OpenDS

An Open-Source Driving Simulation Software for Research

Technical Documentation

翻译人员：赵 谦

目录

软件版本：2.5
文件的创建日期：2014-10-23

1 介绍

由于用于汽车应用评估的成熟的驾驶仿真软件价格高，成本低，模拟器往往缺乏可扩展性。因此，基本驾驶仿真工具包的实现一直被认为是欧盟项“GetHomeSafe”的一部分。该软件被请求在开源代码上在线发布，这将允许世界各地的研究人员根据他们的需要免费使用和扩展该软件。作为回报，它将从社区收集宝贵的贡献。

以往的调查显示，只有少数既定和自由使用的驾驶任务(如 ISO 标准化的“车道改变测试”)可以衡量驾驶性能和驾驶员分心程度，而且这些任务并不适合于各种用例或研究问题。例如，如果需要持续获得指导性能，则应提出不可预见的事件，或者驱动任务困难对性能的调节影响是调查的重点，研究人员需要购买昂贵的工具，而且往往需要大量的建模工作。另一个关键问题是，开始在这一领域开展工作的研究人员或机构(例如车辆排任务、寻路、复杂的十字路口场景)会一次又一次地执行跟踪和场景，从而导致不必要的建模工作，更糟糕的是，这会导致实现和词汇量上的差异。

由于这些原因，一个用于测量驾驶性能和驾驶员分心的驾驶仿真软件必须被实现。由于软件是与研究人员共享的，所以我们不打算为所有类型的驾驶实验提供一个全面的工具包，而是一个可由用户扩展的基本构造工具包。

在下面，我们将提供基本的技术文件。

2 OpenDS 所使用的游戏引擎

OpenDS 基于 jMonkeyEngine[1]（jME），这是一种基于图形 API 的高性能场景图。这个开源框架已经在 Java 中实现，并在游戏开发中赢得了好评。其默认的渲染器是轻量级 Java 游戏库(LWJGL)，完全支持 OpenGL2 到 OpenGL4。在 3.0 版本中，jME 框架使用 jBullet 物理库的 Java 端口，提供高级行业开发人员使用。将 Bullet 物理库封装到 jME3 对象中可以保证简单的交互以及将来的更新。更新包括对本机 Bullet 的支持以及 GPU 加速。JBullet 是一种多线的物理引擎，它在模拟过程中可以创造出精确网格的碰撞图形，可以模拟如加速度、摩擦力、扭矩、重力和离心力等。它支持任何常用的模型格式在模拟器上加载 3D 环境。

JME 渲染器进一步的功能是支持不同的照明选项（像素照明、多通道照明、Phong 照明、正切阴影和反射）、纹理（通过着色器的多重纹理）、以及对诸如烟雾、火、雨、雪等特殊效果进行建模的能力。软件的后期处理和 2D 效果包括双面效果、阴影贴图、高动态光照渲染、屏幕空间环境光遮蔽、光线散射、雾和景深效果。

漂亮的 GUI 集成支持一个易于使用的工具包，用于在呈现框架内部设计平台的独立的图形用户界面，即模拟过程中使用的菜单栏和消息框。关于 OpenDS 的开发，JME 的 GUI 节点（用于速度表和里程面板）、多视图窗口（用于后视镜）和基本的音频支持（用于播放位置和方向声音）都是非常有用的功能。

[1] http://jmonkeyengine.org

3 关于 OpenDS

OpenDS 由两个主要部件组成：模拟器和驾驶分析仪。

模拟器可以加载一个通常为 XML 格式可运行的驾驶任务。这个任务描述了一个设定的地图模型将如何配置额外的道路对象（如交通、标志、信号灯、障碍物等）、事件和其他几个参数。所有被处理的驾驶任务中的可见元素都将被添加到场景图中，并将所有物理对象连接到 jBullet 物理模拟的主物理节点上。可扩展模拟器实现的主要用功能是控制交通灯(预先定义的循环、红色/绿色接近、交互式外部控制)、交通和天气条件的模拟，以及考虑到克服滚动阻力、空气阻力、惯性和势能以及发动机内部摩擦所需要的能量，建立了一种现实的发动机和传动模型，可用于计算当前踏板状态下的油耗。此外，在给定的条件下，可以触发已在驾驶任务中定义的事件。例如，设置驾驶车辆的位置，让物体出现/消失，移动车辆，进行反应测量，播放声音文件等。以这种方式记录的反应数据可以被可视化，例如，做成一个带有集成 Jasper Report 模块的条形图，并导出为文本或 PDF 格式。

OpenDS 的第二个主要组成部分是驱动器分析仪，它能够以每秒几次的速度显示驾驶中记录的汽车数据；例如，位置、方向、速度和踏板状态。它使实验主试能够从先前的驾驶中重新构建精确的模拟环境，以分析汽车在任何比较，以计算偏差，可将其作为衡量驾驶性能的指标。

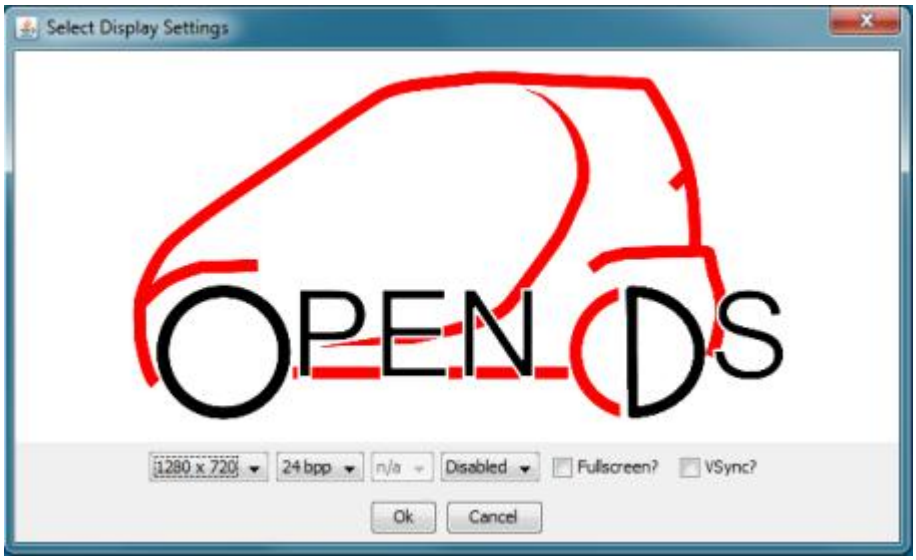
为实现一个接近真实的模拟环境，OpenDS 不仅提供了一个游戏控制器的接口，而且还提供了一个 CAN 接口来连接到真实的汽车，这使得模拟器能够请求到汽车的属性；比如，转向角度和踏板状态，并为车内设备提供模拟数据。为了增加驾驶体验，OpenDS 支持柱形摄像头的概念，它允许圆柱形投影和多屏投影。

4 运行 OpenDS

4.1 运行构建场景

1. 双击 OpenDS. jar

2. 当配置窗口弹出，选择分辨率和是否全屏，点击 OK。



3. 设置司机的姓名（选填）。

4. 选择地图模型，并点击开始。

4.2 从源代码进行运行

下面的步骤展示了如何在 Eclipse 使用源代码运行 OpenDS。但是，在其他开发环境中也可以。

1. 启动 Eclipse 并创建一个新的 Java 项目（File—New—Java Project）。指定适当的指定一个适当的名称（例如 OpenDS），点击‘Finish’

2. 从 OpenDS 主页[2]上下载源代码，并将其中的内容移动到新项目的‘src’文件夹中。

3. 向项目中添加一个新的文件夹‘lib’，并将库文件夹中的内容移动到这个文件夹中。

4. 在您的项目中添加一个‘assets’文件，并将 assets.zip 的内容复制到这个文件夹中。

5. 添加所有可以在‘lib’中找到的或它的子文件夹中找到构建路径的‘jar’文件（总数超过 100）。

6. 右键点击该项目并选择‘Build Path’ → ‘Configure Build Path’，打开‘Properties’对话框，切换到‘Libraries’选项卡，点击‘Add Class Folder’。选择文件夹的‘Logo’复选框，它可以在‘assets/Textures’中找到，也可以在‘assets/Jasper Reports’中找到‘log4j’的复选框。点击‘OK’，关闭两个对话框。

