

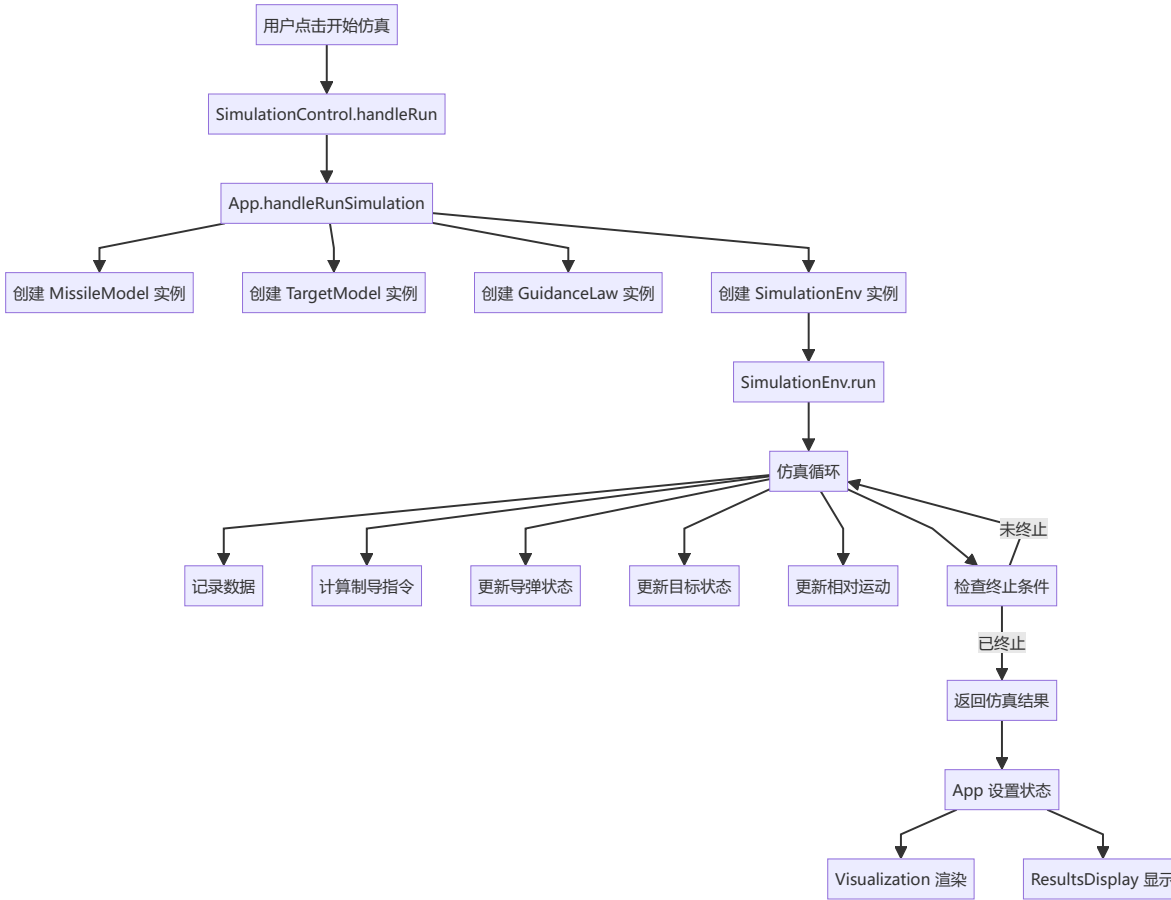
导弹制导律验证程序 - 函数关系流程图

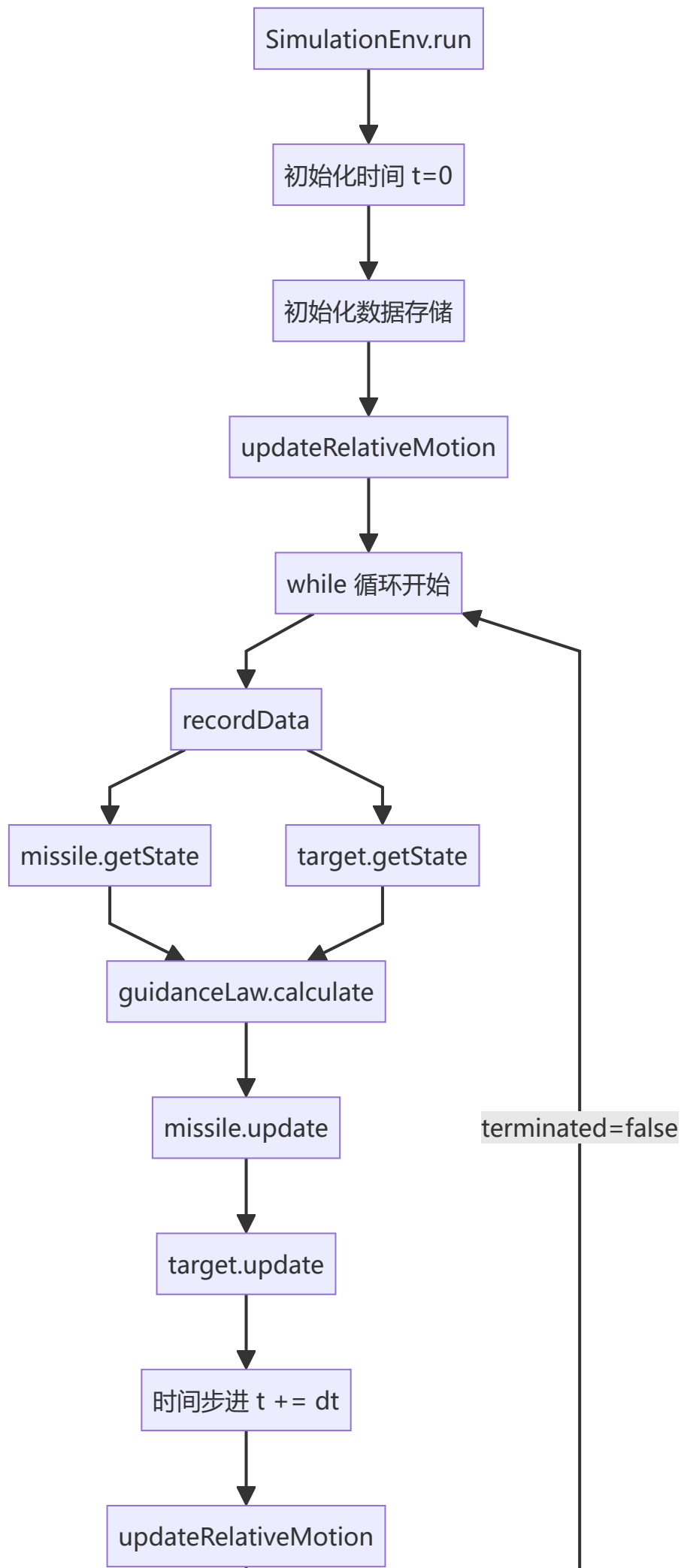
系统架构概览

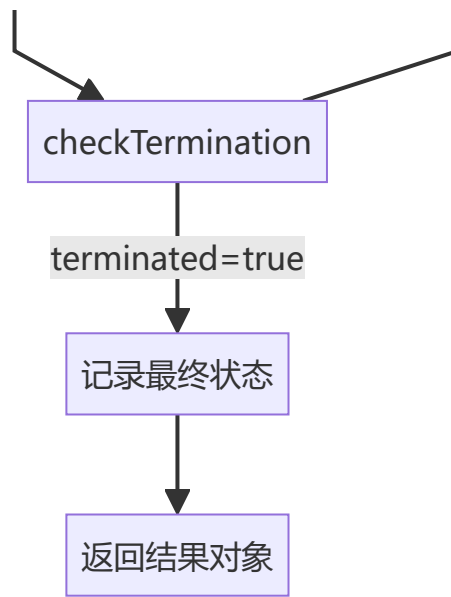
本程序采用模块化设计，主要包含以下核心模块：

1. **App.jsx** - 主应用组件，协调所有子组件
2. **SimulationControl** - 仿真控制面板，收集用户输入
3. **SimulationEnv** - 仿真环境，控制仿真循环
4. **GuidanceLaw** - 制导律模块，计算制导指令
5. **MissileModel** - 导弹运动学模型
6. **TargetModel** - 目标运动学模型
7. **Visualization** - 可视化组件，显示仿真结果
8. **ResultsDisplay** - 结果显示组件

