物理仿真及PeriDyno开源引擎架构简介

何小伟

中国科学院软件研究所

2023.3.26

大纲

- 课程安排
- 物理仿真概述
- 物理仿真引擎概述
- 工欲善其事,必先利其器
  - Git: SmartGit/SourceTree
  - CMake
  - Visual Studio Code
  - Visual Studio 2017/2019/2021
- Hello PeriDyno!
  - PeriDyno引擎架构
  - 仿真案例展示

课程安排

	名称	讲者
第一讲	物理仿真及PeriDyno开源引擎架构简介	何小伟 副研究员 中科院软件所
第二讲	GPU硬件架构简介及CUDA编程基础	
第三讲	计算机图形学常用几何工具及数学原理	
第四讲	刚体动力学并行编程与实践	
第五讲	光滑粒子动力学（SPH）并行编程与实践	
第六讲	近场动力学（Peridynamics）并行编程与实践	
第七讲	PeriDyno插件与功能拓展	
第八讲	Vulkan编程原理及通用并行计算	
第九讲	工程CAE仿真连续介质力学基础	蔡勇 副教授 湖南大学
第十讲	工程CAE仿真中的有限元分析原理	
第十一讲	从PeriDyno到MxSimLab	
第十二讲	基于MxSimLab的CAE软件增量集成开发	
第十三讲	基于MxSimLab的多物理场问题CAE分析应用	

2023年3月26日起，北京时间 每周日 20:00-21:30

课程安排

- 编程、答疑

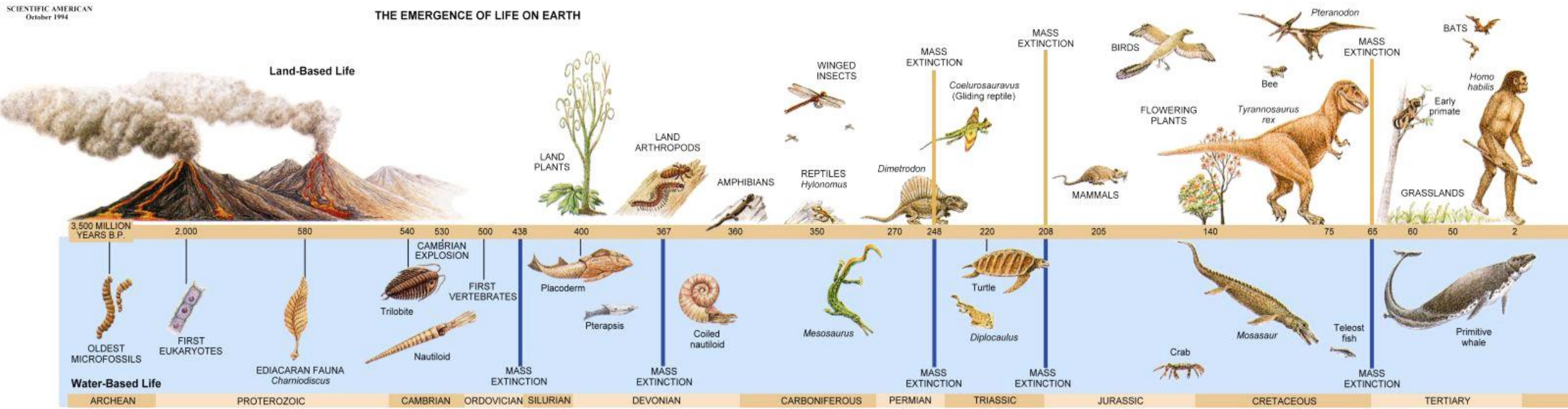
课程面向对象

- 适合
  - 对仿真感兴趣，希望有一个能快速上手做科研的平台
  - 对开源有热情，可以容忍当前的不完美，对未知充满求知欲
  - 喜欢交朋友，希望通过该课程找到志同道合的小伙伴
  - 数理基础好，几何直观强

- 不适合
  - 对编程不感兴趣，或者只熟悉python，不想学C/C++等
  - 看到数学公式就头疼
  - 寻找一个成熟的仿真解决方案

物理仿真概述

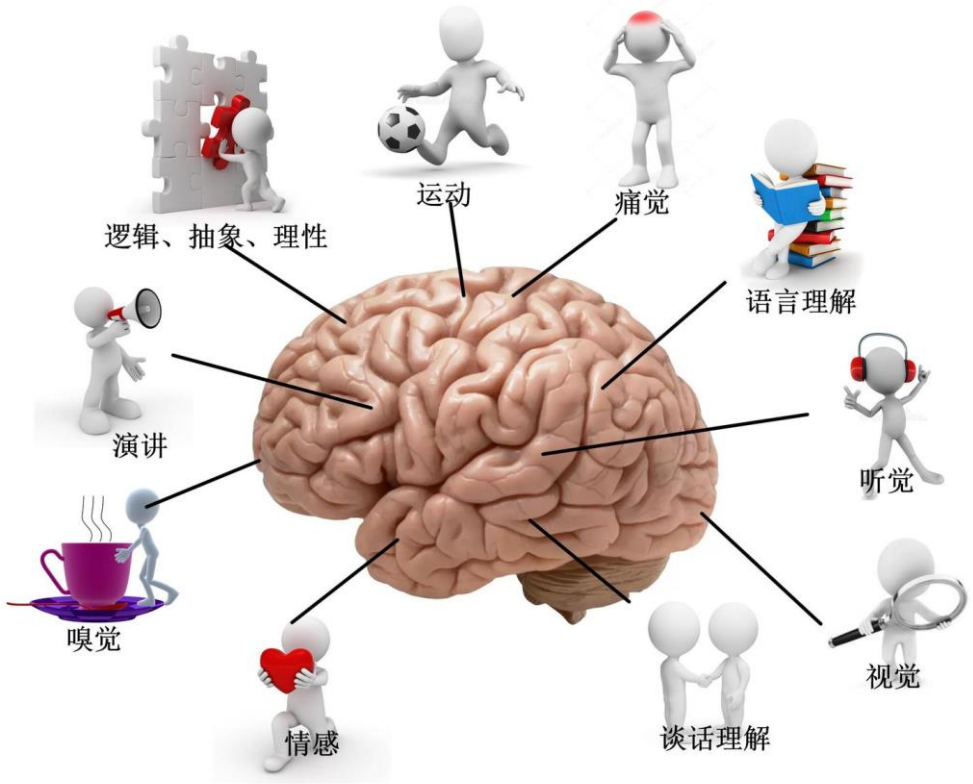
精神世界（人工智能）



物质世界（物理仿真）

物理仿真概述

精神世界



物质世界

人工智能

物理仿真


物理仿真能做什么?
@命运2

物理仿真概述

农场里的火鸡?

The world is not real

The world is Simulated

物理仿真概述


@黑客帝国

物理仿真引擎概述

- 物理仿真引擎的作用是什么
  - 我是小白，要实现一个仿真算法该从哪开始入手；
  - 我是做仿真算法的，不想写{渲染、UI、导入导出。。。} 算法；
  - 审稿意见回来了，Reviewer让和某某论文做一下对比；

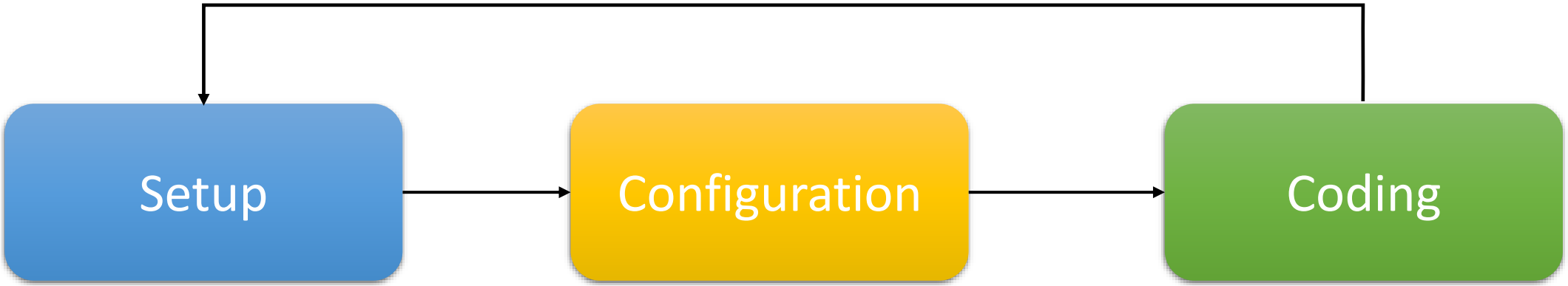
- 物理仿真引擎能提供什么
  - 基础的脚手架工程
  - 与仿真无关的功能模块
  - 其他仿真算法

物理仿真引擎概述

PeriDyno
系统架构图

工欲善其事，必先利其器

- 开源三部曲（以Windows平台为例）



Git

CMake

Visual Studio Code

Visual Studio 2017/2019/2022

Linux平台参见： http://peridyno.com/zh/installation/linux/

推荐配置

- Windows 10+
- 显卡：GeForce RTX 10系列 / 20系列 / 30系列 / 40系列
  - CUDA Toolkit 11.0+（推荐CUDA Toolkit 11.5）

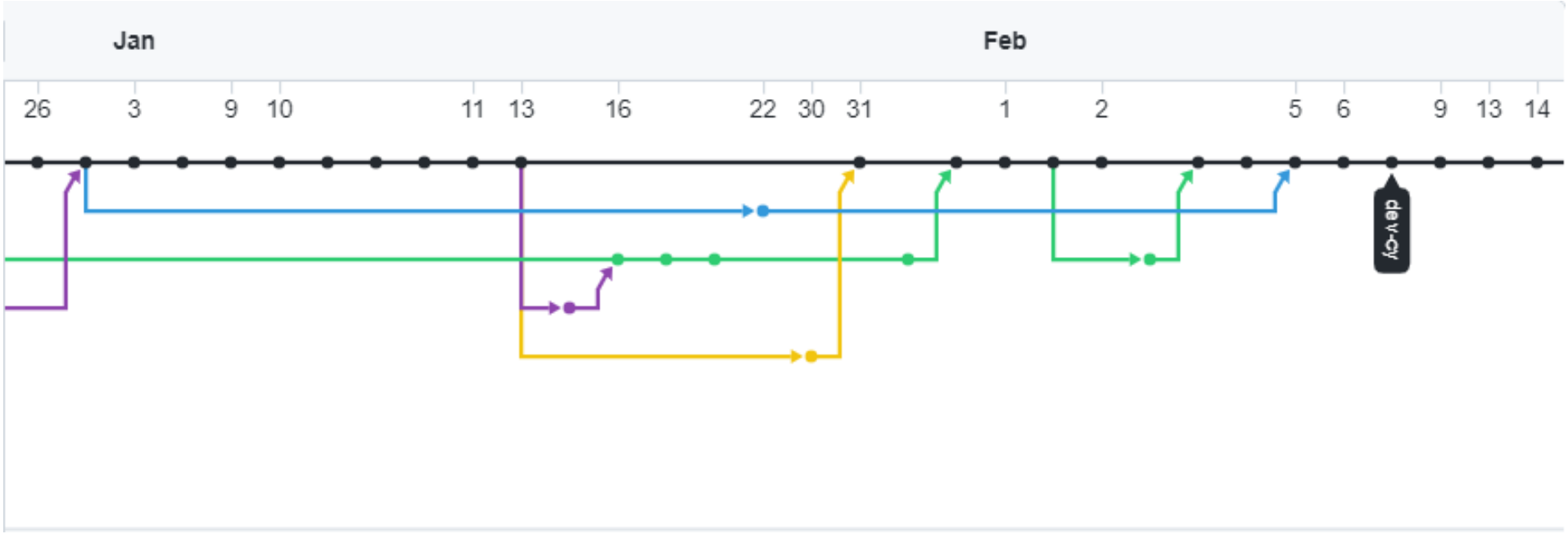
Git

- **Git** is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
  - A complete long-term change history of every file.
  - Branching and merging.
  - Traceability.

https://git-scm.com/downloads

Git

- **Git** is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.



Git Commands

- Setting up a repository
  - Git init/clone
  - git clone --recursive https://github.com/peridyno/peridyno.git
  - git clone -recursive https://gitee.com/peridyno/peridyno.git
- Collaborating
  - Syncing: git remote/fetch/pull/push
  - Using branches: git branch/checkout/merge

https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud

Git clone

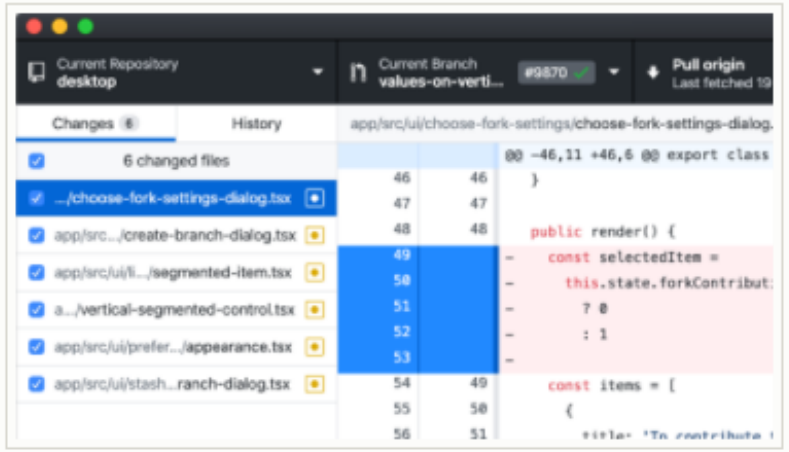
- cmake //Cmake脚本
- data //数据
- examples //仿真样例
- external //第三方库
- src
  - Core //基础数据结构和算法库
  - Dynamics //仿真算法库
  - Framework //仿真框架
  - Plugin //插件
  - Rendering //渲染引擎、GUI
  - Topology //拓扑结构
- tests //测试样例

Git GUI

Git comes with built-in GUI tools for committing (git-gui) and browsing (gitk), but there are several third-party tools for users looking for platform-specific experience.

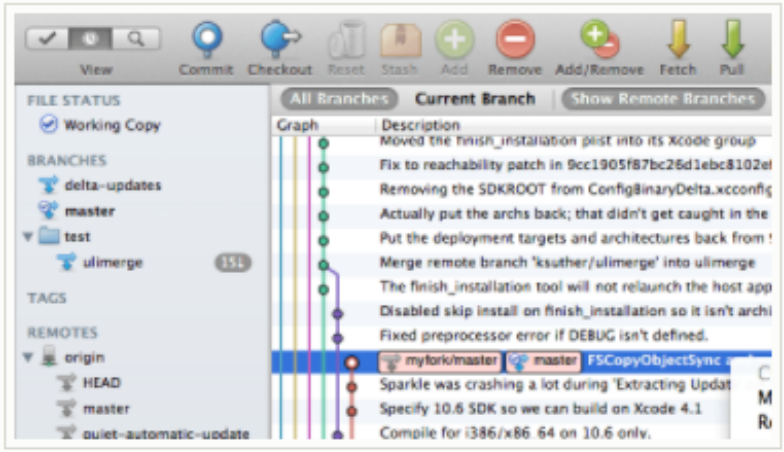
If you want to add another GUI tool to this list, just follow the instructions.

All | Windows | Mac | Linux | Android | iOS



GitHub Desktop

Platforms: Mac, Windows
Price: Free
License: MIT



SourceTree

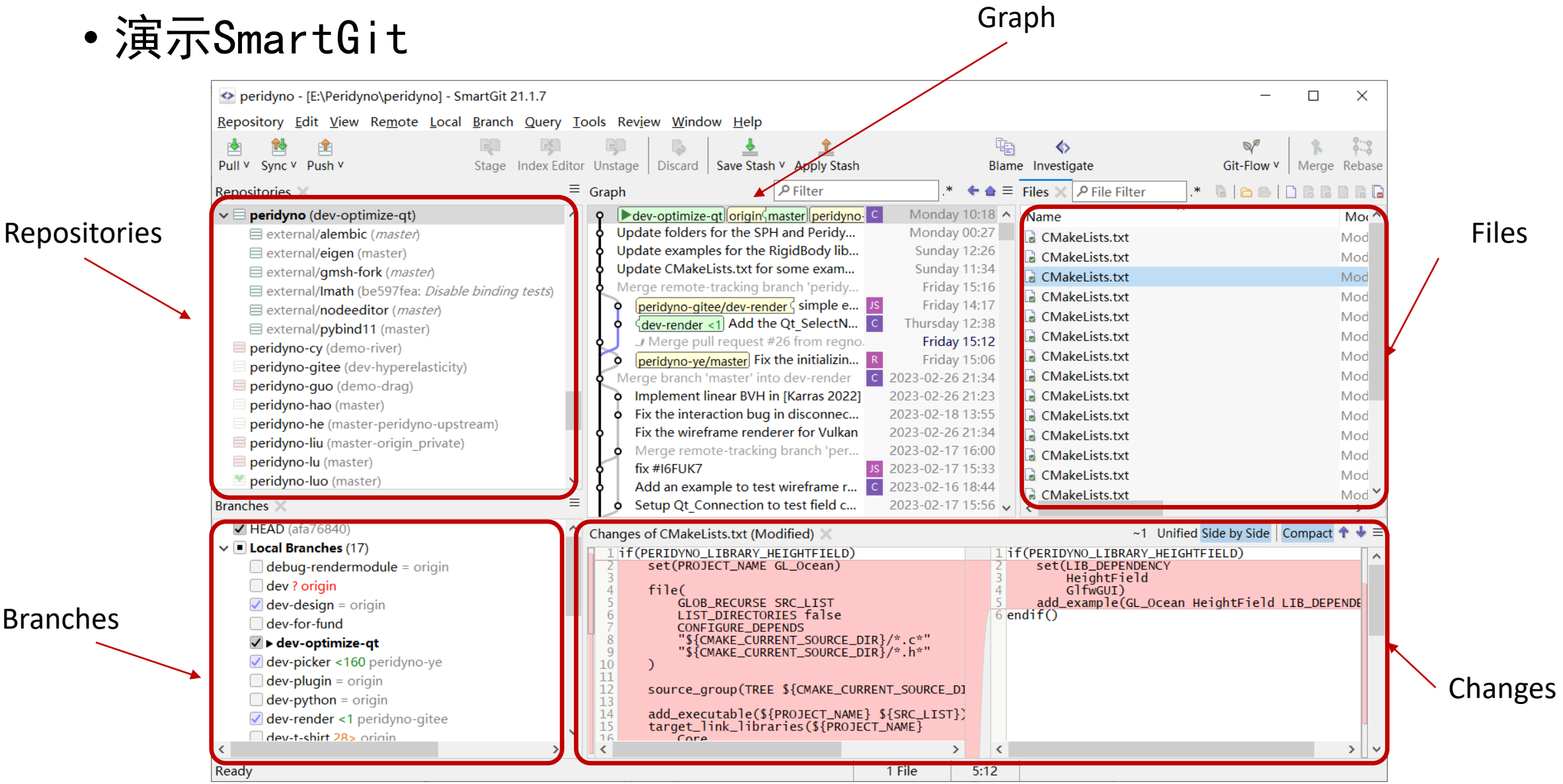
Platforms: Mac, Windows
Price: Free
License: Proprietary