7. 右键点击 Eclipse 的 Package Explorer 中‘eu.opends.main.Simulator’运行 OpenDS，并选择'Run As' → 'Java Application'。

8. 当操作界面出现时选择分辨率，然后点击’OK’。（运行构建版本 3.1

9. 指定驾驶员名称。

10. 选择地图模型并点击’Strat’。

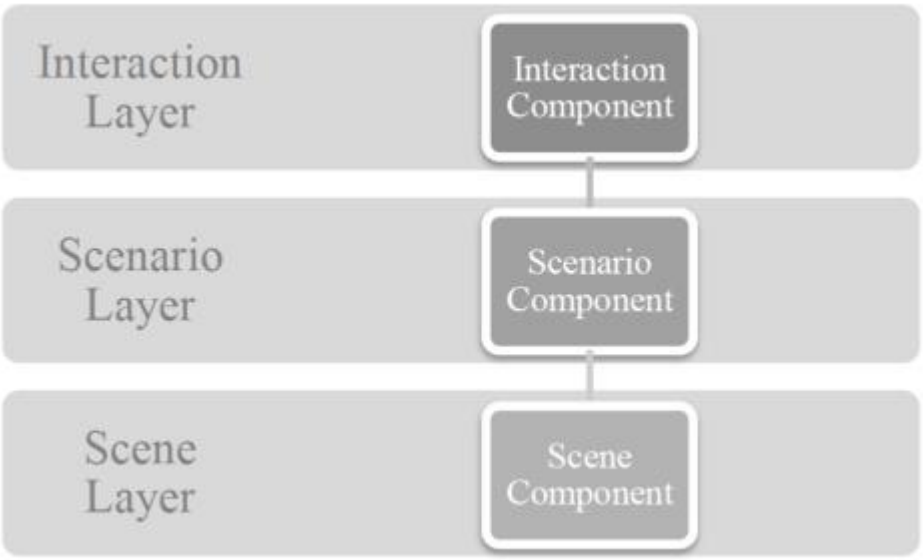
[2] http://www.opends.eu

5 模拟器

本章包括 OpenDS 的模拟组建的用户手册。在以下对 JavaDoc 源代码文档3的引用中可以找到驾驶任务文档(包括注释 XML 模式文件和示例)和默认的按键快捷表。

5.1 驾驶任务

一个 OpenDS 的驾驶任务描述总是由以下三个概念组成：scene、scenario、interaction。为了运行模拟情景，一个驾驶任务描述通常可作为 XML-file（分别是 scene.xml, scenario.xml, interaction.xml）和模拟器设置的文件(settings.xml)。



所有的文件都可以在每个项目文件夹中找到。

assets/DrivingTasks/Projects/<projectfolder>/

[3] http://opends.eu/JavaDoc/

5.1.1 场景(Scene)

在 OpenDS 中，场景的概念是指与驾驶相关的所以物体，如司机的汽车、道路标志、交通信号灯、建筑物、街道、交通车辆等等。每个对象都有一些可以在相应场景文件中(xml)中指定的属性。这些属性描述了一个元素的位置、大小、形状和质量。相关技术细节见下文。

每个 scene.xml 文件包含了与驾驶任务相关的所有元素的说明。说明的部分有：声音、图像、3D 模型、几何图形、复位点、重力和光线。

```
<scene>
   <sounds>…</sounds>
   <pictures>…</pictures>
   <models>…</models>
   <geometries>…</geometries>
   <resetPoints>…</resetPoints>
   <gravity>…</gravity>
   <lights>…</lights>
</scene>
```

5.1.1.1 声音(Sound)

音频文件可以在<sounds>元素中声明，并且必须包含至少一个唯一的标识符和文件路径。标识符作为引用是必需的，例如，从交互层开始播放声音文件。其他属性，如音量和音调，无论是声音的位置或方向，以及它是否循环播放，都可以被指定。所有在运行时可用的声音文件都必须在该元素中声明。

下面的示例显示了一个位置声音“beep”，它具有设定位置和 50%的音量，并且没有对音调进行更改。当启动回放时，这个文件将只播放一次(loop = false)

```
<sound id="beep" key="Sounds/Effects/Beep.ogg">
   <positional value="true" >
      <translation>
```

```
            <vector>
                <entry>0</entry>
                <entry>-5</entry>
                <entry>3.5</entry>
            </vector>
        </translation>
    </positional>
    <directional value="false" />
    <loop>false</loop>
    <volume>0.5</volume>
    <pitch>1</pitch>
</sound>
```

5.1.1.2 图片(Pictures)

在<pictures>元素中，可以在模拟器的运行时定义在显示框中显示的一些图像。除了基本属性：标识符、文件路径以及水平前背景/背景)之外，用户还可以选择指定图像的位置(绝对/相对)、尺寸和透明度。此外，还可以设置图像在启动时的可见性。使用唯一标识符，可以在运行时对图像进行处理，例如更改其可见性。

下面的例子显示了一个透明的 100*100 像素的图片“image01”，它是可见的，位于模拟屏幕顶部 10 像素和左侧 20%。

```
<picture id="image01" key="Textures/image01.png" level="1">
    <vPosition>
        <fromTop unit="px" value="10" />
    </vPosition>
    <hPosition>
        <fromLeft unit="%" value="20" />
    </hPosition>
    <width>100</width>
    <height>100</height>
    <useAlpha>true</useAlpha>
    <visible>true</visible>
</picture>
```

5.1.1.3 模型(Models)

<models>元素的作用是：定义所有在驾驶环境中可应用的模型。至少要指定一个惟一的标识符和一个文件路径(或引用到下面的几何图形)。其他可选参数包括质量、初始能否可视、碰撞形状、比例、旋转和初始位置（称为“平移”）。如果这些参数没有设置，则将应用默认值。参数质量，可视性和碰撞形状都是简单的浮点型，布尔型和字符串型，但是，对于比例，旋转和翻译，这是三个浮点值的向量。如果使用四元数来描述旋转，您可以选择指定一个包含四个浮点值的向量。该元素中列出的所有模型将在启动时进行处理:根据可视性和碰撞形状值，将向场景图或物理节点添加一个模型。

下面的例子展示了一个可视的“可碰撞”模拟“驾驶车”，它具有 1200 千克的质量，并且设定一个的旋转值（绕上轴 95°）和位置（单位：米）。

```
<model id="driverCar" key="Models/Cars/drivingCars/Mercedes/Car.scene">
    <mass>1200</mass>
    <visible>true</visible>
    <collisionShape>meshShape</collisionShape>
    <scale>
        <vector jtype="java_lang_Float" size="3">
            <entry>1</entry>
            <entry>1</entry>
            <entry>1</entry>
        </vector>
    </scale>
    <rotation quaternion="false">
        <vector jtype="java_lang_Float" size="3">
            <entry>0</entry>
            <entry>95</entry>
            <entry>0</entry>
      </vector>
    </rotation>
    <translation>
        <vector jtype="java_lang_Float" size="3">
            <entry>-792.395</entry>
            <entry>0.969</entry>
```

```
            <entry>-33.835</entry>
        </vector>
    </translation>
</model>
```

5.1.1.4 几何图形(Geometries)

<geometries>元素的作用是定义预先定义的形状:立方体、球体、圆柱体和点。

所有这些几何图形都需要规范一个唯一标识符。另需定义箱体尺寸(高度、宽度、深度);球面的定义要求对曲面的半径和样本编号进行规范,以近似于曲面;柱面的定义要求对曲线表面的半径、高度、样本编号进行说明,以及是否关闭;点(point)的定义需要 translation 的规范。

