北京大学硕士研究生学位论文答辩

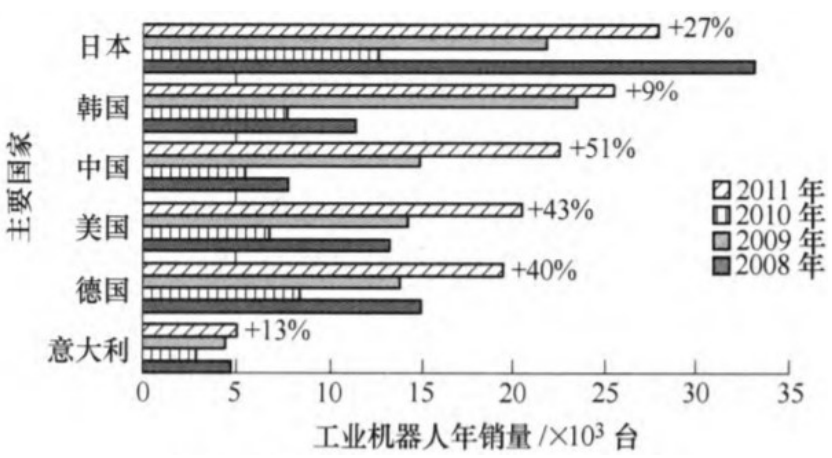
一种可编程智能机器人模拟器的设计与实现

Peking University

目录

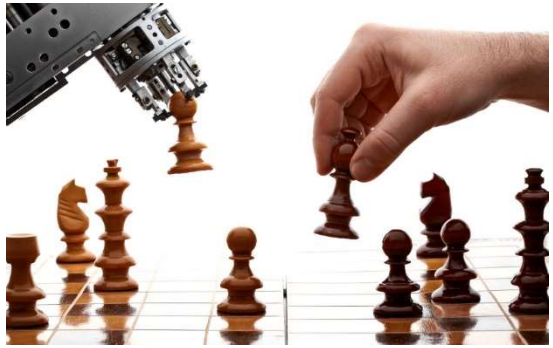
选题背景（1）

- 智能机器人（*Intelligent Robot*）是国内外近年的热门研究领域
  - 20世纪90年代后，世界机器人产品年销售量增长率平均约为10%
  - 2006-2010年间，我国对工业机器人的需求增长迅猛
  - 2016年，国家三部委印发《机器人产业发展规划（2016—2020 年）》
  - 2017年，国务院印发了《新一代人工智能发展规划》



选题背景（2）

- 机器人
  - 可编程的机械装置
  - 通常具有感知、规划和行动等模块
- 智能机器人
  - 具有适应环境和自主学习的能力
  - 能执行基于人工智能算法的应用程序
  - 需要在应用程序的驱动下执行人们布置的任务



问题的提出

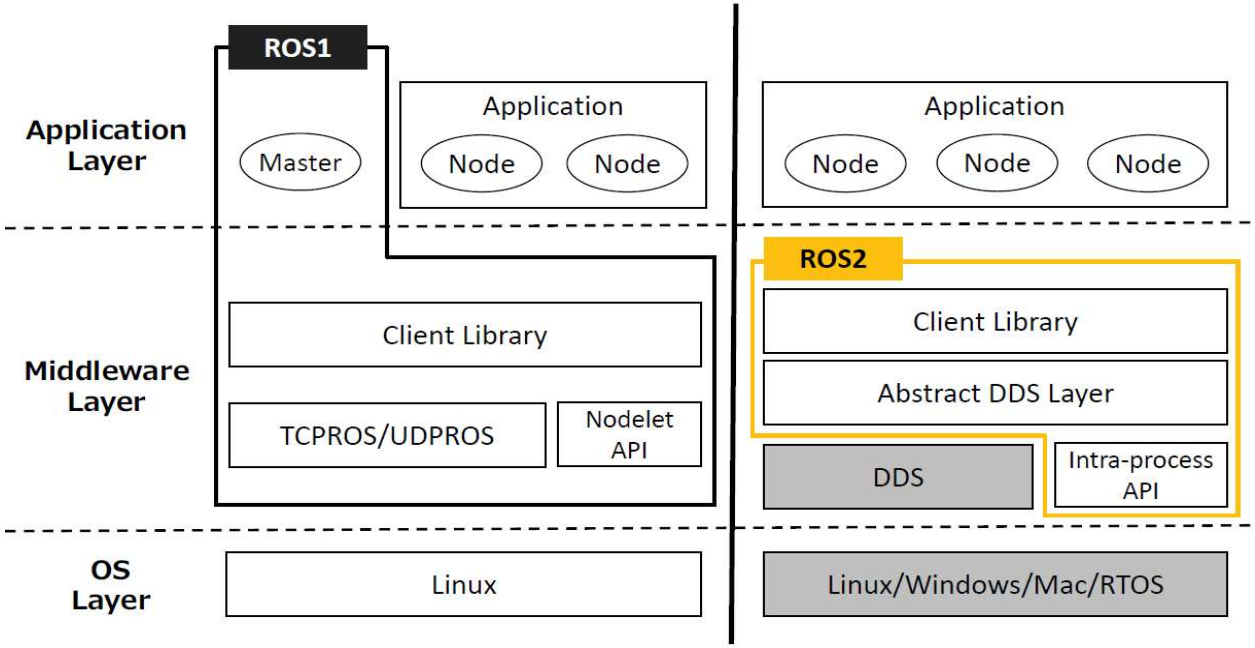
- 智能机器人与智能机器人应用程序一体化
  - 不便直接在真实智能机器人上执行应用程序
    - 对于厂商：智能机器人本身的研发和智能机器人应用程序的开发通常**同步**进行
    - 对于用户：在购买智能机器人前，希望能先**试验**智能机器人应用程序
    - 对于智能机器人：直接执行复杂的或难以从理论上预测结果的应用程序是**不安全**的
  - 不同类型的智能机器人应用程序的代码**可复用性**较差
    - 不同类型智能机器人在结构和功能上存在差异
    - 不同厂商使用不同的硬件或驱动

挑战

- 需要一种便捷和低成本的测试和验证智能机器人应用程序的虚拟环境
  - 在虚拟环境下经过**测试**和**验证**的智能机器人应用程序可以便捷地**移植**到真实智能机器人
  - 高度**还原**真实智能机器人执行应用程序时的环境
  - **扩展**至支持不同场景下的不同类型的智能机器人