SmartGit

Platforms: Linux, Mac, Windows
Price: Free for non-commercial use / $59/user annually
License: Proprietary

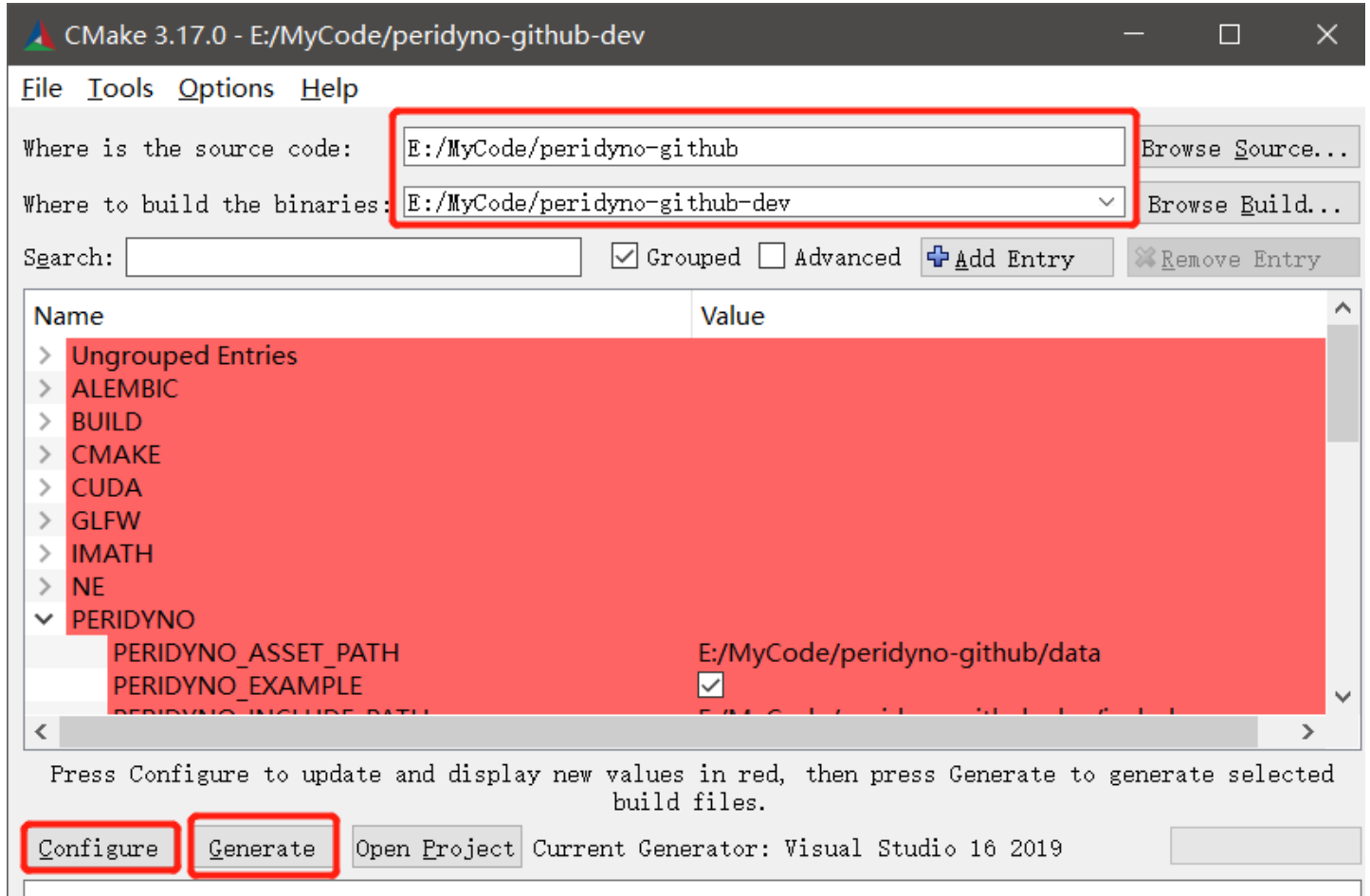
Git

- 演示SmartGit



Configuration

- CMake is an open-source, **cross-platform** family of tools designed to build, test and package software.
  - 设置文件源码路径和编译路径
  - 点击Configure
  - 点击Generate



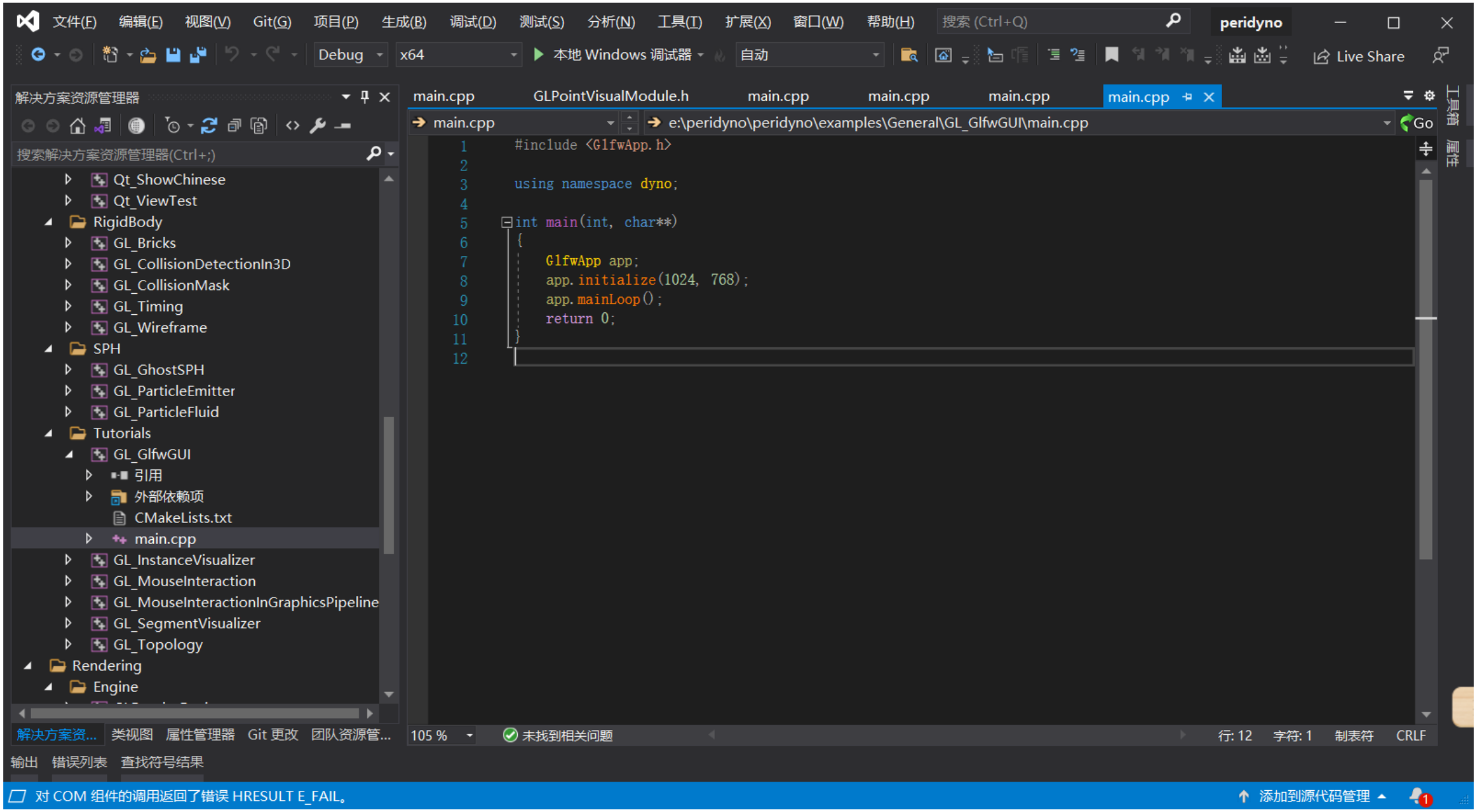
Configuration

- Examples/General/GL_GlfwGUI/CMakeLists.txt

```
set(PROJECT_NAME GL_GlfwGUI)

set(LIB_SRC main.cpp)
source_group(TREE ${CMAKE_CURRENT_SOURCE_DIR} FILES ${LIB_SRC})

add_executable(${PROJECT_NAME} ${LIB_SRC})

set(GUI_ONLY_BINARIES ${PROJECT_NAME})

target_link_libraries(${PROJECT_NAME} GlfwGUI)
```

- Visual Studio Code

Coding


文件(F) 编辑(E) 视图(V) Git(G) 项目(P) 生成(B) 调试(D) 测试(S) 分析(N) 工具(T) 扩展(X) 窗口(W) 帮助(H)
搜索 (Ctrl+Q)
peridyno
Debug
x64
本地 Windows 调试器
自动
Live Share
解决方案资源管理器
搜索解决方案资源管理器(Ctrl+;)
Qt_ShowChinese
Qt_ViewTest
RigidBody
GL_Bricks
GL_CollisionDetectionIn3D
GL_CollisionMask
GL_Timing
GL_Wireframe
SPH
GL_GhostSPH
GL_ParticleEmitter
GL_ParticleFluid
Tutorials
GL_GlfwGUI
引用
外部依赖项
CMakeLists.txt
main.cpp
GL_InstanceVisualizer
GL_MouseInteraction
GL_MouseInteractionInGraphicsPipeline
GL_SegmentVisualizer
GL_Topology
Rendering
Engine
main.cpp
GLPointVisualModule.h
main.cpp
main.cpp
main.cpp
main.cpp
e:\peridyno\peridyno\examples\General\GL_GlfwGUI\main.cpp
Go
#include <GlfwApp.h>
using namespace dyno;
int main(int, char**)
{
GlfwApp app;
app.initialize(1024, 768);
app.mainLoop();
return 0;
}
105 %
未找到相关问题
行: 12
字符: 1
制表符
CRLF
解决方案资...
类视图
属性管理器
Git 更改
团队资源管...
输出
错误列表
查找符号结果
对 COM 组件的调用返回了错误 HRESULT E_FAIL。
添加到源代码管理
工具箱
属性

Hello PeriDyno !!!

- 创建文件夹
  - */examples/Cuda/Tutorials/Hello

Hello PeriDyno !!!

- 创建文件夹
  - */examples/Cuda/Tutorials/Hello
- 创建main.cpp

```cpp
#include <GlfwApp.h>

using namespace dyno;

int main(int, char**)
{
    GlfwApp app;
    app.initialize(1024, 768);
    app.mainLoop();
    return 0;
}
```

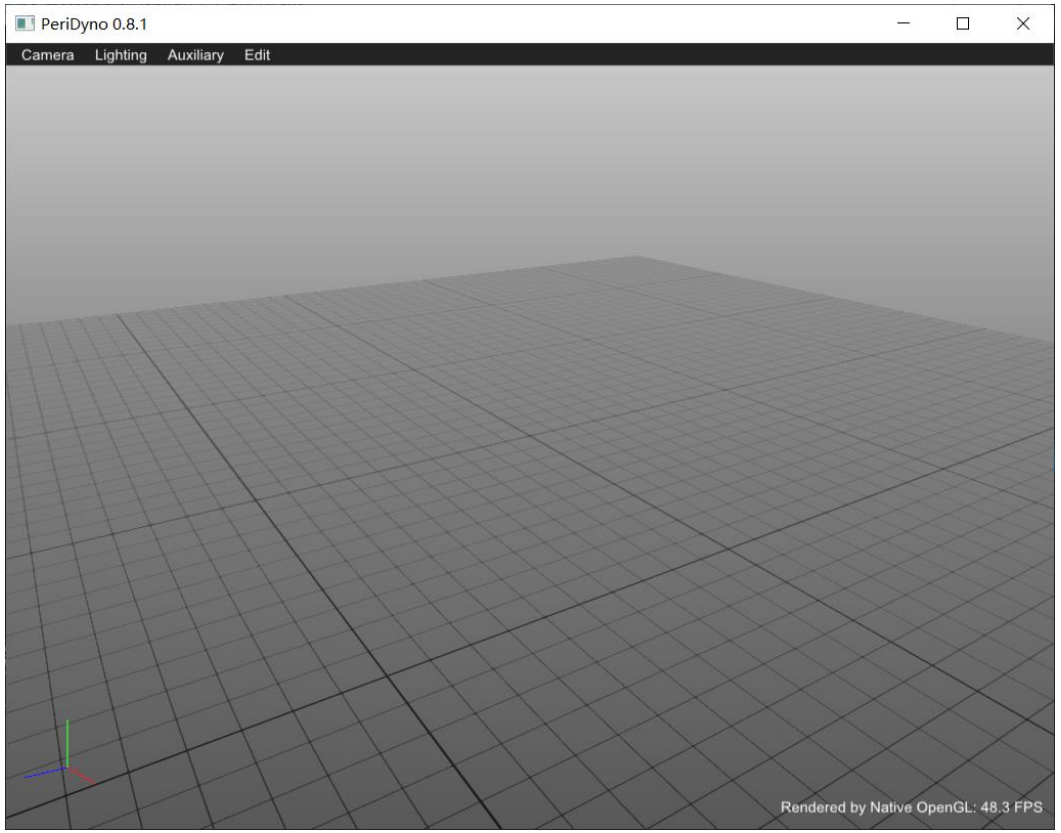
Hello PeriDyno !!!

- 创建文件夹
  - */examples/Cuda/Tutorials/Hello
- 创建main.cpp
- 创建CMakeLists.txt

```
set(LIB_DEPENDENCY GlfwGUI)
add_example(Hello Tutorials LIB_DEPENDENCY)
```

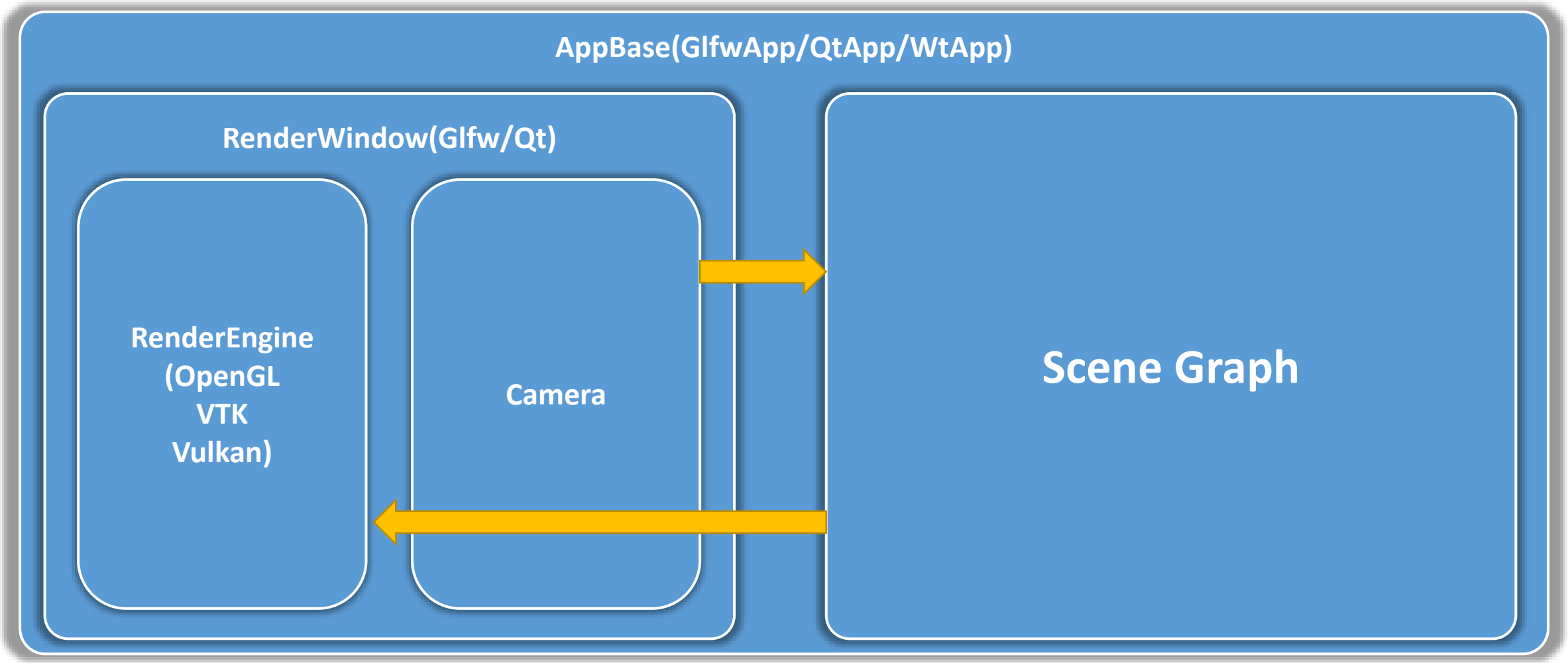
Hello PeriDyno !!!