这些几何图形(除了点)都可以在模型中声明引用,而不用指定文件路径。点只能从场景层引用,以便将它们用作计算机控制车辆的理想点或路径点。

这个例子定义了一个立方体“box01”,宽 2m,深 0.1m,高 11.5m。

```
<box id="box01">
    <width>2</width>
    <depth>0.1</depth>
    <height>11.5</height>
</box>
```

这个例子定义了一个半径为 5m 的球体。

```
<sphere id="sphere02">
    <samples axis="10" radial="10"/>
    <radius>5</radius>
</sphere>
```

这个例子定义了一个封闭的圆柱体“cylinder03”，半径为 0.5m，高为 5m。

```
<cylinder id="cylinder03">
        <samples axis="20" radial="20" />
        <radius>0.5</radius>
        <height>5</height>
        <closed>true</closed>
</cylinder>
```

这个例子定义了点“point04”的设定位置。

```
<point id="point04">
      <translation>
          <vector jtype="java_lang_Float"size="3">
              <entry>2</entry>
              <entry>0.4</entry>
              <entry>-86</entry>
          </vector>
      </translation>
</point>
```

5.1.1.5 复位点(Reset Points)

<reset Points>元素的作用是将一个由初始位置和选择的复位点标记给一个唯一标识符。在运行时，驾驶汽车可被重置到指定的位置和方向，并在运行时触发相关事件。

```
<resetPoint id="reset01">
      <translation>
          <vector jtype="java_lang_Float" size="3">
              <entry>-61</entry>
              <entry>0</entry>
              <entry>38</entry>
          </vector>
      </translation>
      <rotation quaternion="true">
          <vector jtype="java_lang_Float" size="4">
              <entry>0</entry>
```

```
            <entry>0.707107</entry>
            <entry>0</entry>
            <entry>-0.707107</entry>
        </vector>
    </rotation>
</resetPoint>
```

这个例子显示了一个重置点“reset01”，它以一个给定的位置（“translation”）和方向（“rotation”）作为四元数。

5.1.1.6 重力(Gravity)

<gravity>元素可以定义重力加速度。（m/s2）

5.1.1.7 光线(Lights)

<light>元素用于声明不同的光源：点光源（例如路灯）、定向光（例如太阳光）和带有特定参数的环境光。

点光源需要定义位置矢量、半径和颜色矢量；定向光需要定义方向矢量和颜色矢量；环境光只需要定义颜色矢量。

这个例子定义了一个点光源，它是具有一个给定的位置，半径为 2m 的红色(=RGBA[1,0,0,1])光。

```
<pointLight>
    <position>
        <vector jtype="java_lang_Float" size="3">
            <entry>5</entry>
            <entry>-20.5</entry>
            <entry>101.75</entry>
        </vector>
    </position>
    <radius>2</radius>
    <color>
```

```
            <vector jtype="java_lang_Float" size="4">
                <entry>1.0</entry>
                <entry>0.0</entry>
                <entry>0.0</entry>
                <entry>1.0</entry>
            </vector>
        </color>
</pointLight>
```

这个例子定义了一个定向光源，它往一个给定的方向发射白(=RGBA[1,1,1,1])光,亮度为 0.7。

```
<directionalLight>
        <direction>
            <vector jtype="java_lang_Float" size="3">
                <entry>1</entry>
                <entry>0.5</entry>
                <entry>-5.5</entry>
            </vector>
        </direction>
        <color>
            <vector jtype="java_lang_Float" size="4">
                <entry>0.7</entry>
                    <entry>0.7</entry>
                    <entry>0.7</entry>
                    <entry>1.0</entry>
            </vector>
        </color>
</directionalLight>
```

这个例子显示了一个全局蓝(=RGBA[0,0,1,1])色的环境光。

```
<ambientLight>
        <color>
            <vector jtype="java_lang_Float" size="4">
                <entry>0</entry>
                <entry>0</entry>
```

```
                <entry>1</entry>
                <entry>1</entry>
            </vector>
        </color>
</ambientLight>
```

其他更多详细描述可以在 description[4]的 scene.xml 文件中找到。

5.1.2 情节(Scenario)

在 OpenDS 中，情节的定义与剧院里的场景相似，因为它描述了每个演员必须扮演的角色。它可以为在场景层中定义的任何对象提供语义信息。相关技术细节见下文。

每个 scenario.xml 文件包含了驾驶相关的语义信息。到目前为止，所支持的语义信息包括：天气状况、驾驶汽车和交通工具。在之后的阶段，这个文件还将用于注释关于基础设施的语义信息（道路标识、交通信号灯等）。

```
<scenario>
    <environment>…</environment>
    <driver>…</driver>
    <traffic>…</traffic>
    <road>…</road>
</scenario>
```

5.1.2.1 环境(Environment)

<environment>元素可以对不同的天气进行设置调整；例如雪、雨和雾的强度可以用百分比进行设置。

[4] http://opends.eu/drivingtask/sceneXSD.html

```
<weather>
    <snowingPercentage>100</snowingPercentage>
    <rainingPercentage>0</rainingPercentage>
    <fogPercentage>52</fogPercentage>
</weather>
```

5.1.2.2 司机(Driver)

<driver>元素可以更详细地描述驾驶汽车（<car>元素），也可以描述飞行视角（<camera Flight>元素）

```
<cameraFlight>
    <speed>50</speed>
    <automaticStart>true</automaticStart>
    <automaticStop>true</automaticStop>
    <track>
        <point ref="startPoint" />
        <point ref="waypoint01" />
        <point ref="waypoint02" />
        <point ref="waypoint03" />
        <point ref="endPoint" />
    </track>
</cameraFlight>
```

<car>元素的作用是通过将对应的 ID 传递给“ref”-属性来引用底层的 3D 对象。此外，还可以设置轮胎、发动机、传动、悬挂、车轮和制动装置这些的复位点。

```
<car ref="driverCar">
    <resetPoints>
        <resetPoint ref="reset01" />
        <resetPoint ref="reset02" />
    </resetPoints>
    <engine>
        <engineOn>true</engineOn>
        <minSpeed>0</minSpeed>
        <maxSpeed>180</maxSpeed>
        <acceleration>3.3</acceleration>
        <minRPM>750</minRPM>
        <maxRPM>7500</maxRPM>
    </engine>
    <transmission>
        <automatic>true</automatic>
        <reverse>3.182</reverse>
        <forward>
            <vector jtype="java_lang_Float" size="6">
                <entry>3.615</entry>
                <entry>1.955</entry>
                <entry>1.281</entry>
                <entry>0.973</entry>
                <entry>0.778</entry>
                <entry>0.646</entry>
            </vector>
        </forward>
    </transmission>
    <suspension>
        <stiffness>120</stiffness>
        <compression>0.2</compression>
        <damping>0.3</damping>
    </suspension>
    <wheel>
        <frictionSlip>50</frictionSlip>
    </wheel>
    <brake>
        <decelerationFreeWheel>2.0</decelerationFreeWheel>
        <decelerationBrake>8.7</decelerationBrake>
    </brake>
</car>
```

此外，一个预定义的轨迹的点（<ideal Track>元素）可以在<driver>元素的作用下指定为有序的列表。这条理想的线可以被分析工具用来与驾驶的轨迹进行比对，可以计算司机的驾驶表现。

```
<idealTrack>
      <point ref="point_01" />
      <point ref="point_02" />
      <point ref="point_03" />
      <point ref="point_04" />
      <point ref="point_05" />
      <point ref="point_06" />
</idealTrack>
```

5.1.2.3 交通(Traffic)

