

Exploring the Metaverse

Unveiling a New Dimension of Virtual Experiences and Applications

Project submitted to the
SRM University – AP, Andhra Pradesh
for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology

In

School of Engineering and Sciences

Submitted by

Candidate Name

Registration Number

P. PRUDHVI KRISHNA
P.N.B.S. ESWAR

AP20110020021
AP20110010011



Under the Guidance of

(Prof. Dr. Udaya Shankar)

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

[Dec, 2023]

Acknowledgement

We would like to express my special thanks of gratitude to my Professor, Dr. Udaya Shankar sir, and Mentor, Dr. Vishnu Ram Sir, who gave us the golden opportunity to do this wonderful project on Metaverse and its applications, who has also helped us in completing our project. I came to know about so many new things, I am really thankful to them.

Our sincere thanks to Omacs company, for the support and resources provided to us for completing this project.

Table of Contents

- Acknowledgement
- Table of Contents
- Abstract
- Introduction
 - I. Background
 - II. Objective of the study
- Methodology
 - I. Study of different tools
 - II. Exploration of Nvidia Omniverse
 - III. Setting up an Omniverse Development Environment
 - IV. The platform for creating and operating metaverse applications.
- Applications

Abstract

The metaverse, a conceptual space where digital and physical realities converge, has emerged as a transformative paradigm in the realm of virtual experiences and applications. This study delves into the multifaceted dimensions of the metaverse, aiming to unravel its underlying technologies, implications, and diverse applications across various domains.

The research delves into the practical applications of the metaverse in different industries. It scrutinizes how businesses leverage the metaverse for virtual meetings, collaborative workspaces, and immersive customer experiences. Additionally, the study examines the role of the metaverse in healthcare, education, and training, exploring its potential to revolutionize remote learning, medical simulations, and professional development.

Introduction

The rapid advancement of digital technologies has ushered in a new era in the virtual landscape, giving rise to the concept of the metaverse. The metaverse represents a collective virtual shared space that is created by the convergence of physical and virtual reality, providing users with immersive, interactive, and interconnected experiences. As we stand on the precipice of this digital frontier, it is imperative to understand the metaverse's genesis, its underlying technologies, and the myriad applications that have the potential to reshape the way we live, work, and interact.

Background:

The concept of the metaverse has evolved from science fiction to a tangible, albeit nascent, reality. Coined by Neal Stephenson in his 1992 science fiction novel "Snow Crash," the metaverse was initially envisioned as a virtual reality space where users could interact with each other and digital environments. Fast forward to the present, and the metaverse has transcended its fictional origins, propelled by advancements in technologies such as virtual reality (VR), augmented reality (AR), blockchain, and artificial intelligence (AI). The metaverse is not merely a speculative concept but an unfolding phenomenon with real-world applications that span across diverse industries.

Objectives of the Study:

This study aims to comprehensively explore the metaverse and its applications, addressing key questions surrounding its development, technologies, and potential impact. The primary objectives include:

- 1. Unraveling the technological foundations of the metaverse:** Understanding the building blocks, such as VR, AR, blockchain, and AI, that enable the creation of immersive virtual spaces.
- 2. Examining the societal implications:** Investigating how the metaverse may influence communication, commerce, education, and entertainment on a global scale.
- 3. Exploring practical applications:** Delving into how businesses and industries leverage the metaverse for collaborative work, customer experiences, healthcare, education, and training.
- 4. Analyzing ethical considerations and challenges:** Scrutinizing the potential risks and benefits associated with the metaverse, including issues of privacy, security, and digital inequality.

Methodology

The metaverse concept:

Metaverse also called “WEB 3.0”

- i. Web 1.0 → WWW (World Wide Web)
- ii. Web 2.0 → Social Media

Metaverse depends on several essential technologies: **Virtual Reality** (VR) and **Augmented Reality** (AR) that facilitate the entrance to the three-dimensional online environment through dedicated headsets and other devices connected to computers or games consoles. Artificial Intelligence helps create a virtual version of each user, called an ‘**avatar**’, who is the main player inside the metaverse, and is also used for seamless communications along with the Internet of Things technology.

Data protection and privacy concerns in the metaverse:

- Metaverse will bring new dimensions to the data protection and privacy scene.
- Regulations of data protection have been so far tackling physical data about users, and its movement between countries.
- Metaverse world will create totally new actors (avatars) in addition to the original users with massive amounts of data generated from new sources such as the data collected from facial and eye expressions, moving between different metaverses.

It carries many complications, concerns and policy issues to consider in terms of privacy and data protection.

Study of different tools and platforms in Metaverse:

1. Oculus
2. Unity
3. Nvidia Omniverse

➤ **Oculus:** Oculus is a brand of virtual reality (VR) hardware and software products developed by Oculus VR, a subsidiary of Meta Platforms (formerly known as Facebook). The company's primary focus is on creating VR headsets that allow users to experience immersive virtual environments and interact with digital content as if they were physically present within the virtual world.

Example: Reaching out your hand in VR to pick up a virtual object or ducking to avoid an incoming virtual projectile. Oculus is a brand that specializes in creating VR hardware and software products, primarily VR headsets, that enable users to immerse themselves in virtual environments and interact with digital content in a highly immersive and interactive way.

Oculus role in Metaverse

In the context of the "**metaverse**," which is an interconnected virtual reality space where users can interact, socialize, and engage in various digital experiences, **Oculus** plays a crucial role. It contributes to the metaverse by offering a platform for users to access and participate in a wide range of virtual experiences, social interactions, entertainment, and productivity applications.

➤ **Unity:** Unity is a powerful and widely used real-time 3D development platform, it is also used for various other interactive 3D applications, simulations, **virtual reality (VR)**, and **augmented reality (AR)** experiences.

1. **Cross-Platform Support**
2. **Real-Time 3D Rendering**
3. **Physics Simulation**
4. **VR and AR Support**
5. **Extensibility**

How is unity helpful in metaverse?

Unity plays a crucial role in the development and realization of the **metaverse** by providing a powerful and versatile platform for creating immersive 3D experiences, simulations, and interactive applications. It offers various features and tools that enable developers to build virtual worlds, games, social spaces, and other content that can be integrated into the metaverse.

1. **Real-Time 3D Rendering:** Unity's real-time rendering engine allows developers to create visually stunning and interactive 3D environments.
2. **Content Creation and Customization:** Unity provides developers with robust tools for content creation, including modeling, animation, physics simulation, audio, and more.
3. **Multiplayer Networking:** This is essential for enabling social interactions within the metaverse, where users can collaborate, play together, and communicate in real-time.
4. **Virtual Reality (VR) and Augmented Reality (AR) Support:** Unity natively supports VR and AR development, making it a versatile choice for creating immersive experiences in the metaverse. It allows developers to build applications and games for VR headsets and AR glasses, contributing to the spatial computing aspect of the metaverse.
5. **Integration of External Technologies:** Unity can be integrated with various external technologies, APIs, and SDKs, allowing developers to incorporate **AI**, **machine learning**, **blockchain**, and other cutting-edge technologies to enhance the functionality and interactivity of the metaverse.
6. **Extensibility and Third-Party Plugins:** Developers can create and integrate their custom tools, plugins, and extensions to tailor Unity to their specific needs and enhance the metaverse experience.

NVIDIA OMNIVERSE

OMNIVERSE

Omniverse refers to **NVIDIA Omniverse**, a platform developed by NVIDIA, a technology company known for its graphics processing units (GPUs) and artificial intelligence (AI) technologies. NVIDIA Omniverse is an ambitious and cutting-edge platform designed to revolutionize the way content is created, simulated, and experienced in the fields of 3D content creation, virtual production, and collaborative workflows.

NVIDIA Omniverse and the metaverse are related in the sense that both concepts revolve around creating immersive, interconnected, and collaborative virtual experiences. However, it's essential to understand that **NVIDIA Omniverse and the metaverse are not the same thing.**

Omniverse is a platform for collaboration and simulation

- **OMNIVERSE FOR EVERYONE:**
 - i. Omniverse for Individuals:**
 1. It is free for individuals
 2. It supports public forums and training videos
 3. It has only nucleus work station
 4. Use of all connectors including beta
 - ii. Omniverse Enterprise**
 1. It has a license per creator
 2. Full enterprise support and training videos
 3. Use of all LTS connectors
 4. Nucleus enterprise [Private Cloud]
 5. Omniverse create [License per creator]
 6. Omniverse View [Unlimited]

Relationship b/w Omniverse and Metaverse

1. **Shared Virtual Spaces:** Both NVIDIA Omniverse and the metaverse involve the creation of shared virtual spaces. The metaverse is a broader concept that envisions a collective virtual shared space where users can interact, socialize, work, and engage in various digital experiences. NVIDIA Omniverse, on the other hand, is a specific platform developed by NVIDIA, designed to enable real-time collaboration and content creation in 3D environments.
2. **Interoperability and Collaboration:** Both concepts emphasize interoperability and collaboration. In the metaverse, users should be able to move seamlessly between different virtual worlds and experiences. Similarly, NVIDIA Omniverse facilitates real-time collaboration between multiple artists and creators, enabling them to work together on the same 3D scene simultaneously.
3. **Content Creation and Simulation:** Both NVIDIA Omniverse and the metaverse involve content creation and simulation. In the metaverse, users can create and share virtual experiences, objects, and content. NVIDIA Omniverse provides tools for real-time 3D content creation and physically accurate simulations, catering to industries like film, gaming, and architecture.
4. **Immersive Experiences:** Both concepts aim to deliver immersive experiences. The metaverse seeks to provide users with a sense of presence and immersion in virtual environments. NVIDIA Omniverse utilizes real-time ray tracing, AI, and other advanced technologies to create visually stunning and realistic 3D environments, enhancing the sense of immersion for users.

NVIDIA Omniverse is a powerful platform for real-time collaboration and **content creation in 3D environments**, it is just one of the many technologies and platforms that contribute to the realization of the broader metaverse concept.

Relationship b/w NVIDIA and METAVERSE

NVIDIA is not a platform or creator of the metaverse itself, its advanced graphics processing units (GPUs) and related technologies are crucial components that contribute to the creation and realization of the metaverse concept.

Omniverse has five main components:

- i. **NUCLEUS:** Nucleus is how we manage the data between all of these applications. It manages the delta of change between the different users.
- ii. **CONNECT:** Connect is how we connect to the third-party products. So, as we build more and more connectors, more and more third-party tools become live on the Omniverse platform.
- iii. **KIT:** Kit is a toolkit that's on the platform that allows developers to build their own custom applications, or for advanced users to enhance the platform and customize it for their own workflows.
- iv. **SIMULATION:** Simulation tools are used and famous for real time simulating fire, water, smoke, etc.
- v. **RTX RENDER:** It was so advanced; it applies multi-GPU technology using RTX for real time photorealistic rendering.

OMNIVERSE HAS CORE APPLICATIONS

- i. **Create:**
 - 1. Create is an application built on **kit**
 - 2. Create allows you to do things like create 3D objects, animate those objects, add lights, and shadows.
- ii. **View:**
 - 1. It allows you to view that world, and view it in real time.