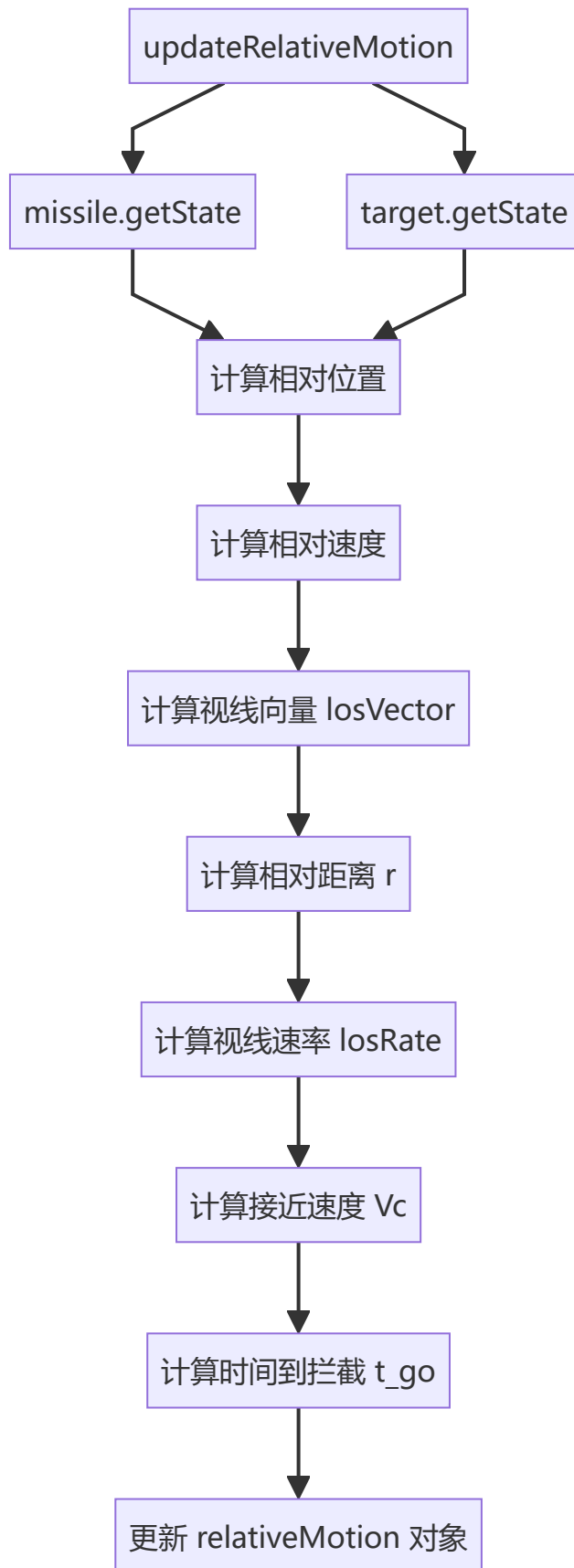
主流程图



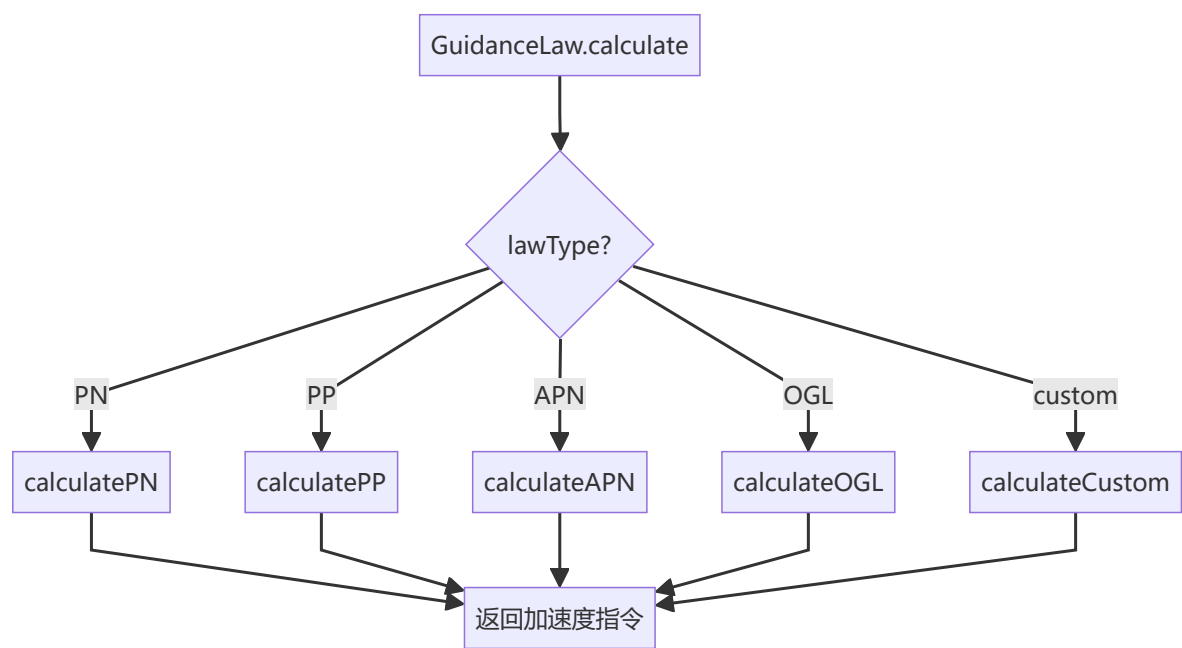




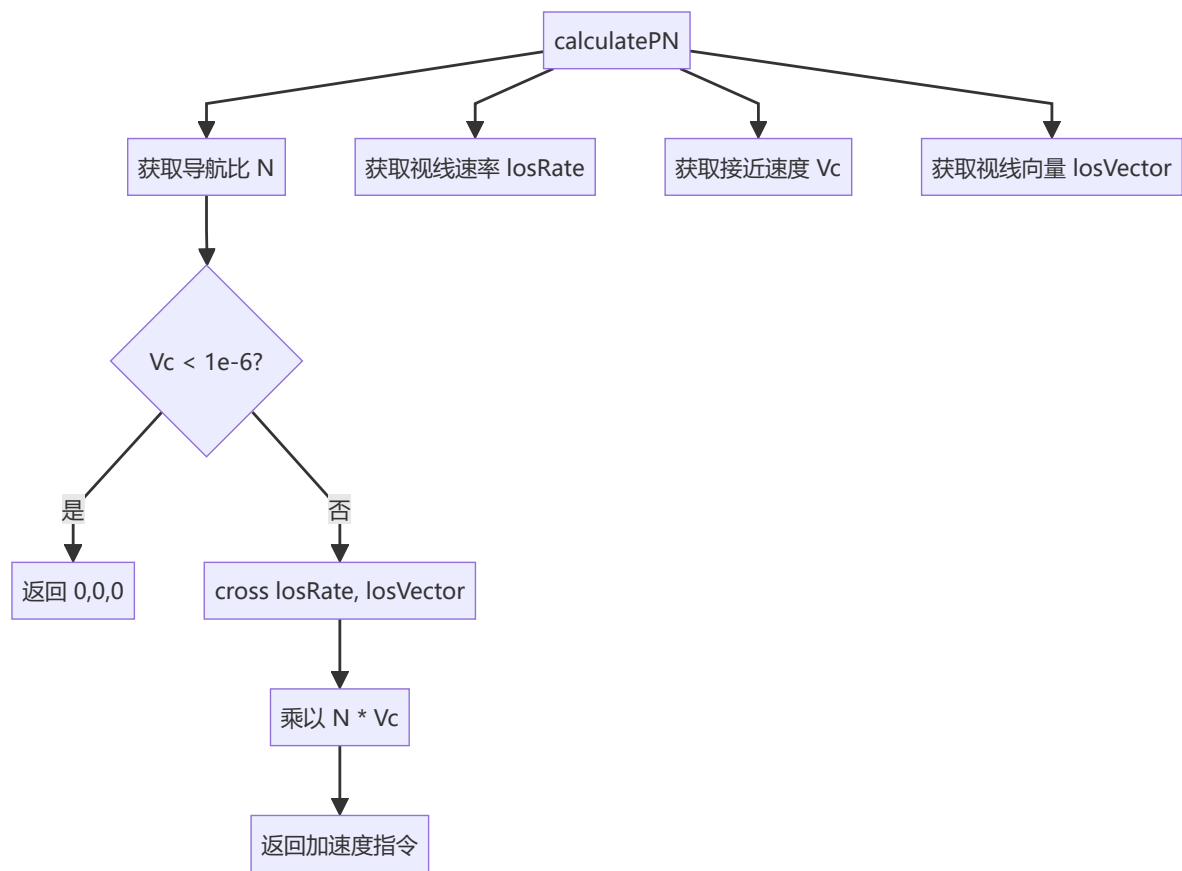
updateRelativeMotion 函数流程



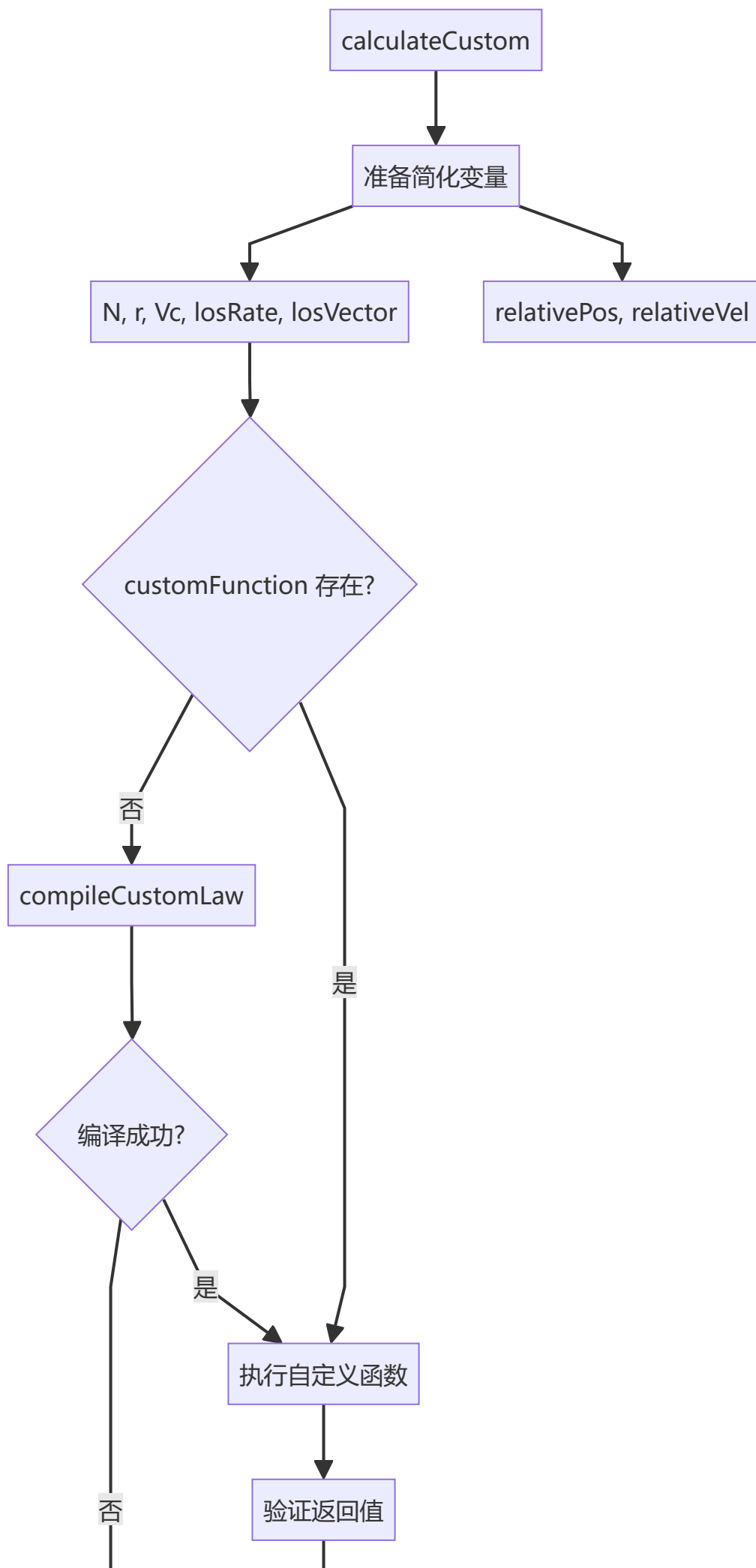
GuidanceLaw 详细流程图

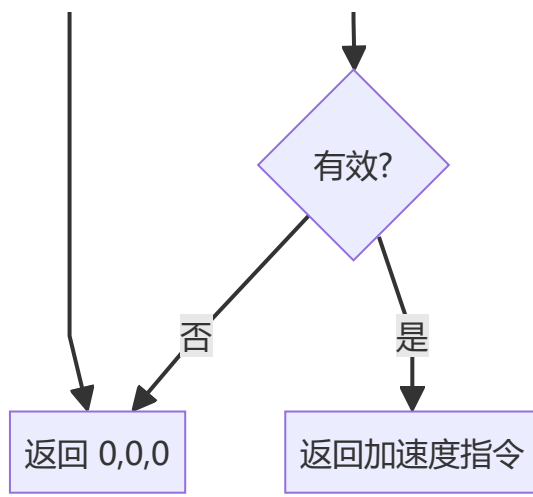


calculatePN 流程

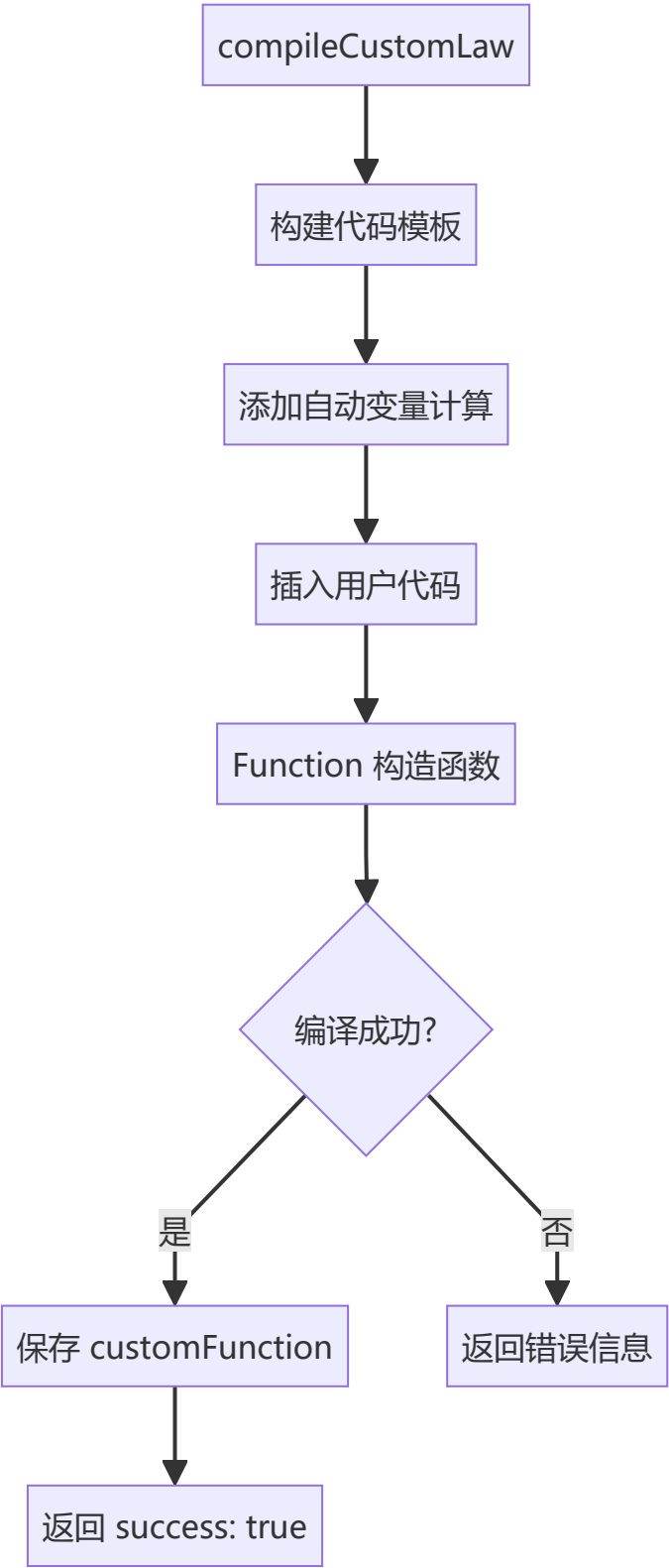


calculateCustom 流程

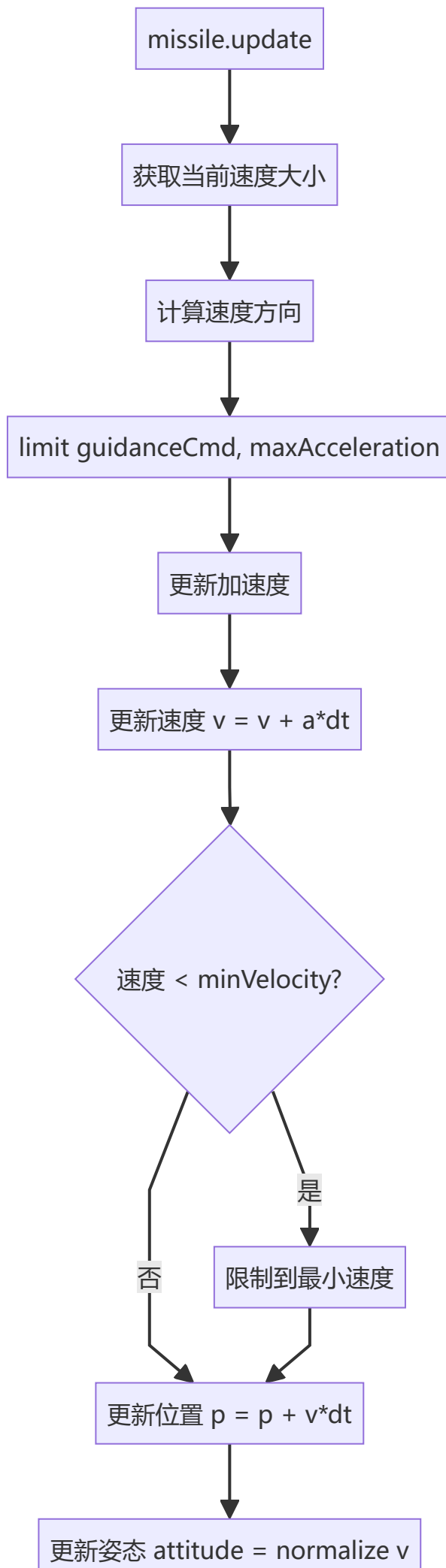