<traffic>元素是一个包含了所有车辆的列表（在未来也将包括行人、自行车等），这些车辆围绕所附连的<waypoints>元素中的路点来自行移动。可以在<vehicle>元素中指定的车辆的属性：移动的 3D 物体的文件路径、质量、加减速值，以及引擎是否开始运行。与车辆行驶路径相关的属性有:最大预见距离、曲线张力、路径是否循环、路径是否可见(调试模式)、起点，以及 waypoint 列表中包含 3d 坐标和在相应的路径点不被超过限制速度路径点的有序集合。

```
<vehicle id="car01">
      <modelPath>Models/Cars/drivingCars/CarGreen/Car.scene</modelPath>
      <mass>800</mass>
      <acceleration>3.3</acceleration>
      <decelerationBrake>8.7</decelerationBrake>
      <decelerationFreeWheel>2.0</decelerationFreeWheel>
      <engineOn>true</engineOn>
      <maxDistanceFromPath>3.0</maxDistanceFromPath>
      <curveTension>0.05</curveTension>
      <pathIsCycle>false</pathIsCycle>
```

```
    <pathIsVisible>true</pathIsVisible>
    <startWayPoint>WayPoint_01</startWayPoint>
    <wayPoints>
        <wayPoint id="WayPoint_1">
            <translation>
                <vector jtype="java_lang_Float" size="3">
                    <entry>72.61561</entry>
                    <entry>0.108792834</entry>
                    <entry>-277.24188</entry>
                </vector>
            </translation>
            <speed>50</speed>
        </wayPoint>
        <wayPoint id="WayPoint_2">
            <translation>
                <vector jtype="java_lang_Float" size="3">
                    <entry>38.26012</entry>
                    <entry>0.108847</entry>
                    <entry>-229.40056</entry>
                </vector>
            </translation>
            <speed>50</speed>
        </wayPoint>
    </wayPoints>
</vehicle>
```

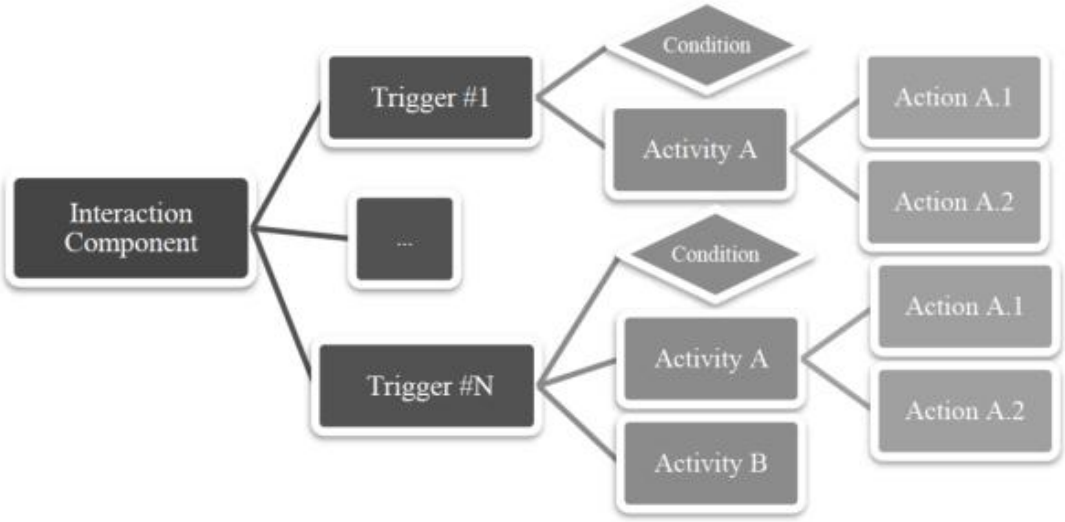
更多细节见 description[5]的 scenario.xml 文件。

5.1.3 交互(Interaction)

在场景层和情节层之上，交互层在模拟过程中任何时候都可以改变由 met 条件触发的对象的行为参数。交互层被划分为两个部分：活动声明和触发器声明。在触发器声明部分中，可以将一个条件分配给一个活动列表（活动部分的声明），该列表将在触发时按序列的顺序执行。每个活动由一系列操作组成，这些操作构成了可以由 OpenDS 处理的最原子级的指令。每个操作由一组参数-值对组成，描述了在执行相应操作时要更新的参数。在模拟运行的同时，连续检查每个触发器

[5] http://opends.eu/drivingtask/scenarioXSD.html

的状态，一旦条件被评估为真，相关的活动将按照它们的顺序执行。下图描述了交互组件的结构



详细的技术参照下文

interaction.xml 文件可以允许定义在运行时触发的各种事件。为此，可以指定事件触发的条件和需要执行的操作。例如车辆按照设定的行程与道路中特定对象相撞。可以触发的操作包括：操作对象、暂停模拟、复位汽车、移动通信、播放声音文件等。完整的列表可以在附录中找到

交互层被划分为活动声明和触发器声明。

```
<interaction>
    <activities>…</activities>
    <triggers>…</triggers>
</interaction>
```

5.1.3.1 活动(Activities)

<activities>元素可以自定义活动，它由一系列操作组成的，可以被 OpenDS

处理的最细微的指令。每一个操作本身由一组参数值对组成，描述在执行操作时要更新的参数。接下来，将重点介绍所描述的结构：

```
<activities>
    <activity id="activity01">
        <action id="" delay="" repeat="">
            <parameter name="" value="" />
            <parameter name="" value="" />
            <parameter name="" value="" />
            ...
        </action>
        <action id="" delay="" repeat="">
                <parameter name="" value="" />
                <parameter name="" value="" />
                <parameter name="" value="" />
                ...
        </action>
        ...
    </activity>
    <activity id="activity02">
        <action id="" delay="" repeat="">
            <parameter name="" value="" />
            <parameter name="" value="" />
            <parameter name="" value="" />
            ...
        </action>
        <action id="" delay="" repeat="">
            <parameter name="" value="" />
            <parameter name="" value="" />
            <parameter name="" value="" />
            ...
        </action>
        ...
    </activity>
    ...
</activities>
```

<activities>元素中可以包含任意数量的<activities>元素，这些元素可以包含任意数量的<action>元素。为了给活动分配条件，它们需要一个唯一标识符

（也就是代码中的 id）。

在附录中，有 OpenDS 可以处理的预定义动作的完整集合。它们可能由必需的、可选的和互斥的参数组成。除了特定操作的参数外，还可以为所有操作指定通用参数“delay”和“repeat”。这些参数用于延迟执行相应的操作，或限制最大执行的数量。

通用参数	类型	是否必需	默认	描述
delay	浮点	不	0	延迟执行（以秒为单位）
repeat	整数	不	0	一个触发器可使用的次数（0=无限）

下面的例子显示了“manipulate Object”动作，它将移动、旋转、缩放，并且更改已在 scene.xml 文件中定义的对象“RoadworksSign1”的可见性。此操作可以在触发后立即执行，在这个操作可重复 4 次后失效。

```
<action id="manipulateObject" delay="0" repeat="4">
    <parameter name="id" value="RoadworksSign1" />
    <parameter name="translationX" value="-81" />
    <parameter name="translationY" value="-1.693" />
    <parameter name="translationZ" value="-48" />
    <parameter name="rotationX" value="0" />
    <parameter name="rotationY" value="135" />
    <parameter name="rotationZ" value="0" />
    <parameter name="scaleX" value="0.02" />
    <parameter name="scaleY" value="0.02" />
    <parameter name="scaleZ" value="0.02" />
    <parameter name="visible" value="true" />
</action>
```

5.1.3.2 触发器(Triggers)

在<triggers>元素中，由<activities>元素定义的单个活动或者一系列活动可以通过引用相应的标识符来指定一个条件另外，活动也可以在本地进行声明，

不需要引用它们。当模拟器运行时，将不断检查所有条件。一旦一个条件为 true，相关的活动将按照它的顺序执行。以下是触发器声明部分的结构：

```
<triggers>
    <trigger id="collide" priority="1">
        <activities>
            <!—引用全局活动-->
            <activity ref="activity01" />
            <activity ref="activity02" />
            <activity ref="activity03" />
             ...
             <!-- 声明本地活动 -->
             <activity id="localActivity01">
                 <action id="manipulateObject" delay="0" repeat="1">
                 <parameter name="id" value="RoadWorksSign01" />
                 <parameter name="visible" value="true" />
             </action>
        </activity>
        ...
        </activities>
        <condition>collideWith:redBox</condition>
    </trigger>
    <trigger id="pressKey" priority="2">
        <activities>
             <activity ref="activity01" />
             <activity ref="activity04" />
             <activity ref="activity05" />
        </activities>
        <condition>pressKey:KEY_X</condition>
    </trigger>
    ...
</triggers>
```