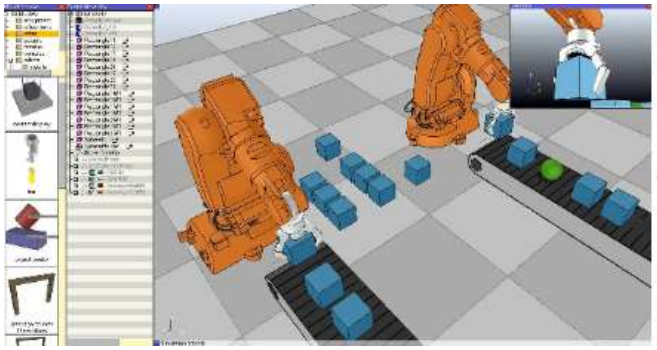
目录

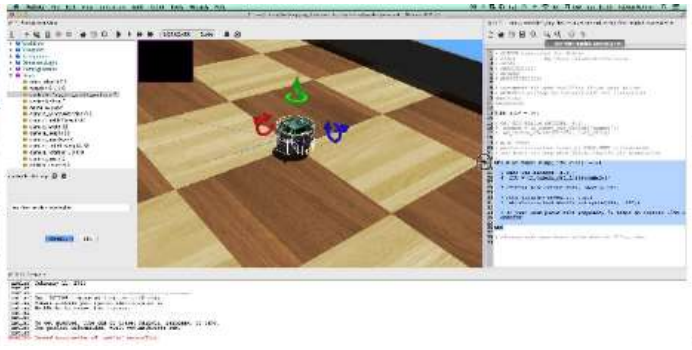
- **ROS**（*Robot Operating System*）是目前最流行的智能机器人编程模型之一 *[ICRA'09]*
  - 设计理念：提高机器人应用程序开发领域代码的复用率
  - ROS的应用程序由一系列称为“节点”的独立进程组成
  - 节点之间的通信基于发布/订阅模型
    - 节点通过话题传递消息

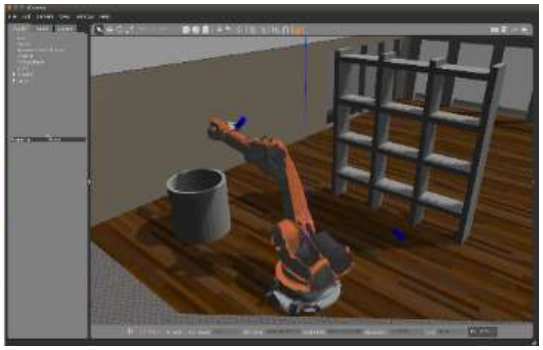


流行的智能机器人模拟器

- 机器人仿真平台**V-REP** *[IROS'13]*
- 移动机器人仿真软件**Webots** *[IJARS'04]*
- 机器人仿真软件**Gazebo** *[IROS'04]*
- 人型机器人平台**OpenHRP** *[IJRR'04]*
- 多机器人仿真平台**MORSE** *[ICRA'11]*
- 工业机器人仿真平台**OpenRAVE** *[CMU-RI-TR'08]*







- 用户在考虑选用某一款智能机器人模拟器时，仿真效果通常是最重要的 *[arXiv'14]*

基于游戏引擎的智能机器人模拟

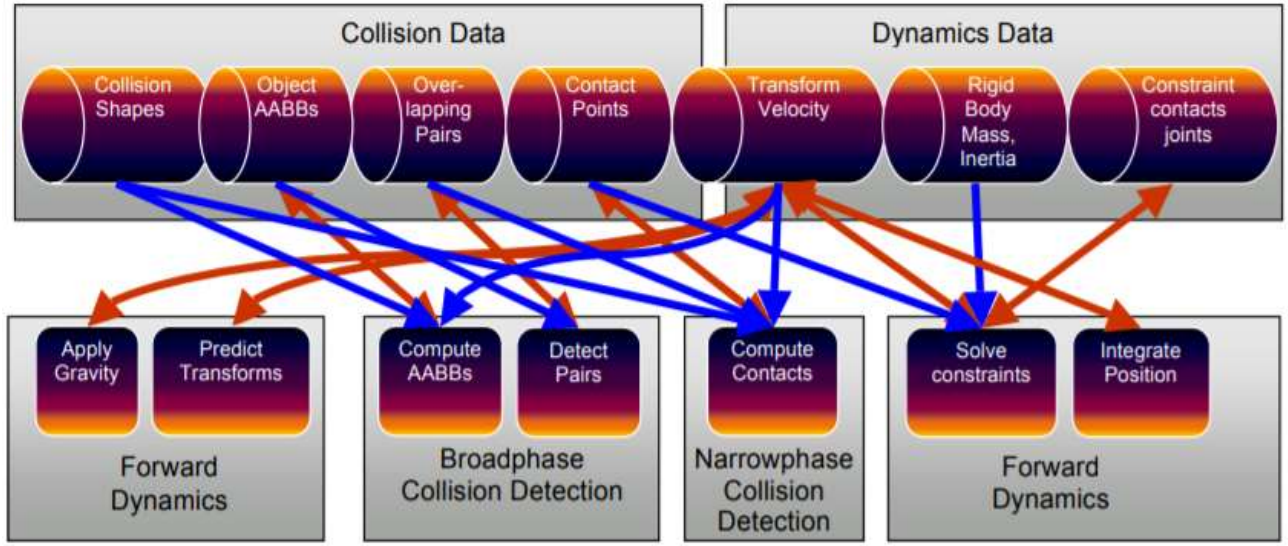
❑ 游戏引擎

- ➢ 具有一定**通用性**的交互式实时图形应用程序的核心组件
- ➢ 流行的游戏引擎：**Unity**和**虚幻**引擎（*Unreal Engine*）等
- ➢ 常用架构：**ECS**（*Entity-Component-System*）

❑ 基于游戏引擎的智能机器人模拟相关工作

- ➢ 基于**虚幻引擎2**的多机器人仿真平台**USARSim** ***[ROBOT'07]***
- ➢ 基于ROS和**Unity**的无人机集群导航和控制模拟器**ROS+Unity** ***[IES'15]***
- ➢ 基于**虚幻引擎4**的城市自动驾驶模拟器**CARLA** ***[CoRL'17]***