3D Workflows are essential for every Industry

- i. Architecture, Construction
- ii. Media, Game Development
- iii. Product design and Manufacturing
- iv. Scientific Visualization
- v. Robotics
- vi. Autonomous Vehicles

There will be many challenges over these likes:

- i. Too many vendors, too many tools
- ii. Complexity, Compute Power

Omniverse is all about solving some of these challenges.

Evolution of Omniverse

- i. GTC 2017 Holodeck
- ii. SIGGRAPH 2018 RTX
- iii. GTC 2019 Omniverse
- iv. **GTC 2020 Omniverse Open Beta**

They have made Omniverse based on USD

USD: Universal Scene Description

- This is developed by **Pixar**
- Foundation for NVIDIA Omniverse
- Open-Source API and the file framework for complex scene graphs.

How 3D Simulation Assets Are Used

i. Robotics

1. These require thousands of assets to build a photorealistic and physically-accurate environment

2. Assets will accelerate the development, testing, and management of AI-based robots.

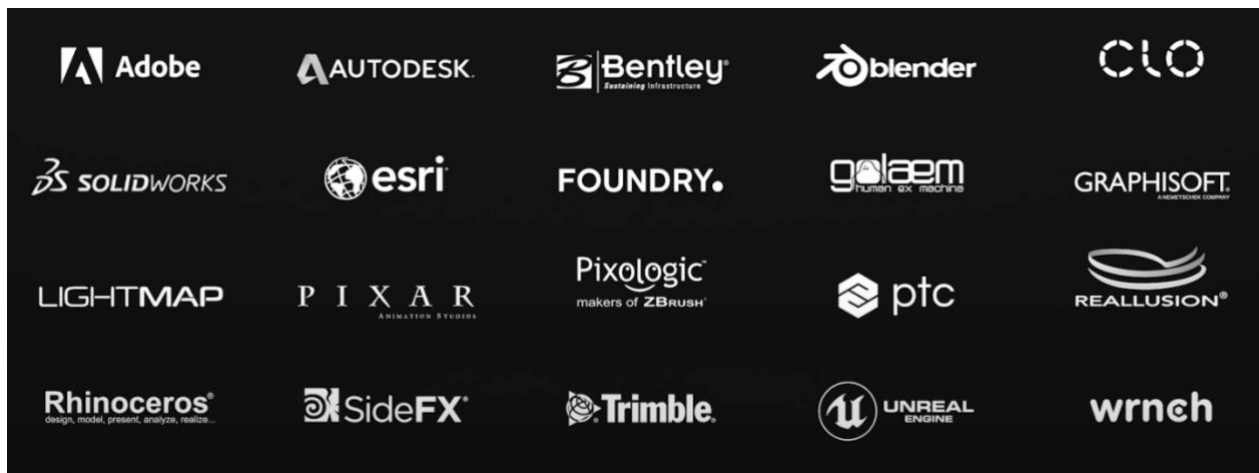
ii. Digital Twin Warehouses

1. 3D assets like forklifts, railings, racks, and robots can be used for simulations to optimize warehouse design and flow.
2. Train intelligent robot assistants, and improve productivity.

iii. Autonomous Driving

1. Immediately identify SimReady objects in a scene with semantic labels and physical materials to accelerate the sensor development for autonomous vehicles.

Companies involved in these are creating connectors to connect their products to the omniverse platform.



Setting up an Omniverse Development Environment

Step – 1:

1. [Download Visual Studio Code](#).
2. Perform the standard installation.

Step – 2: Tutorial

Developing on Nvidia Omniverse - How to Build an App

What Is an App?

An Omniverse Kit app is a *.kit file*. It is a single file extension. The Kit executable (e.g., Kit.exe) is the engine that runs the app based on the configuration defined in the *Kit file*.

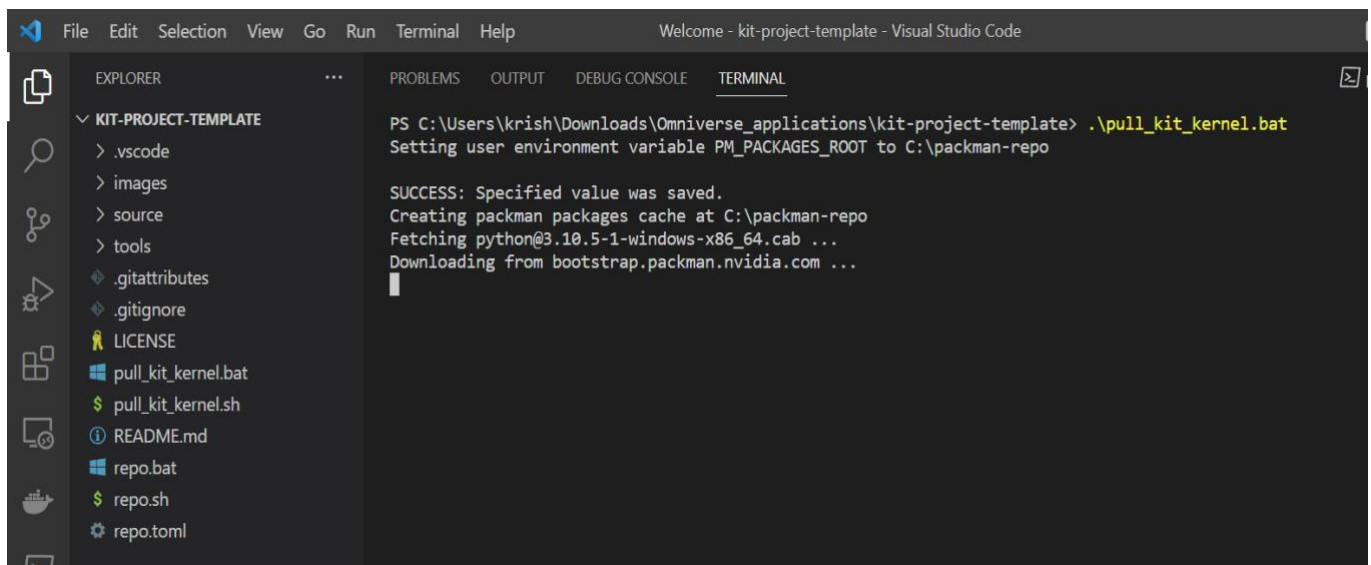
Step 1: Create an Omniverse App

Step 1.1: Clone the Repo

- i. **Open** VSCode
- ii. **Open** the command palette using Ctrl + Shift + P
- iii. In the palette prompt **enter** gitcl then **select** Git: Clone command
- iv. **Paste** <https://github.com/NVIDIA-Omniverse/kit-project-template> into the repository URL then **select** Clone from URL
- v. **Select (or create)** the local directory into which you want to clone the project
- vi. Once it has finished cloning it will ask if you want to open the cloned repository, select **Open**

Step 1.2: Pulling Kit Code

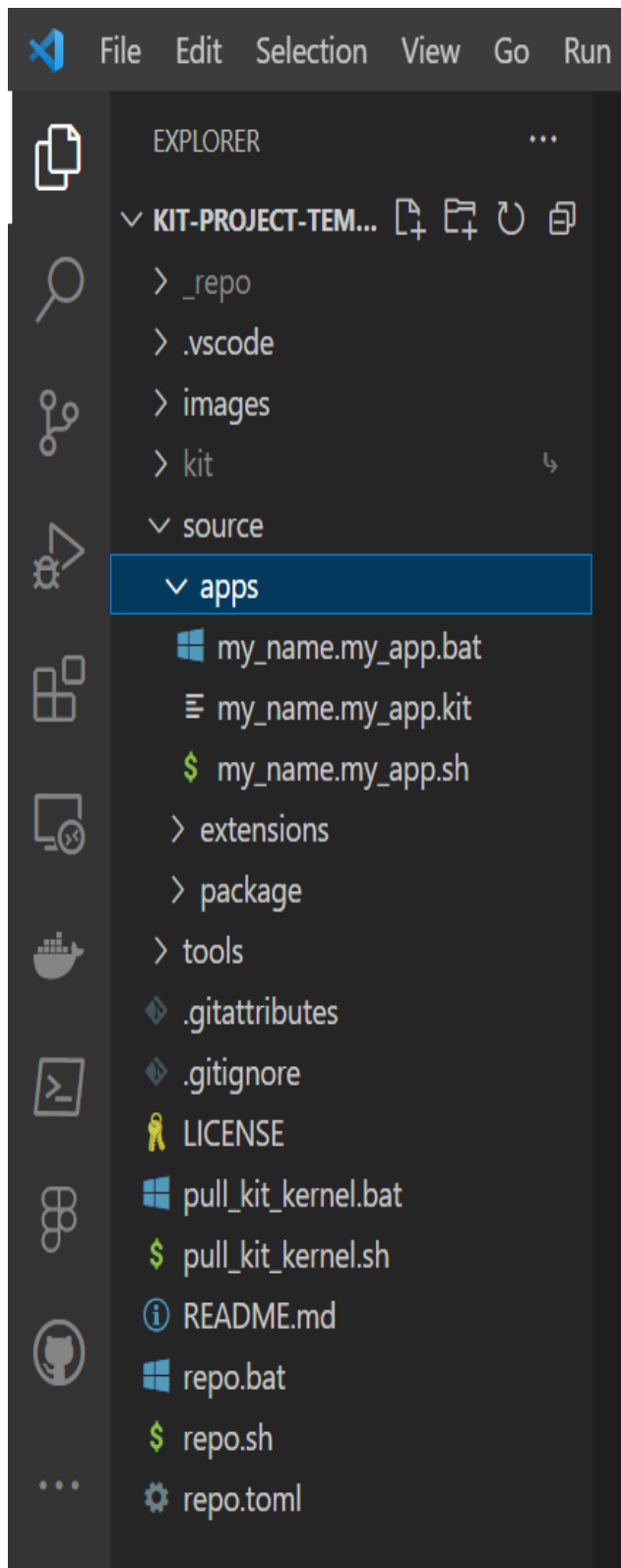
- i. **Open** a new terminal in *VSCode*
- ii. **Type** `.\pull_kit_kernel.bat` (windows) or `.\pull_kit_kernel.sh` (linux)
- iii. **Hit enter** to run



The screenshot shows the Visual Studio Code interface with the Explorer, Search, Source Control, and Run and Debug views on the left. The Explorer view shows the file structure of the 'KIT-PROJECT-TEMPLATE' project, including files like .vscode, images, source, tools, .gitattributes, .gitignore, LICENSE, pull_kit_kernel.bat, pull_kit_kernel.sh, README.md, repo.bat, repo.sh, and repo.toml. The Terminal view on the right shows the output of the command `.\pull_kit_kernel.bat` executed in a PowerShell prompt. The output indicates that the user environment variable `PM_PACKAGES_ROOT` is being set to `C:\packman-repo`, and that the packman packages cache is being created at `C:\packman-repo`. It also shows the process of fetching python@3.10.5-1-windows-x86_64.cab and downloading from bootstrap.packman.nvidia.com.

```
PS C:\Users\krish\Downloads\Omniverse_applications\kit-project-template> .\pull_kit_kernel.bat
Setting user environment variable PM_PACKAGES_ROOT to C:\packman-repo

SUCCESS: Specified value was saved.
Creating packman packages cache at C:\packman-repo
Fetching python@3.10.5-1-windows-x86_64.cab ...
Downloading from bootstrap.packman.nvidia.com ...
```



