

Omniverse 性能优化系列（一）：Tracy Profiler

原创 丽台科技 丽台科技 2025年10月29日 17:42 上海

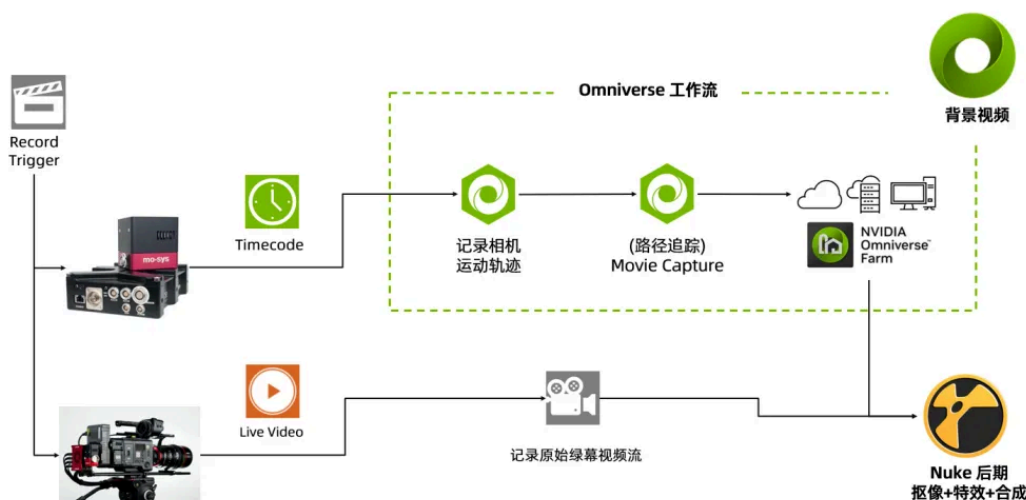
前文回顾：

实战项目拆解！当 FreeD 定位系统遇上 Omniverse Extension

NVIDIA Omniverse Extension 开发秘籍：Python/C++ 实战，附完整代码

UI 卡顿问题复现及分析

在进行 NVIDIA Omniverse™ USD 插件开发的时候遇到了一个性能卡顿的问题，这个功能的初衷是通过路径追踪和原始的绿幕视频，借助 Omniverse Farm 来实现高质量的后期自动化流程，实现思路是记录相机定位 FreeD 的运动轨迹，并记录保存到一个 USD 的 sublayer 当中，根据时间码（Timecode）进行后期自动化合成的流程，流程图如下：



在外部摄影机记录原始的绿幕影片素材的时候，点击开始 / 结束分别会触发一个时间码（Timecode）的信号，信号可以在 BMD 采集卡当中通过 SDK 获得，这样我们把从开始到结束的相机定位轨迹记录存至 buffer 中，然后更新到 USD 的 stage sublayer 中。

首先通过 Python API 创建一个 Sublayer，把记录的 sequence 通过 USD time sampler 记录到相机 prim 的 attribute 下面，仅对一万个 time sampler 进行记录并统一写入该 sublayer，后观察到该写入过程耗时达数十秒，且造成 Omniverse 主线程 UI 出现卡顿。经测试，无论采用同步、异步 AsyncIO 或线程方式执行操作，均未使情况得到改善，UI 卡死现象始终存在。（可以查看代码文件中注释的 1、2、3）