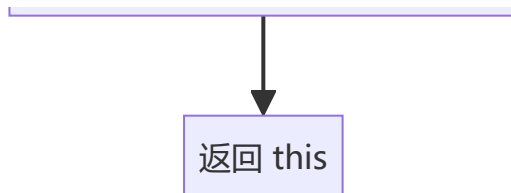


compileCustomLaw 流程

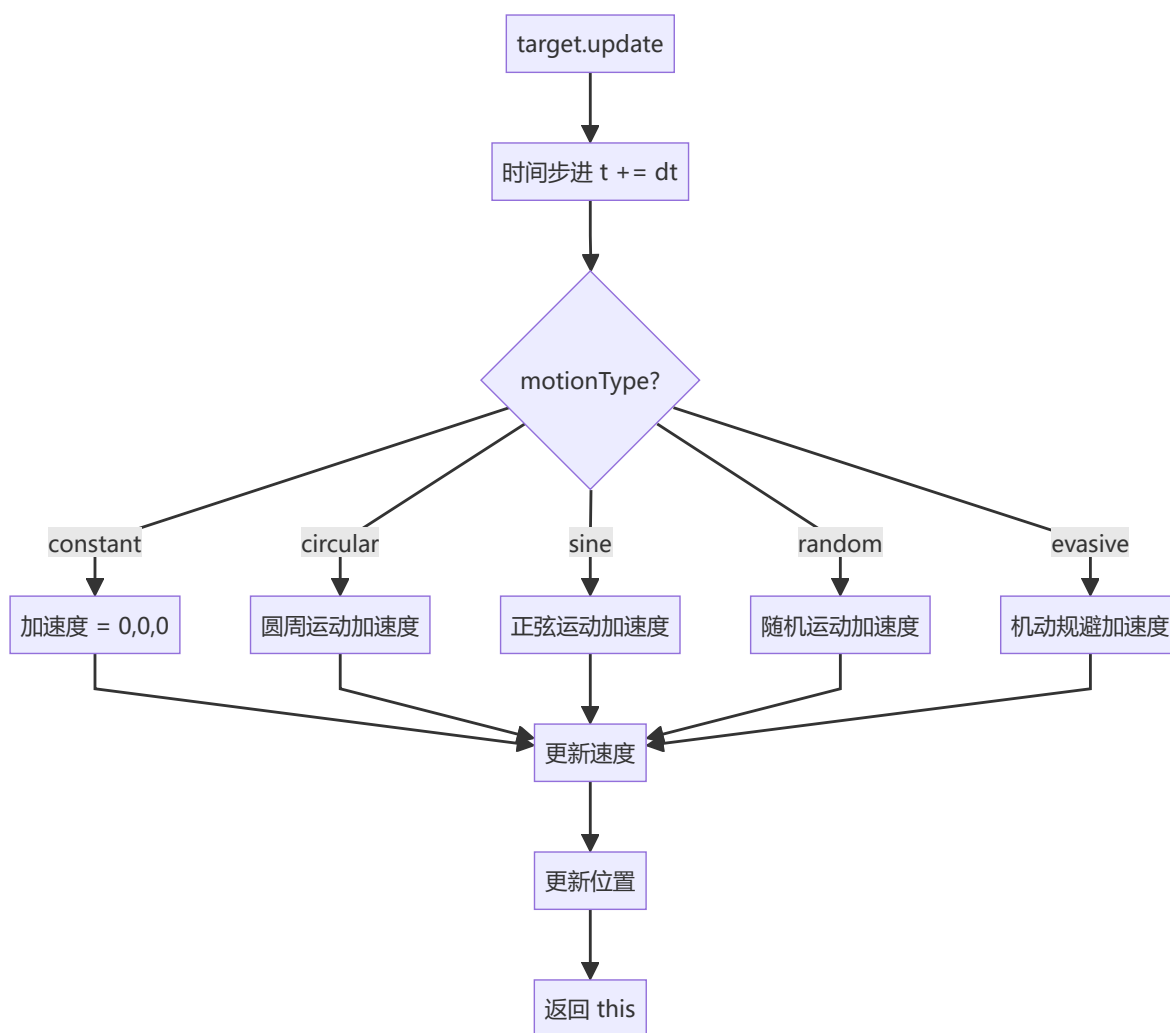


MissileModel 详细流程图

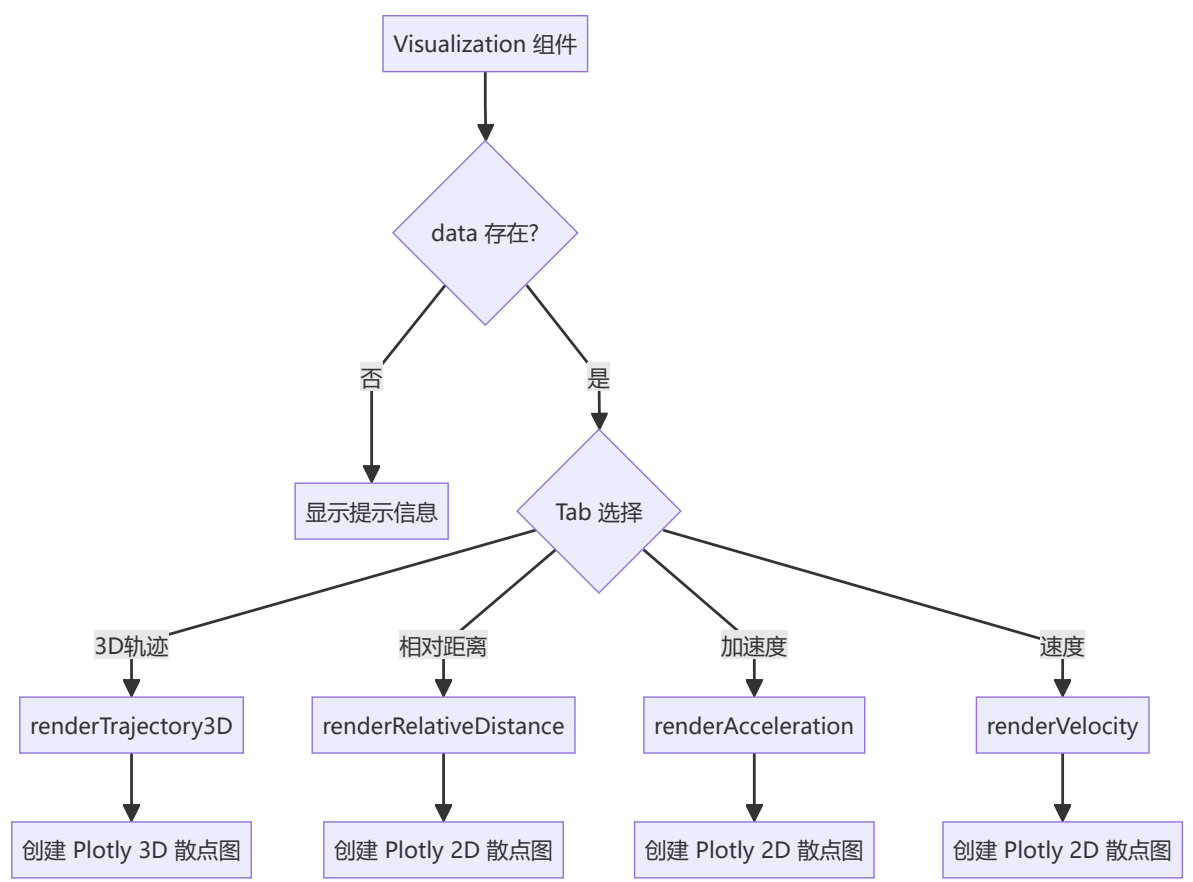




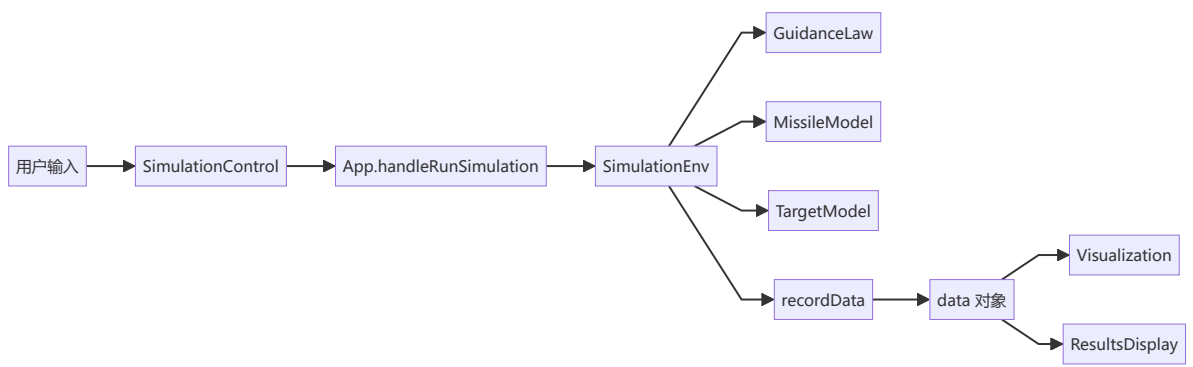
TargetModel 详细流程图



Visualization 组件流程图



数据流向图



函数调用关系表

调用者	被调用函数	说明
App	MissileModel	创建导弹模型实例
App	TargetModel	创建目标模型实例
App	GuidanceLaw	创建制导律实例
App	SimulationEnv	创建仿真环境实例