物理仿真系统

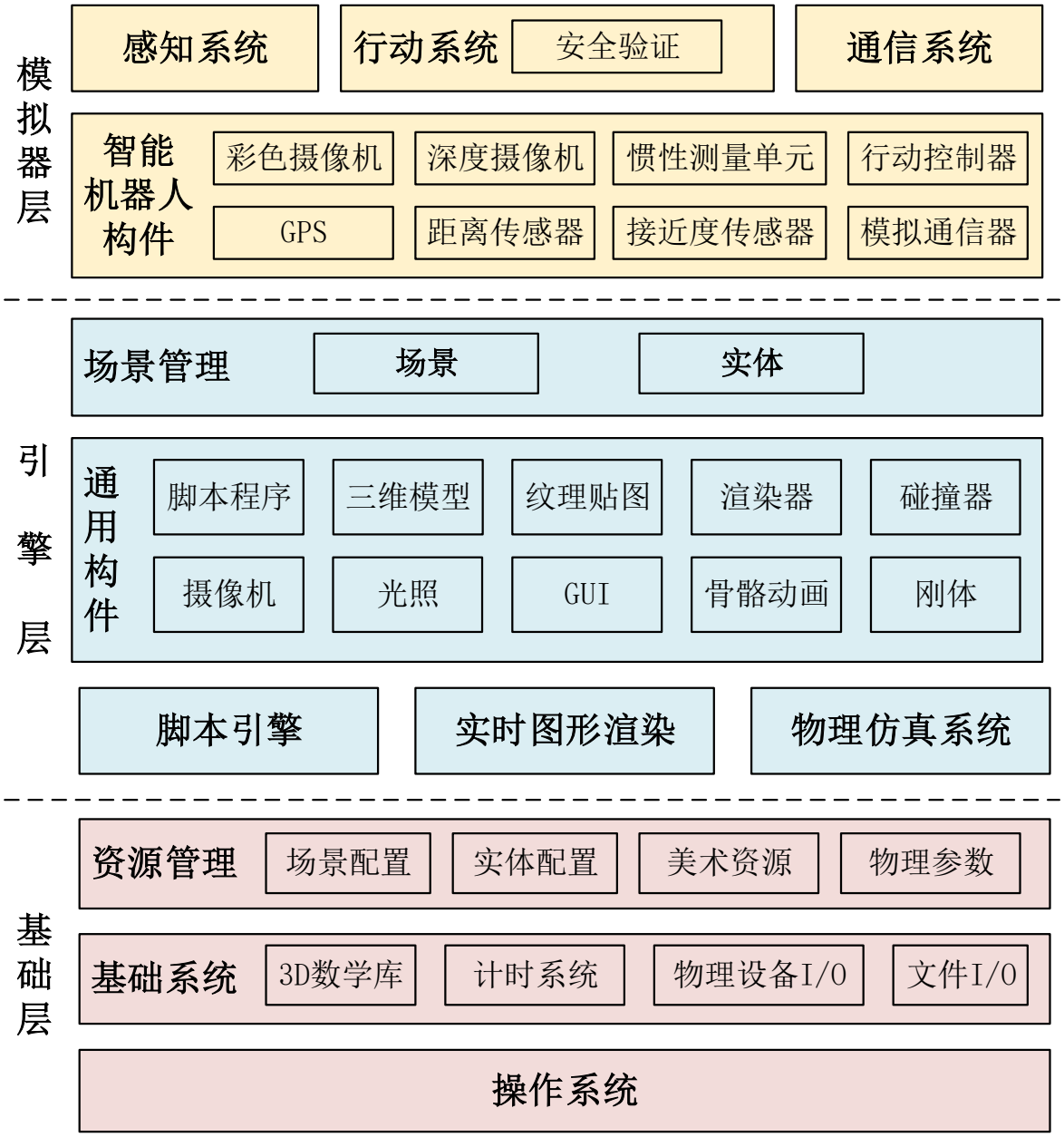
- 在机器人模拟中，物理仿真通常不可或缺
  - 物理仿真系统是一种在计算机软件中进行**碰撞检测**和**牛顿力学模拟**的系统
  - 流行的物理仿真系统：Havok、PhysX、Bullet、ODE和Mujoco
    - 结合对各款物理仿真系统的仿真效果、运算效率、易用性、使用成本等因素的对比分析 *[ICRA'15]*，本文将使用**Bullet**物理仿真系统
    - 

目录

主要工作

- 可编程智能机器人模拟器
  - 模拟真实智能机器人及其所处场景
  - 测试、验证与移植智能机器人应用程序
  - 支持不同场景下不同类型的智能机器人

系统架构


模拟器层
感知系统
行动系统
安全验证
通信系统
智能机器人构件
彩色摄像机
深度摄像机
惯性测量单元
行动控制器
GPS
距离传感器
接近度传感器
模拟通信器
引擎层
场景管理
场景
实体
通用构件
脚本程序
三维模型
纹理贴图
渲染器
碰撞器
摄像机
光照
GUI
骨骼动画
刚体
脚本引擎
实时图形渲染
物理仿真系统
基础层
资源管理
场景配置
实体配置
美术资源
物理参数
基础系统
3D数学库
计时系统
物理设备I/O
文件I/O
操作系统

模拟真实智能机器人及其所处场景（1）

- 物理与视觉模拟
  - 基于物理仿真系统模拟物理属性
    - 碰撞检测
    - 刚体动力学模拟
  - 基于3D实时图形渲染表现物体视觉
    - 渲染应具有真实感
      - 光照模型
      - 阴影绘制
    - 多个虚拟智能机器人的摄像机视角
      - 层次细节算法和视锥体裁剪算法等图形学算法
      - 图形渲染线程

模拟真实智能机器人及其所处场景（2）

❑ 智能机器人特性模拟

➢ 单体智能机器人的感知

传感器		在虚拟环境的工作原理
视觉	彩色摄像机	实时图形渲染产生的彩色图，GPU与CPU同步
	深度摄像机	实时图形渲染产生的深度缓冲图，GPU与CPU同步
距离	距离传感器	物理仿真系统的射线投射检测或扫描体检测
	接近度传感器	物理仿真系统的射线投射检测或扫描体检测
运动学	惯性测量单元	物理仿真系统的刚体动力学模拟的运动参数