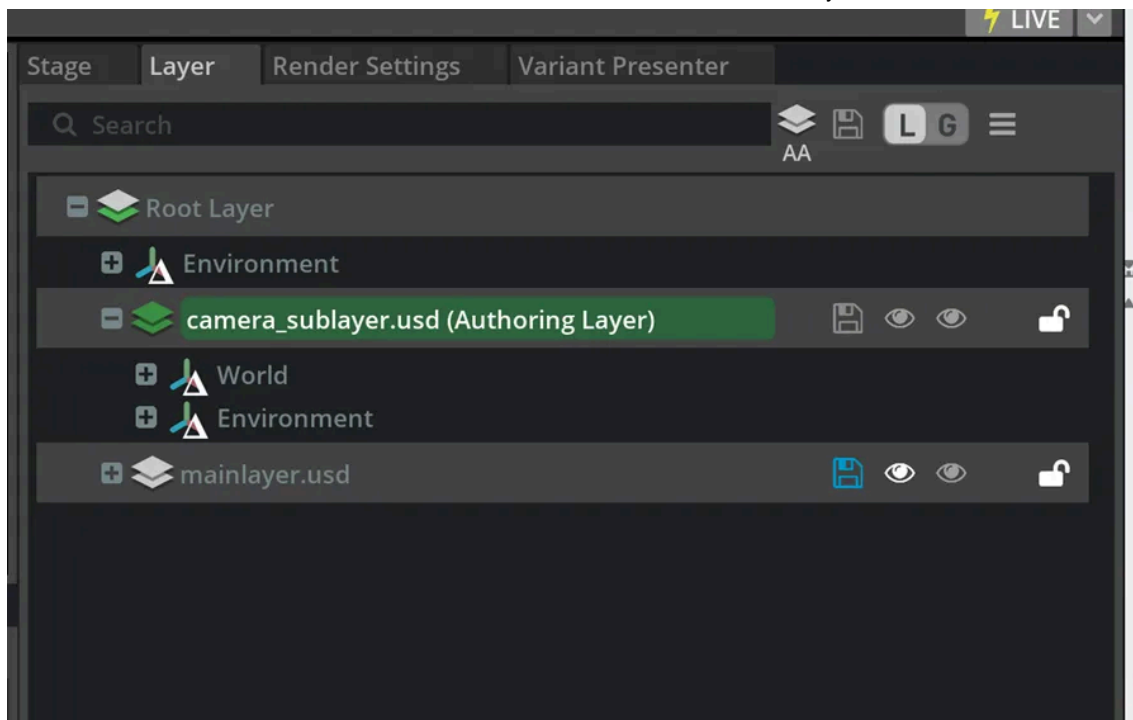
Tracy.py 的源代码如下：

```
1 import omni.kit.app
2 import time
3 import asyncio
4 from pxr import Sdf, Usd, UsdGeom, Gf
5 from omni.kit.usd.layers import LayerUtils, get_layers, LayerEditMode
6 import omni.kit.commands
7 from collections import deque
8 import os
9 from typing import List, Tuple
10 from omni.kit.widget.layers.path_utils import PathUtils
11 import carb
12 from typing import List, Tuple
13 from enum import Enum
14 import asyncio
15 from pxr import Sdf, Usd, UsdGeom
16 from pxr import Usd, UsdGeom, Gf
17 from omni.kit.async_engine import run_coroutine
18 from concurrent.futures import ALL_COMPLETED, ThreadPoolExecutor, wait
19
20 def create_prim(stage, prim_path="/World/Camera"):
21     prim = stage.GetPrimAtPath(prim_path)
22     if prim and prim.GetTypeName() == "Camera":
23         carb.log_info(f"Camera already exists at: {prim_path}")
24     return prim
```

