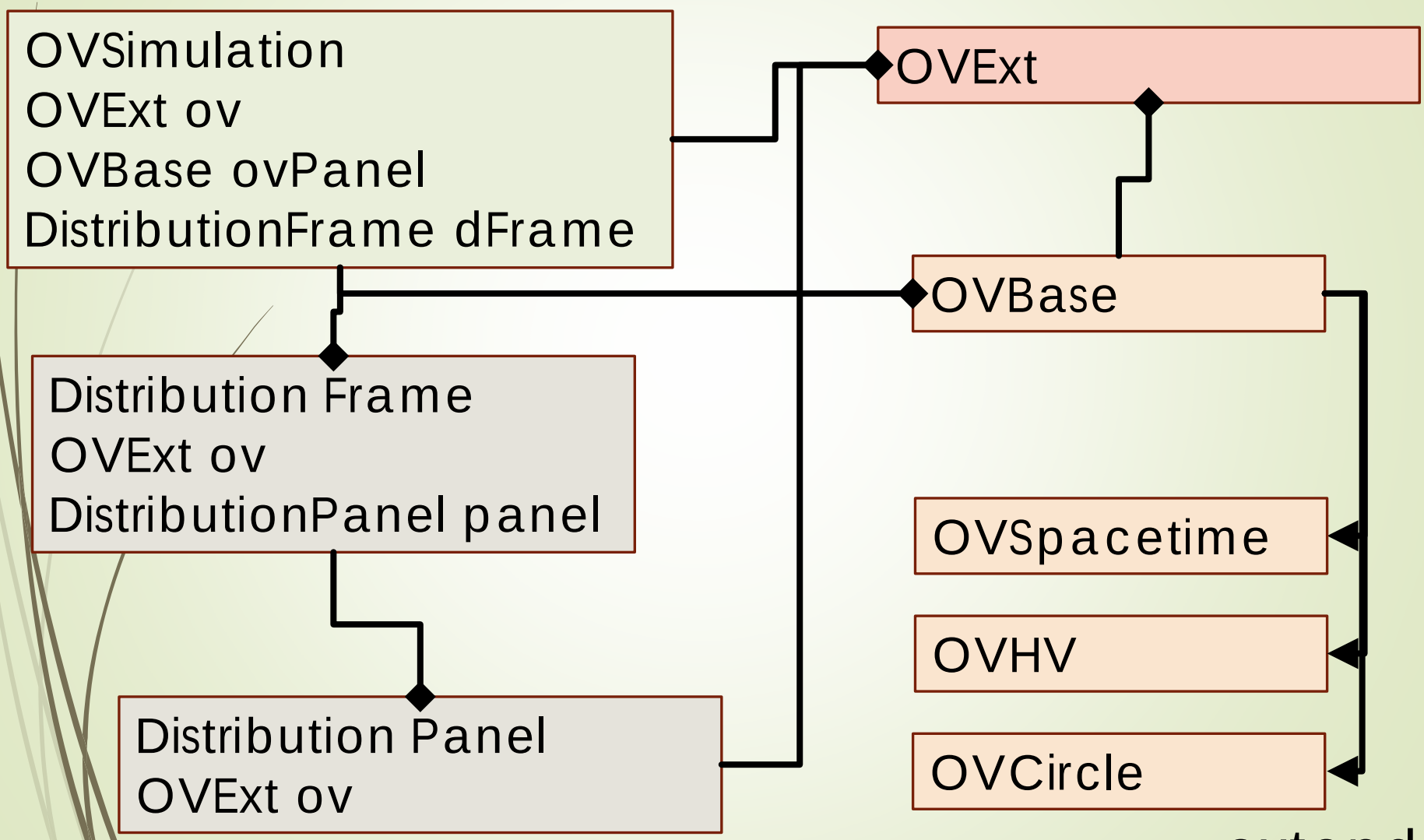
GUIの概要

- 全体のフレーム
  - ボタンパネル
    - ボタン
    - ラベル
    - スライダー
    - コンボボックス
- 描画用パネル




QUIT
START
STOP
RESTART
Stop&Save
UNIFORM
# of cars:35
sleep
variety in Vmax
variety in W
時空図
t=648.8000

GUIのクラス関連


OVSimulation
OVExt ov
OVBase ovPanel
DistributionFrame dFrame
OVExt
OVBase
Distribution Frame
OVExt ov
DistributionPanel panel
OVSpacetime
OVHV
Distribution Panel
OVExt ov
OVCircle
extends

描画パネルの階層

MainFrameクラスからすべてOVBaseのインスタンスに見せる

イベント送信