➢ 单体智能机器人的行动

- 行动模块由一系列行动单元组成
  - 移动：改变虚拟智能机器人在物理仿真系统中对应的刚体的运动状态

➢ 智能机器人集群的通信

- 自定义通信规则

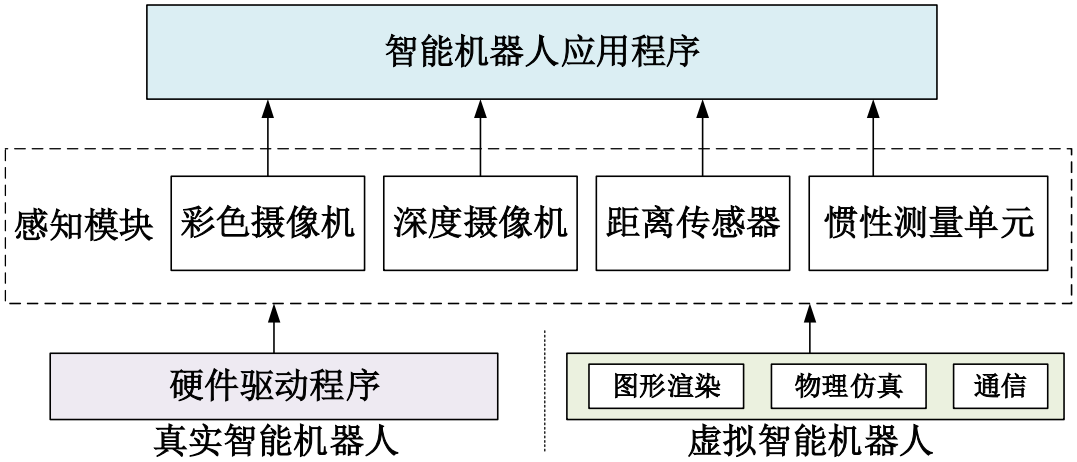
测试、验证和移植智能机器人应用程序（1）

- 设计理念
  - 智能机器人受应用的驱动
  - 智能机器人应用程序相互独立

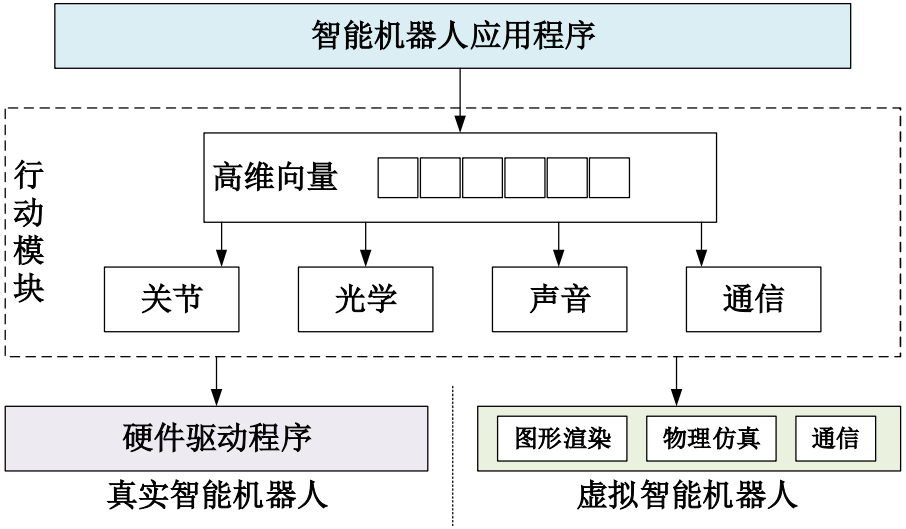
- 相互独立的脚本程序
  - 系统通过脚本引擎执行脚本程序
    - 提供编程库
  - 每种应用程序作为一个类
    - 执行该应用程序的智能机器人拥有该类的一个对象
    - 应用程序的逻辑在类的成员方法中实现