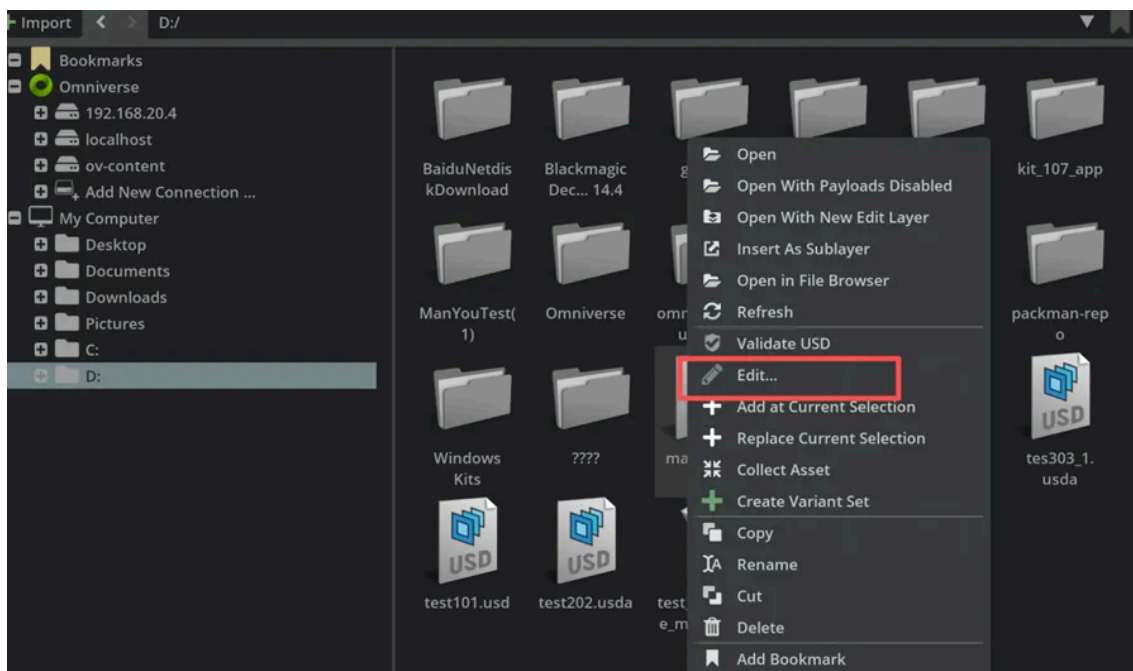
* 附代码链接: https://github.com/slayersong/OVPerf_Tracy/blob/main/tracy_profiler.py (复制链接至浏览器打开)

复现问题: 打开菜单中的 Developer -- Script Editor, 打开 tray_profiler.py 文件, 然后点击 Run, 可以看到创建了一个 camera_sublayer, 并且主 UI 卡住了几十秒无响应。

需要注意的是, 在 Omniverse USD 的 layer 层级继承覆盖当中, 在上层的 Layer 的行为会覆盖下层的 layer, 关于 USD layer 层级的关系, 请查看本文结尾提供的 DLI 课程链接。



然后在 Content Browser 中单击鼠标右键，选择 Edit，可以看到数据成功写入了 USD 文件，只是中间卡顿的时间过长。



分析：在遇到 Profiler 的时候不要盲猜，可能是 memory、IO Bound、Compute Bound 或者一些不太能想到的情况，这时候则需要利用专业化的工具进行分析定位，找到问题所在并解决，比如可以利用著名工具 **Tracy** (<https://github.com/wolfpld/tracy>)，该工具可以分析 CPU / GPU 性能瓶颈，并支持主流 Graphics API：DX、Vulkan、OpenGL、CUDA 等，且 Omniverse 已经把该工具与 Omniverse Kit 进行了集成。因此可以利用 Tracy 去看底层的 CallStack 里什么影响了这个操作，在 Omniverse 当中，Tracy 已经配置好了 Symbol 符号表，可以看到底层的代码函数调用堆栈，而后寻找具体是什么情况卡住了不正常的几十秒时间。