调用者	被调用函数	说明
SimulationEnv.run	missile.getState	获取导弹状态
SimulationEnv.run	target.getState	获取目标状态
SimulationEnv.run	guidanceLaw.calculate	计算制导指令
SimulationEnv.run	missile.update	更新导弹状态
SimulationEnv.run	target.update	更新目标状态
SimulationEnv.run	updateRelativeMotion	更新相对运动
SimulationEnv.run	recordData	记录数据
SimulationEnv.run	checkTermination	检查终止条件
GuidanceLaw.calculate	calculatePN	计算比例导引
GuidanceLaw.calculate	calculatePP	计算纯追踪
GuidanceLaw.calculate	calculateAPN	计算扩展比例导引
GuidanceLaw.calculate	calculateOGL	计算最优制导律
GuidanceLaw.calculate	calculateCustom	计算自定义制导律
calculateCustom	compileCustomLaw	编译自定义代码
MissileModel.update	limit	限制向量大小
MissileModel.update	normalize	归一化向量
TargetModel.update	-	根据运动类型更新

关键数据结构

SimulationEnv.data

```
1  {
2    time: [],           // 时间序列
3    missilePosition: [[], [], []], // 导弹位置 [x, y, z]
4    targetPosition: [[], [], []],  // 目标位置 [x, y, z]
5    missileVelocity: [[], [], []], // 导弹速度 [vx, vy, vz]
6    targetVelocity: [[], [], []],  // 目标速度 [vx, vy, vz]
7    missileAcceleration: [[], [], []], // 导弹加速度 [ax, ay, az]
8    relativeDistance: [],           // 相对距离
9    closingVelocity: []             // 接近速度
10 }
```