<triggers>元素可以包含任意数量的<trigger>元素，每个<trigger>元素都包含一个<activities>元素和一个<condition>元素，以便将每个条件分配到对

应每个活动中。活动可以在本地<activities>元素中进行声明，或者从上面的活动声明部分引用（使用“ref”属性）。每个触发器都有一个唯一标识符（所谓的 id）和一个优先级，当两个触发器同时被触发时，它将决定执行的顺序（目前尚未实现）。

下表显示了 OpenDS 当前可以处理的预定义条件。

条件词头	描述	例子
CollideWith:	当车与特点对象相撞时，触发器启动	CollideWith : redBox
PressKey:	当给定按键被按下时，触发器启动	PressKey: KEY _ X

可以在以下网址中找到所有按键：

http://jmonkeyengine.org/javadoc/com/jme3/input/KeyInput.html

更多详细内容可以在 description[6]的 interaction.xml 文件中找到。

5.1.4 设置(Setting)

每个驾驶任务项目都包含一个单独的设置文件，该文件可以给模拟器定制一个特定的任务。这个文件与之前的场景、情节和交互不同，因为它不影响模拟器环境，而是影响模拟器的外观。相关技术细节见下文。

settings.xml 文件用于配置相应的驾驶任务的设定。例如，可以指定是否显示后视镜像。此外，所有的外部连接都是从该文件进行控制：已经可用的是一个外部可视化接口、一个用于从外部设备控制驾驶汽车的 CAN 服务器接口，以及一个用于从外部应用程序控制模拟器设置的接口。此外，settings.xml 文件可以用来配置输入设备，比如方向盘轴的设置，以及它们的灵敏度和键盘上的按键的

[6] http://opends.eu/drivingtask/interactionXSD.html

设置。此外，还可以配置鼠标滚轮，用于设置外部摄像机模式下的距离缩放。

settings.xml 文件的结构：

```
<settings>
    <general>…</general>
    <analyzer>…</analyzer>
    <externalVisualization>… </externalVisualization>
    <CANInterface>…</CANInterface>
    <settingsControllerServer>…</settingsControllerServer>
    <reactionMeasurement>…</reactionMeasurement>
    <controllers>
        <joystick>…</joystick>
        <keyboard>…</keyboard>
        <mouse>…</mouse>
    </controllers>
</settings>
```

在下面的示例中，我们将详细列出所有可用的参数。

总体	**总体设置**		
参数	**类型**	**默认值**	**描述**
driverName （驾驶者姓名）	字符串	“”	驾驶者的名字
showRearviewMirror （显示后视镜）	布尔类型	false	后视镜在启动时是看得见的
showStats **（显示数据）**	布尔类型	false	统计的描绘器信息（三角形数，帧频，等等）
showAnalogIndicators （显示模拟指示器）	布尔类型	true	指示器上可看见模拟的速度和每分钟转速

showDigitalIndicators （显示数字指示器）	布尔类型	false	指示器上可看见数字的速度和每分钟转速
showFuelConsumption （显示耗油量）	布尔类型	false	指示器上可以看见数字的耗油量
numberOfScreens （屏幕数量）	整数	1	把屏幕垂直地分成几个部分
angleBetweenAdjacentCameras （邻近相机间的角度）	单精度	40	两个邻近相机之间水平旋转角度（屏幕数＞1）
frameOfView （视野的框架）	单精度	30.5	每台相机的张开角度（屏幕数＞1）
rearviewMirror/viewPortBottom （后视镜/视口末端）	单精度	0.78	后视镜底部结构的位置（0.0=屏幕底部，1.0=屏幕顶部）
rearviewMirror/viewPortTop （后视镜/视口顶端）	单精度	0.98	后视镜顶部结构的位置（0.0=屏幕底部，1.0=屏幕顶部）
rearviewMirror/viewPortLeft （后视镜/视口左端）	单精度	0.3	后视镜左边结构的位置（0.0=屏幕左边，1.0=屏幕右边）
rearviewMirror/viewPortRight （后视镜/视口右端）	单精度	0.7	后视镜右边结构的位置（0.0=屏幕左边，1.0=屏幕右边）

例子：

```
<general>
    <driverName>Peter</driverName>
    <showRearviewMirror>false</showRearviewMirror>
    <showStats>false</showStats>
    <showAnalogIndicators>true</showAnalogIndicators
    <showDigitalIndicators>false</showDigitalIndicators>
    <showFuelConsumption>false</showFuelConsumption>
    <numberOfScreens>3</numberOfScreens>
    <angleBetweenAdjacentCameras>40</angleBetweenAdjacentCameras>
```

```
        <frameOfView>30</frameOfView>
        <rearviewMirror>
            <viewPortBottom>0.75</viewPortBottom>
            <viewPortTop>0.95</viewPortTop>
            <viewPortLeft>0.4</viewPortLeft>
            <viewPortRight>0.6</viewPortRight>
        </rearviewMirror>
</general>
```

分析仪		分析仪设置	
参数	**类型**	**系统默认值**	**描述**
suppressPDFPopup（出错时弹出 pdf）	布尔	false	分析器上不能在反应措施实施后，自动弹出 PDF 格式文件（PDF 格式文件将会被储存在分析器数据中）

例子：

```
<analyzer>
   <suppressPDFPopup>false</suppressPDFPopup>
</analyzer >
```

外部可视化	与一个外部可视相关的设置 相机数据（位置和方向）将会被传送出去		
参数	**类型**	**默认值**	**描述**
enableConnection（启动联系）	布尔	false	启动时建立连接
Ip（互联网地址）	字符串	“192.168.0.1”	可视化服务器的协议地址
Port（端口）	整数	1234	连到能够处理的可视化服务器上的端口号

updateRate （更新速率）	整数	25	每秒的更新数量
scalingFactor （缩放要素）	单精度	1.0	相机位置的数据中将会用到缩放要素
sendPosOriAsOneString （发送位置方向）	布尔	false	将位置和方向数据合并成一个字符串。否则，两条字符串都会被发送。

例子：

```
<externalVisualization>
    <enableConnection>false</enableConnection>
    <ip>192.168.0.1</ip>
    <port>1234</port>
    <updateRate>20</updateRate>
    <scalingFactor>1</scalingFactor>
    <sendPosOriAsOneString>false</sendPosOriAsOneString>
</externalVisualization>
```

CAN 界面	**与外部应用服务器建立联系的设置，外部应用服务器可从汽车的 CAN 总线技术中提供数据。可以得到转向角和踩踏板的情况，还可以通过反馈信道获取速度和位置数据。**		
参数	**类型**	**系统默认值**	**描述**
enableConnection （启动联系）	布尔	false	启动时建立连接
Ip （互联网地址）	字符串	“192.168.0.2”	可视化服务器的协议地址
Port （端口）	整数	5678	连到能够处理的可视化服务器上的端口号

updateRate （更新速率）	整数	20	每秒的更新数量（只是反馈信道）
maxSteeringAngle （最大转向角）	单精度	270.0	由转向停止时 CAN 服务器提供数值（用来计算转动强度）

例子：

```
<CANInterface>
    <enableConnection>false</enableConnection>
    <ip>192.168.0.2</ip>
    <port>5678</port>
    <updateRate>20</updateRate>
    <maxSteeringAngle>180</maxSteeringAngle>
</CANInterface>
```

控制服务器设置	**为了和在进行模拟时控制实验设置的外部应用服务器建立联系**		
参数	**类型**	**系统默认值**	**描述**
startServer （开始服务器）	布尔	错误	当模拟器启动时服务器会开始运行
Port （端口）	整数	1000	服务器守听端口

例子：

```
<settingsControllerServer>
    <startServer>false</startServer>
    <port>1000</port>
</settingsControllerServer>
```