2.1 操作介绍

Omniverse 已经集成了 Tracy 的开发集成插件：

https://docs.omniverse.nvidia.com/extensions/latest/ext_profiler_tracy.html

Tracy 本身是一个著名的分析工具，具体的菜单操作可以参考知乎这个帖子：

<https://zhuanlan.zhihu.com/p/1915041165033607442>

UI 操作的详细讲解可参考如下视频：

07:27

Tracy 讲解文档

视频参考：

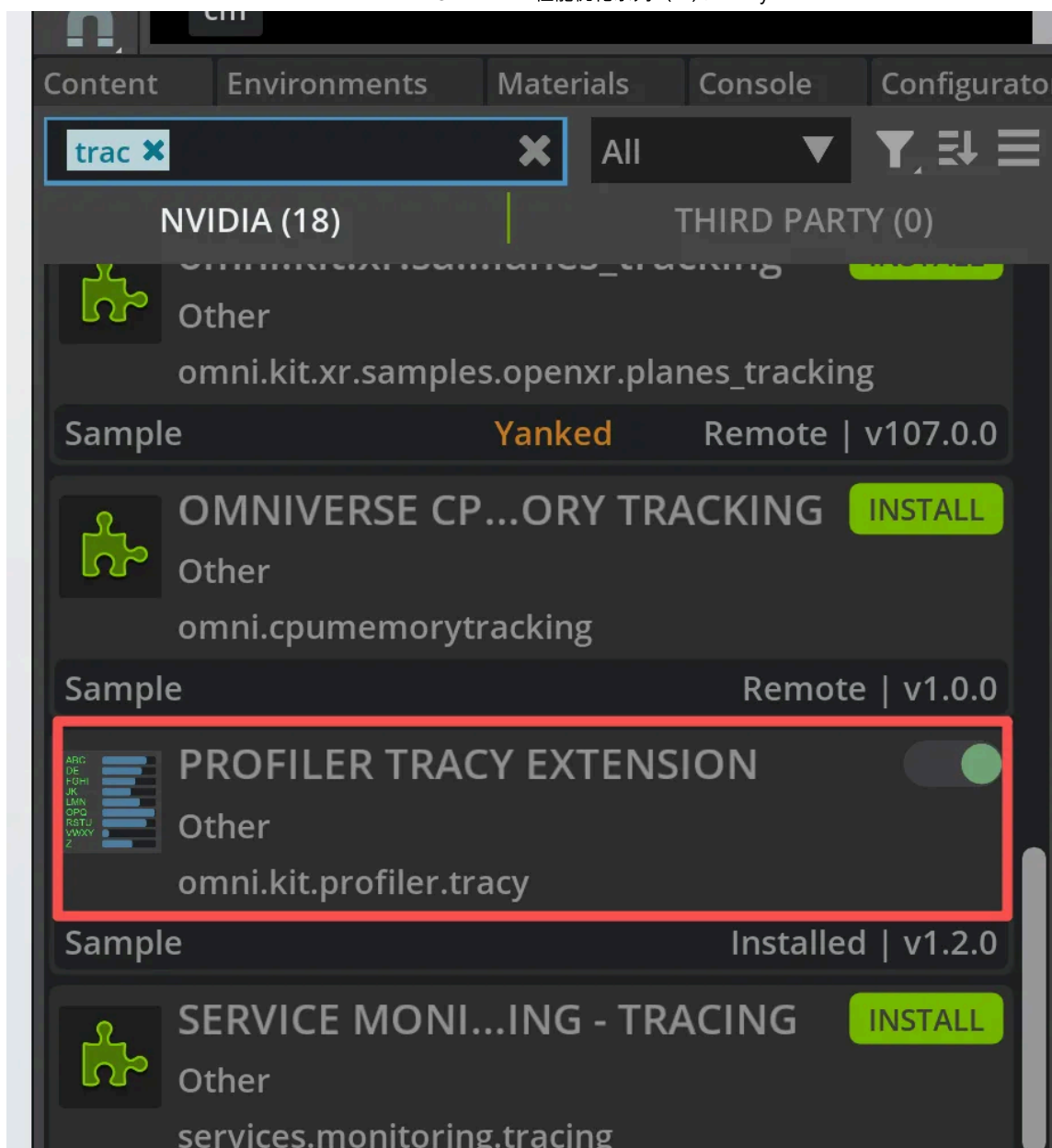
https://www.bilibili.com/video/BV1or421J7Du/?spm_id_from=333.337.search-card.all.click