- 创建文件夹
  - */examples/Cuda/Tutorials/Hello
- 创建main.cpp
- 创建CMakeLists.txt
- CMake GUI
  - Configure->Generate->Open Project



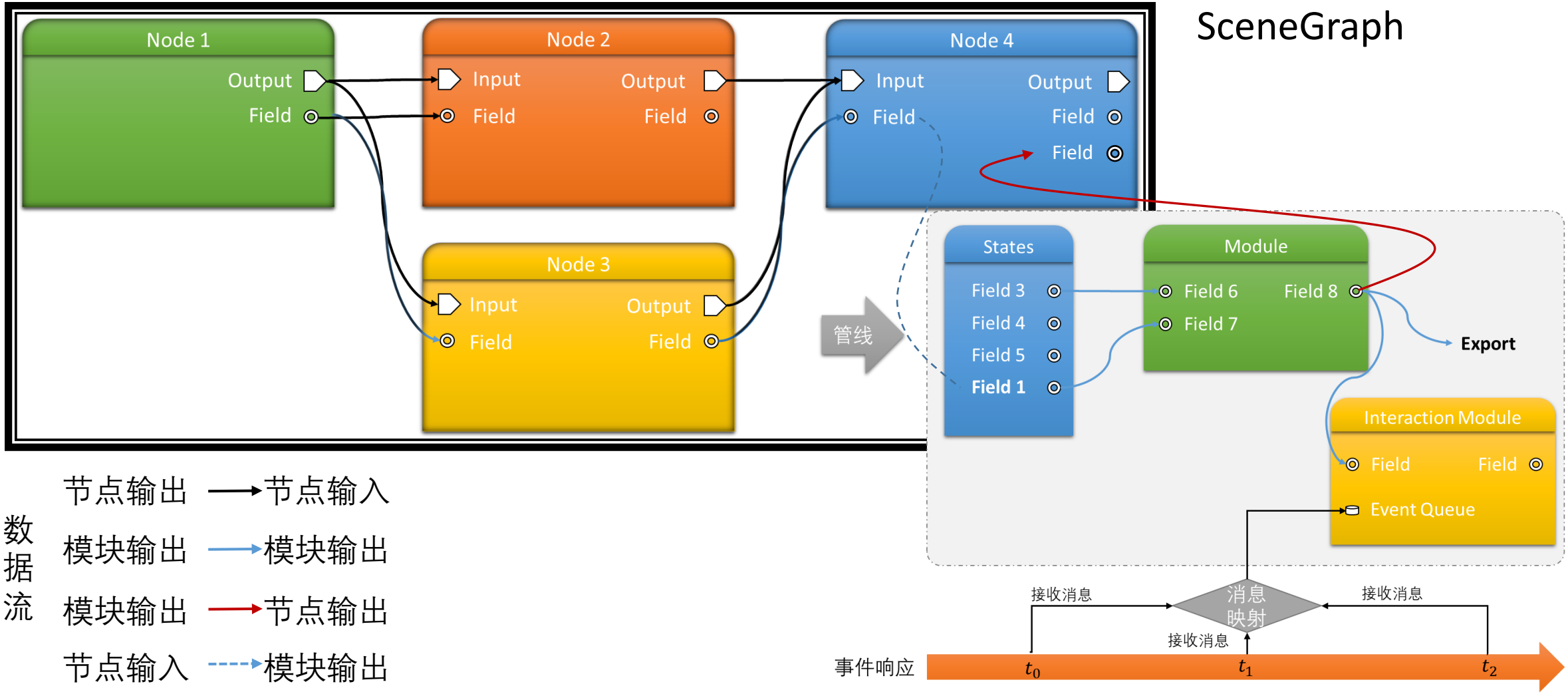
PeriDyno物理仿真框架

- void AppBase::setSceneGraph(std::shared_ptr<SceneGraph> scene);



PeriDyno物理仿真框架

- Field -> Module -> Node -> SceneGraph



Field类型

- 对基本数据类型的封装，用于实现节点以及模块之间的数据传递。按数据类型划分包括：
  - **基本类型**：指数据量比较小，可以在CPU和GPU进行直接传递的数据，典型的包括bool、int、float、double等标量数据以及Vec3f、Vec3d、Mat3f、Mat3d等张量数据，Field可通过DEF_VAR、DEF_ENUM等宏定义来完成数据的定义和传递；
  - **静态数组**：指数据尺寸固定的连续存储空间，包括一维数组、二维数组、三维数组等，Field宏定义采用DEF_ARRAY{2D, 3D}_{*}等宏定义。
  - **动态数组**：指数据尺寸动态变化的连续存储空间，包括List、Map、Set、MultiSet、Pair、Stack等类型的数组表示，Field宏定义采用DEF_ARRAYLIST_{*}完成定义。
  - **引用类型**：针对复杂类型，采用类似C++指针的方式进行定义的数据，Field宏定义采用DEF_INSTANCE_{*}完成定义。

Field类型

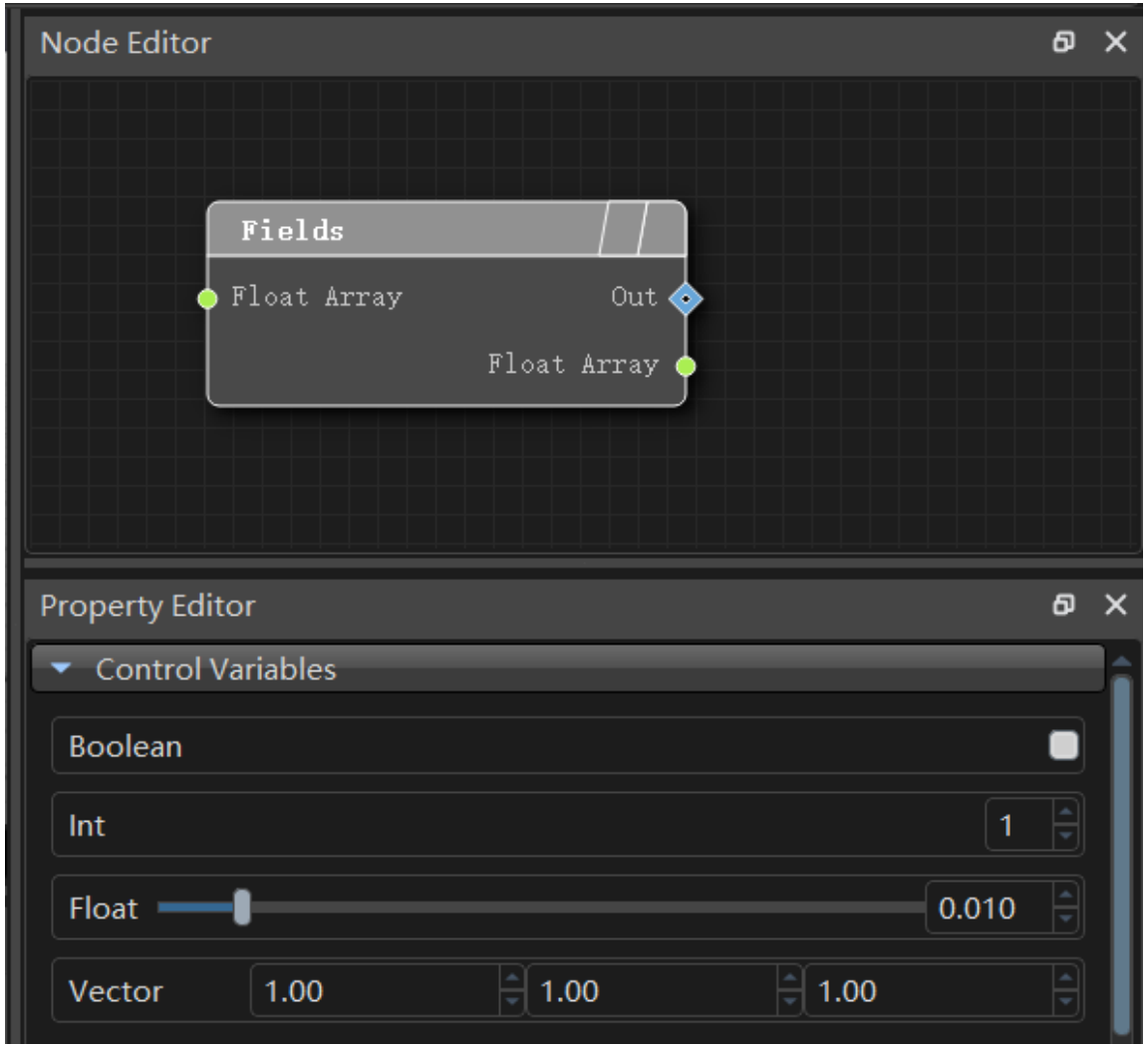
- 按用途划分包括：
  - 变量

    DEF_VAR (type, name, value, description);

    DEF_ENUM (type, name, value, description);
  - 输入
    - DEF_{*}_IN (type, name, description);
  - 输出
    - DEF_{*}_OUT (type, name, description);
  - 状态
    - DEF_{*}_STATE (type, name, description);

Field类型

```
class Fields : public Node
{
    DECLARE_CLASS(Fields);
public:
    Fields() {
        this->varFloat()->setRange(0.01, 10.0f);
        this->inFloatArray()->tagOptional(true);
    };
    ~Fields() {};

public:
    DEF_VAR(bool, Boolean, false, "Define a boolean field");

    DEF_VAR(int, Int, 1, "Define an int");

    DEF_VAR(float, Float, 1.0f, "Define a float field");

    DEF_VAR(Vec3f, Vector, Vec3f(1.0f), "Define a vector field");

    DEF_ARRAY_IN(float, FloatArray, DeviceType::GPU, "Define a float array as input");

    DEF_ARRAY_OUT(float, FloatArray, DeviceType::GPU, "Define a float array as output");

    DEF_ARRAY_STATE(float, Value, DeviceType::GPU, "Define a float array as state");

protected:
    void resetStates() {
        std::cout << "resetStates() " << " is called " << std::endl;
    }
};

IMPLEMENT_CLASS(Fields);
```



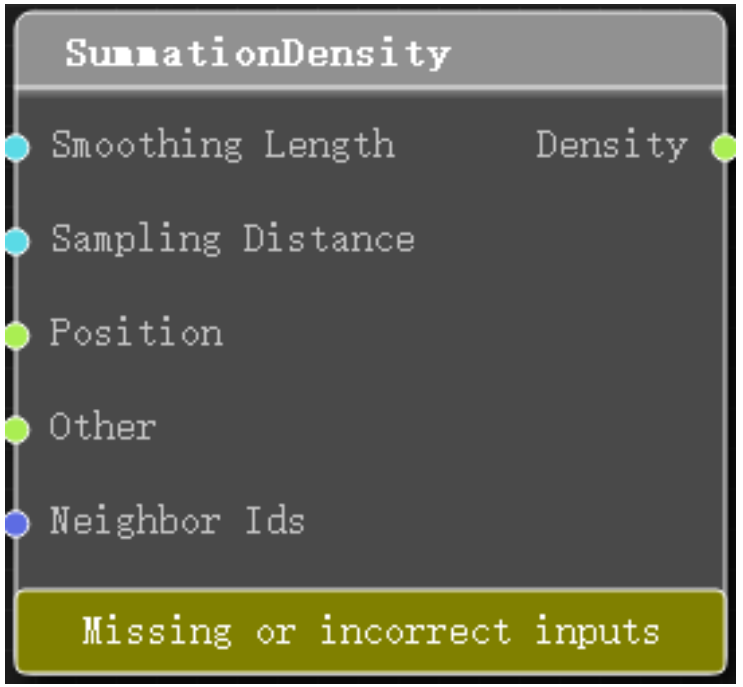
src/examples/Cuda/QtGUI/Qt_Fields

Module

- 模块(Module)定义为供节点内部调用的功能独立的算法单元，其由**输入数据**、**输出数据**和**控制变量**构成。其与节点最大的差别在于输入输出类型只接受Field类型。
  - **控制变量**：采用DEF_VAR宏定义数据；
  - **输入数据**：采用DEF_ *_ IN形式宏定义定义的数据
  - **输出数据**：采用DEF_ *_ OUT形式宏定义定义的数据。

Module

```
public:
    DEF_VAR(Real, RestDensity, 1000, "Rest Density");

    ///Define inputs
    /**
     * @brief Particle positions
     */
    DEF_ARRAY_IN(Coord, Position, DeviceType::GPU, "Particle position");

    /**
     * @brief Particle positions
     */
    DEF_ARRAY_IN(Coord, Other, DeviceType::GPU, "Particle position");

    /**
     * @brief Neighboring particles
     *
     */
    DEF_ARRAYLIST_IN(int, NeighborIds, DeviceType::GPU, "Neighboring particles' ids");

    ///Define outputs
    /**
     * @brief Particle densities
     */
    DEF_ARRAY_OUT(Real, Density, DeviceType::GPU, "Return the particle density");
```



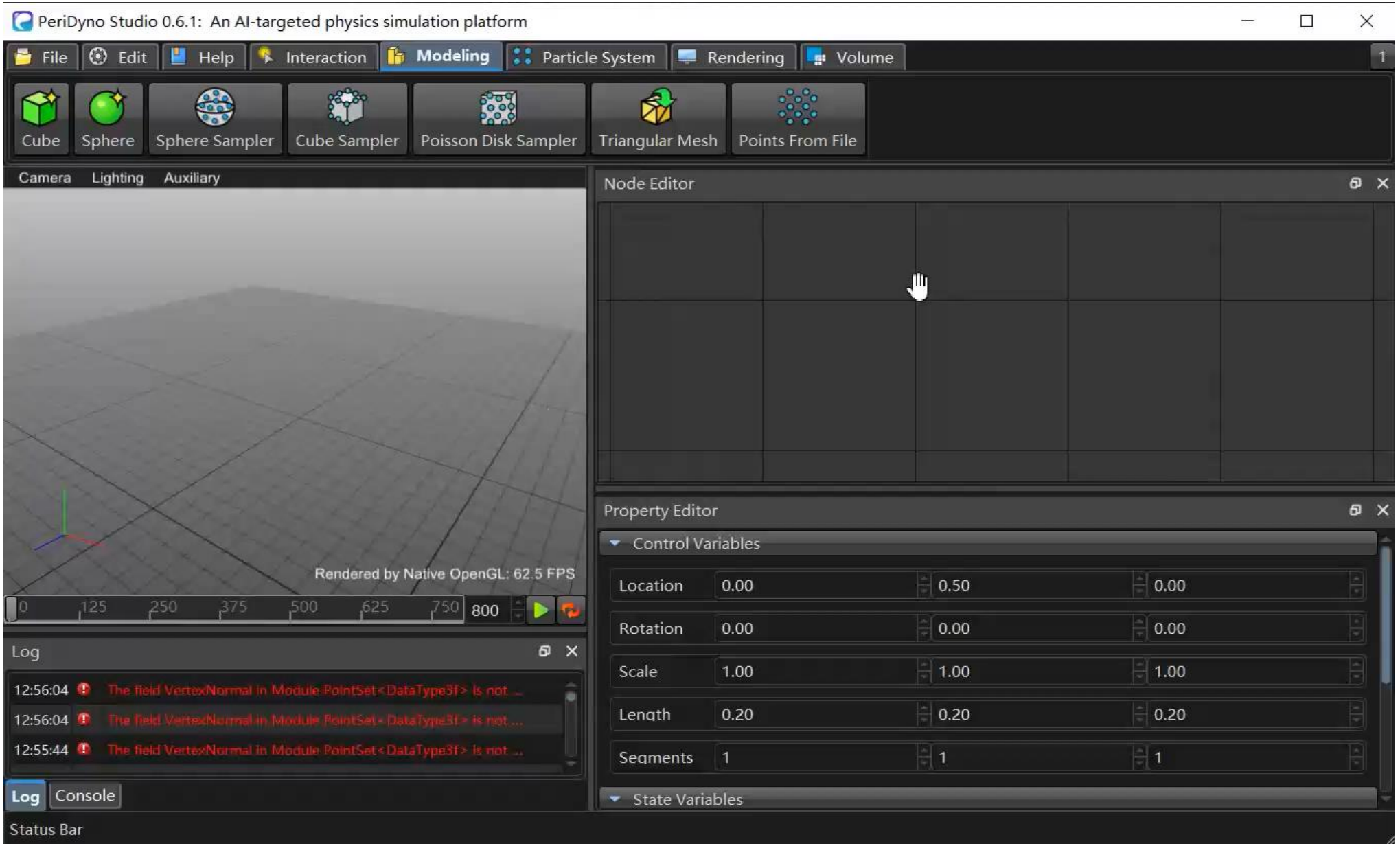
src/Dynamics/Cuda/ParticleSystem/Module/SummationDensity.h

Node

- 节点为功能相对独立的、封装了特定功能模块的算法集合，主要包含如下几个部分：
  - **控制变量**：采用DEF_VAR宏定义数据；
  - **输入节点**：利用宏定义DEF_NODE_PORT或者DEF_NODE_PORTS定义；
  - **输出节点**：当前实现只支持节点自身作为默认输出，不需显式定义；
  - **输入数据**：采用DEF_ *_ IN形式宏定义定义的数据
  - **输出数据**：采用DEF_ *_ OUT形式宏定义定义的数据；
  - **状态变量**：采用DEF_ *_ STATE形式宏定义定义的数据；
  - **模块管线**：主要包含仿真管线和渲染管线各一条，分别采用animationPipeline()和graphicsPipeline()函数进行调用。

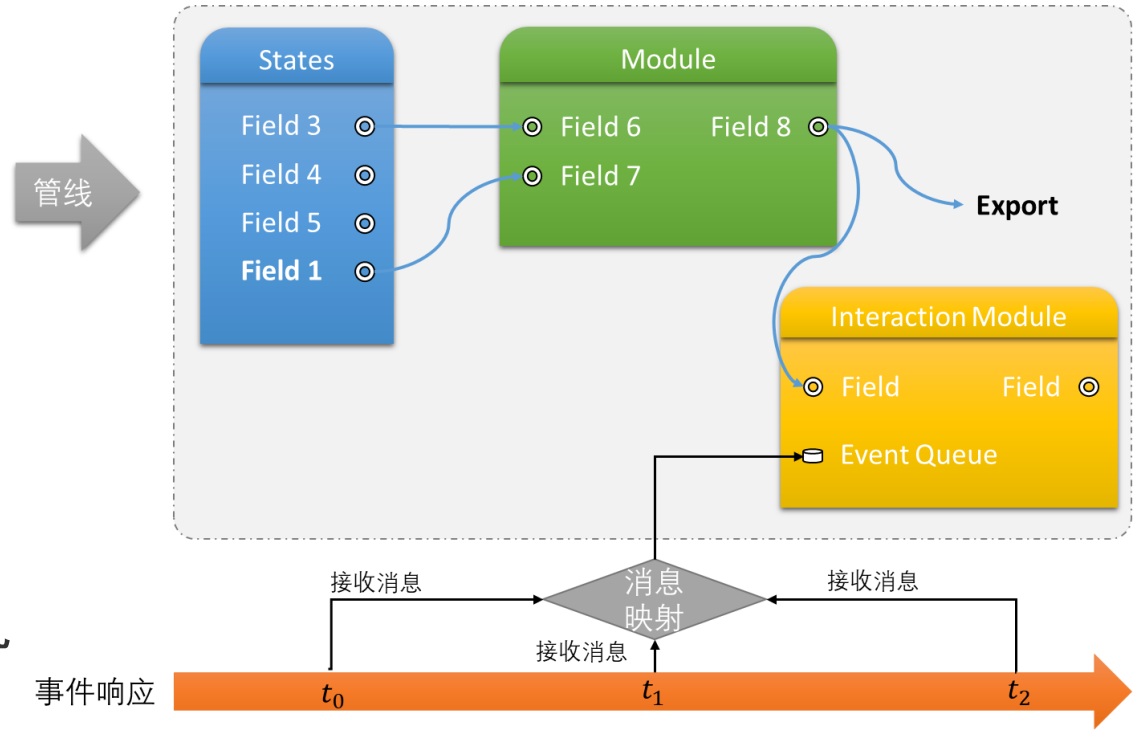
src/examples/Cuda/QtGUI/Qt_Connection

SceneGraph


PeriDyno Studio 0.6.1: An AI-targeted physics simulation platform
File
Edit
Help
Interaction
Modeling
Particle System
Rendering
Volume
Cube
Sphere
Sphere Sampler
Cube Sampler
Poisson Disk Sampler
Triangular Mesh
Points From File
Camera
Lighting
Auxiliary
Rendered by Native OpenGL: 62.5 FPS
0
125
250
375
500
625
750
800
Node Editor
Property Editor
Control Variables
Location 0.00 0.50 0.00
Rotation 0.00 0.00 0.00
Scale 1.00 1.00 1.00
Length 0.20 0.20 0.20
Segments 1 1 1
State Variables
Log
12:56:04 The field VertexNormal in Module PointSet<DataType3f> is not ...
12:56:04 The field VertexNormal in Module PointSet<DataType3f> is not ...
12:55:44 The field VertexNormal in Module PointSet<DataType3f> is not ...
Log
Console
Status Bar