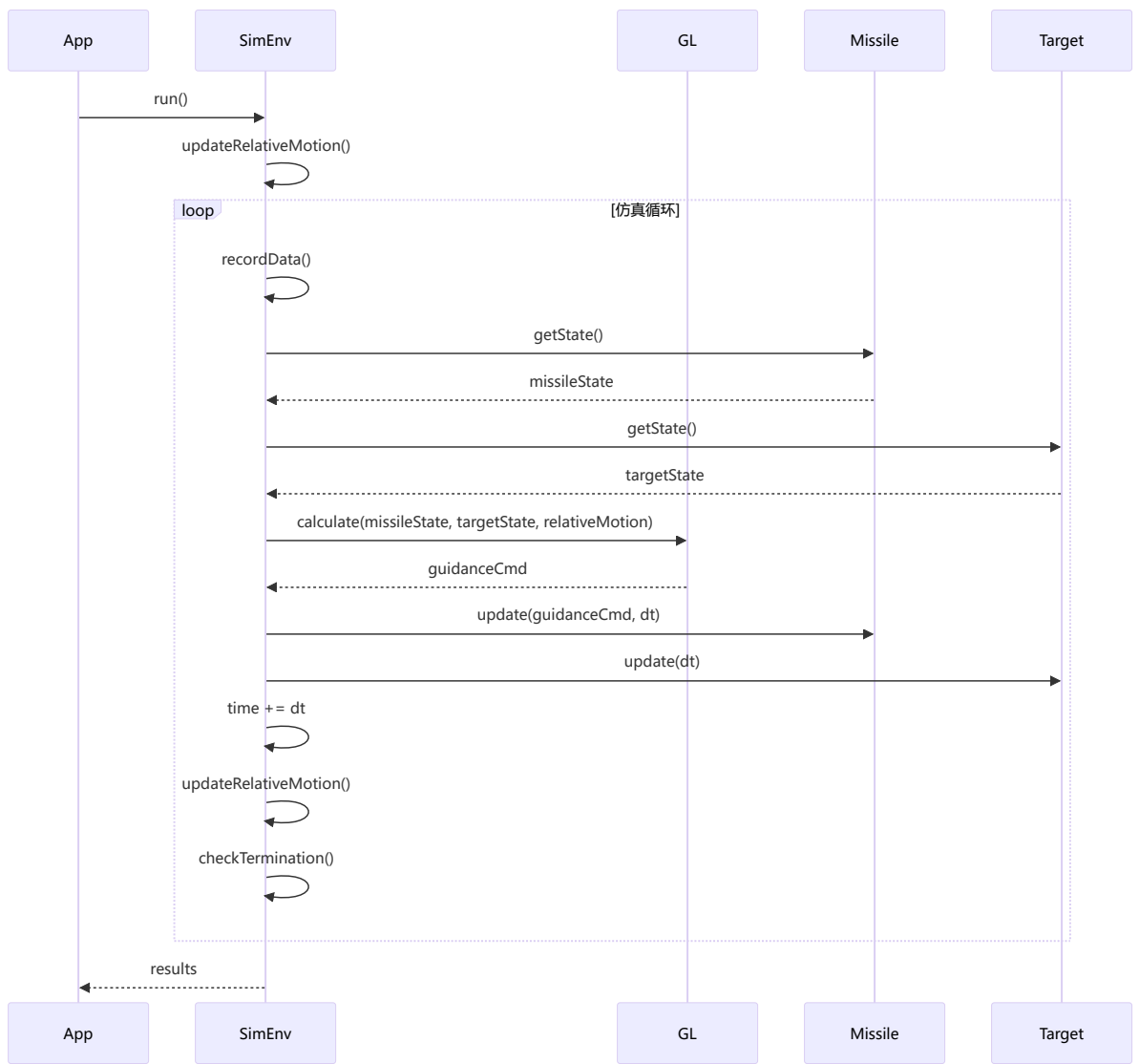
relativeMotion

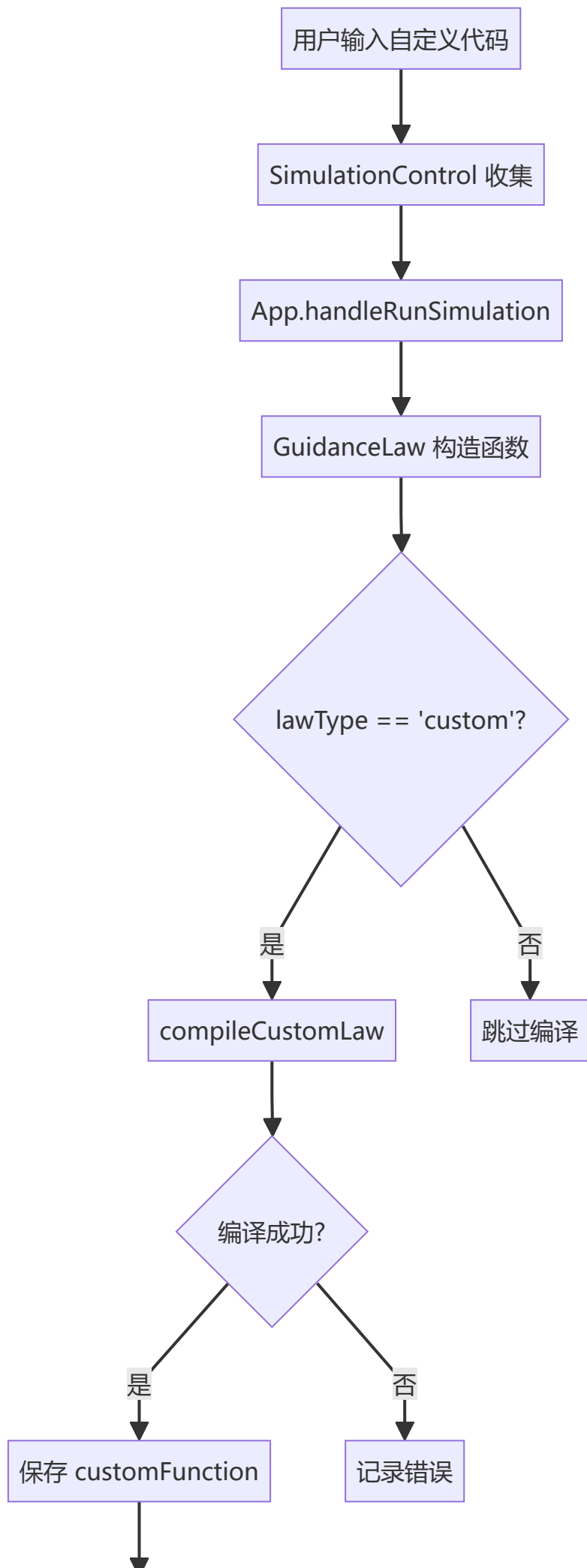
```
1 {  
2   relativePosition: [rx, ry, rz],    // 相对位置  
3   relativeVelocity: [vx, vy, vz],    // 相对速度  
4   losVector: [nx, ny, nz],           // 视线向量（归一化）  
5   losRate: [wx, wy, wz],             // 视线速率  
6   closingVelocity: Vc,                // 接近速度  
7   timeToGo: t_go,                    // 时间到拦截  
8   relativeDistance: r                // 相对距离  
9 }
```

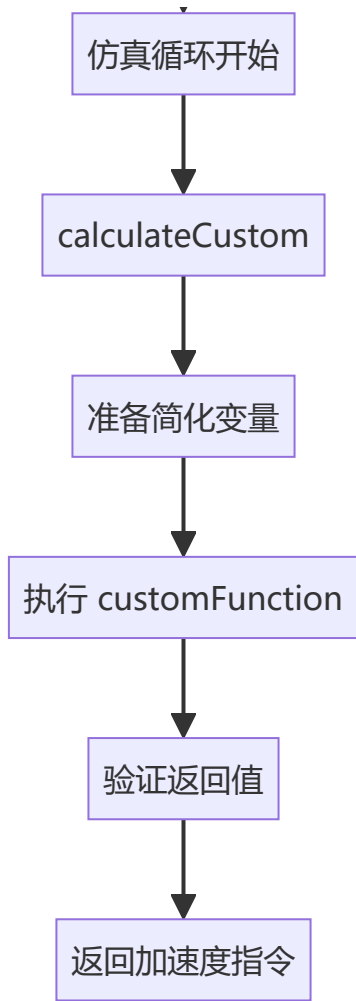
missileState / targetState

```
1 {  
2   position: [x, y, z],                // 位置  
3   velocity: [vx, vy, vz],            // 速度  
4   acceleration: [ax, ay, az],        // 加速度  
5   speed: v                            // 速度大小（仅导弹）  
6 }
```

仿真循环时序图







总结

本程序采用清晰的模块化架构，各模块职责明确：

1. **App.jsx** - 应用主控制器，协调各组件
2. **SimulationControl** - 用户界面，收集参数
3. **SimulationEnv** - 仿真引擎，控制循环
4. **GuidanceLaw** - 制导算法，计算指令
5. **MissileModel** - 导弹模型，更新状态
6. **TargetModel** - 目标模型，更新状态
7. **Visualization** - 可视化展示
8. **ResultsDisplay** - 结果统计

数据流向清晰：用户输入 → 仿真计算 → 数据记录 → 可视化展示