Overview of Files

1. **pull_kit_kernel.bat (Windows)**
A script that pulls the Omniverse Kit Kernel. By running this, a kit folder link will be created.
2. **repo.bat (Windows)**
Script file which runs the Repo Tools
3. **repo.toml**
Configuration file used by Repo Tools
4. **source/apps/my_name.my_app.kit**
A template *Kit file*. This will be our working file throughout the tutorial
5. **source/apps/my_name.my_app.bat** This script runs `kit.exe` and passes it **my_name.my_app.kit**
6. **source/extensions/**
Location of local extensions. Currently there is a template extension called hello world
7. **/kit**
Contains reference to Kit directory after running `pull_kit_kernel.bat/sh`
8. **/kit/apps**

Contains documented sample *Kit files* that can be used as a reference

Step 1.3: Running the App

- i. **Open** a new terminal if one isn't already open
- ii. **Type** `.\source\apps\my_name.my_app.bat` (windows) or `.\source\apps\my_name.my_app.bat` (linux)
- iii. **Hit** Enter

Step 1.4: Adding an Extension

- i. **Close** any running instance of your project, then head back to *VSCode*
- ii. **Navigate** to `source/apps/` and **Open** `my_name.my_app.kit`
This is your working *Kit file* that contains your configurations for your App
- iii. **Locate** the line `[dependencies]` and **add** a new line after it
Extensions will go under `[dependencies]` this tells the project which Extensions are required. Order in which it gets placed does not matter.
- iv. **Add** the following lines of code under `[dependencies]`

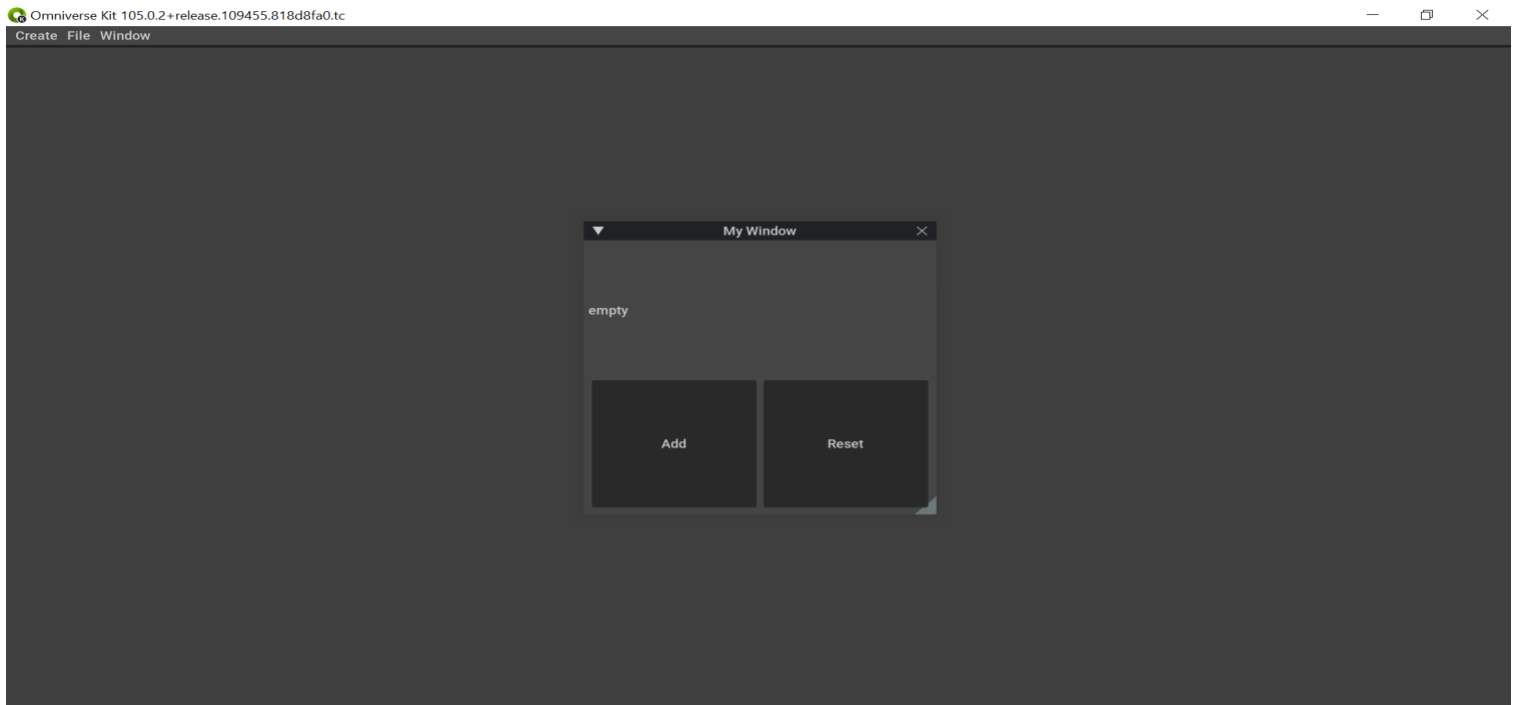
```
# Extensions window  
"omni.kit. window.extensions" = {}  
"omni.app.setup" = {}
```
- v. **Remove** the following lines:

```
# Create Kit UI Based applications  
"omni.app.editor.base" = {}
```
- vi. **Save** the file

Step 1.5: Finding Extensions

- i. **Run** the project
`.\source\apps\my_name.my_app.bat`
- ii. **Click on** Window > Extensions in the App
You can drag and dock the window
- iii. In the Extensions window, type "extensions" in the search bar

Note: i.e. `omni.kit.welcome.extensions`, `omni.kit.windows.extensions`, etc., These are the package names for each extension.



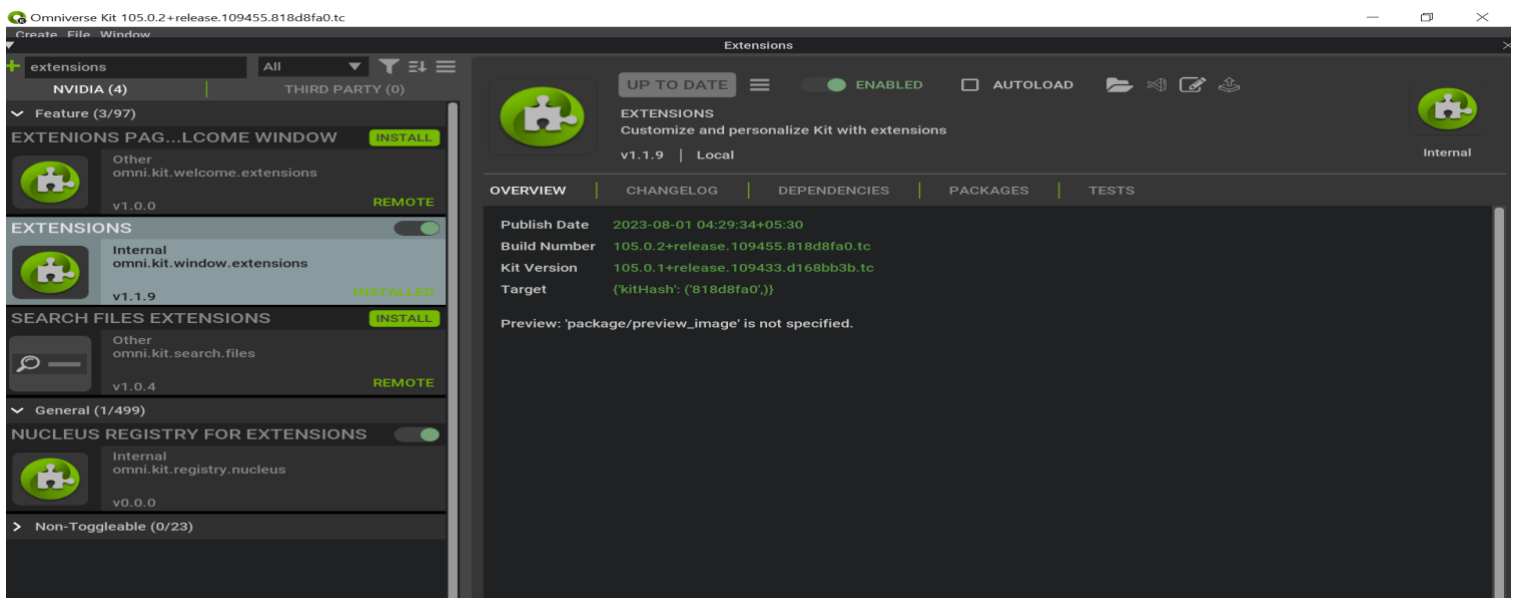
iv. **Select / Click** on *Extensions*

v. **Click** on *Dependencies*

vi. For a list view you can select the *Toggle View button* in the bottom right of the window.

For node view, you can zoom with the mouse wheel and use the middle mouse wheel to pan around.

vii. **Close** the project and head back to *VSCode*



Simply by adding in **omni.kit.window.extensions** we have also imported all the extensions listed in the dependency graph. It is good to note what the dependencies of each extension are. That way you can simplify your *Kit file* and not have to add every extension.

Step 1.6: Adding the Viewport

- i. Back in VSCode add the following lines in the [dependencies] section of **my_name.my_app.kit**:
Viewport Bundle
"omni.kit.viewport.bundle" = {}
Enable Pixar Storm for your viewport
"omni.hydra.pxr" = {}
"omni.hydra.pxr.settings" = {}
- ii. Next, you need to **enable** Pixar Storm through settings. To do this, **add** the following lines after [settings.app]:
content.emptyStageOnStart = true
exts."omni.kit.viewport.window".startup.windowName = "Viewport"
exts."omni.kit.renderer.core".compatibilityMode = true
here you can choose what renderers are enabled
renderer.enabled = "pxr"
renderer.active = "pxr"
- iii. Make sure you **save** your *Kit file*
- i. **Let's view the results of adding an extension:**
 - **Close** any instance of the project and **run** the project again. You should now see a Viewport window
 - **Move** the 'My Window' off to the side
 - **Go to** *Create > Mesh > Cone*
 - Notice how the manipulator is enabled. You can also navigate the camera in the viewport as well
 - Press the Right Mouse button to Orbit around
 - Use the Scroll Wheel or Press Right Mouse Button + "OPT"/"Alt" to zoom in
 - Press the Middle Mouse Button to Pan

With just a few extensions, you already have the capability to view, create and manipulate prims in the scene.- Prim is short for "**primitive**", a fundamental unit in Omniverse. Anything imported or created in a USD scene is a prim. This includes, cameras, sounds, lights, meshes, and more.