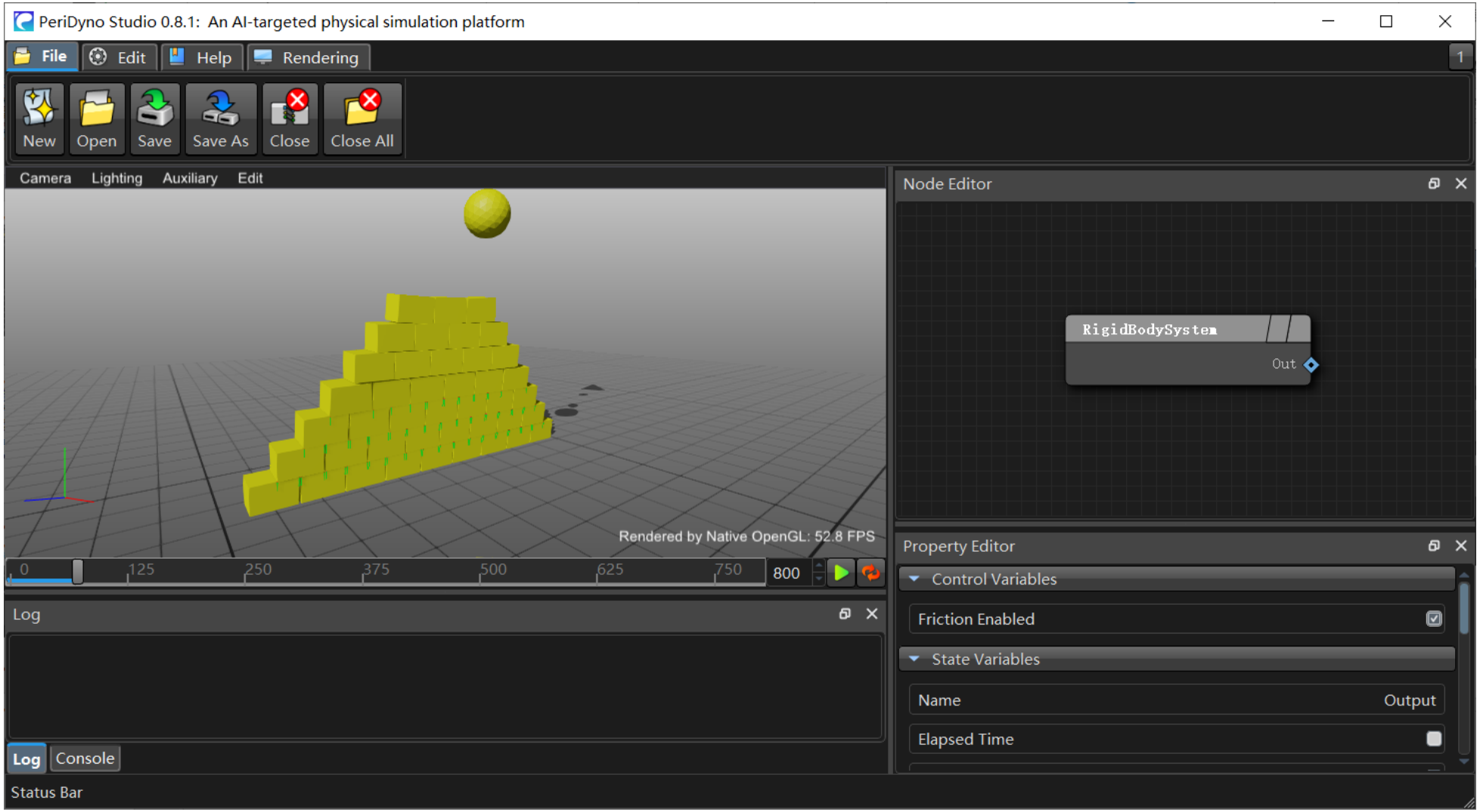
场景图管线

- 管线
  - **管线入口**：统一以节点的状态变量作为管线入口，支持直接对模块状态变量的更新；
  - **模块类型**：管线支持的功能模块既可以是仿真计算模块，也可以是渲染模块或者交互模块，不同功能模块可以自由排列和组合，从而支持复杂实时可交互场景的建模与仿真；
  - **管线类型**：当前实现支持仿真管线和渲染管线两种
  - **更新机制**：利用Field的tick()、tack()机制实现数据时间戳同步，保证每个模块只在输入数据或者控制变量更新之后才执行，从而降低冗余计算，具体参见数据更新。