文档参考：

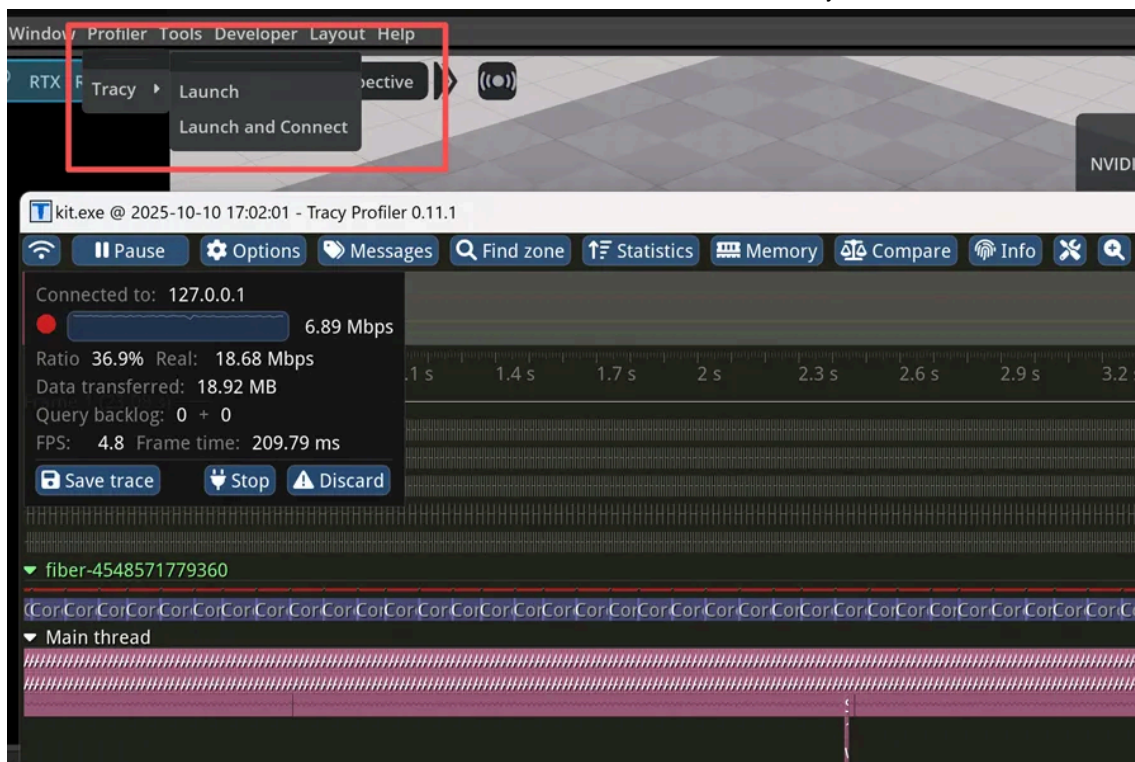
https://github.com/CppCon/CppCon2023/blob/main/Presentations/Tracy_Profiler_2024.pdf