File Edit Selection View Go Run Terminal Helpmy_name.my_app.bat - kit-project-template - Visual Studio Code

EXPLORERKIT-PROJECT-TEMPLATE> .repo> .vscode> images> kit> source> appsmy_name.my_app.batmy_name.my_app.kit 1, M\$ my_name.my_app.shextensions\omni.hello.world> config> data> docs> omni\hello\world> __pycache__> tests> __init__.py> extension.py> package> .gitattributes> .gitignore> LICENSE> pull_kit_kernel.bat\$ pull_kit_kernel.sh> README.md> OUTLINE> TIMELINE

my_name.my_app.batXmy_name.my_app.kit (Index)

source > apps > my_name.my_app.bat1 @echo off2 setlocal3 call "%~dp0...\kit\kit.exe" "%~dp0my_name.my_app.kit" %*4

PROBLEMS 3OUTPUTDEBUG CONSOLETERMINAL

[406.894s] [ext: omni.kit.widget.versioning-1.4.2] startup[406.929s] [ext: omni.ui.scene-1.6.6] startup[406.978s] [ext: omni.kit.clipboard-1.0.2] startup[406.987s] [ext: omni.iray.libs-0.0.0] startup

-----Driver Version: 532.09 | Graphics API: D3D12-----GPU | Name | Active | LDA | GPU Memory | Vendor-ID | LUIDDevice-ID | UUID-----0 | NVIDIA GeForce MX230 | Yes: 0 | | 1982 MB | 10de1d11 | f3ae0100..0-----1 | Intel(R) UHD Graphics | | | 128 MB | 80869b41 | 9aaa0100..0-----OS: Windows 10 Home Single Language, Version: 10.0 (22H2), Build: 19045, Kernel: 10.0.19041.3324Processor: Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz | Cores: 4 | Logical: 8-----Total Memory (MB): 7942 | Free Memory: 1030Total Page/Swap (MB): 25350 | Free Page/Swap: 10015-----2023-08-23 18:16:50 [432,271ms] [Warning] [omni.gpu.foundation.factory.plugin] RT-capable GPU not found, switching to compatibility mode[437.286s] [ext: omni.kit.collaboration.telemetry-1.0.0] startup[437.293s] [ext: omni.kit.window.filepicker-2.7.44] startup[437.387s] [ext: omni.mdl.neuraylib-0.2.0] startup

Ln 4, Col 1 Spaces: 4 UTF-8 CRLF Batch

File Edit Selection View Go Run Terminal Helpmy_name.my_app.bat - kit-project-template - Visual Studio Code

EXPLORERKIT-PROJECT-TEMPLATE> .repo> .vscode> images> kit> source> appsmy_name.my_app.batmy_name.my_app.kit 1, M\$ my_name.my_app.shextensions\omni.hello.world> config> data> docs> omni\hello\world> __pycache__> tests> __init__.py> extension.py> package> .gitattributes> .gitignore> LICENSE> pull_kit_kernel.bat\$ pull_kit_kernel.sh> README.md> OUTLINE> TIMELINE

my_name.my_app.batXmy_name.my_app.kit (Index)

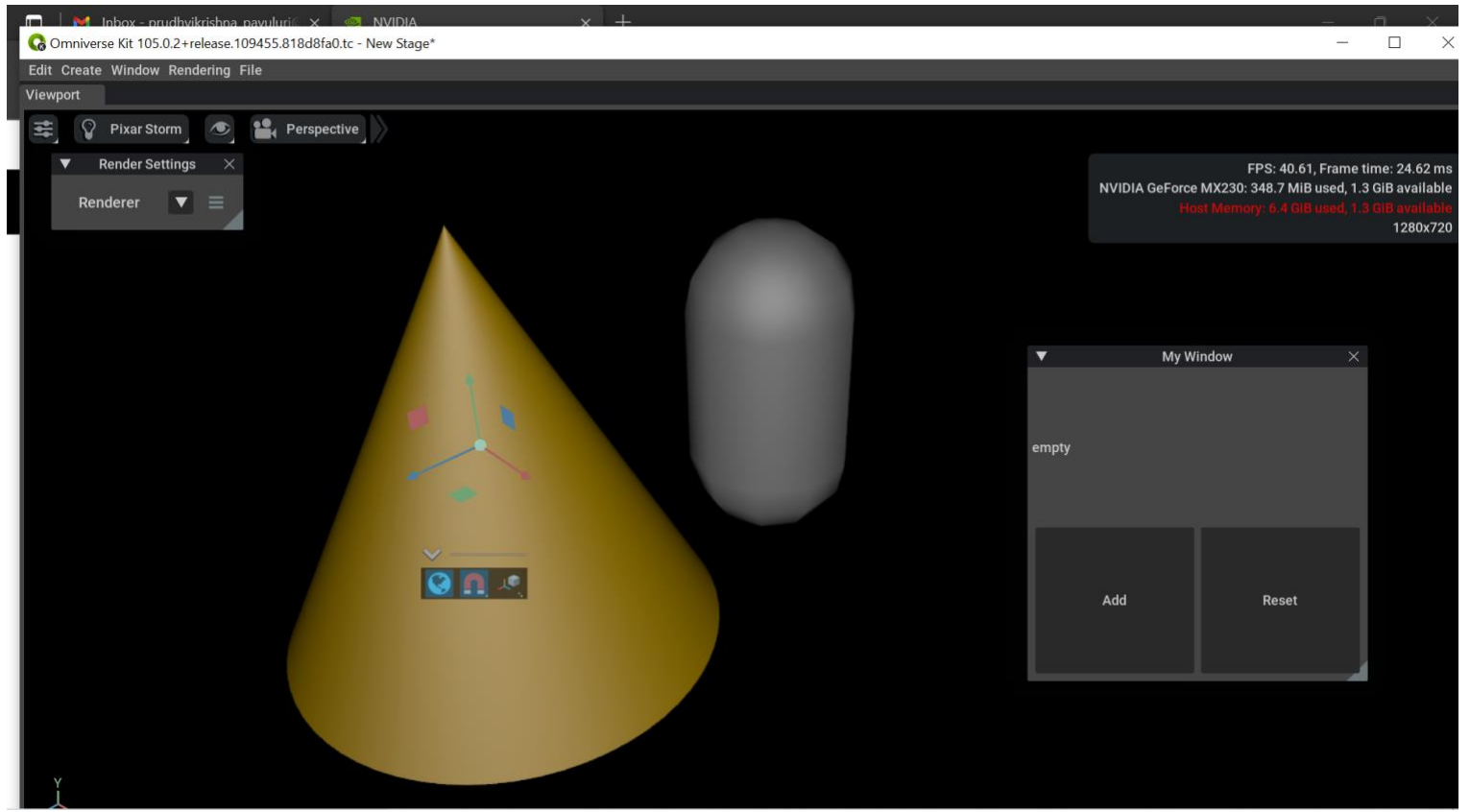
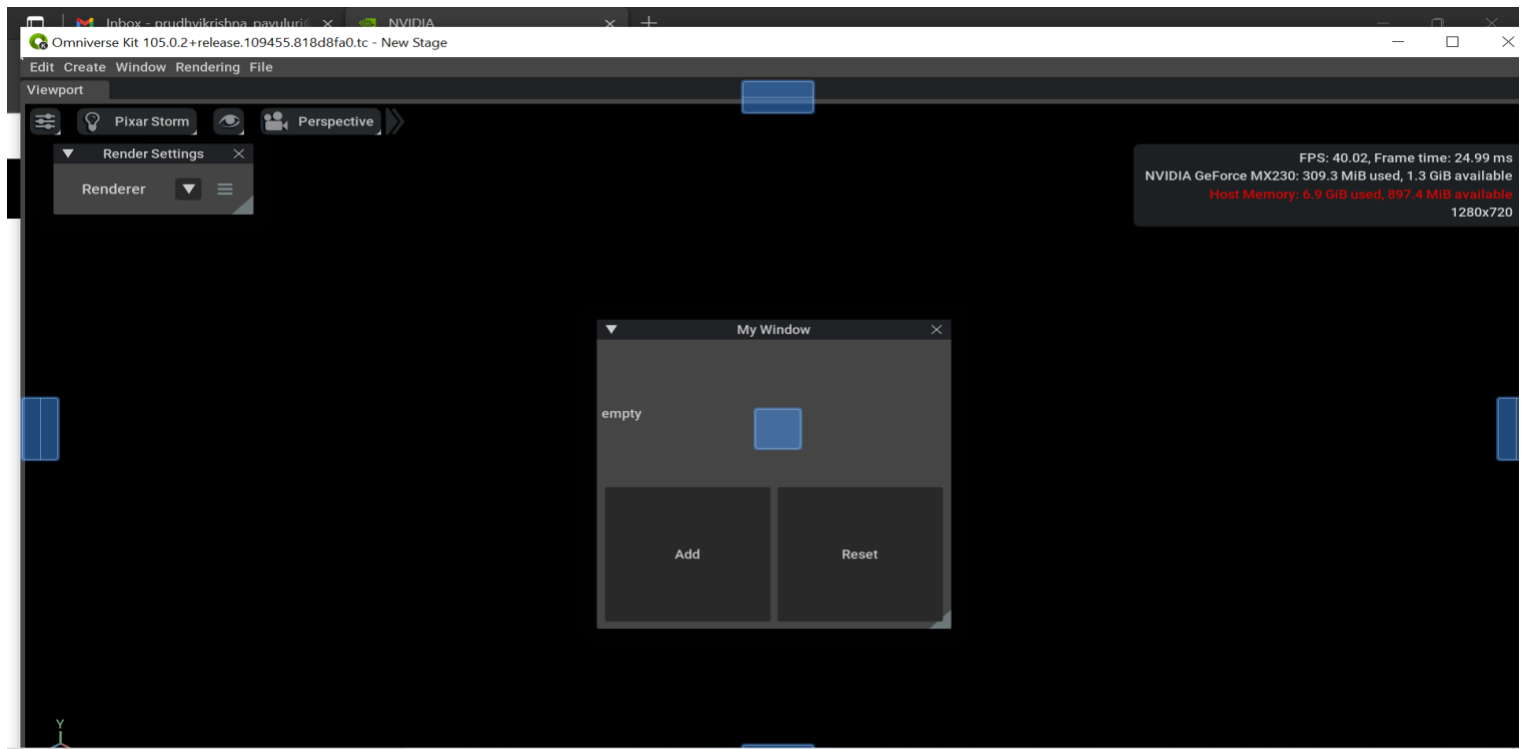
source > apps > my_name.my_app.bat1 @echo off2 setlocal3 call "%~dp0...\kit\kit.exe" "%~dp0my_name.my_app.kit" %*4

PROBLEMS 3OUTPUTDEBUG CONSOLETERMINAL

[406.894s] [ext: omni.kit.widget.versioning-1.4.2] startup[406.929s] [ext: omni.ui.scene-1.6.6] startup[406.978s] [ext: omni.kit.clipboard-1.0.2] startup[406.987s] [ext: omni.iray.libs-0.0.0] startup