反应方式	为了直方图美观简洁，反应组群（集合相似的反应方式之间的相互作用）可能被分配到预定义的颜色。		
参数	**类型**	**系统默认值**	**描述**
groupRed （红色组）	字符串	“”	这种反应组会标注成红色
groupGreen （绿色组）	字符串	“”	这种反应组会标注成绿色
groupYellow （黄色组）	字符串	“”	这种反应组会标注成黄色
groupCyan （蓝绿色组）	字符串	“”	这种反应组会标注成蓝绿色
groupBlue （蓝色组）	字符串	“”	这种反应组会标注成蓝色
groupMagenta （洋红色组）	字符串	“”	这种反应组会标注成洋红色

例子：

```
<reactionMeasurement>
    <groupRed>reactionGroup01</groupRed>
    <groupGreen>reactionGroup02</groupGreen>
    <groupYellow>brakeReaction</groupYellow>
    <groupCyan>steeringReaction</groupCyan>
    <groupBlue>noReaction</groupBlue>
    <groupMagenta></groupMagenta>
</reactionMeasurement>
```

键盘	设置和键盘相连接		
参数	**类型**	**默认值**	**描述**
keyAssignments（按键任务）	功能列表/密钥对		游戏控制器的 ID 将被使用（只有当一台以上的设备接通电源时）
注意： 预定义的模拟器功能被定义到一系列按键上。如果某个功能已经被定义到一个按键上，那么这个按键的默认值就会被覆盖。这可以用于替换系统的默认值。 注意：如果你用一个空的字符串取代一个键，功能则不能被启动。 完整的功能列表请参考附录中的 “系统默认按键” 可用的按键列表：http://jmonkeyengine.org/javadoc/com/jme3/input/KeyInput.html			

例子：

```
<keyboard>
    <keyAssignments>
        <!-- E 键启动引擎 -->
        <keyAssignment function="start_engine" key="KEY_E" />
        <!-- 使用 V 键和 C 键切换视觉 -->
        <keyAssignment function="toggle_cam" key="KEY_V,KEY_C" />
        <!-- 禁用喇叭 -->
        <keyAssignment function="horn" key="" />
    </keyAssignments>
</keyboard>
```

操纵杆	与游戏控制器有关的设置（如：方向盘）		
参数	**类型**	**默认值**	**描述**

controllerID （控制器 ID）	整数	0	游戏控制器的 ID 将被使用（超过一台设备接通电源时）
steeringAxis （方向盘轴线）	整数	1	指定方向盘轴线是哪一根
invertSteeringAxis （倒置方向盘轴）	布尔	false	从转向轴返回的反转值
steeringSensitivityFactor （转向灵敏度系数）	浮点数	1.0	增加/减少转向灵敏度系数
pedalAxis （踩踏板的轴）	整数	2	指定哪一根轴是踩踏板的轴
invertPedalAxis （倒置踏板轴线）	布尔	false	从踏板轴返回的倒置值
pedalSensitivityFactor （踩踏板灵敏度系数）	浮点数	1.0	增加/减少踩踏板灵敏度系数
注意： 确保在你的游戏控制器中，油门和刹车踩踏板已经被调为相同的轴线。 未来，将会把模拟器各个功能分配到游戏控制器按钮上（就像已经可以实现的键盘一样）			

例子：

```
<joystick>
    <controllerID>0</controllerID>
    <steeringAxis>1</steeringAxis>
    <invertSteeringAxis>false</invertSteeringAxis>
    <steeringSensitivityFactor>1.0</steeringSensitivityFactor>
    <pedalAxis>2</pedalAxis>
    <invertPedalAxis>false</invertPedalAxis>
    <pedalSensitivityFactor>1.0</pedalSensitivityFactor>
    <!-- button assignment not available yet -->
```

```
    <keyAssignments>
        <keyAssignment function="start_engine" key="JOY_1" />
        <keyAssignment function="stop_engine" key="JOY_2" />
    </keyAssignments>
</joystick>
```

鼠标	**与鼠标有关的设置（除非相机运动到视野外围）**		
参数	**类型**	**默认值**	**描述**
minScrollZoom （最小滚动变焦值）	浮点数	1.0	以米为单位，相机到车的最小距离（当使用滚轮时）
maxScrollZoom （最大滚动变焦值）	浮点数	40.0	以米为单位，相机到车的最大距离（当使用滚轮时）
scrollSensitivityFactor （滚动的敏感度系数）	浮点数	5.0	滚轮的敏感度系数

例子：

```
<mouse>
    <minScrollZoom>1.0</minScrollZoom>
    <maxScrollZoom>40.0</maxScrollZoom>
    <scrollSensitivityFactor>5.0</scrollSensitivityFactor>
</mouse>
```

更多的详细信息可以到 settings.xml [7]文件中查看。

[7] http://opends.eu/drivingtask/settingsXSD.html

6 其他各项（Miscellaneous）

6.1 转化模型

为了将 3D 模型转换为与模拟器兼容的文件格式(OgreXML)，可以使用 Blender 中的 OgreXML- exports 导出文件。使其运行如下:

1. 安装Blender[8]。因新版本存在一些毛病，所以要确保安装的是2.49版本。

2. 安装Python2.66[9]。对于Blender本身来说，这不是必需的，但是对于导出脚本来说。必须选择与你的Blender版本相匹配的Python版本(在启动Blender时您将看到它)。

3. 将相关的文件[10]复制到Blender scripts文件夹下。
   注：Blender2.49版和Windows 7的这个文件都存储
   在:C:\Users\<name>\AppData\Roaming\BlenderFoundation\Blender\.blender\scripts

4. 将场景导出程序，并且复制到与网格导出程序[11]相同的文件夹中。。

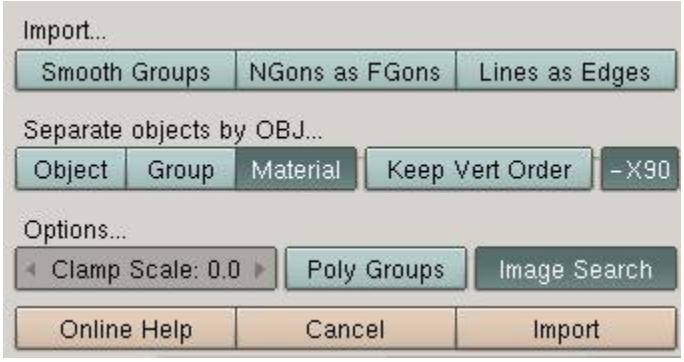
关于如何使用网格和场景导出的更多细节可以在这里找到:

http://www.ogre3d.org/tikiwiki/Blender+Exporter

http://www.ogre3d.org/tikiwiki/Blender+dotScene+Exporter

[8] http://blender.org
[9] http://www.python.org/download/releases
[10] http://www.xullum.net/lefthand/downloads/temp/BlenderExport.zip
[11] http://ogreaddons.svn.sourceforge.net/viewvc/ogreaddons/trunk/blendersceneexporter/ogredotscene.py

当Blender在导出*.obj模型过程中，将*.obj格式转化成XML模型时必须使用下面的设置方式。



继续导出操作，在OgreXML的所有部件中，选择你想要导出的部件。然后像上面学习教程中描述的那样继续 scene（场景）导出(File→Export →OGRE Scene)和meshes（网格）导出(File→Export →OGREMeshes)。

6.2 CAN 接口（CAN interface）

为了建立模拟器与汽车的链接，您必须提供一个 TCP-server（tcp 服务器），该服务器接收来自汽车 CAN-bus 的数据并将这些数据转发给模拟器。在 setting.xml 文件中通过<CANInterface>元素阔以指定模拟器（客户端）必须设置的服务器 ip 和端口。建立连接后，模拟器等待以下指令:

转向角度(从-x 到+x，其中 x 为最大值。在 settings.xml 中指定的转向角度:

```
<message>
      <action name="steering">-92.5</action>
</message>
```

加速度(值从 0 到 1.0):

```
<message>
      <action name="acceleration">0.5</action>
</message>
```

煞车(值从 0 到 1.0):

```
<message>
      <action name="brake">0.3</action>
</message>
```