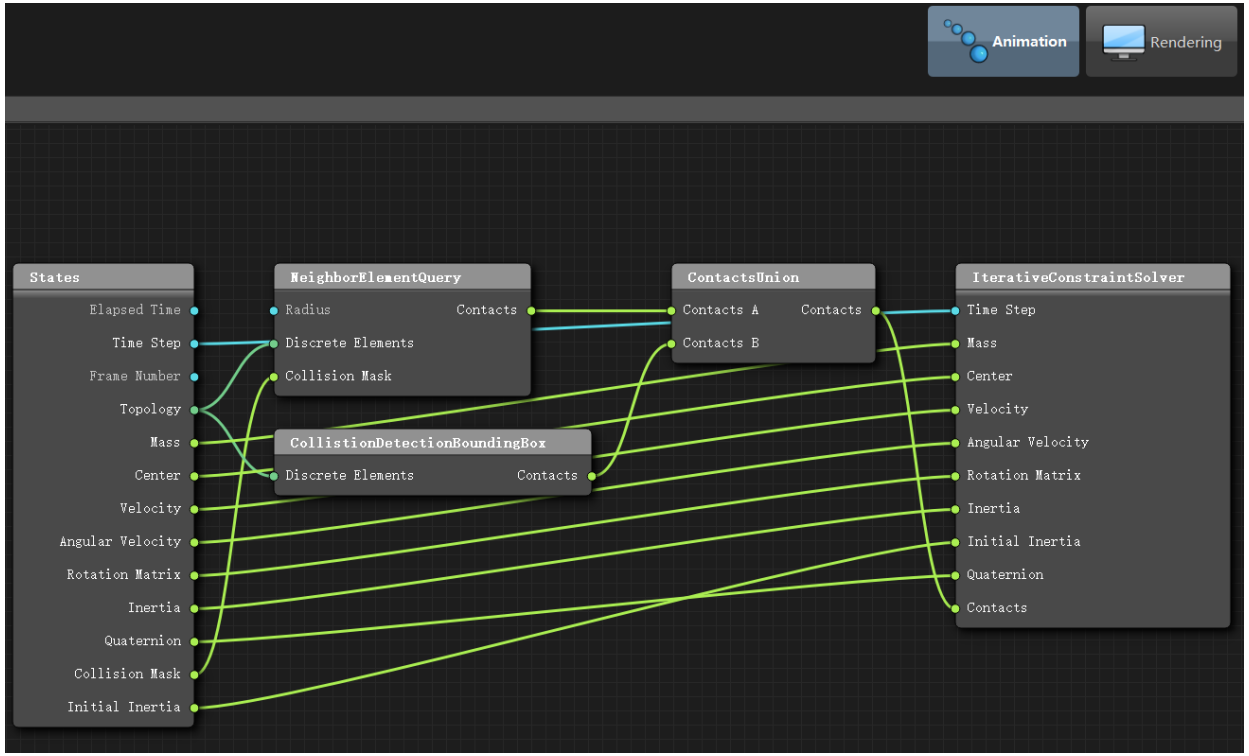
场景图管线

- examples/Cuda/QtGUI/Qt_Bricks

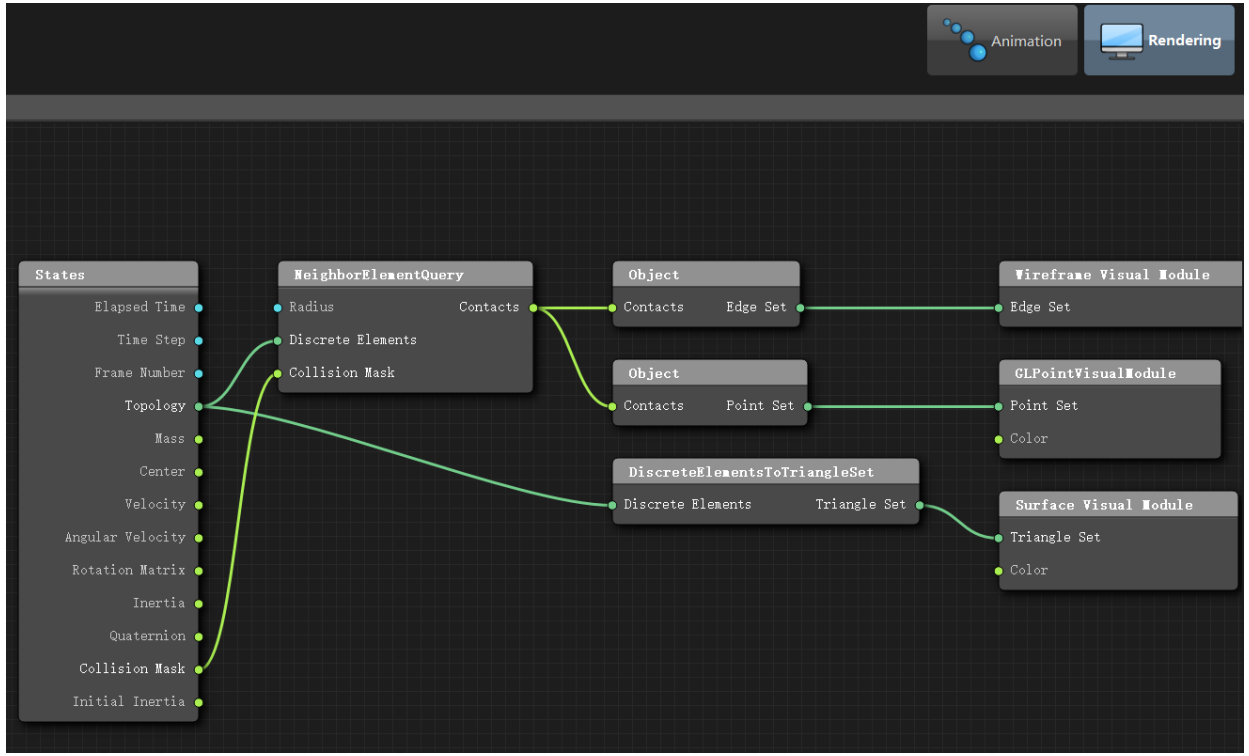


场景图管线

- examples/Cuda/QtGUI/Qt_Bricks



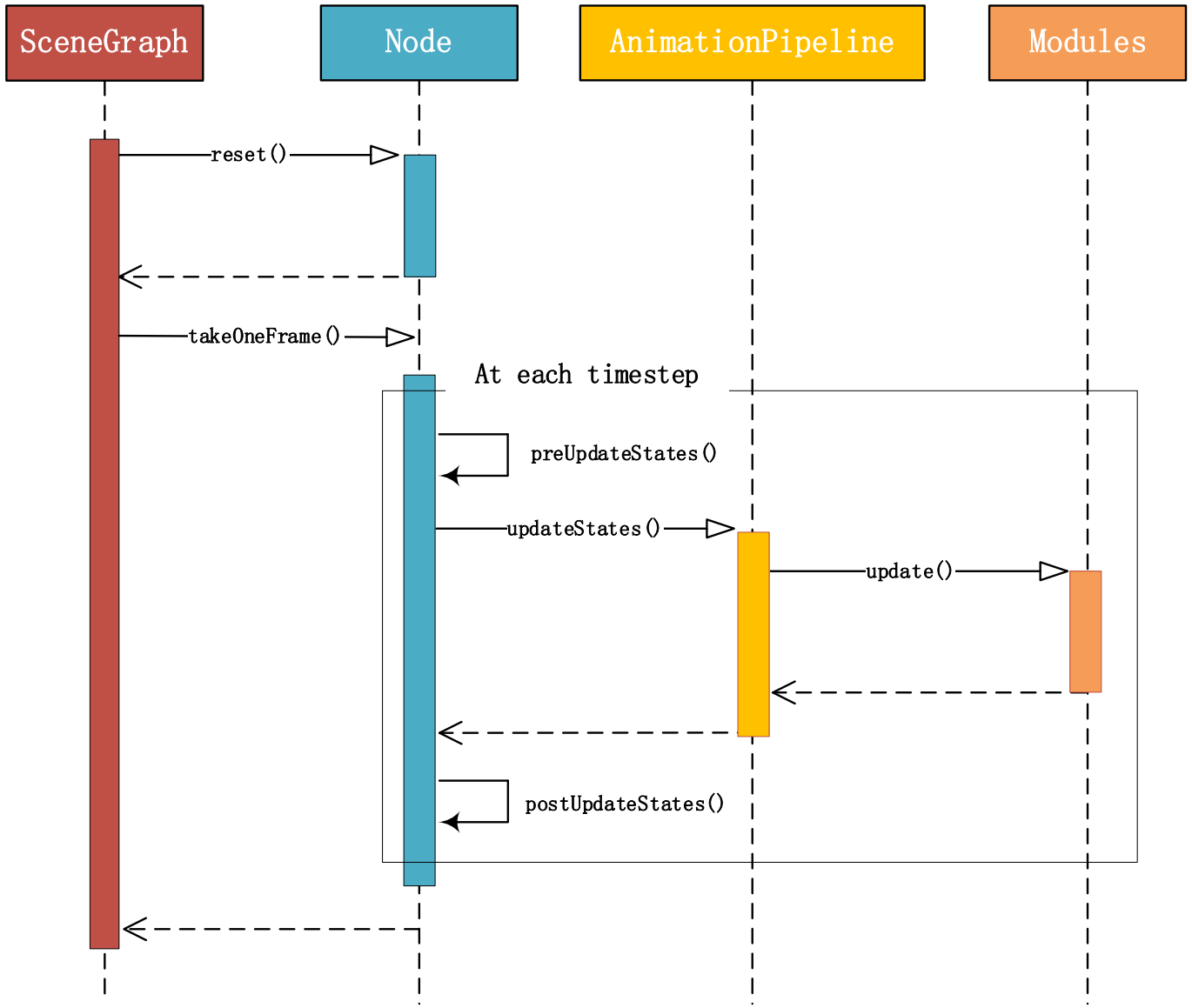
仿真管线



渲染管线

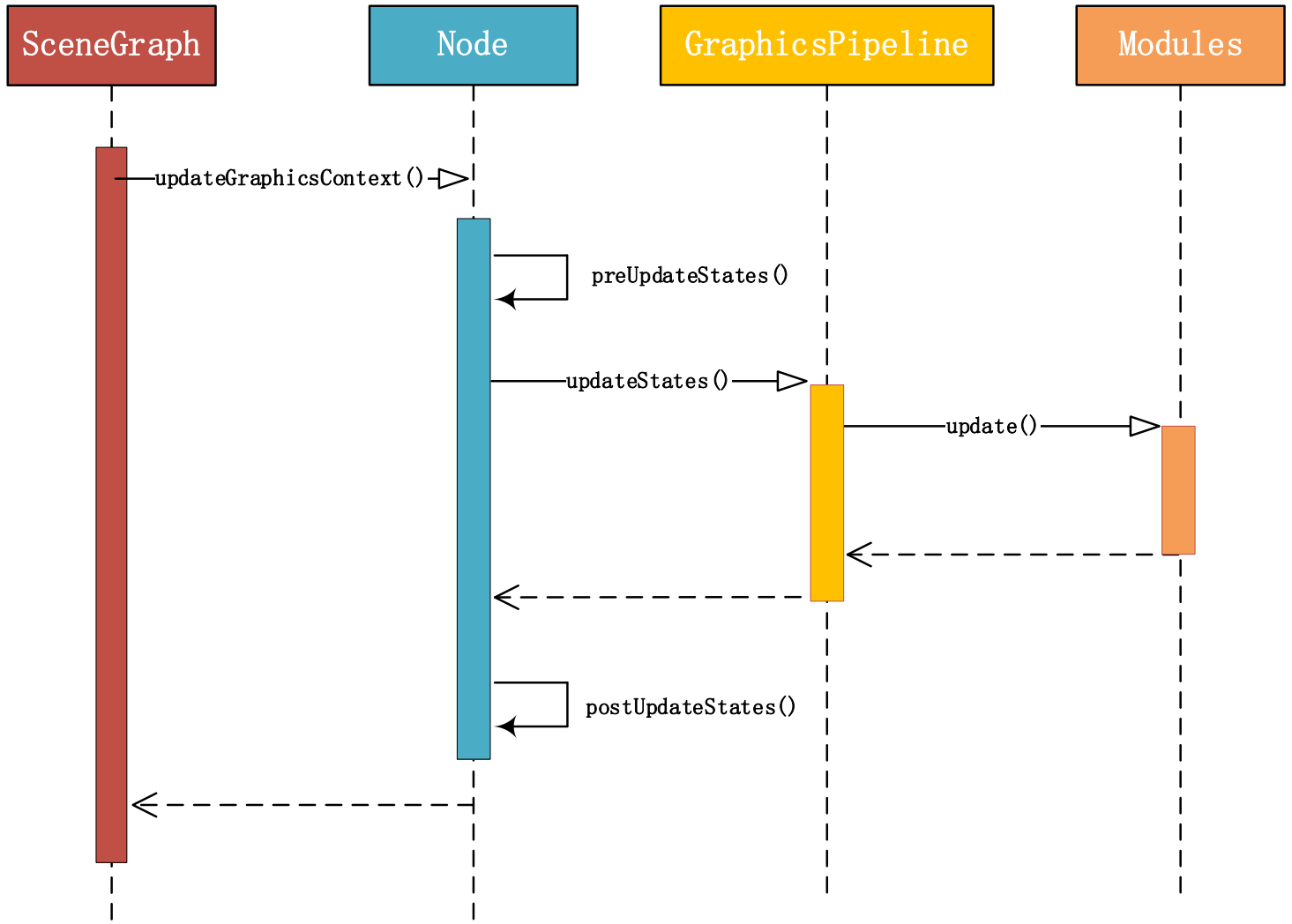
场景图管线

- 仿真管线运行时序图

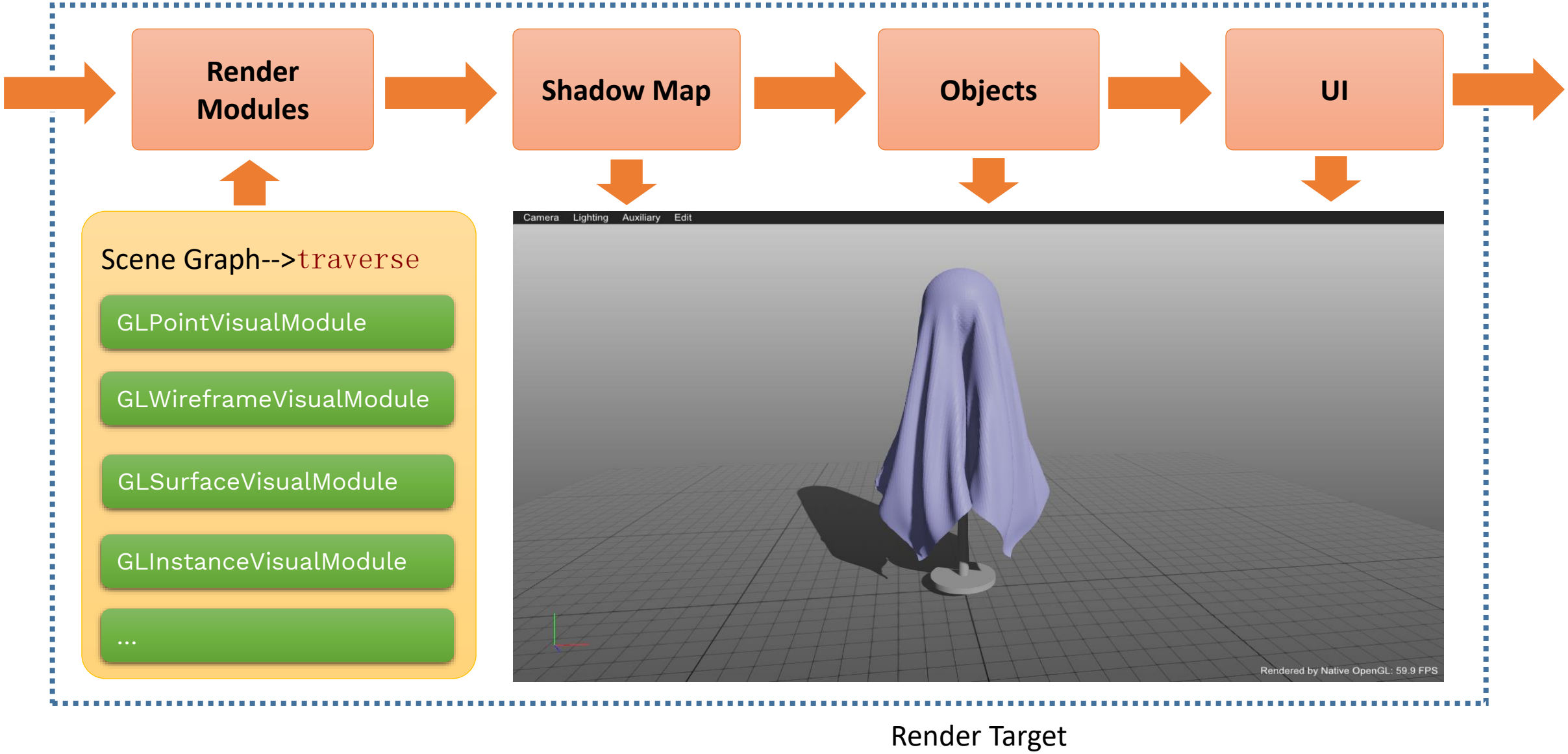


场景图管线

- 渲染管线运行时序图



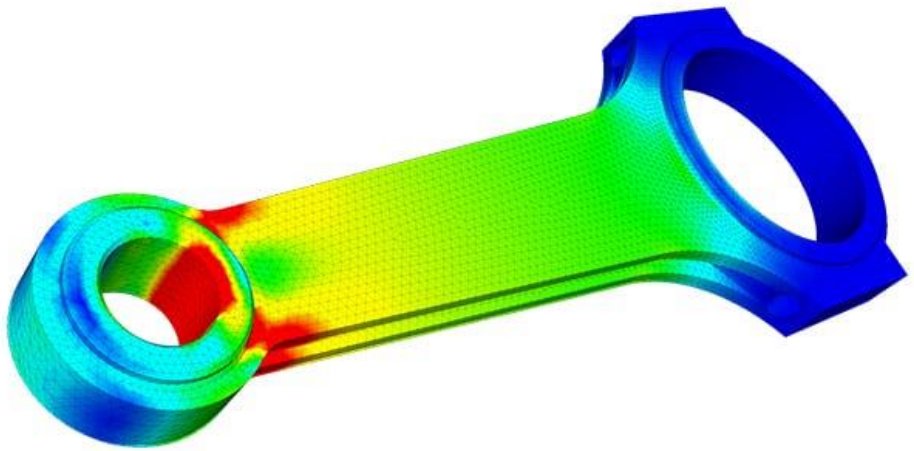
渲染


Render Modules
Shadow Map
Objects
UI
Scene Graph-->traverse
GLPointVisualModule
GLWireframeVisualModule
GLSurfaceVisualModule
GLInstanceVisualModule
...
Camera
Lighting
Auxiliary
Edit
Rendered by Native OpenGL: 59.9 FPS
Render Target

为什么要设计成四层架构

- 仿真计算与数据 解耦
- 仿真计算与渲染 解耦
- 仿真计算与交互 解耦

仿真计算与数据的解耦



Topology

- Tetrahedral Mesh

Fields

- Displacement / Stress / Strain

Numerical Algorithm

- Jacobi / Gauss-Sedel / Newton / Quasi-Newton

Finite Element Method

仿真计算与数据的解耦

Topology

- Particles

Fields

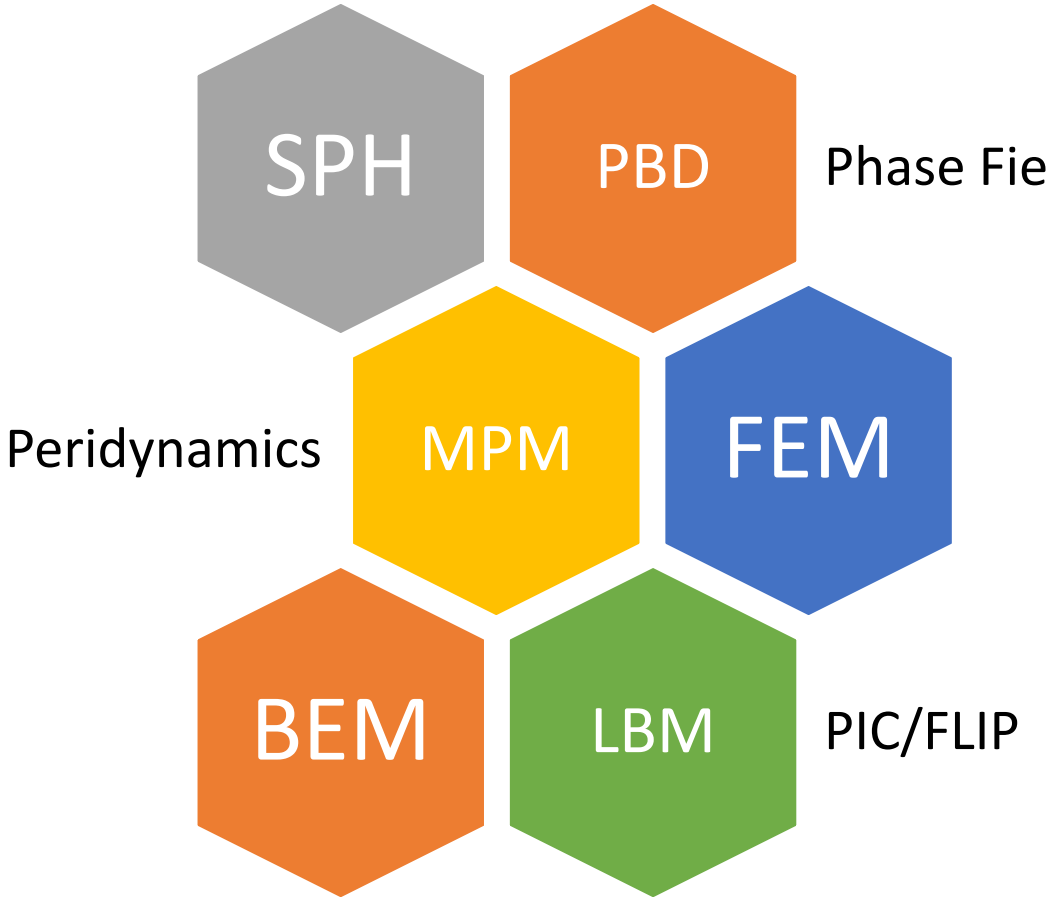
- Position / Velocity / Stress

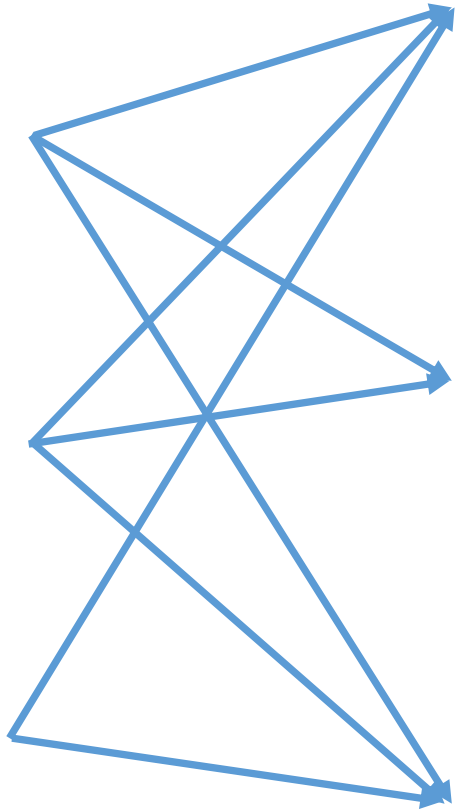
Numerical Algorithm

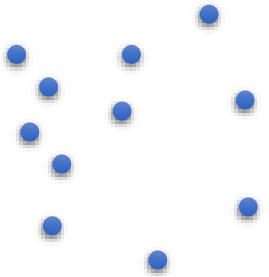
- Jacobi / Gauss-Sedel / Newton / Quasi-Newton

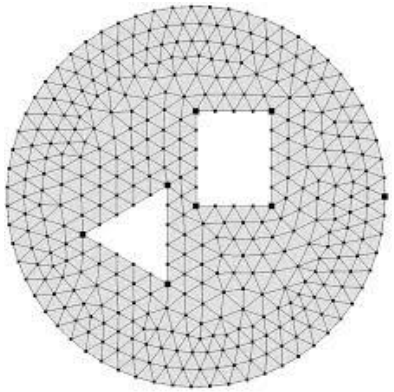
Smoothed Particle Hydrodynamics

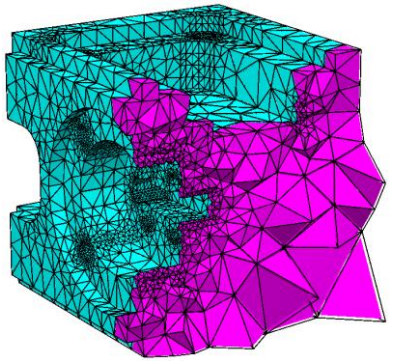
仿真计算与数据的解耦



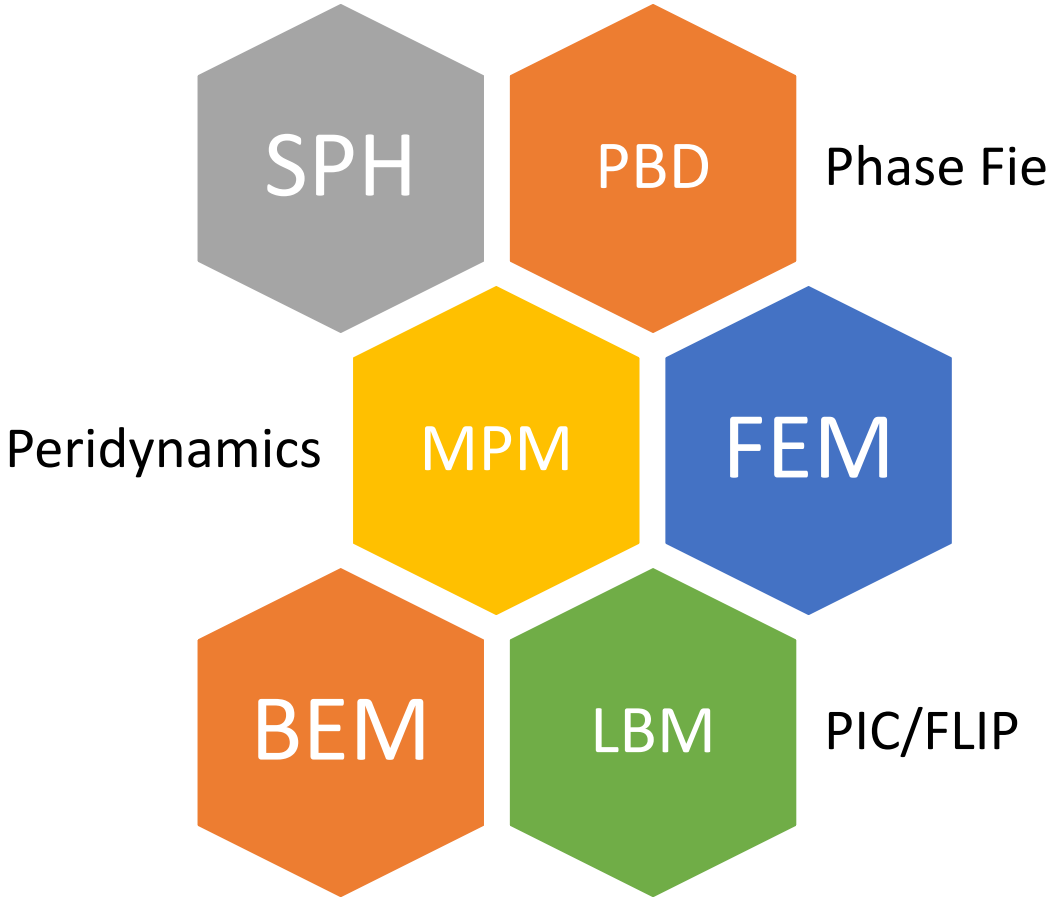


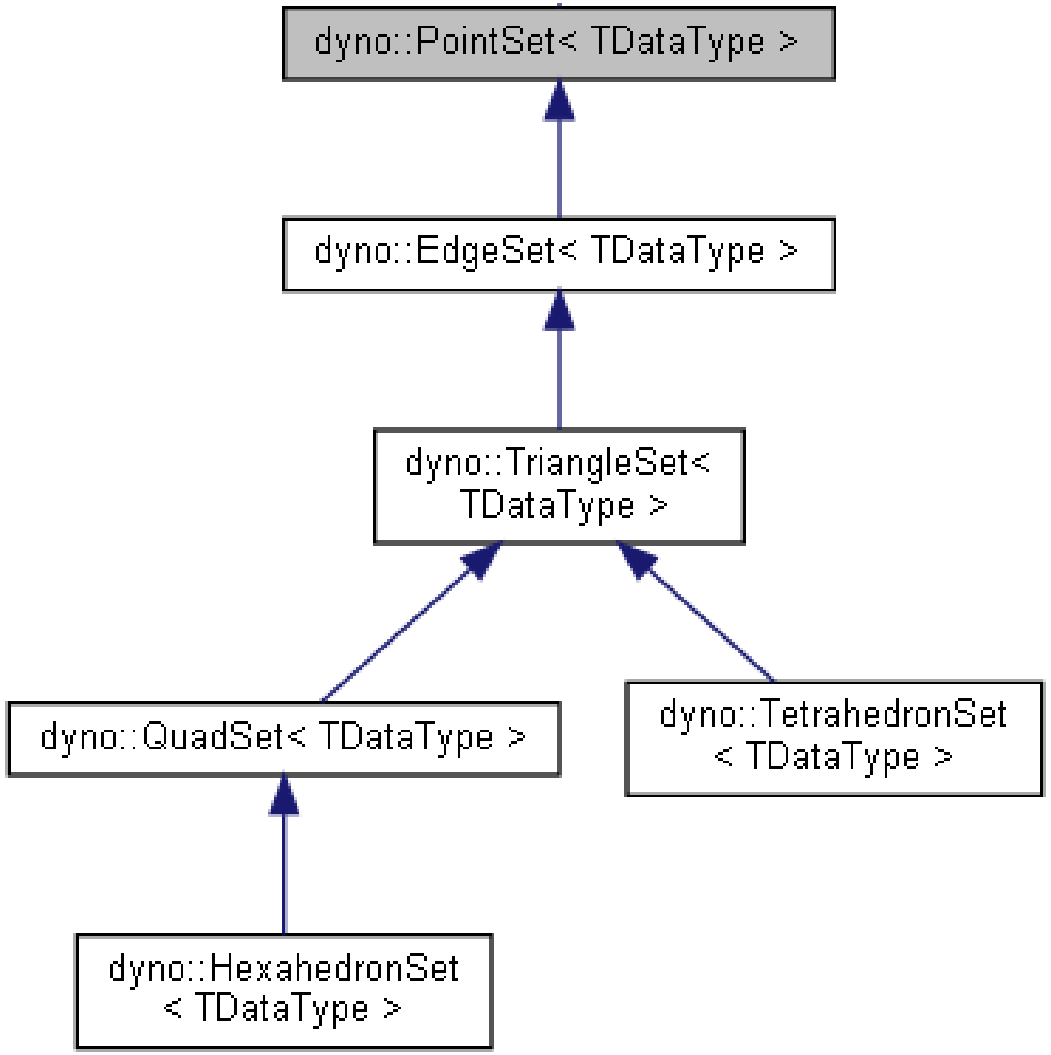




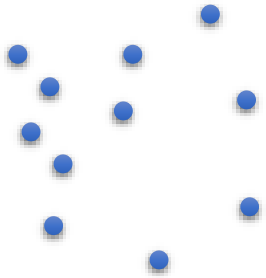
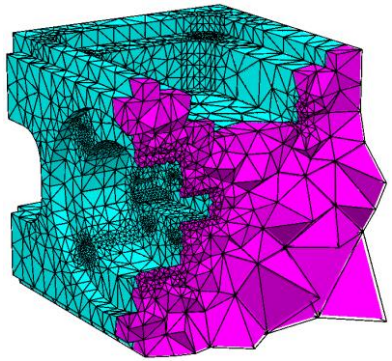