-----Driver Version: 532.09 | Graphics API: D3D12-----GPU | Name | Active | LDA | GPU Memory | Vendor-ID | LUIDDevice-ID | UUID-----0 | NVIDIA GeForce MX230 | Yes: 0 | | 1982 MB | 10de1d11 | f3ae0100..0-----1 | Intel(R) UHD Graphics | | | 128 MB | 80869b41 | 9aaa0100..0-----OS: Windows 10 Home Single Language, Version: 10.0 (22H2), Build: 19045, Kernel: 10.0.19041.3324Processor: Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz | Cores: 4 | Logical: 8-----Total Memory (MB): 7942 | Free Memory: 1030Total Page/Swap (MB): 25350 | Free Page/Swap: 10015-----2023-08-23 18:16:50 [432,271ms] [Warning] [omni.gpu.foundation.factory.plugin] RT-capable GPU not found, switching to compatibility mode[437.286s] [ext: omni.kit.collaboration.telemetry-1.0.0] startup[437.293s] [ext: omni.kit.window.filepicker-2.7.44] startup[437.387s] [ext: omni.mdl.neuraylib-0.2.0] startup

Ln 4, Col 1 Spaces: 4 UTF-8 CRLF Batch

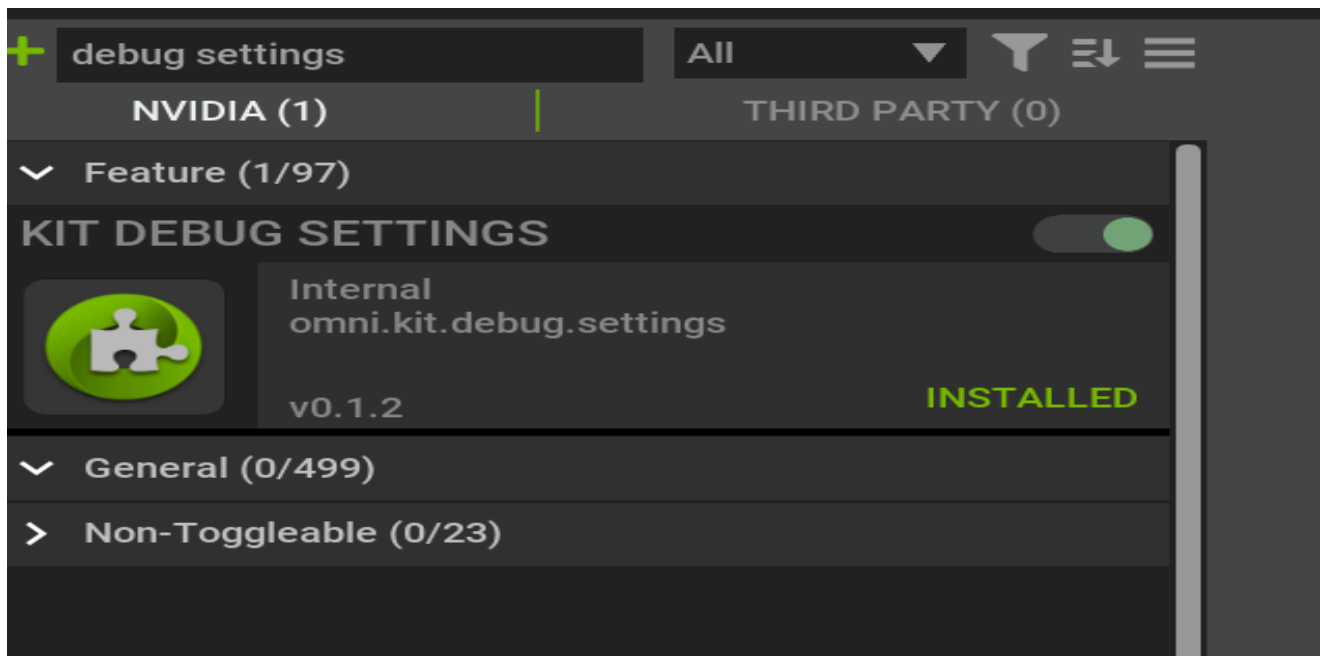


Viewport Settings:

- This is the explanation to the above code we added.
- i. **content.emptyStageOnStart = true**
This will create an empty stage during the startup process
- ii. **exts."omni.kit.viewport.window".startup.windowName = "Viewport"**
Setup Kit to create an 'omni.kit.viewport.window' Window named Viewport
- iii. **exts."omni.kit.renderer.core".compatibilityMode = true**
This forces Compatibility Mode on. Compatibility mode allows the user to run the project if they do not have RTX.
- iv. **renderer.enabled = "pxr"**
Renderers that the user can pick between
- v. **renderer.active = "pxr"**
Default renderer that is active upon launching

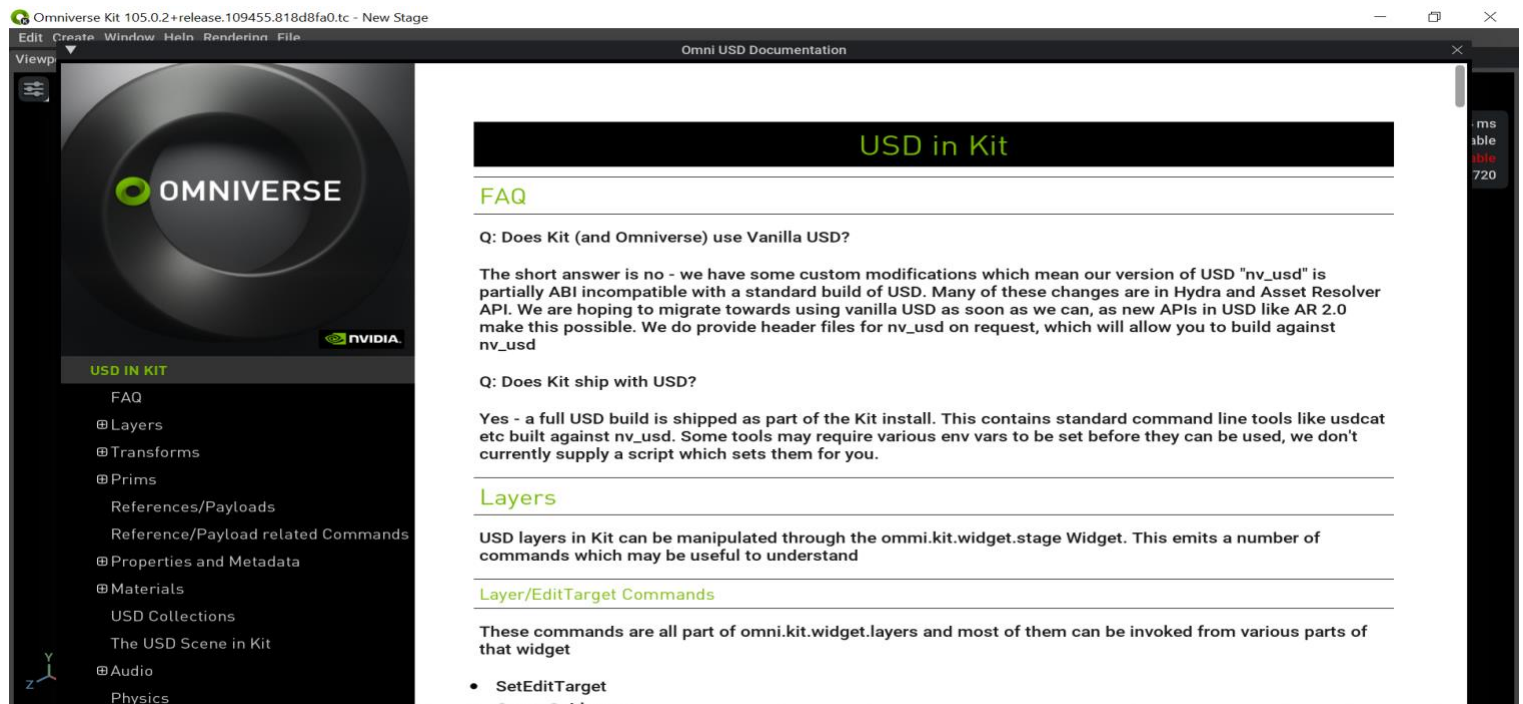
Additional Settings

1. **Run** the project
2. **Go to** Window > Extensions
3. **Search** for debug settings
4. **Install and Enable** "omni.kit.debug.settings"
5. Each setting can be hovered over so you can see the path.
6. For example, if we look at audio/enabled the path is /app/audio/enabled. To add this setting in the *Kit file* we add it to [settings.app].



Step 1.7: Finding an Extension Name

1. Inside the App **Open** the *Extension Window*.
 - Window > Extensions
 - Notice for each Extension box it includes the Title, and inside the box it has the category and a short name i.e. omni.graph.bundle.action, omni.kit.window.environment
2. In the *search bar* **type** `usd_doc`.
3. **Take note** the package name is `omni.kit.usd_docs`.
4. **Close** the App and head back to *VSCode* and **add** the following line in our [dependencies] section. Order does not matter as long as it's under [dependencies].
 - `"omni.kit.usd_docs" = {}`
5. **Save** the *Kit file* and **rerun** the project.
6. **Go to Help** in the *menu bar*, you should see *USD Docs* in the list.



Step 1.8: Hello World

1. Underneath `"omni.hydra.pxr.settings" = {}` add the following lines:

File Menu

```
"omni.kit.menu.file" = {}
```

Stage

```
"omni.kit.window.stage" = {}
```

Content Browser

```
"omni.kit.window.content_browser" = {}
```

Edit Menu

```
"omni.kit.menu.edit" = {}
```

ToolBar

```
"omni.kit.window.toolbar" = {}
```

Property Window

```
"omni.kit.property.bundle" = {}
```

Variant presenter

```
"omni.kit.variant.presenter" = {}
```

USD Docs

```
"omni.kit.usd_docs" = {}
```

USD Paths

```
"omni.kit.window.usd_paths" = {}
```

USD Collection window

```
"omni.kit.window.collection" = {}
```

2. **Save** the *Kit file* and **close** any instances of Omniverse.
3. **Run** the project, you should now see the extensions added.

Note: If you have more than the listed extensions in the previous steps, your project will look different from the image.

Extensions?