切换视觉:

```
<message>
      <action name="button">cs</action>
</message>
```

重置驾驶汽车:

```
<message>
   <action name="button">return</action>
</message>
```

6.3 图形用户界面（Graphical User Interfaces）

在 OpenDS 中可以使用的图形用户界面是通过集成到 jMonkeyEngine 的 Nifty GUItoolkit 创建的。用户界面的布局以 xml 格式存储在“assets”文件夹的子文件夹“Interfaces”中:

文件(.xml)	描述
AnalyzerFileSelectionGUI.xml (分析器文件选择的图形用户界面)	在为分析器组件时，对驾驶数据文件的选择的图形用户界面
DrivingTaskSelectionGUI.xml (驾驶任务选择的图形用户界面)	在为模拟组件方面，对驾驶任务文件的选择的图形用户界面
InstructionScreenGUI.xml (指导屏幕图形用户界面)	显示指导的图形用户界面
KeyMappingGUI.xml (键盘映射图形用户界面)	显示所有按键代表的任务（当 F1 被按下时）的图形用户界面
MessageBoxGUI.xml (对话框图形用户界面)	对话框适用于模拟中的对话演示
ShutDownGUI.xml (关闭的图形用户界面)	显示关闭对话框的图形用户界面（当 ESC 被按下时）

在 OpenDS 源代码中的“eu.opends.niftyGui”包里，可以找到图形用户界面的事件处理器。

更多关于 Nifty 图形化用户界面的使用信息都可以在 jMonkeyEngine[12] 网站和 Nifty GUI[13]用户手册上找到。

[12] http://jmonkeyengine.org/wiki/doku.php/jme3:advanced:nifty_gui
[13] http://sourceforge.net/projects/nifty-gui/files/nifty-gui/1.3.2/nifty-gui-the-manual-1.3.2.pdf/download

7 附录

7.1 模拟器中默认按键

下表显示了模拟器中可用的默认按键。根据函数方法的 ID 可以在 settings.Xml 文件中手动更默认按键。

默认按键	函数方法的 ID	描述
UP	Accelerate	让车加快速度
DOWN	accelerate_back	让车减慢速度
SPACE	Brake	刹住车
F5	close_instruction_screen	关掉指示屏幕
F	hazard_lights	让转向灯闪光
H	Horn	喇叭
SPACE	report_landmark	报告路标
G	report_reaction	报告反应
R	reset_car	重新设定车辆到下一个重置位置
1	reset_car_pos1	将车重新设定到 1 号位置（起始位置）
2	reset_car_pos2	将车重新设定到 2 号位置（被设定的）
3	reset_car_pos3	将车重新设定到 3 号位置（被设定的）
4	reset_car_pos4	将车重新设定到 4 号位置（被设定的）
5	reset_car_pos5	将车重新设定到 5 号位置（被设定的）
6	reset_car_pos6	将车重新设定到 6 号位置（被设定的）

7	reset_car_pos7	将车重新设定到 7 号位置（被设定的）
8	reset_car_pos8	将车重新设定到 8 号位置（被设定的）
9	reset_car_pos9	将车重新设定到 9 号位置（被设定的）
0	reset_car_pos10	将车重新设定到 10 号位置（被设定的）
T	reset_fuel_consumption	重新设定燃料消耗
F9	rotate_object_left	目标位置：将目标旋转到左侧
F7	rotate_object_left_fast	目标位置：将目标快速旋转到左侧
F10	rotate_object_right	目标位置：将目标旋转到右侧
F8	rotate_object_right_fast	目标位置：将目标快速旋转到右侧
F11	set_object	目标位置：把目标放在地图上
PGDN	shift_down	减慢传动装置
PGUP	shift_up	加快传动装饰
ESCAPE	Shutdown	退出模拟器
O	start_pause	暂停模拟
LEFT	steer_left	将车向左开
RIGHT	steer_right	将车向右边开
I	stop_pause	继续（驾驶）模拟
END	toggle_automatic	在自动传输和手动传输之间切换
BACK	toggle_backmirror	显示/隐藏后视镜
V	toggle_cam	改变摄像机视图

RETURN	toggle_cinematics	能/不能让相机飞行
E	toggle_engine	引擎开启/关闭
L	toggle_headlight	头灯切换状态（off-1-2）
F1	toggle_keymapping	显示/隐藏键盘映射
M	toggle_messagebox	显示/隐藏对话框
D	toggle_min_speed	巡航控制开启/关闭
F12	toggle_object	目标位置：选择下一个对象
P	toggle_pause	停顿/继续（驾驶）模拟
F6	toggle_physics_debug	显示/隐藏物理调试
S	toggle_record_data	开始/停止汽车数据的记录
F4	toggle_stats	显示/隐藏统计渲染器信息
A	toggle_trafficlightmode	改变交通灯的模式（trigger-program-external-blinking-off）
W	toggle_wireframe	显示/隐藏线框
J	turn_left	打左转信号灯
K	turn_right	打右转信号灯

7.2 分析仪默认按键

下表显示了在模拟器中可使用的默认按键。

默认按键	**描述**

F1	显示/隐藏键盘映射
ESCAPE	退出分析器
V	改变相机视觉
1	显示/隐藏点
2	显示/隐藏线
3	显示/隐藏圆锥体
UP	移动相机位置至下一个数据点
DOWN	移动相机位置至上一个数据点
RIGHT	向前移动相机位置

7.3 可用事件

下面显示了可用事件的完整列表。这些事件可以在 interaction.xml 文件交互中引用。（ <action>元素）

7.3.1 发送信息（sendMessage）

发送信息	**将文本输出至屏幕以获得给定的秒数**			
参数	**类型**	**是否必填**	**默认值**	**描述**
Text （文本）	字符串	Yes		屏幕上显示的文本
Duration （持续时间）	整数	No	1	文本显示的秒数（0=无限）

7.3.2 操作对象（manipulateObject）

操作对象	操作转化，旋转，规模和/或者给定模型的可视化程度			
参数	**类型**	**是否必填**	**默认值**	**描述**
Id	字符串	Yes		需操作模型的 ID
setTranslationX	浮点数	No	0.0	将模型转换为 X 坐标
setTranslationY	浮点数	No	0.0	将模型转换为 y 坐标
setTranslationZ	浮点数	No	0.0	将模型转换为 z 坐标
setRotationX	浮点数	No	0.0	围绕 X 轴旋转模型
setRotationY	浮点数	No	0.0	围绕 y 轴旋转模型
setRotationZ	浮点数	No	0.0	围绕 z 轴旋转模型
setScaleX	浮点数	No	0.0	缩放模型至 X 坐标
setScaleY	浮点数	No	0.0	缩放模型至 y 坐标
setScaleZ	浮点数	No	0.0	缩放模型至 z 坐标
addTranslationX	浮点数	No	0.0	将这个值添加到模型的 X 坐标中
addTranslationY	浮点数	No	0.0	将这个值添加到模型的 y 坐标中
addTranslationZ	浮点数	No	0.0	将这个值添加到模型的 z 坐标中
addRotationX	浮点数	No	0.0	将这个值添加到模型绕 X 轴旋转
addRotationY	浮点数	No	0.0	将这个值添加到模型绕 y 轴旋转
addRotationZ	浮点数	No	0.0	将这个值添加到模型绕 z 轴旋转
addScaleX	浮点数	No	0.0	将这个值添加到模型 X 坐标范围内
addScaleY	浮点数	No	0.0	将这个值添加到模型 y 坐标范围内

addScaleZ	浮点数	No	0.0	将这个值添加到模型 z 坐标范围内
visible	布尔	No	true	使得模型可视化

7.3.3 操作图片（manipulatePicture）

操作图片	**操纵给定图片的可见性**			
参数	**类型**	**必填的**	**默认值**	**描述**
Id	字符串	Yes		需操纵图像的 ID
visible	布尔	Yes		使图像是否可见

7.3.4 暂停模拟（pauseSimulation）

暂停模拟	**在给定时间里暂停模拟**			
参数	**类型**	**是否必填**	**默认值**	**描述**
Duration (持续时间)	整数	No	1	停顿的秒数（0=无限）