仿真计算与数据的解耦

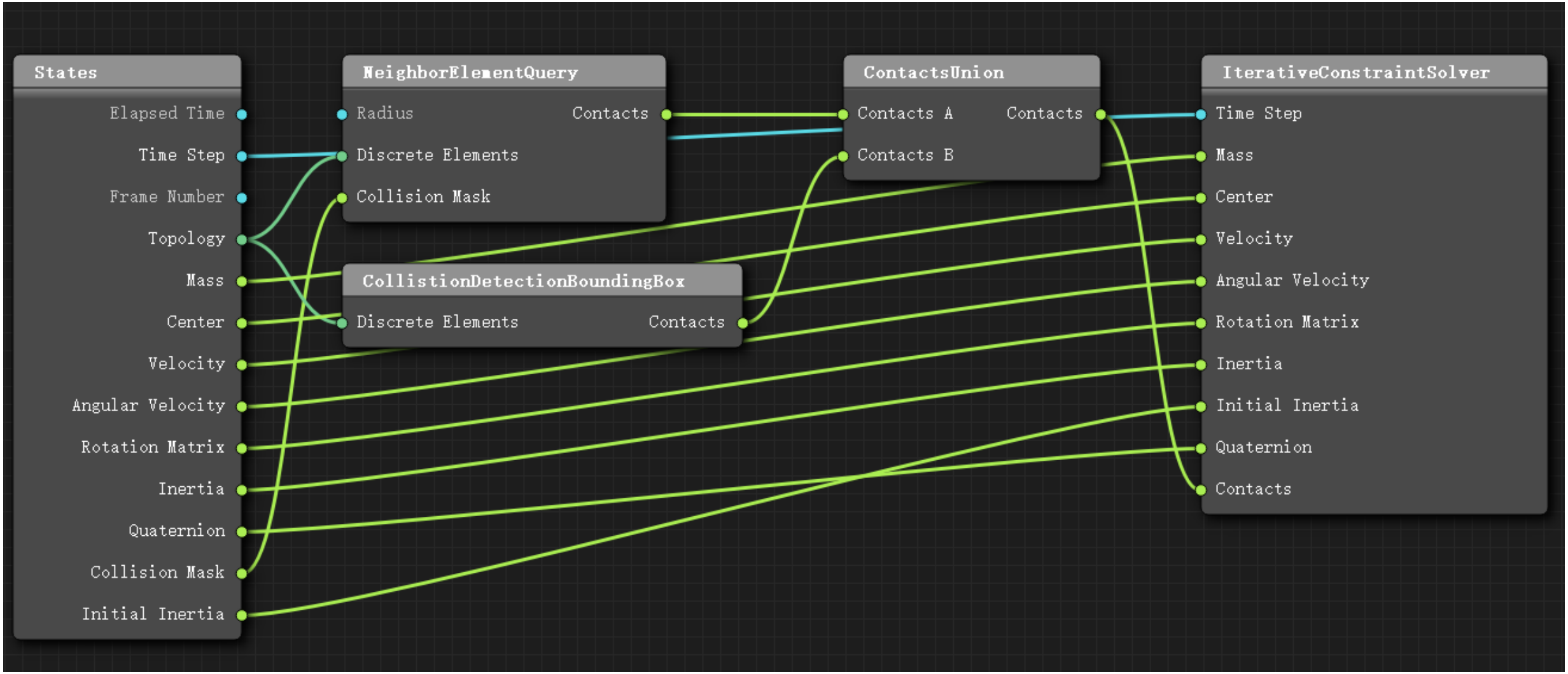

SPH
PBD
Phase Field
Peridynamics
MPM
FEM
BEM
LBM
PIC/FLIP


dyno::PointSet< TDataType >
dyno::EdgeSet< TDataType >
dyno::TriangleSet< TDataType >
dyno::QuadSet< TDataType >
dyno::TetrahedronSet < TDataType >
dyno::HexahedronSet < TDataType >

仿真计算与数据的解耦

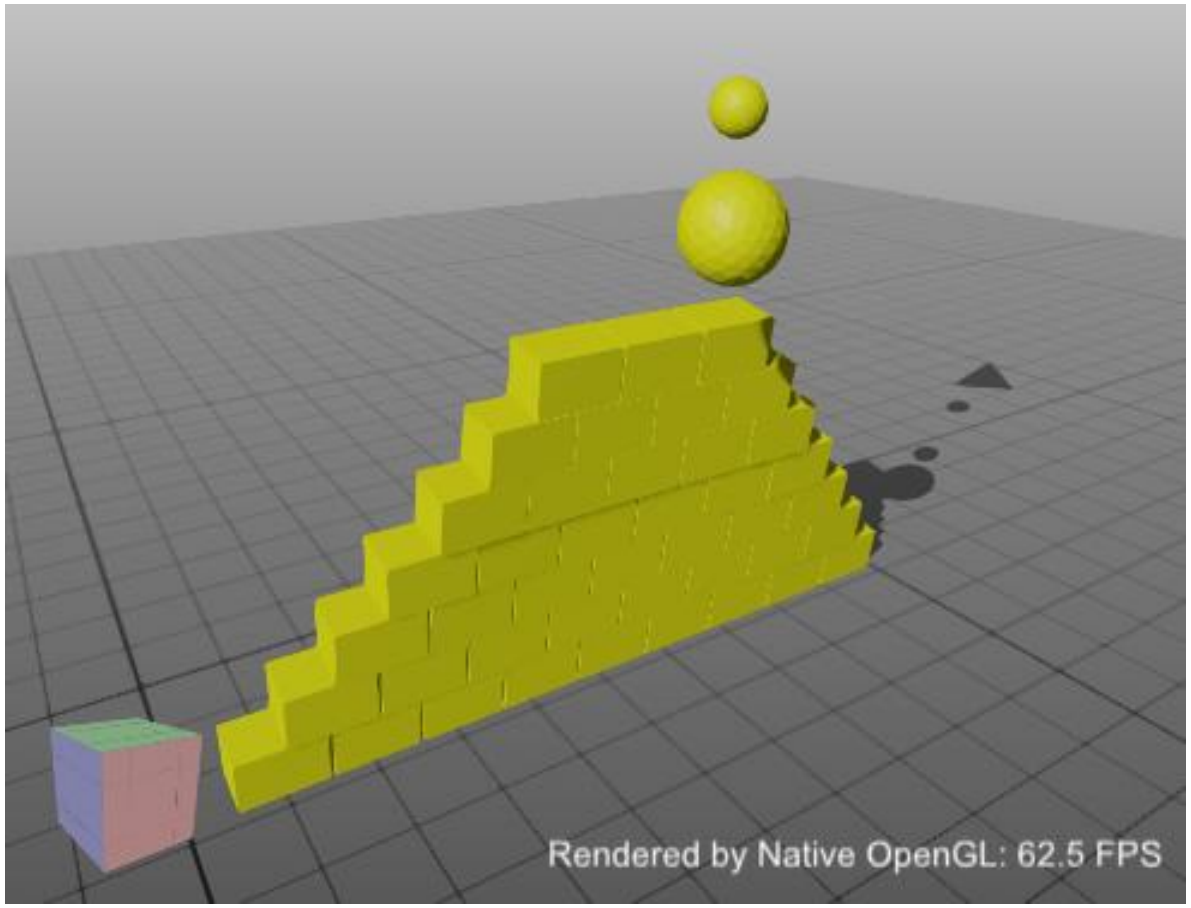
	Create Topology	Define Fields	Apply Operations
SPH		$\mathbf{x}$: Position $\mathbf{v}$: Velocity V: Volume p: Pressure …	Gradient: $\nabla\mathbf{v}$ Divergence: $\nabla\cdot\mathbf{v}$ Laplacian: Δp …
FEM		$\mathbf{x}$: Position $\mathbf{v}$: Velocity V: Volume $\boldsymbol{\varepsilon}$: Strain $\boldsymbol{\tau}$: Stress …	Gradient: $\nabla\mathbf{v}$ Divergence: $\nabla\cdot\mathbf{v}$ Laplacian: Δp …

仿真计算与数据的解耦

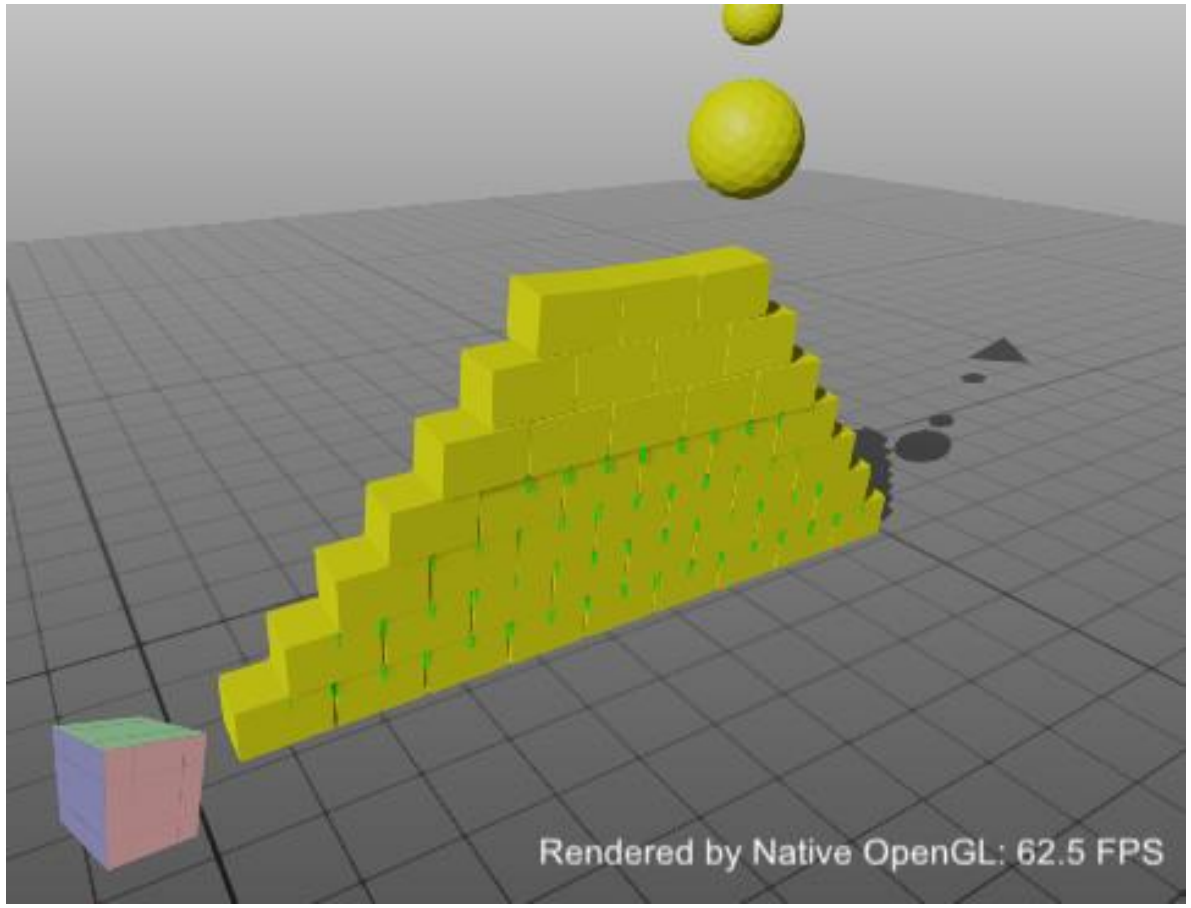


发现运行结果不对怎么办?