- **File Menu & Edit Menu:** File menu offers access to file operations both locally and on your Nucleus. Edit Menu contains editing tools like Undo, Redo, Select, along with other more advanced capabilities.
- **Stage:** Here you will be able to manage all the assets in your USD Scene, listed in a hierarchical (parent/child) order. This is essentially for large scenes.
- **Content Browser:** Items listed in the Content Browser provide meaningful information about the asset without having to open it.
- **Tool Bar:** Provides easy access to common commands needed when editing a USD Scene.
- **Property Window:** Provides you a way of configuring properties for the prim selected.
- **Variant Presenter:** Gives you convenient access to all the USD Variants in a scene. Can also be used to organize variants into custom groups using USD Collections.
- **USD Docs:** Interactive Documentation with coding examples for omni.kit.usd
- **USD Paths:** Easy tool for you to search and replace paths of prims in the scene.
- **USD Collection Window:** Collections are a way to group/organize prims and properties in USD

Step 2: Styling

Some of the methods in styling:

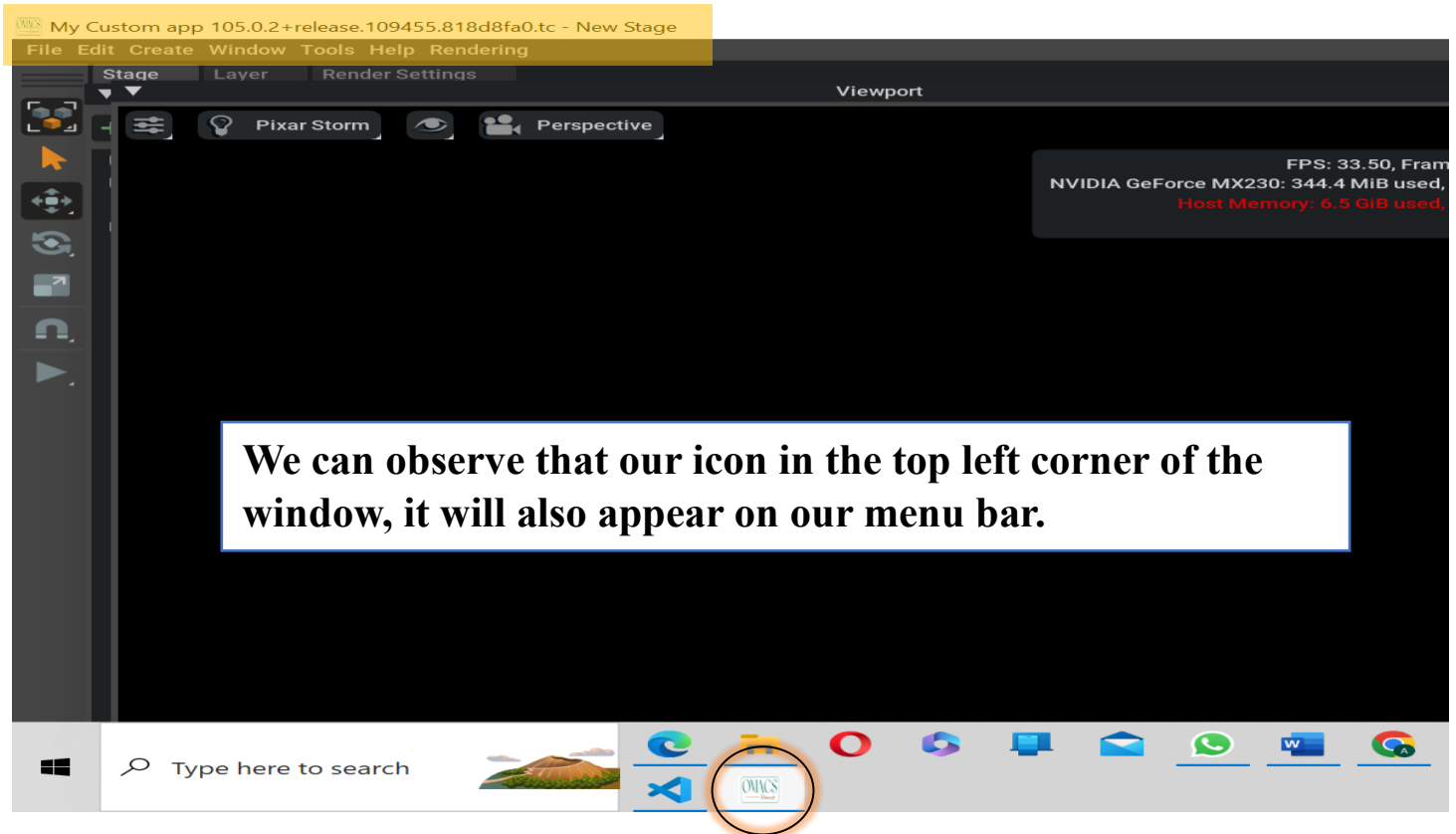
1. **Changing the Icon that appears in the top left corner of the app window**
2. **Customizing the Title Bar color**
3. **Adding Splash Screens**
4. **Changing Fonts**
5. **Shading Extension Colors**

Step 2.1: Change the Icon

1. In *VSCode*, **Create** a new folder in apps called **data**
2. Make sure your *.ico image* is stored in this folder
 - You need to have a *.ico image* with a size of 64x64 and resolution of 72.
3. In *VSCode*, **Add** [settings.app.window] section if you do not have one
4. **Add** the following line under [settings.app.window]:
 - `iconPath = "${app}/data/icon.ico"`

Note: `${app}` refers to `/source/apps` folder

5. **Save and run** the project. You should see your icon in the top left corner of the window, it will also appear on your menu bar.



Step 2.2: Customize the Title Bar

The Title Bar is the Application Window's main menu bar where you can minimize, maximize and close the window.

1. First, we need to **add** the extension that allows us to change the title bar settings. In the dependencies section **add** the following line:
 - o "omni.kit.window.modifier.titlebar" = {}
2. Next, **add** the following section:

```
[settings.exts."omni.kit.window.modifier.titlebar"]
titleFormatString = " My Custom App {verKey:/app/version,font_color=0x909090,font_size=16} {separator} {file,
board=true}"
icon.file = "${app}/data/icon.ico"
icon.size = 64
name = "Arial"
defaultFont.size = 16
defaultFont.color = 0xD0D0D0
separator.color = 0x00B976
separator.width = 1
windowBorder.color = 0x0F0F0F
windowBorder.width = 2
colors.caption = 0x0F0F0F
colors.client = 0x0F0F0F
```

3. **Save** the file and **Rerun** the project.

Customize the Title Bar Deep Dive

1. Title
 - o titleFormatString – Contains the title that will be displayed in the Title Bar
2. Icon
 - o icon.file – This is the Icon file that will appear. The extension will override what we provided previously in iconPath but only for the Title Bar.
 - o icon.size – Defines the Size of the Icon
3. Font
 - o defaultFont.name – Default font for the Title Bar
 - o defaultFont.size - Default Font Size
 - o defaultFont.color – Font Color
4. Separator
 - o separator.color – Color for the Separator
 - o separator.width – The Width of the separator

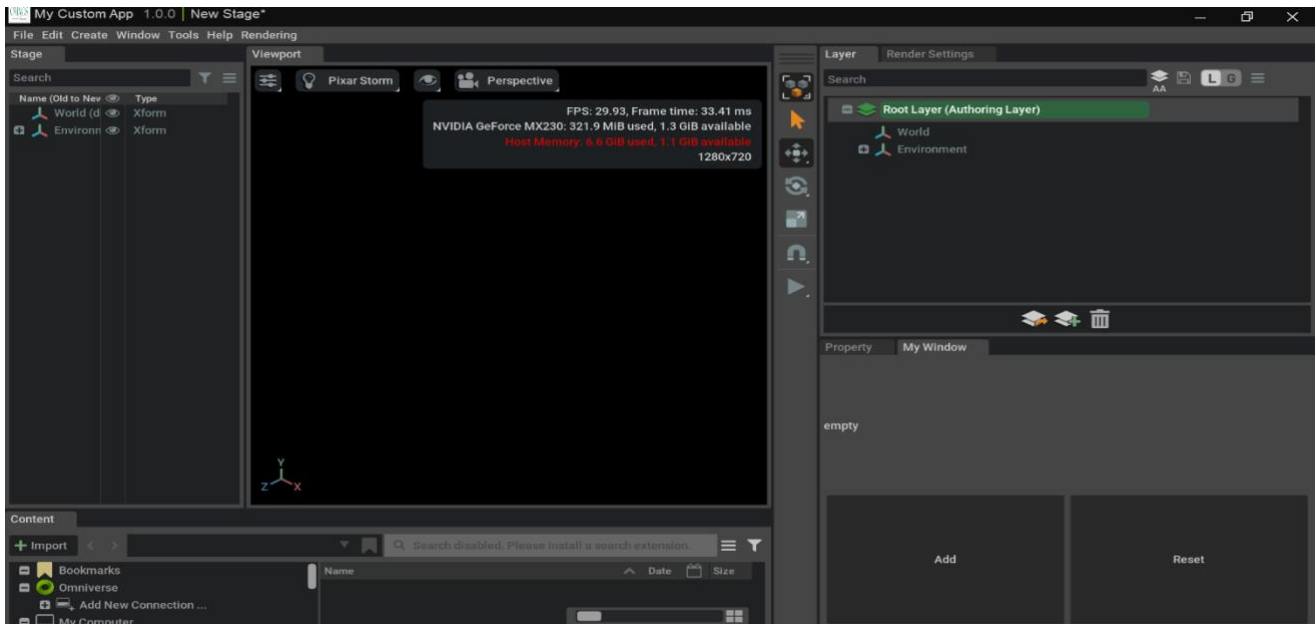
5. Border
 - `windowBorder.color` - Border Color of the window
 - `windowBorder.width` - Width of the window border
6. Window
 - `colors.caption` – The Color of the Title Bar
 - `colors.client` – The Color of the window outline

Step 3: Updating the Layout

Currently, your windows might not be in the appropriate default locations. You can dock the windows where you like them and then save a layout based on the configuration you've created.