7.3.5 开始记录（startRecording）

开始记录	**开始记录驾驶者信息**			
参数	**类型**	是否必填	**默认值**	**描述**

Track （踪迹）	整数	No	1	记录的 ID

7.3.6 停止记录（stopRecording）

停止记录	停止记录驾驶者信息			
参数	类型	是否必填	默认值	描述

7.3.7 重置汽车（resetCar）

重置汽车	将驱动车移动到给定的复位点			
参数	类型	是否必填	默认值	描述
重置点 ID	字符串	Yes		将驱动车移动到重置点的 ID

7.3.8 移动交通（moveTraffic）

移动交通	**将交通工具移动到指定位置**			
参数	**类型**	是否必填	**默认值**	**描述**
trafficObjectID （交通对象 ID）	字符串	Yes		交通工具移动 ID

wayPointID（路标 ID）	字符串	Yes		移动交通工具的路标的 ID

7.3.9 设置当前速度限制（setCurrentSpeedLimit）

设置当前速度限制	**将速度限制设置到给定的数值**			
参数	**类型**	是否必填	**默认值**	**描述**
speedLimit（限速）	整数	No	0	速度限制在千米/时（0=无限）

7.3.10 给启动速度设限（setUpcominigSpeedLimit）

给启动速度设限	**将启动速度设置到给定的值**			
参数	**类型**	是否必填	**默认值**	**描述**
speedLimit（限速）	整数	No	0	速度限制在千米/时（0=无限）

7.3.11 测量刹车时间（measureTimeUntilBrake）

测量刹车时间	**测量刹车反应时**

参数	类型	是否必填	默认值	描述
triggerName （触发名）	字符串	Yes		输出文件中识别的触发器 ID

7.3.12 测量速度改变所需时间（measureTimeUntilSpeedChange）

测量速度改变所需时间	测量速度改变到给定值所需的时间			
参数	**类型**	是否必填	**默认值**	**描述**
triggerName （触发名）	字符串	Yes		输出文件中识别的触发器 ID
speedChange （变速）	整数	Yes		速度以（千米/时）增加或降低

7.3.13 播放声音（playSound）

播放声音	播放场景图层中指定的声音文件			
参数	**类型**	是否必填	**默认值**	**描述**
soundID （声音 ID）	字符串	Yes		声音文件的播放 ID

7.3.14 请求绿色交通灯（requestGreenTrafficLight）

请求绿色交通灯	请求给定交通灯转变为绿色			
参数	**类型**	**是否必填**	**默认值**	**描述**
trafficLightID（交通灯 ID）	字符串	Yes		请求交通灯变绿的 ID

7.3.15 设置按键的反应计时器（setupKeyReactionTimer）

设置按键的反应计时器	设置一个从游戏控制器按钮输入到键盘做出反应的时间计算器			
参数	**类型**	**是否必填**	**默认值**	**描述**
timerID（计时器 ID）	字符串	No	“timer1”	计时器的 ID 是用于测量的。如果计时器正在使用，那么先前的测量结果将导致一个未被响应的反应。
reactionGroup（反应组）	字符串	Yes		将反应测量分配给一个组（例如，在 settings.xml 中“reactionGroup”中定义的颜色表示）
correctReaction（正确反应）	字符串	Yes		触发正确反应的键列表（如“key_H, joy_1”）

failureReaction （错误反应）	字符串	Yes		触发错误反应的键列表（如“key_G，joy_2”）
comment （评价）	字符串	No	“”	评论（将转发到输出）

7.3.16 设置一个车道改变的计时器（setupLaneChangeReactionTimer）

设置一个车道改变的计时器	**设置一个车道改变的反应计时器**			
参数	**类型**	**是否必填**	**默认值**	**描述**
timerID （计时器 ID）	字符串	No	“timer1”	计时器的 ID 是用于测量的。如果计时器正在使用，那么先前的测量结果将导致一个未被响应的反应。
congruenceClass （同等组）	字符串	Yes		将反应测量分配给一个组（例如，在 settings.xml 中“reactionGroup”中定义的颜色表示）
startLane （起始车道）	字符串	Yes		车道改变必须从哪儿开始
targetLane （目标车道）	字符串	Yes		车道改变必须在哪结束
minSteeringAngle （最小转向角）	浮点数	No		最小转向角必须被克服（百分比）
taskCompletionAfterTime （任务完成时）	浮点数	No		任务必须在 X 毫秒后被完成（0=无限制）
taskCompletionAf	浮点数	No		任务必须在 X 米后被完成（0=无限制）

terDistance （完成任务的距离）				
allowBrake （允许刹车）	布尔	No		驾驶者可能在改变车道时刹车吗？（如果是错误的，那么报告失败反应。）
holdLaneFor （保持该车道）	浮点数	No		在目标车道持续的毫秒数
failSound （失败声音）	字符串	No		在车道变换错误或未变换车道时，声音文件会被播放
successSound （成功声音）	字符串	No		在车道变换错误或未变换车道时，声音文件会被播放
Comment （评论）	字符串	No		评论（将转发到输出）

7.3.17 设置刹车反应计时器（setupBrakeReactionTimer）

设置刹车反应计时器	**为刹车设置一个反应计时器**			
参数	**类型**	**是否必填**	**默认值**	**描述**
timerID （计时器 ID）	字符串	No	“timer1”	计时器的 ID 是用于测量的。如果计时器正在使用，那么先前的测量结果将导致一个未被响应的反应。
congruenceClass （同等组）	字符串	Yes		将反应测量分配给一个组（例如，在 settings.xml 中

				“reactionGroup”中定义的颜色表示）
startSpeed（起始速度）	浮点数	No	80.0	汽车行驶到开始反映测量点时的最小速度
targetSpeed（目标速度）	浮点数	No	60.0	汽车行驶到停止反映测量点时的最大速度
mustPressBrakePedal（必须按压刹车踏板）	布尔	No	True	司机必须按下刹车踏板以获得成功的反应
taskCompletionAfterTime（任务完成时）	浮点数	No	0.0	任务必须在 X 毫秒后被完成（0=无限制）
taskCompletionAfterDistance（完成任务的距离）	浮点数	No	0.0	任务必须在 X 米后被完成（0=无限制）
allowLaneChange（允许车道改变）	布尔	No	True	驾驶者可能在刹车时改变车道吗？（如果是错误的，那么报告失败反应。）
holdSpeedFor（保持该速度）	浮点数	No		持续目标速度的毫秒数
failSound（失败声音）	字符串	No		在车道变换错误或未变换车道时，声音文件会被播放
successSound（成功声音）	字符串	No		在车道变换错误或未变换车道时，声音文件会被播放
Comment	字符串	No		评论（将转发到输出）

（评论）				

7.3.18 打开指示屏（openInstructionScreen）

打开指示屏	显示一个带有指导语的屏幕			
参数	**类型**	**是否必填**	**默认值**	**描述**
instructionID（指导 ID）	字符串	Yes		用于显示指示屏的 ID（可以定义 assets/Interface/InstructionScreenGUI.xml）

7.3.19 设置 TVPT 刺激（setTVPTStimulus）

设置 TVPT 刺激	为三车平台测试设置一个刺激			
参数	**类型**	**是否必填**	**默认值**	**描述**
stimulusID（刺激 ID）	字符串	Yes		刺激触发的 ID： -“brakeLight（刹车灯）”（引导车辆制动灯） -“turnSignal（转弯信号）”（跟踪车辆转弯信号） -“speedReduction（减速）”（使车辆减速） -“emergencyBrake（急刹车）”（使车辆两期刹车灯并减速）

7.3.20 写入数据库（writeToKnowledgeBase）

写入数据库	**在知识库中插入/编辑属性**			
参数	**类型**	**是否必填**	**默认值**	**描述**
path （路径）	字符串	Yes		插入/编辑属性的路径
propertyName （属性名）	字符串	Yes		插入/编辑属性的名称
propertyValue （属性值）	字符串	Yes		插入/编辑属性的数值
propertyType （属性类型）	字符串	Yes		插入/编辑属性的类型