仿真计算与渲染的解耦



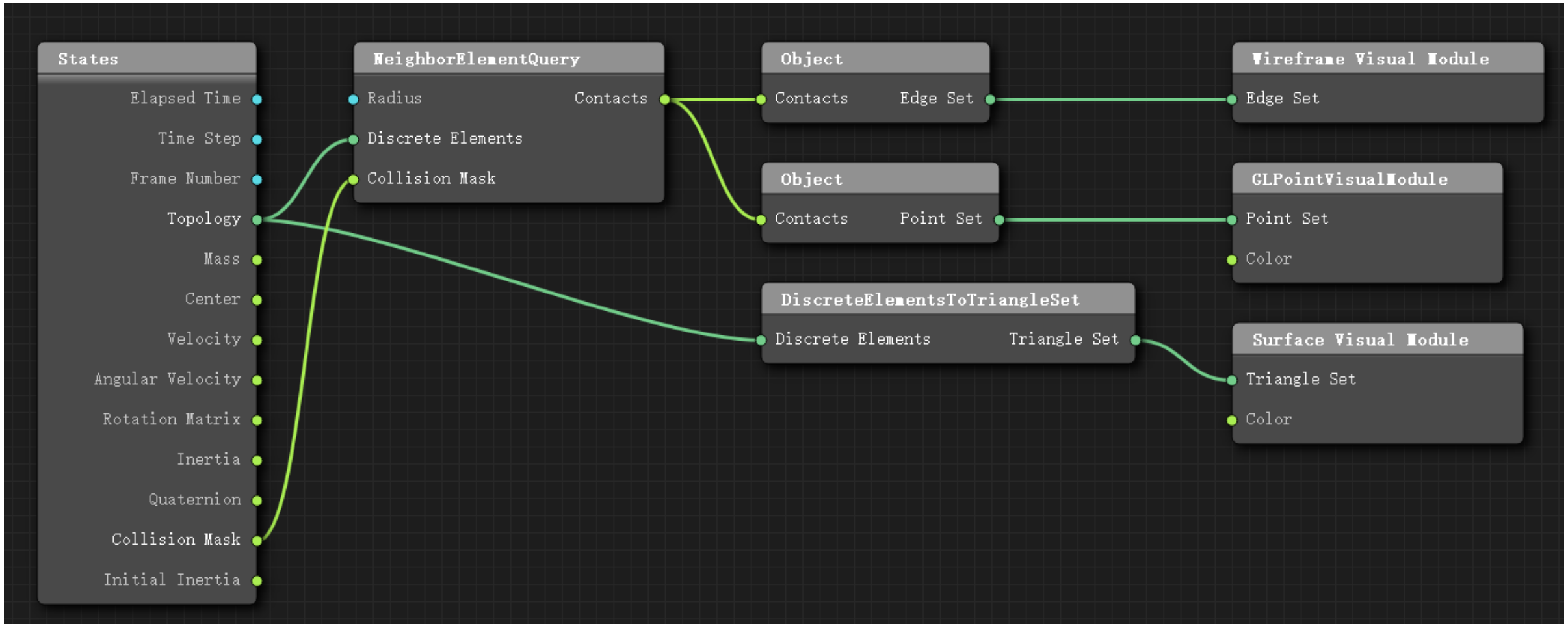
刚体渲染（不带触点和法向）



刚体渲染（带触点和法向）

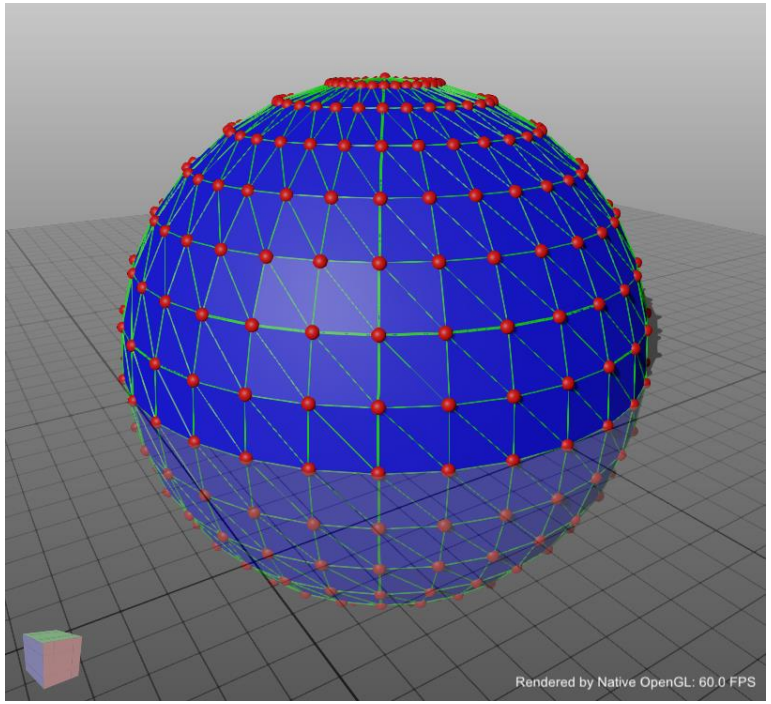
仿真计算与渲染的解耦

- examples/Cuda/QtGUI/Qt_Bricks



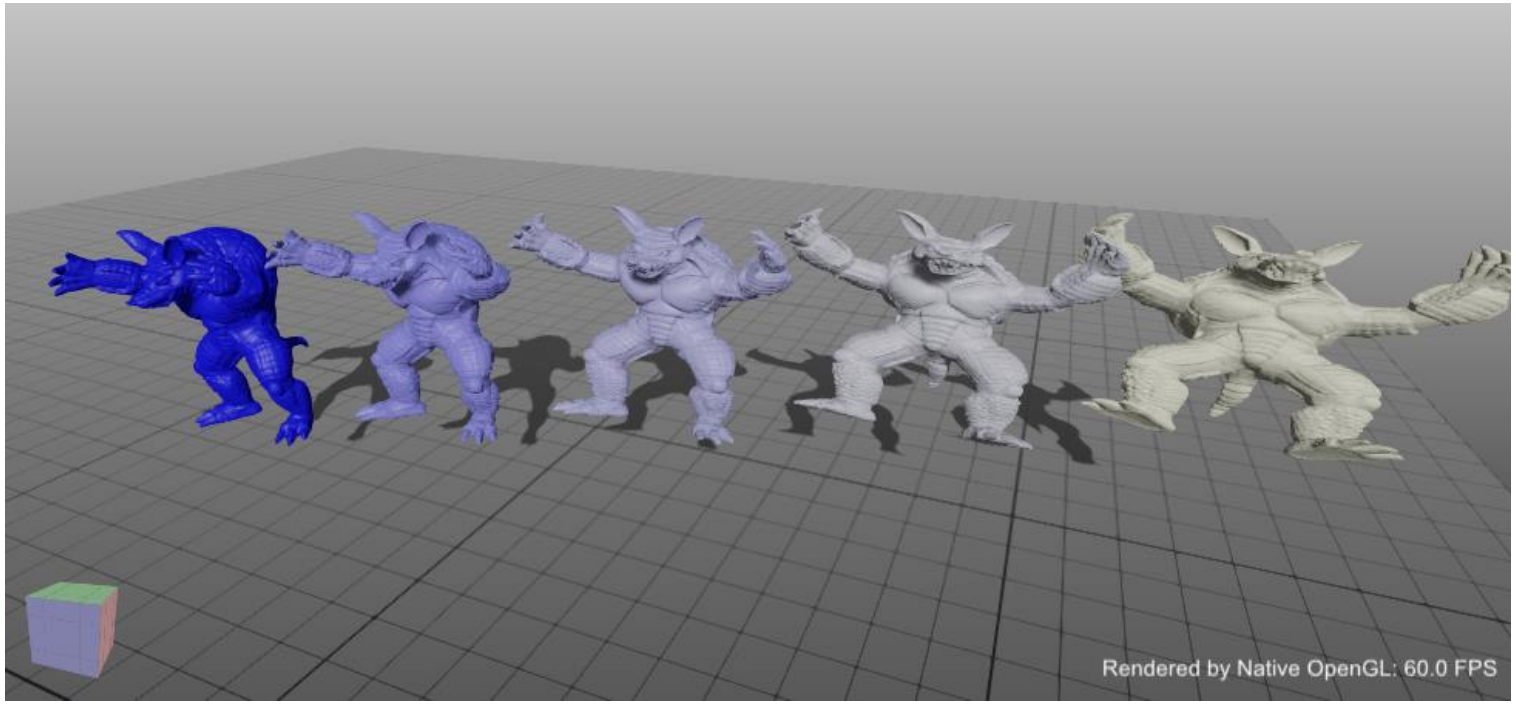
仿真计算与渲染的解耦

点线面渲染



GL_Topology

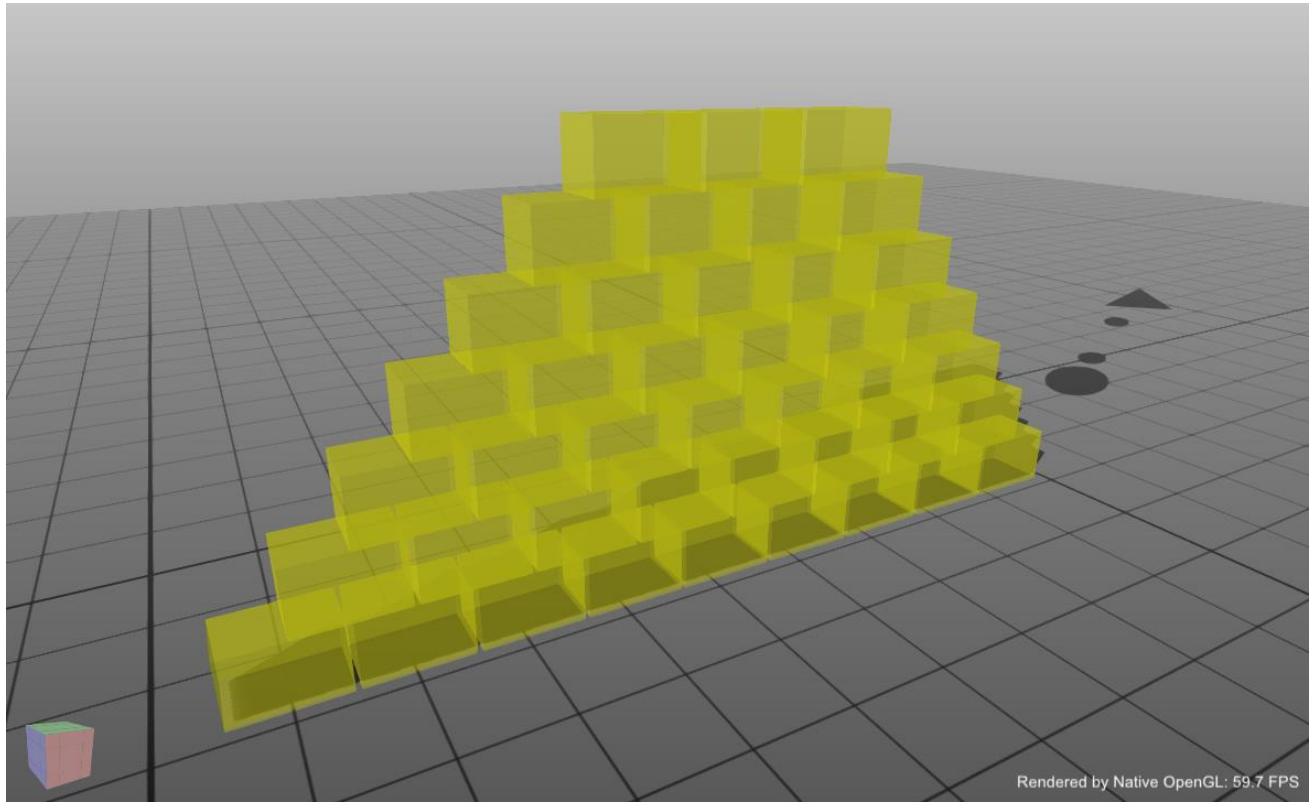
实例渲染



GL_InstanceVisualizer

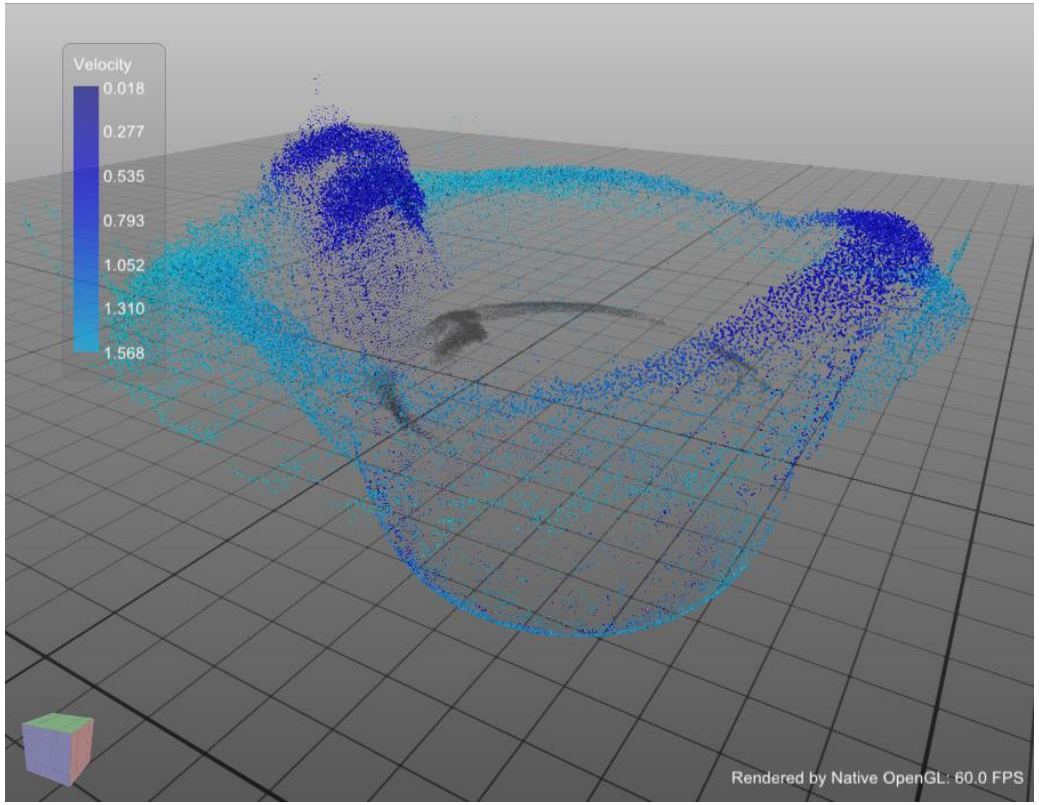
仿真计算与渲染的解耦

半透明



GL_Bricks

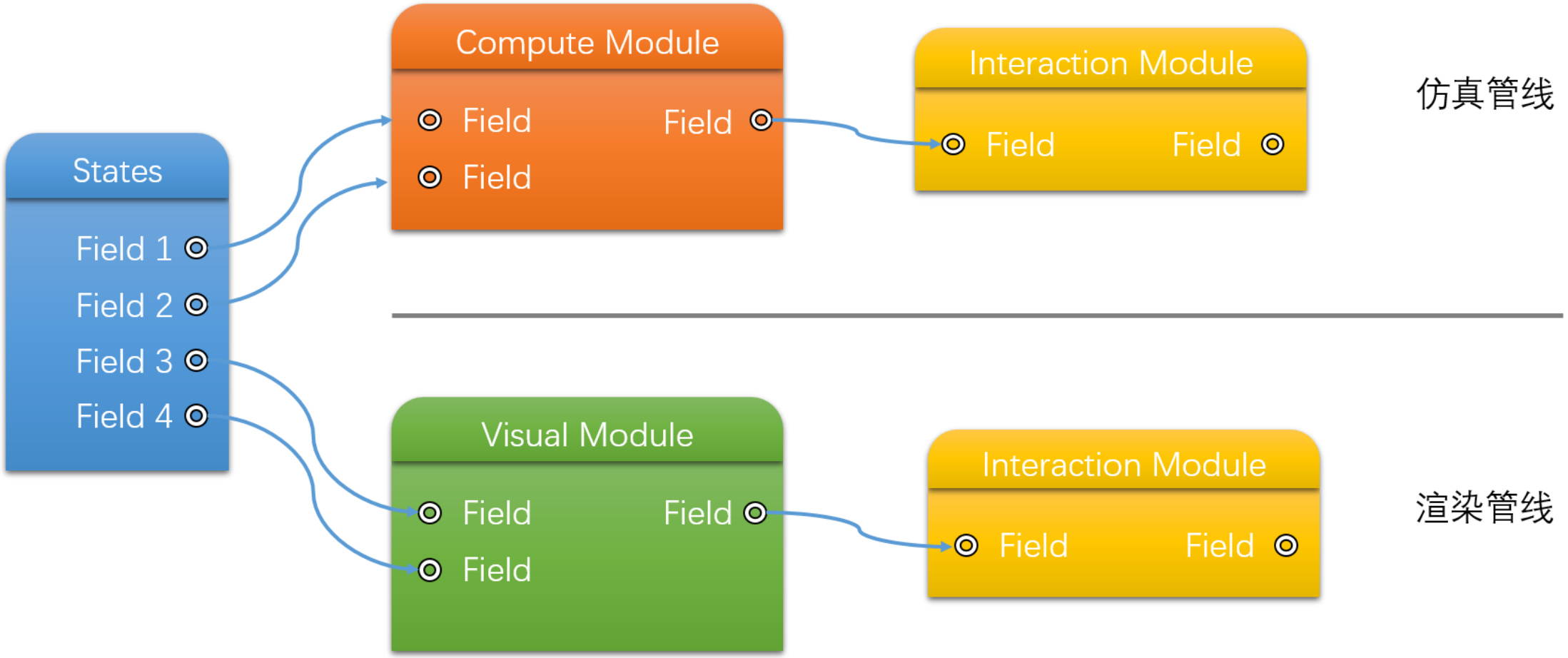
颜色映射



GL_ParticleFluid

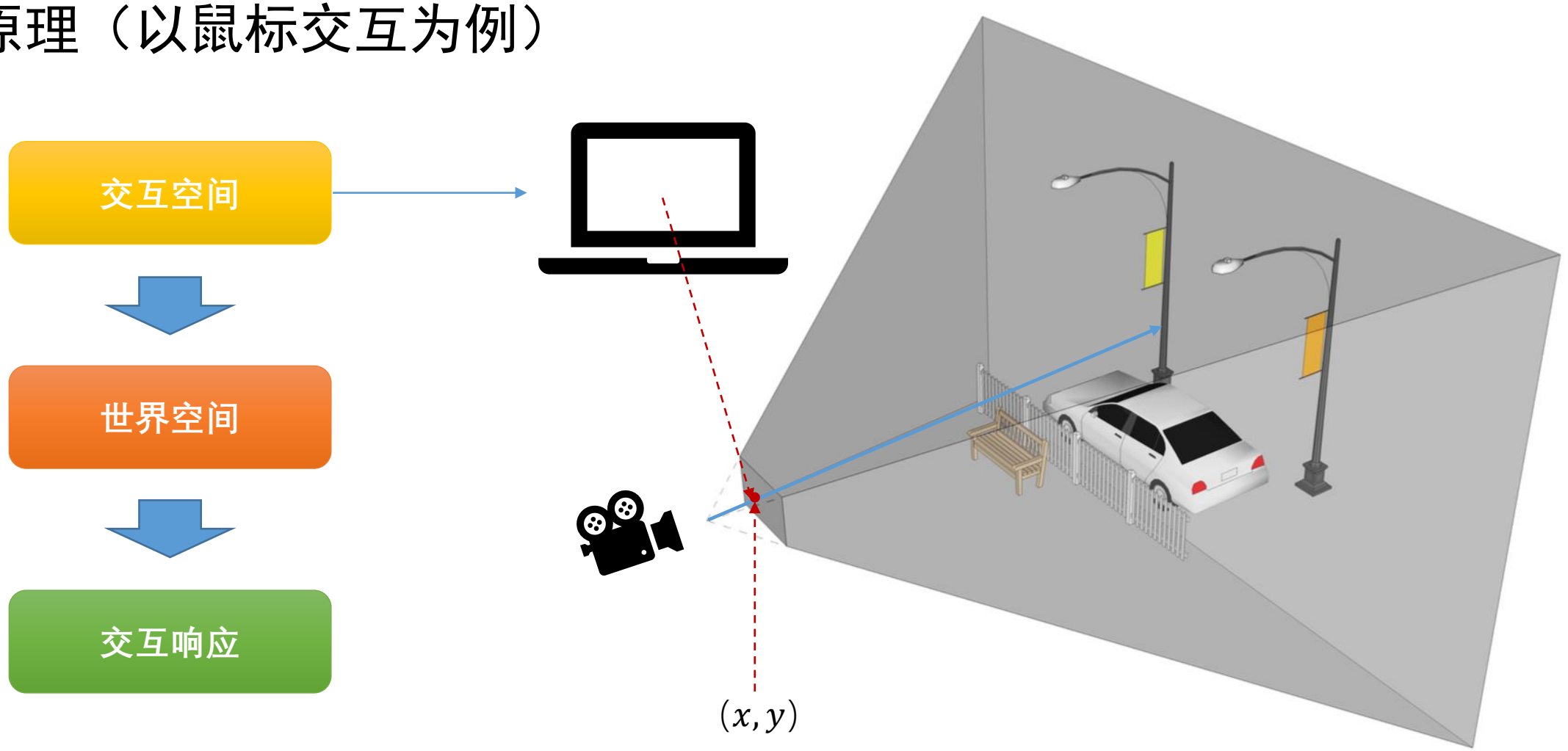
仿真计算与交互的解耦

- 目的：实现交互行为的定制



仿真计算与交互的解耦

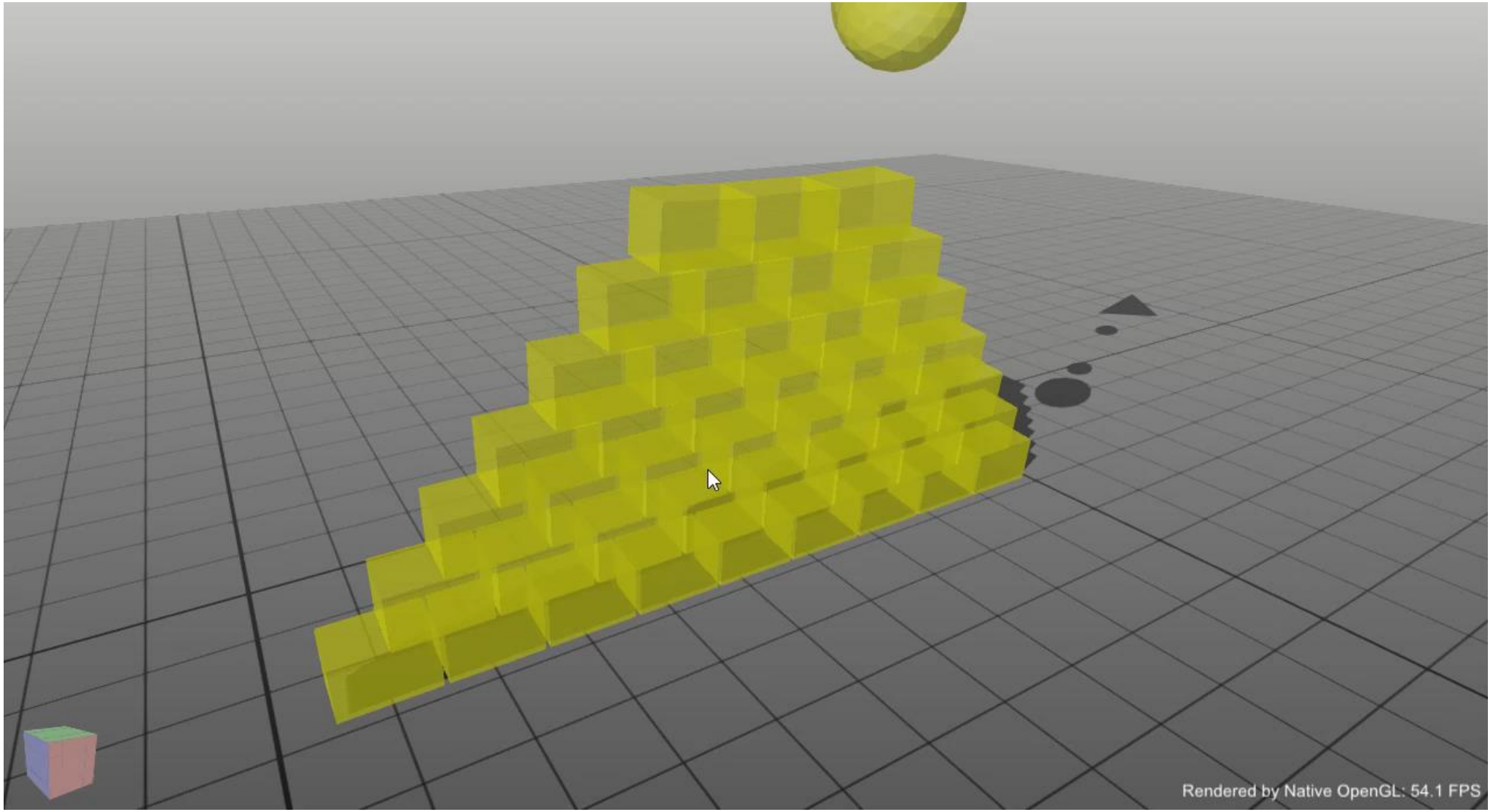
- 原理（以鼠标交互为例）



鼠标交互

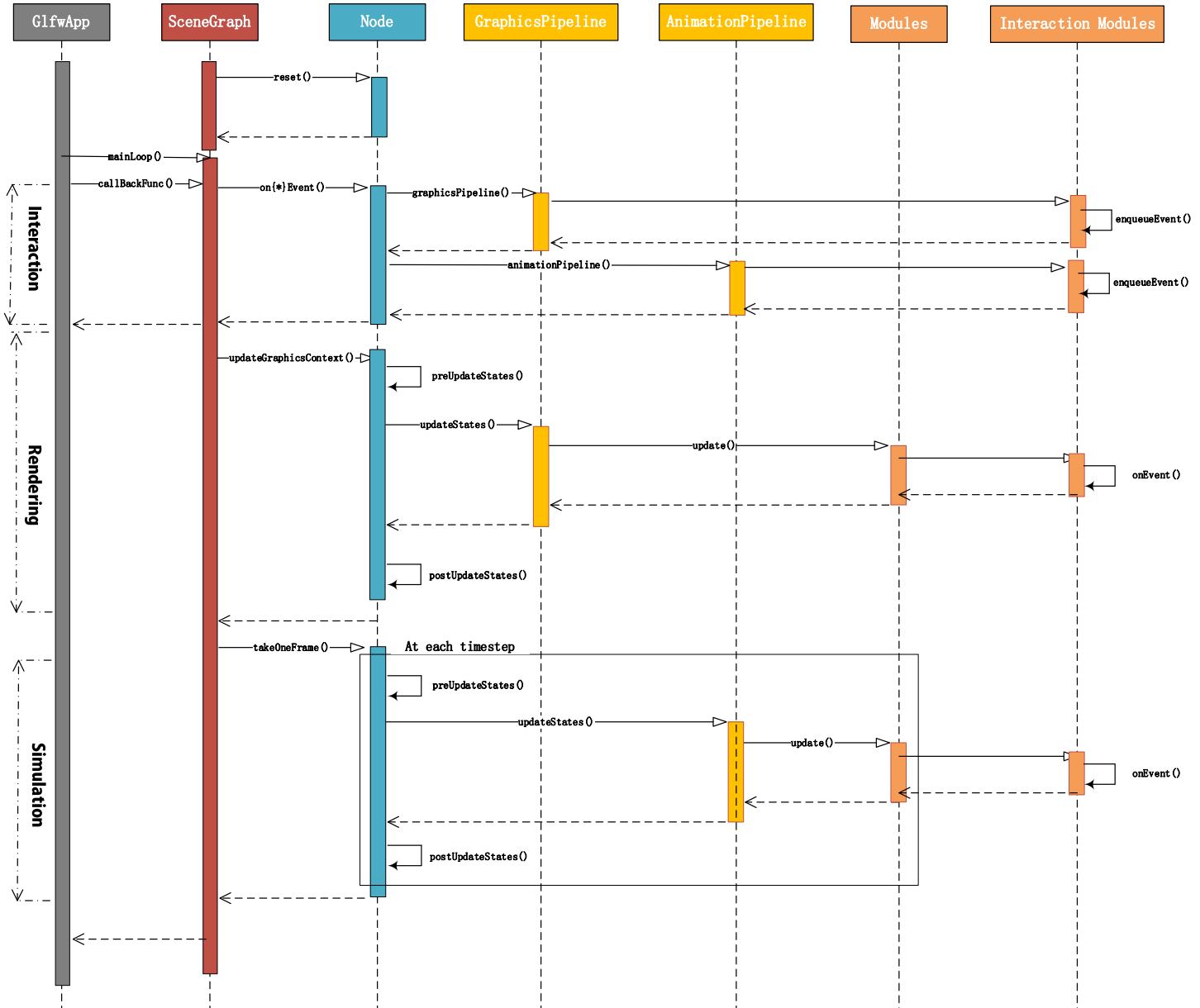