2.2 安装

首先打开菜单 Developer -- Extension 搜索，找到 Profiler Tracy 并且安装。



然后会出现一个新的 Profiler 菜单，点击 Profiler -- Tracy -- Launch and Connect。

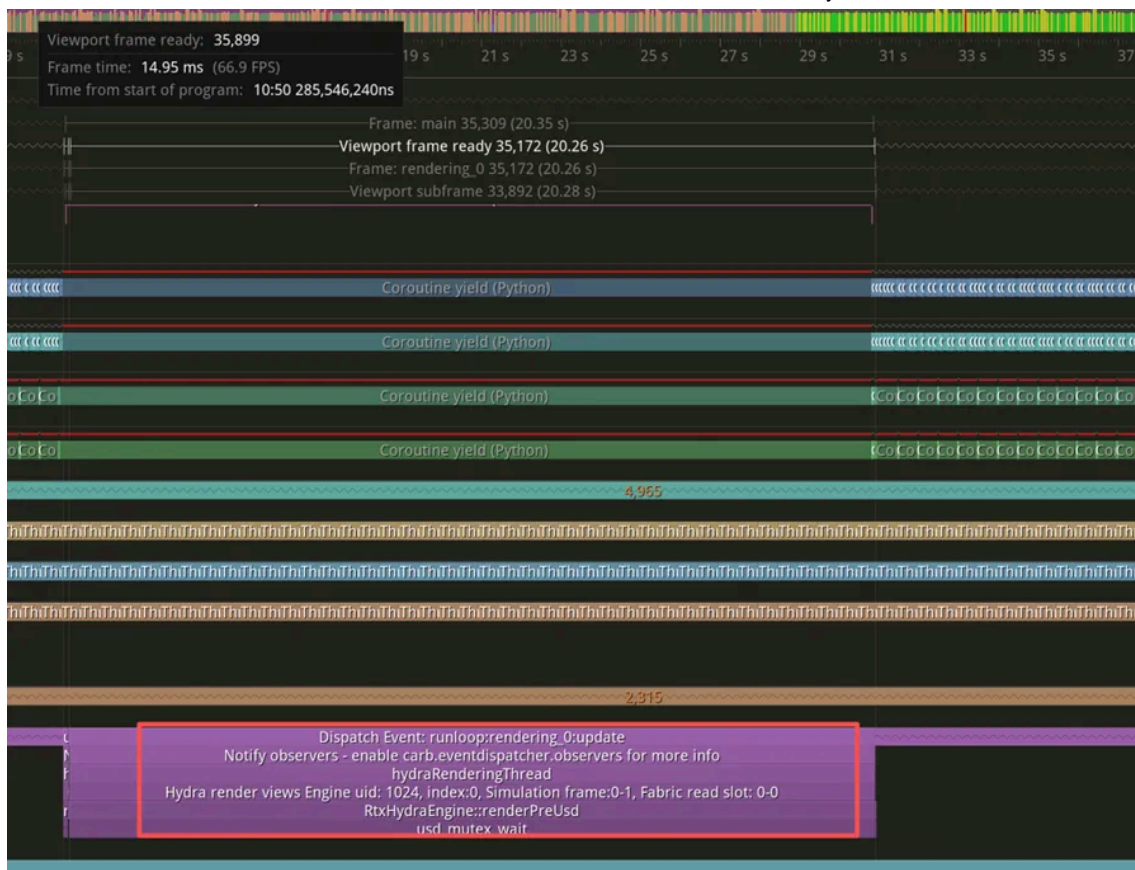


Tracy 基本使用操作：

1. Pause：在实时监测到发生性能瓶颈的事件以后要暂停，否则时间轴会一直向右走
2. 按住鼠标右键可以拖动到你想要的位置
3. 鼠标滚轮：Zoom in / out

2.3 分析问题

运行上述代码，点击 Tracy 中的 Pause 暂停（不暂停 Tracy 会一直记录的一直滚动）。之后按住 Ctrl 和鼠标中间的滚轮，Zoom 缩小操作，可以很容易找到一个最大的耗时，从 11 秒开始到 31 秒，这一个 Frame Render 用了二十几秒（注意：函数的调用堆栈已经正确显示），可以发现卡在了 RenderThread 中的 `usd_mutex_wait` 函数上面：



这样通过 Tracy 的使用就明白了问题卡住的大致原因，简而言之，渲染线程会等待 USD 写入的结束，一直卡在 `usd_mutex_wait`。

2.4 解决问题

分析：该问题的本质其实是 USD 的写入与修改会非常的慢，这是 USD 的基础架构造成的。

单单针对这个问题解决的方法不复杂，可以思考一下，写入的 Sublayer 其实并不需要实时参与 USD Composite，因为我们并不需要实时观察到合成结果，可以创建离线的 Sublayer，等待写入结束以后再自动或者手动把 Sublayer 加入进来，代码如下，看到并没有卡顿这一个过程，那么问题就解决了。

针对此次问题的 Solution 如下：


```

1 import omni.kit.app
2 import time
3 import asyncio
4 from pxr import Sdf, Usd, UsdGeom, Gf
5 from omni.kit.usd.layers import LayerUtils, get_layers, LayerEditMode
6 import omni.kit.commands
7 from collections import deque
8 import os
9 from typing import List, Tuple
10 from omni.kit.widget.layers.path_utils import PathUtils
11 import carb
12 from typing import List, Tuple
13 from enum import Enum
14 import asyncio
15 from pxr import Sdf, Usd, UsdGeom
16 from pxr import Usd, UsdGeom, Gf
17 from omni.kit.async_engine import run_coroutine
18 from concurrent.futures import ALL_COMPLETED, ThreadPoolExecutor, wait
19
20 def create_prim(stage, prim_path="/World/Camera"):
21     prim = stage.GetPrimAtPath(prim_path)
22     if prim and prim.GetTypeName() == "Camera":
23         carb.log_info(f"Camera already exists at: {prim_path}")
24     return prim