Step 3.1: Configuring a Custom Layout

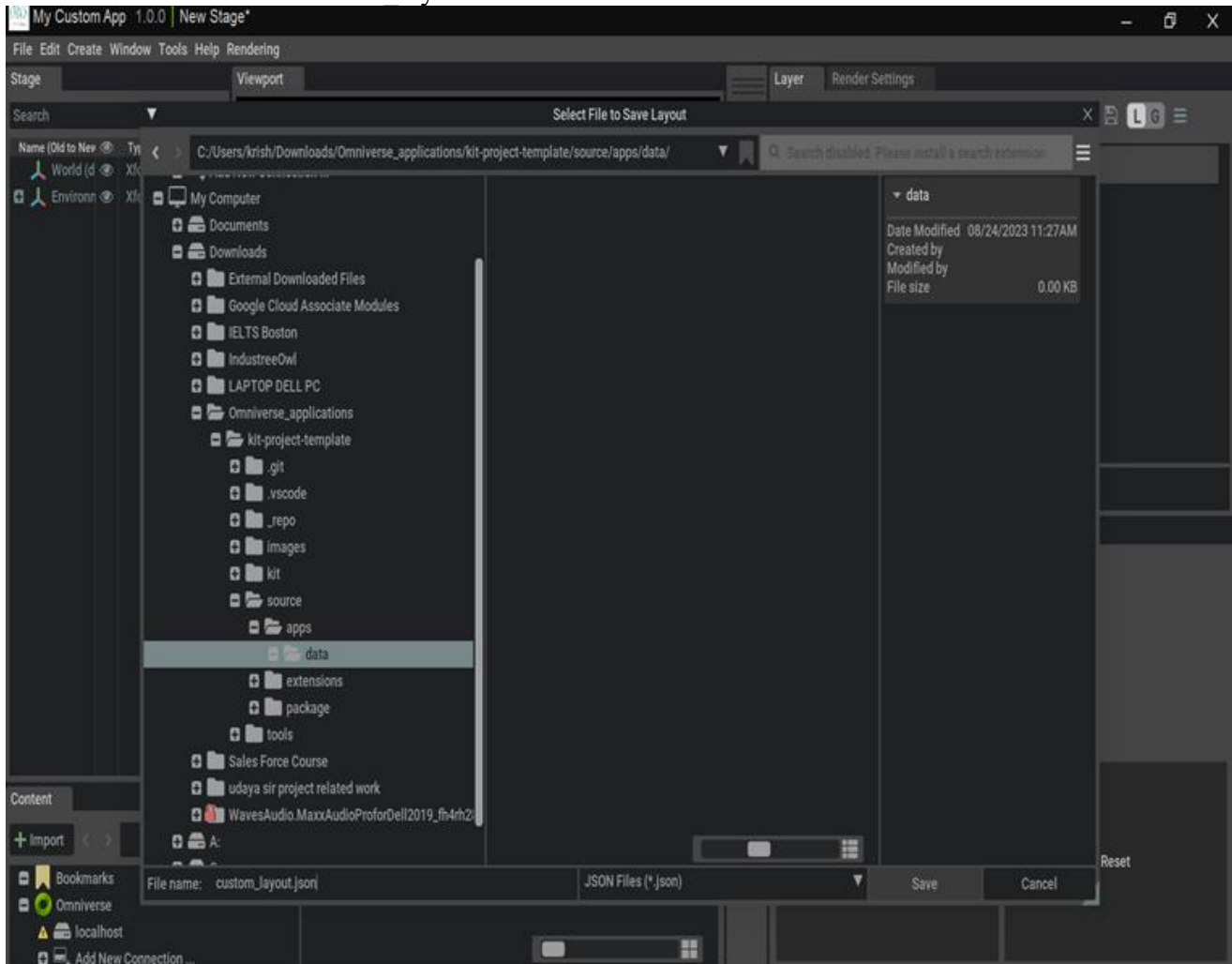
1. **Run** the project if it is not running already
2. **Dock** your windows in the locations that appeal to you.
To position the window Left-Click + Drag near the blue preview images.
3. **Repeat** for any other extension windows



Arrange the tabs accordingly

Step 3.2: Saving the Layout

1. After setting up the layout *go to the Menu Bar* click on Window > Layout > Save Layout....
2. A new prompt window will open. **Navigate to** your project folder. From your project folder go to source > apps > data.
3. Under filename name it custom layout then click Save.



Step 3.3: Updating Your Application's Layout Reference

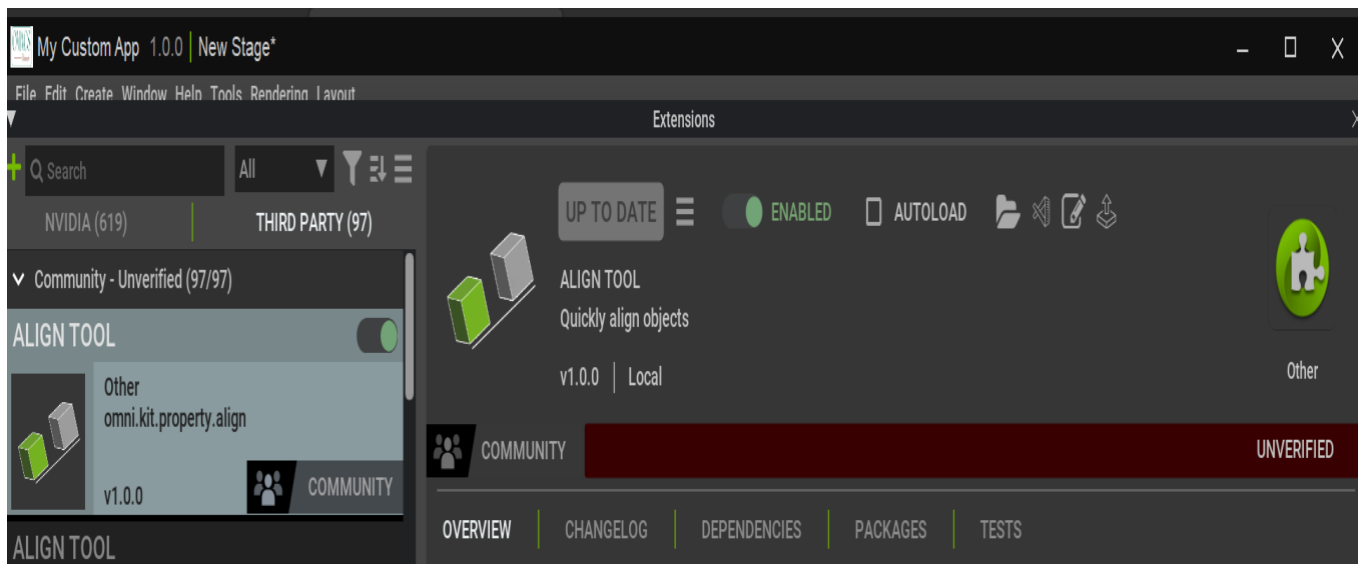
1. **Close** the application. **Go back** to *VSCode* and in your *Kit file*.
2. **Add** the following line to the [settings.app] section: This will use the newly created .json layout file.
 - o layout.default = "\${app}/data/custom_layout.json"
3. **Save** the *Kit file*.
4. **Run** the project. Now every time you open your App it will retain the same layout.

Step 4: Adding Third Party Extensions

So far, you have added Extensions that come with Omniverse. You can also add Third Party Extensions, these are extensions other community developers create and publish on GitHub publicly.

Step 4.1: Add Third Party Extensions

1. In our *Kit file* in *VSCode*, **scroll down** to the `[settings.exts."omni.kit.registry.nucleus"]` section.
2. **Add** the following inside the *registries dictionary*, between `{ name="kit/sdk....}` and `]`:
 - `{ name = "kit/community", url = "https://dw290v42wisod.cloudfront.net/exts/kit/community" }`
3. **Add a comma** at the end of `{ name="kit/sdk..."}`
4. **Save** the file and **run** the project.
5. **Open** the Extension Manager
 - Window > Extensions
6. **Click on Third-Party tab.** With the registry now in our *Kit file* you can now pull all available extensions that are live on Github.
7. Notice the first extension, *Align Tool* (omni.kit.property.align). You will be adding this as an example.
8. **Close** the App and **go back** to *VSCode*.
9. **Locate** the [dependencies] section and **add** the following line:
 - `"omni.kit.property.align" = {}`
10. By default, all *Untrusted Extensions* will **NOT** install. To override this, you must change the setting. *Use this wisely.* **Locate** [settings.app] section and **add** the following line:
 - `extensions.installUntrustedExtensions = true`
11. **Save** the file and **run** the project.
12. **Head back to** the *Extension Manager* and under the *Third-Party tab* you should see that the *Align Tool* is installed and enabled.



Step 4.2: Add Extensions Locally

1. **Take** any locally created extension and **copy ONLY** the extension. The project directory should not be included.
 - i.e. my.local.extension, omni.hello.world, omni.spawn.prims, etc.
2. In the project folder, **add/paste** your extension to source/extensions. Currently, there is one extension in the folder, omni.hello.world
3. In your *Kit file*, **add** your extension name. For example, omni.example.spawn_prims has been added to my extension folder so in my *Kit file* I will add the following line under [dependencies]:
 - "omni.example.spawn_prims" = {}
4. **Save** the file and **run** the project. You should now see your extension has been added to your App.

Step 5: Packaging and Distribution

Step 5.1: Change the App Name

Step 5.2: How to Package the App

1. To package an app, **go to VSCode** and **Open a Terminal**
2. **Type** the following in the *Terminal*:
 - .\tools\package.bat (windows) or .\tools\package.sh (linux)
 - OR
 - repo package
3. **Hit** Enter

The package will be created in the _build/packages folder

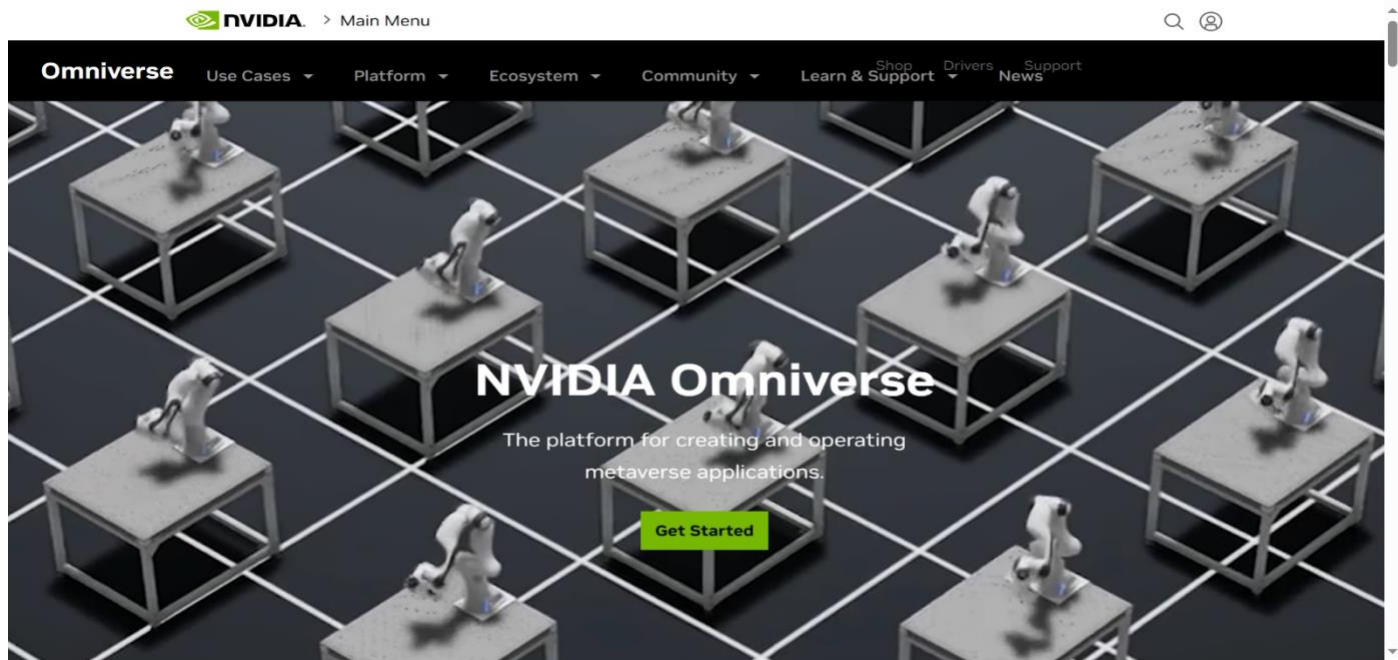
Step 5.3: Packaging and Distribution

1. After Downloading the Zipped folder, **Extract it**
2. **Run** pull_kit_kernel.bat (windows) or pull_kit_kernel.sh (linux) inside the package
3. **Navigate** and **run** source/apps/[app_file].bat (windows) or source/apps/[app_file].bat (windows)

NVIDIA Omniverse

The platform for creating and operating metaverse applications.