测试、验证和移植智能机器人应用程序（2）

- 便于移植智能机器人应用程序的智能机器人编程模型
  - 感知模块的抽象：面向不同类型传感器的抽象

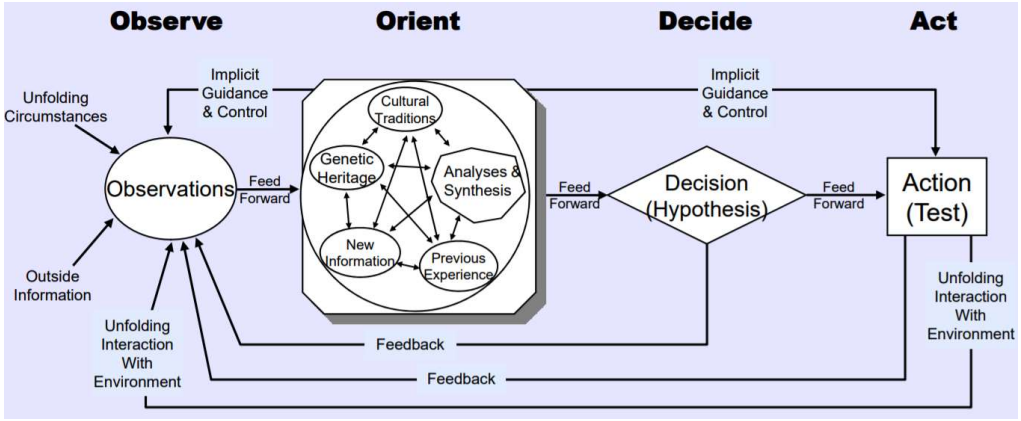


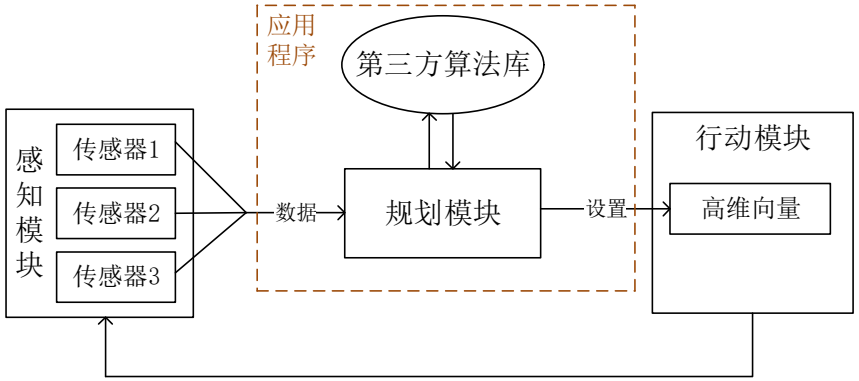
  - 行动模块的抽象：面向整体行动的抽象

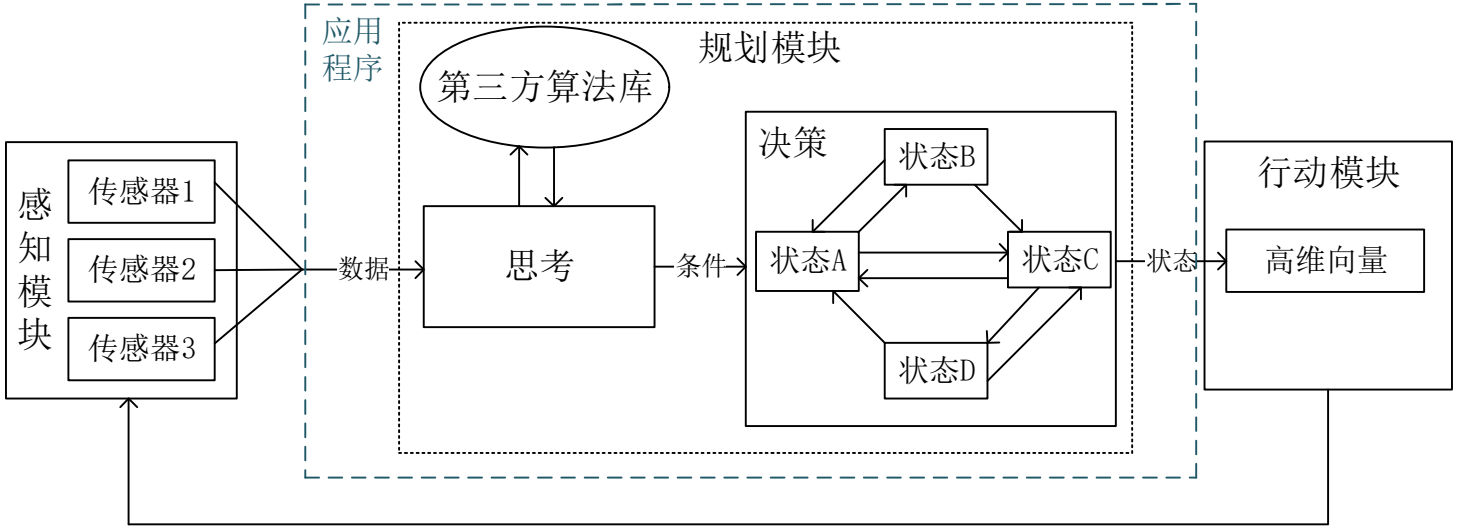


测试、验证和移植智能机器人应用程序（3）

- OODA（*Observe-Orient-Decide-Act Loop*）循环
- 交互式编程思路
- 交互-触发式编程思路





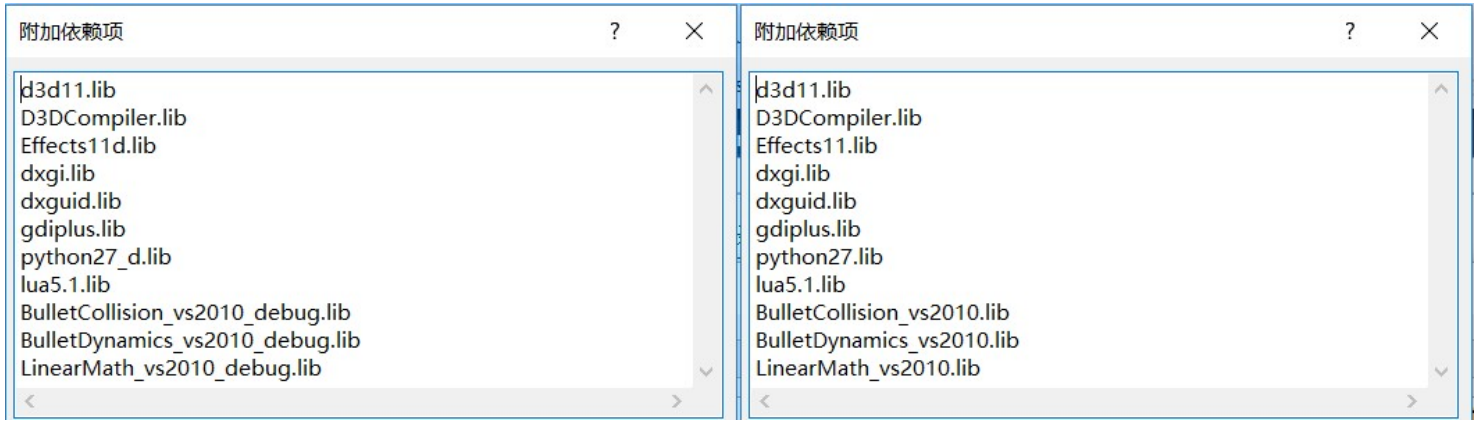


支持不同场景下不同类型智能机器人的模拟

- 基于**ECS**架构的实体和构件，抽象地描述智能机器人及其所处场景
  - 实体：一个真实物体的整体或一部分
    - 实体在场景中以**树状结构**组织
  - 构件：描述实体的属性或行为
    - 实体由若干构件构成
      - 通用构件：美术资源、物理参数、脚本程序等
      - 智能机器人构件：传感器、行动控制器、通信器等
- 场景和实体的配置信息
  - 构件：类型+描述串
  - 实体：子结点实体+构件
  - 场景：若干实体