```

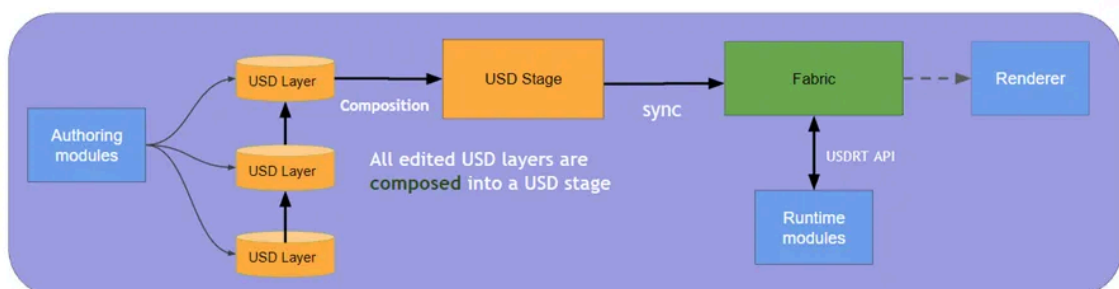
* 附代码链接: https://github.com/slayersong/OVPerf_Tracy/blob/main/solution_%20tracy_profiler.py

但是在一些项目当中一定要实时观察到结果。比如，有很多的数字孪生的工业场景中会存在小车传送带，各种物品都是实时进入到场景管线当中，这其中必定要参与 USD 合成。

所以这里介绍一个 Omniverse 对 USD 进行重构的基本概念 Fabric，USDRT（USDRT 是 Fabric 的 API），NVIDIA 在 Omniverse 当中开发了 Fabric 组件专门处理 USD 实时更改缓慢的问题：

https://docs.omniverse.nvidia.com/kit/docs/usdr/latestdocs/usd_fabric_usdr.html

通过这个官方文档的图也验证了刚才的结论：Render 线程会等待 USD 的合成结果 Composed 后进行渲染。



结束：如果单解决这个问题其实并不复杂，但是其中需要用到很多的基础知识，包括 USD 的合成机制、多线程开发、遇到问题如何去利用工具定位分析等。后面我们将会对 USDRT 与 Fabric 进行更细致的讲解，包括代码的开发使用和 Omniverse 中其他性能工具的使用教程。也希望更多的朋友可以分享在 USD 开发过程当中的心得体会。

附录：

关于前面提到的 USD 的基本开发教程，包括 USD 合成机制，USD 基本动画 TimeSampler 等：

<https://www.nvidia.cn/training/online/?activetab=ctabs-5>

文案提供和技术支持：

宋毅明

NVIDIA Omniverse & OpenUSD 开发者关系经理

*与 NVIDIA 产品相关的图片或视频（完整或部分）的版权均归 NVIDIA Corporation 所有。

END

**Bilibili@丽台科技**

有料有温度的阿婆主

**知乎@丽台科技**

专业又正经的知乎者也

**微博@丽台科技**

搜罗新鲜事的广播站

**视频号@丽台科技**

行业精彩时刻分享者

**抖音@丽台科技**

独具匠心的视频达人

**今日头条@丽台科技**

有价值的信息创造者

**丽台科技公众号**

应有所有的资讯首发站

**丽台科技用户社群**

行业内专家&用户聚集地

丽台（上海）信息科技有限公司

官网：www.leadtek.com.cn邮箱：service@leadtek.com.cn

咨询热线：400 138 8886

服务时间：周一至周五 9:00-17:00（法定节假日除外）