仿真计算与交互的解耦

- examples/Cuda/QtGUI/Qt_HitBricks

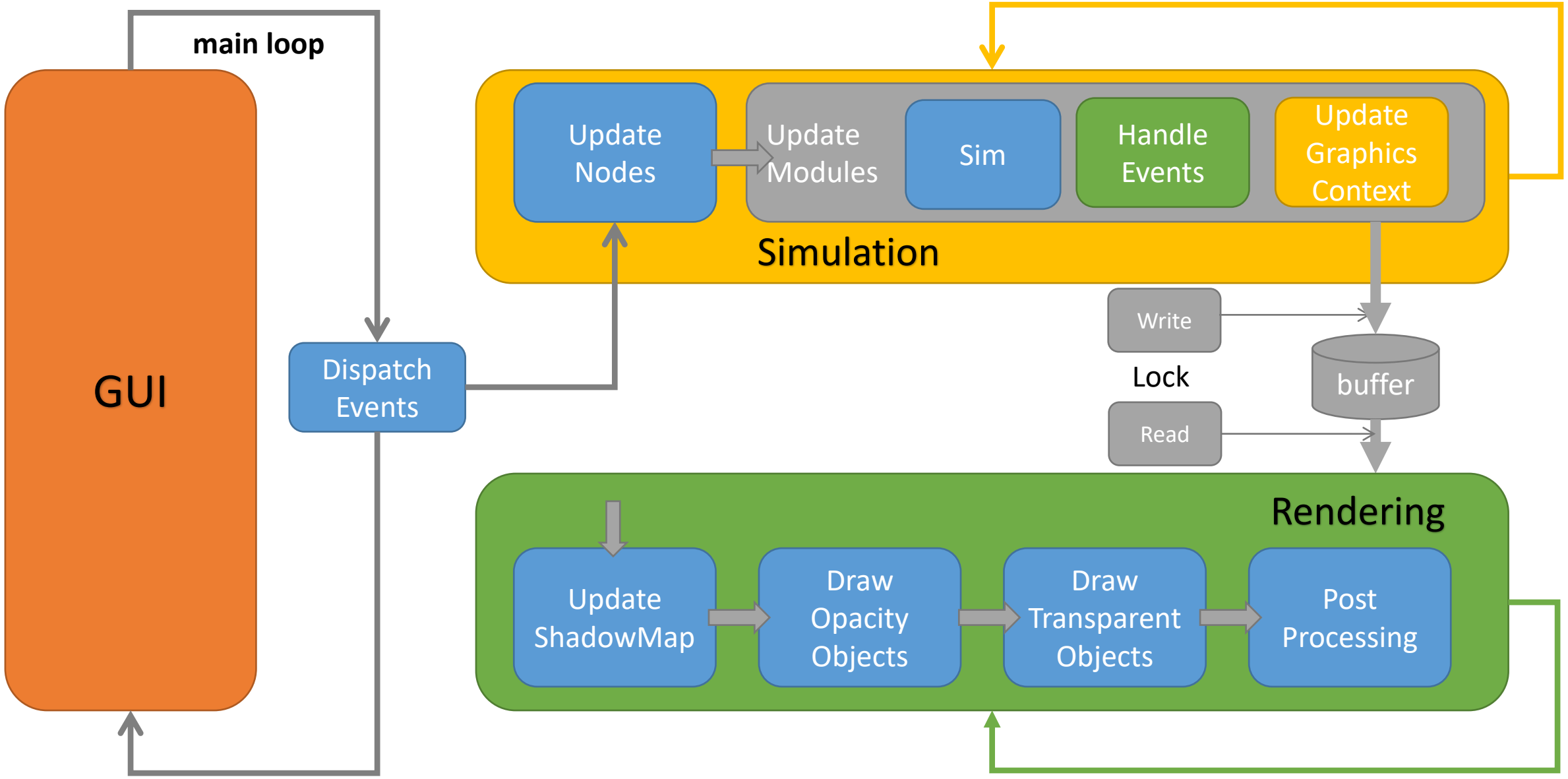


单线程环境仿真、渲染、交互协作图

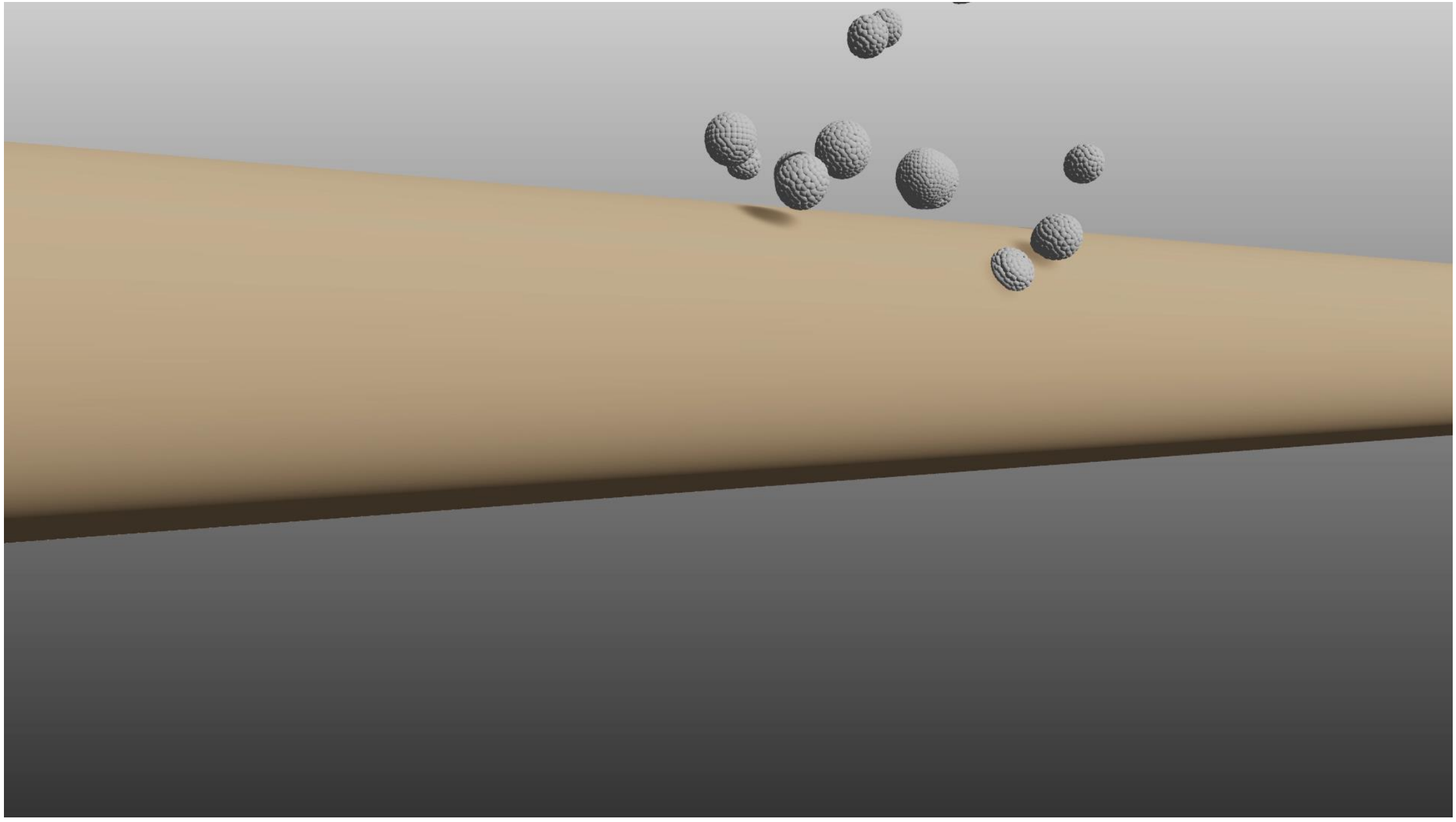
仿真渲染交互运行时序图



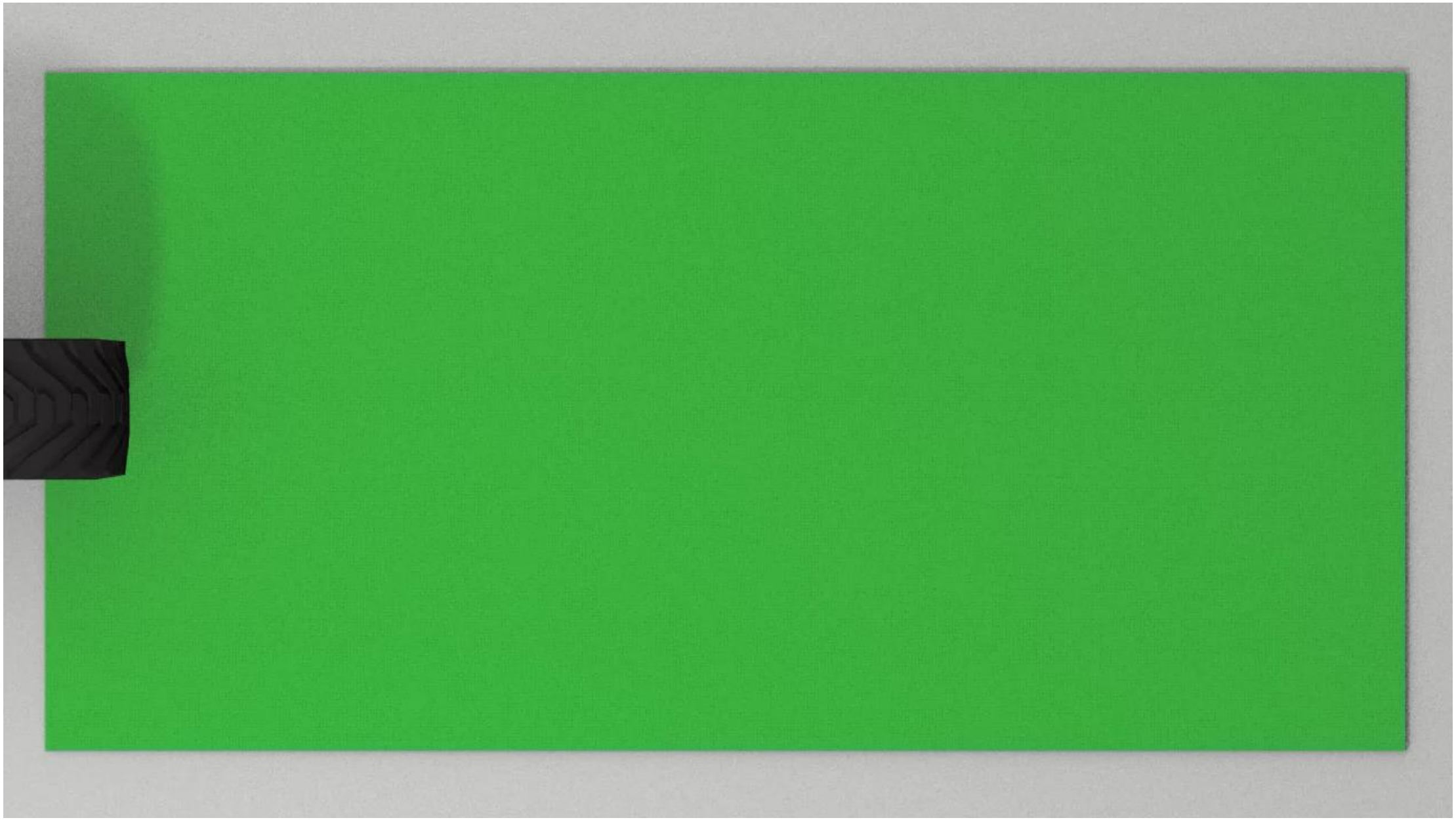
多线程环境仿真、渲染、交互协作图



仿真案例演示：SPH



仿真案例演示：SPH



仿真案例演示：Peridynamics

Thickness = 3.26 mm

Thickness = 1.63 mm

仿真案例演示：Peridynamics

贡献者列表

- 常悦（ University of Toronto ）
- 蔡元添（湖南大学）
- 郭煜中（中科院软件所）
- 郭德闻（北京大学）
- 何浩（中科院软件所）
- 刘树森（中科院软件所）
- 罗旭锟（中科院软件所）
- 卢子璇（中科院软件所）
- 缪冉（中科院软件所）
- 任丽欣（中科院软件所）
- 石剑（中科院自动化所）
- 苏明才（华为）
- 王宇杰（湖南大学）
- 王强（湖南大学）
- 徐力有（字节跳动）
- 夏提（ Vanderbilt University ）
- 叶子萌（中科院软件所）
- 朱威（华为）
- 赵博伟光（湖南大学）
- 。。。

Question