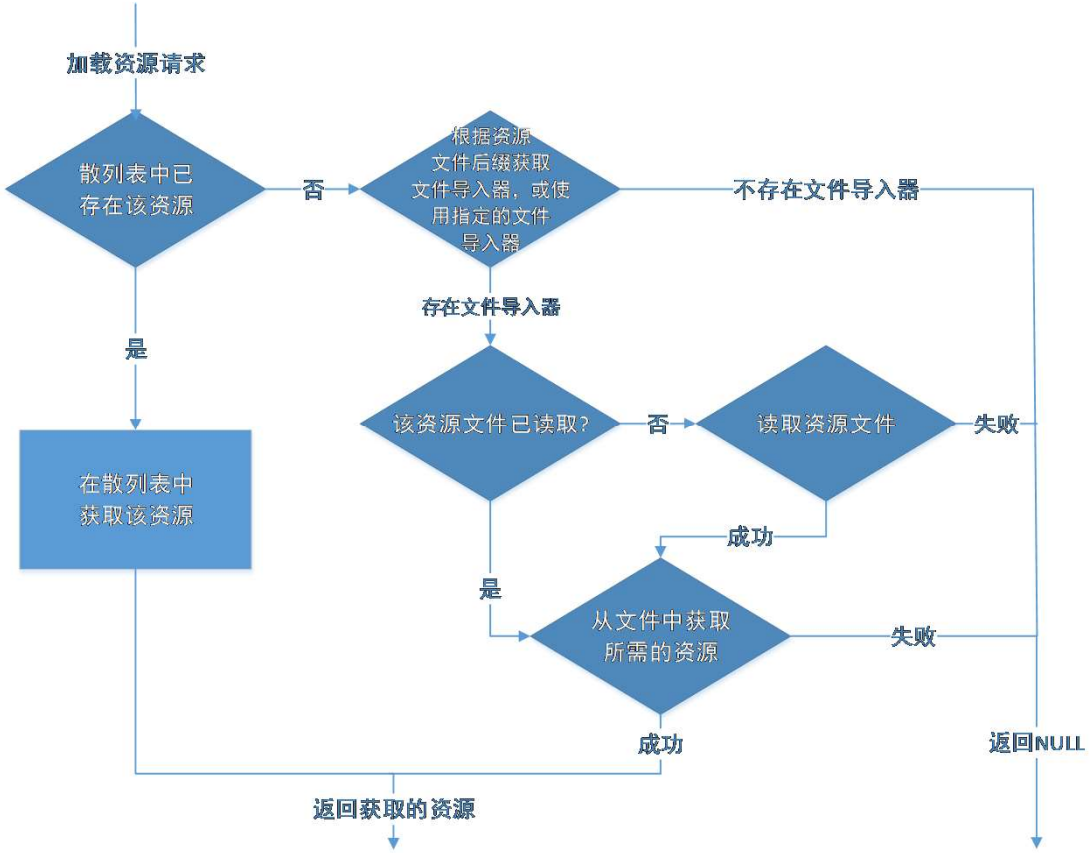
目录

系统实现

- 操作系统：Windows 7/8/10
- 开发工具：Microsoft Visual Studio 2015/2017
  - 语言：C/C++
- 运行所需的第三方库
  - Direct3D 11
  - Python 2.7
  - Lua 5.1
- 开发配置
  - 

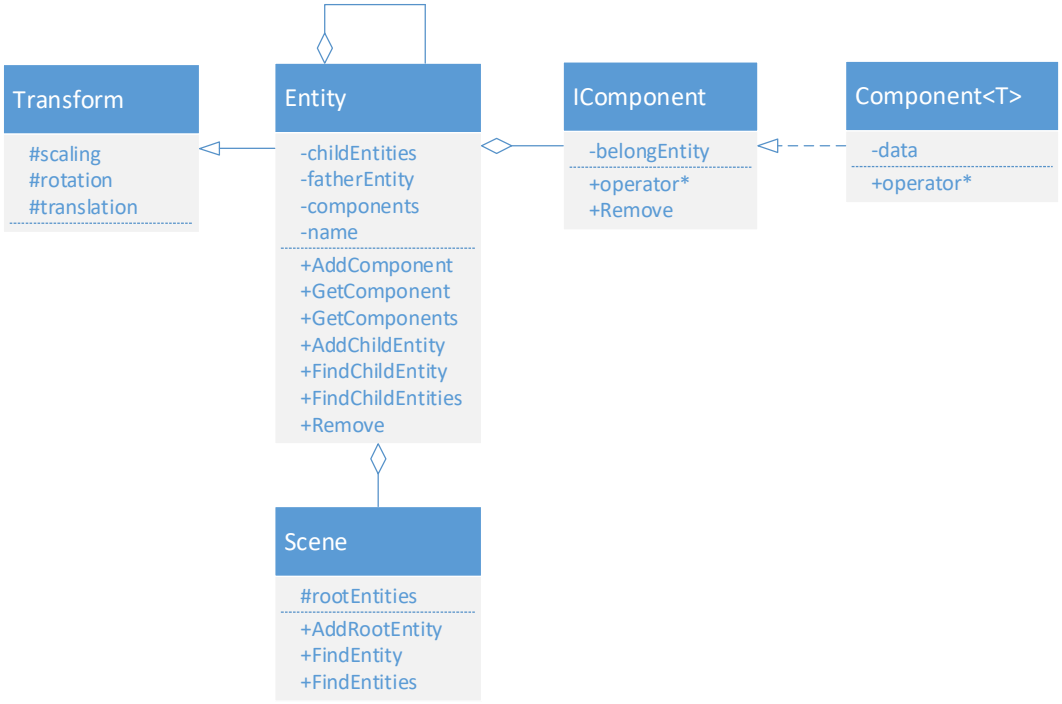
基础层

- 基于DirectXMath的3D数学库
  - 封装常用的数据类型：Vector3、Quaternion、Matrix4x3
- 资源加载与管理
  - 不重复地从磁盘中读取相同的文件和把资源文件中的数据转化为需要用到的数据
  - 配置信息：Lua的table



引擎层

- **ECS架构**
- **基于Direct3D 11的实时图形渲染**
  - Render To Texture
  - 真实感：基于Lambert光照模型和阴影贴图
- **基于Bullet的物理仿真**
  - 碰撞器构件和刚体构件
  - 提供碰撞检测查询的接口和碰撞测试的接口
- **基于Python的脚本引擎**
  - 提供编程模块：coolengine和robotsimulator
  - Script构件与脚本交互
  - Python控制台



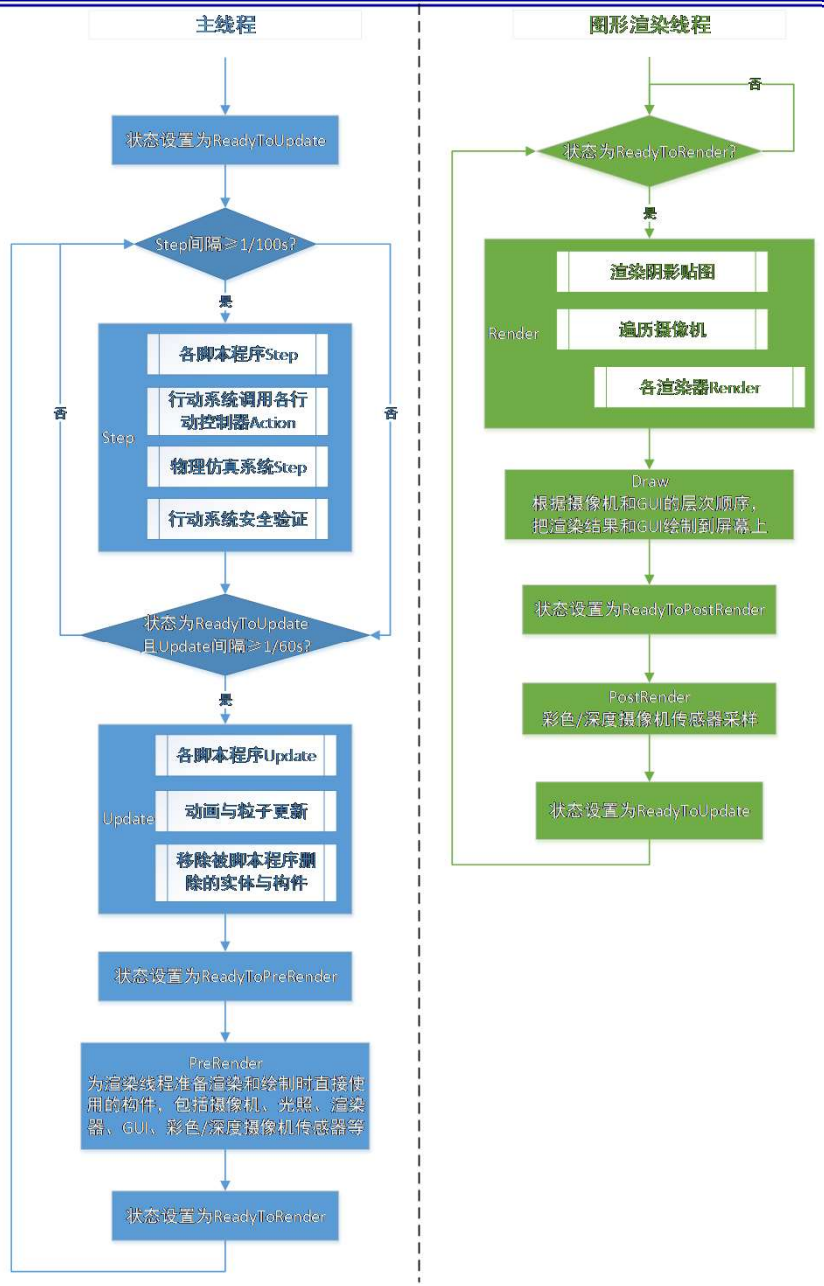

```python
import coolengine as ce
import robotsimulator as sim

class safetyInfo:
    def __init__(self, entity, param = None):
        self.entity = entity

    def start(self):
        self.text = ce.text(self.entity.getComponent("Text", "Safety"))

    def update(self, dt, param = None):
        unsafe = sim.getUnsafe()
        if not len(unsafe):
            self.text.setContent("All Safe")
        else:
            content = "Unsafe:\n"
            for ue in unsafe:
                content += ue.getName()+"\n"
            self.text.setContent(content)
```

模拟器层

- 虚拟智能机器人的感知
  - 每种类型的虚拟传感器分别作为一种构件，分别对外提供获取感知信息的接口
- 虚拟智能机器人的行动
  - 行动控制器作为虚拟智能机器人的构件，通过高维向量描述虚拟智能机器人当前的行动，对外提供设置高维向量的接口
    - 行动控制器基于高维向量调用更底层的系统，驱动虚拟智能机器人的行动
- 虚拟智能机器人集群的通信
  - 通信器作为虚拟智能机器人的一种构件，与通信系统交互
    - 接收消息类似传感器的感知，发送消息由行动控制器进行控制
  - 通信系统允许在外部脚本中自定义通信规则

系统运行架构

- 初始化
  - 引擎层初始化
    - 注册通用构件
    - 资源管理器初始化
    - 渲染管理器初始化
    - 物理管理器初始化
    - Python脚本引擎初始化
  - 模拟器层初始化
    - 注册虚拟智能机器人构件
    - 添加自定义渲染器和着色器
    - 添加自定义行动控制器
  - 加载场景
    - simscene.lua的Scene table
- 主循环
  - 主线程
  - 图形渲染线程



目录

e-puck2移动机器人

- **e-puck2是GCtronic和洛桑联邦理工学院研发的最新款小型移动机器人**
  - 大小与直径7.3cm、高度4.3cm的圆柱体相仿
  - 包含两个轮子，可以向前/向后运动和转向
  - 提供八个分别朝向不同方向的红外接近传感器、惯性测量单元、CMOS摄像机、ToF距离传感器等
- **e-puck2使用C语言编程**
  - 目前只有Webots支持e-puck2的仿真

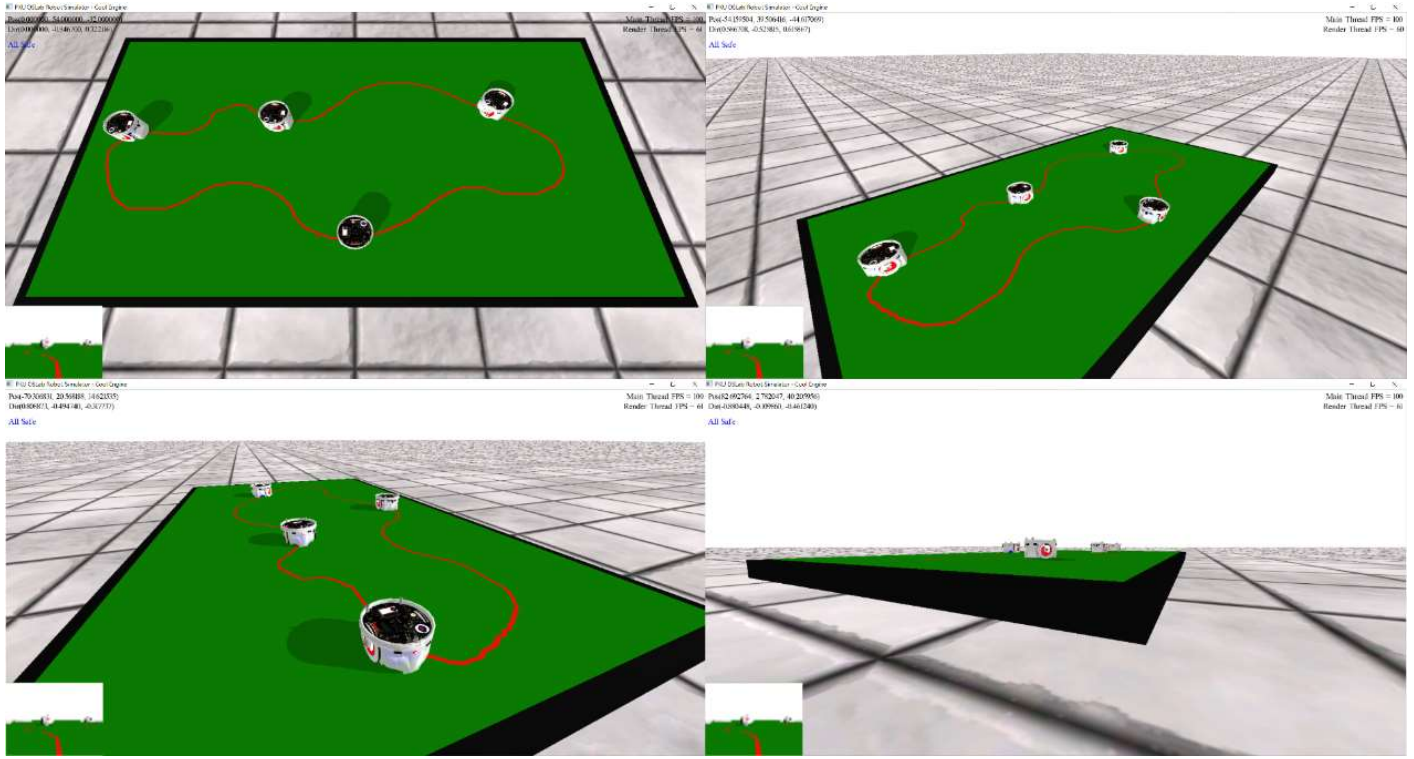
e-puck2的仿真

- **e-puck2实体**
  - 包含由一对轮子组成的子实体
  - 由三维模型、纹理贴图、虚拟传感器、行动控制器、通信器等构件构成

```
Entity = {
    Child = {
        { Config = "@epuck2.lua:Entity_Wheel", Name = "Epuck2Wheel", T = "0 2 0" },
        { Config = "@epuck2.lua:Entity_Camera", Name = "Epuck2Camera", T = "0 2.78 3.475"},
        { Config = "@epuck2.lua:Entity_ToFDist", Name = "Epuck2ToF", T = "0 4 3.63"}, -- Front
        { Config = "@epuck2.lua:Entity_IRProx", Name = "Epuck2IR0", T = "1 3.3 3", Q = "0 0.149832 0 0.9
        { Config = "@epuck2.lua:Entity_IRProx", Name = "Epuck2IR1", T = "2.5 3.3 2.2", Q = "0 0.389785 0
        { Config = "@epuck2.lua:Entity_IRProx", Name = "Epuck2IR2", T = "3.1 3.3 0", Q = "0 0.707107 0 0
        { Config = "@epuck2.lua:Entity_IRProx", Name = "Epuck2IR3", T = "1.5 3.3 -3", Q = "0 -0.969207 0
        { Config = "@epuck2.lua:Entity_IRProx", Name = "Epuck2IR4", T = "-1.5 3.3 -3", Q = "0 -0.968616
        { Config = "@epuck2.lua:Entity_IRProx", Name = "Epuck2IR5", T = "-3.1 3.3 0", Q = "0 -0.707107 0
        { Config = "@epuck2.lua:Entity_IRProx", Name = "Epuck2IR6", T = "-2.5 3.3 2.2", Q = "0 -0.389052
        { Config = "@epuck2.lua:Entity_IRProx", Name = "Epuck2IR7", T = "-1 3.3 3", Q = "0 -0.149044 0 0
    },
    Comp = {
        {"Mesh", "@epuck2body.lua:Mesh"},
        {"Material", "@epuck2.lua:Material_Body"},
        {"Texture", "DiffuseMap", "@epuck2.png"},
        {"Renderer", "EpuckRenderer", "1 1 1050"},
        {"Collider", "@epuck2.lua:Collider_OnGround"},
        {"Rigidbody", "@epuck2.lua:Rigidbody"},
        {"IMUSensor"},
        {"Communicator"},
        {"ActionController", "EpuckActionController", "15 2"},
    }
}
```

模拟实验场景

- 实验平台：模拟飞利浦65PUF6051/T3智能电视
- 若干个虚拟e-puck2
- 全局系统
  - 提供全局光照、用户自由视角摄像机、帧率显示、安全验证信息显示等



e-puck2避障巡线模拟实验

- 实验目标
  - 左上方的虚拟e-puck2（名为“e-puck2_00001”）沿红色曲线前进
  - 其他虚拟e-puck2作为障碍物。e-puck2_00001在前进时需要避开障碍物
- 实验方法
  - 巡线：获取e-puck2彩色摄像机当前图像，根据图像最底一行的R通道值，确定e-puck2当前的行动方向，设置行动控制器的高维向量
  - 避障：通过距离传感器感知到与障碍物的距离，做圆周运动绕过障碍物，并回到线上
- 实验演示
  - e-puck2避障巡线.mp4

e-puck2集群接力巡线模拟实验

- 实验目标
  - 一个e-puck2沿红色曲线前进。当其距离传感器感知到前方存在障碍物时，发出消息并停止前进。接收到消息的e-puck2开始前进
  - 消息只能被限定范围内的e-puck2接收，且有较小延迟
- 实验演示
  - e-puck2集群接力巡线.mp4

目录

总结

- 可编程智能机器人模拟器的设计
  - 模拟真实智能机器人及其所处场景
  - 智能机器人应用程序的测试、验证和移植
  - 支持不同场景下不同类型智能机器人的模拟
- 可编程智能机器人模拟器的实现
- 面向**e-puck2**移动机器人的模拟实验
  - e-puck2避障巡线
  - e-puck2集群接力巡线

未来工作展望

- 扩展引擎层的通用性
  - 操作系统：Linux、macOS等
  - 图形API：OpenGL等
- 完善引擎层的功能
  - 加入音效系统
  - 支持更多的物理仿真系统，如Havok、PhysX等
- 更便捷的模型与场景编辑
  - 可视化编辑工具
- 预设更多的虚拟智能机器人
  - NAO、Kobuki等
- 支持流行的机器人编程模型
  - ROS等