[Omniverse Platform for OpenUSD Development and Collaboration | NVIDIA](#)



Click on Get Started

Get Started with NVIDIA Omniverse

NVIDIA Omniverse Licensing Options			
Features	Standard	Enterprise	Cloud
	For Individuals, Deploy On Premises	For Teams, Deploy On Premises	For Teams, Cloud Platform- as-a-Service
	Download Free	Get in Touch	Get in Touch
Subscription Model	Named User	Named User	Unlimited Users

Register to Download

I am a (select all that apply):

☐ Creator ☒ Developer

First Name
PRUDHVI

Last Name
KRISHNA

Business Email Address
prudhvikrishna_pavuluri@srmap.edu.in

Organization / University Name
SRM University-AP

Industry
Academia / Education

Job Title
Student: Undergraduate

Location
India

Preferred Language
English (US)

Industry
Academia / Education

Job Title
Student: Undergraduate

Location
India

Preferred Language
English (US)

Send me the latest news, announcements, and more from NVIDIA about:

- ☒ Enterprise Business Solutions
- ☒ Developer Technology & Tools

(Optional). You can unsubscribe at any time.

[NVIDIA Privacy Policy](#)

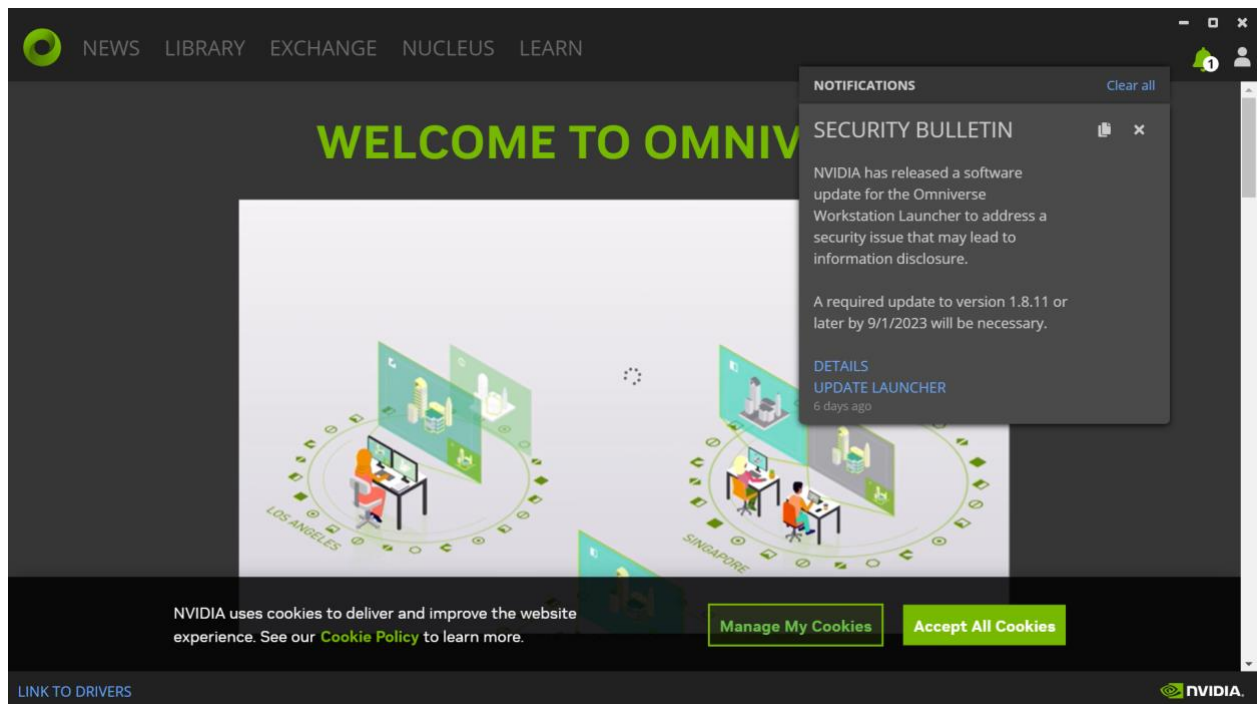
Submit

See You in Omniverse

Thank you for your submission.
Download here for [Windows](#) or [Linux](#).

Click on Windows to get downloaded in your laptops.

Login or Create Account



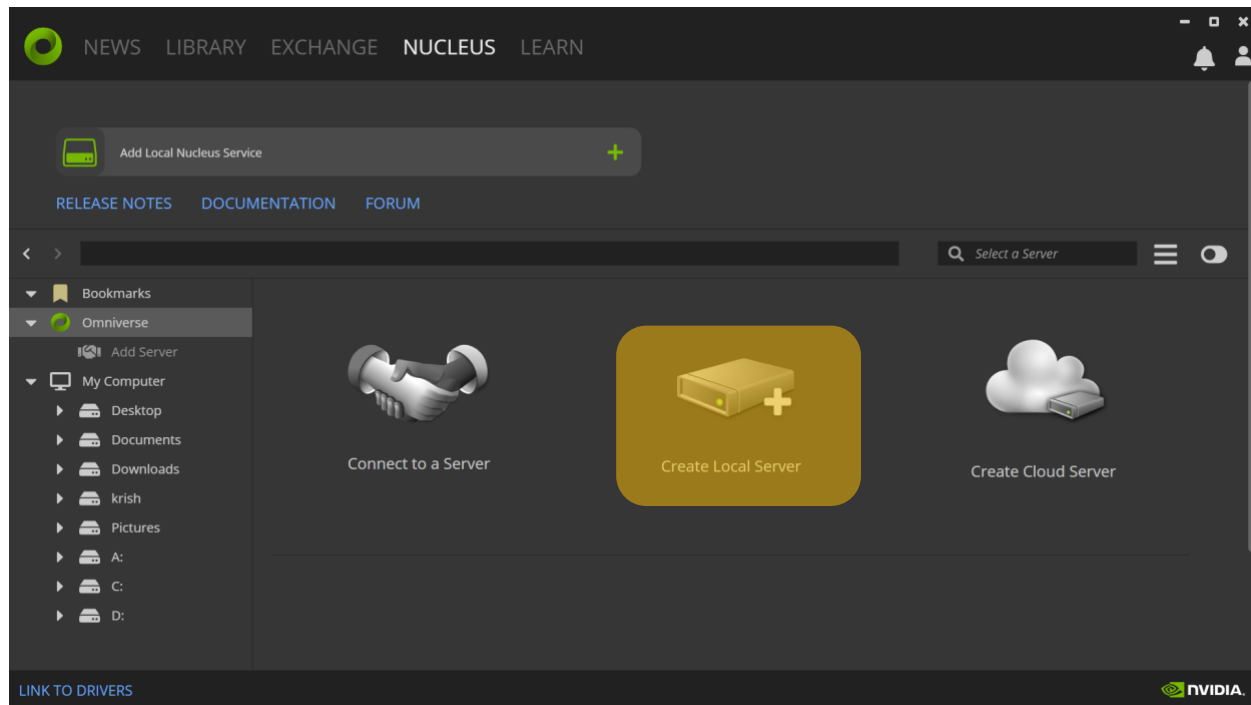
SECURITY BULLETIN

NVIDIA has released a software update for the Omniverse Workstation Launcher to address a security issue that may lead to information disclosure.

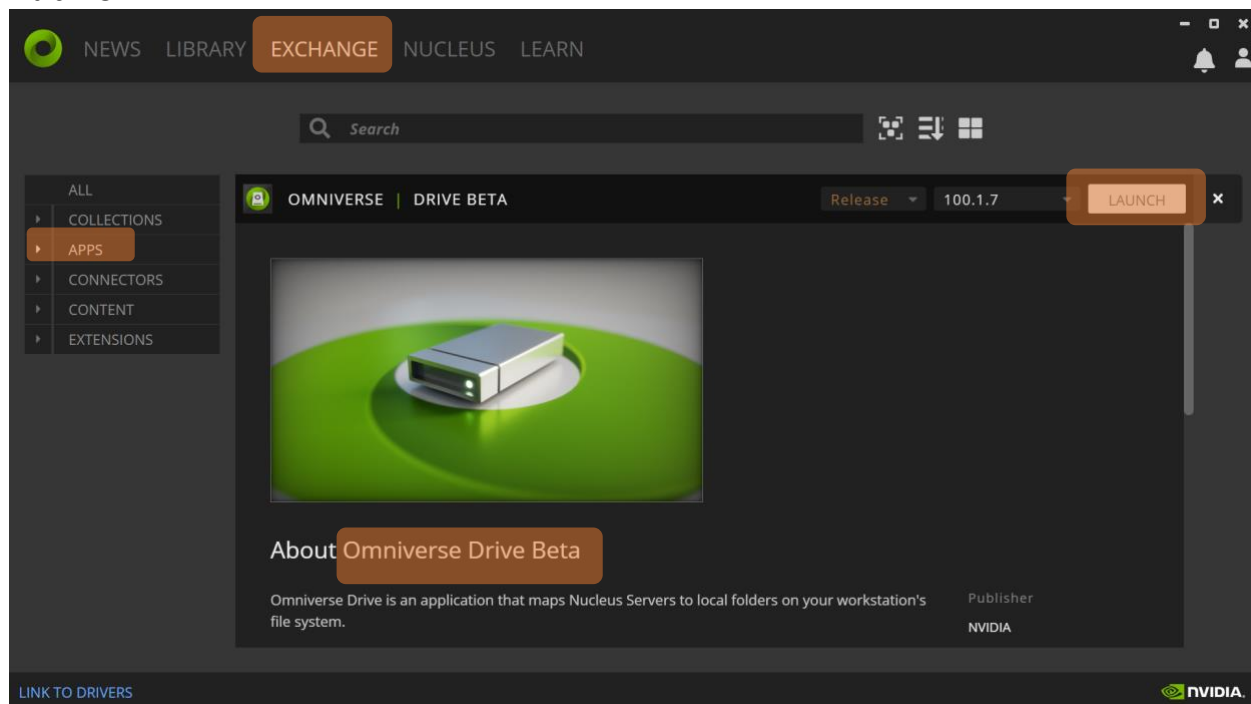
A required update to version 1.8.11 or later by 9/1/2023 will be necessary.

[Details](https://nvidia.custhelp.com/app/answers/detail/a_id/5472)

[Update Launcher](<https://www.nvidia.com/en-us/omniverse/download/>)
NUCLEUS

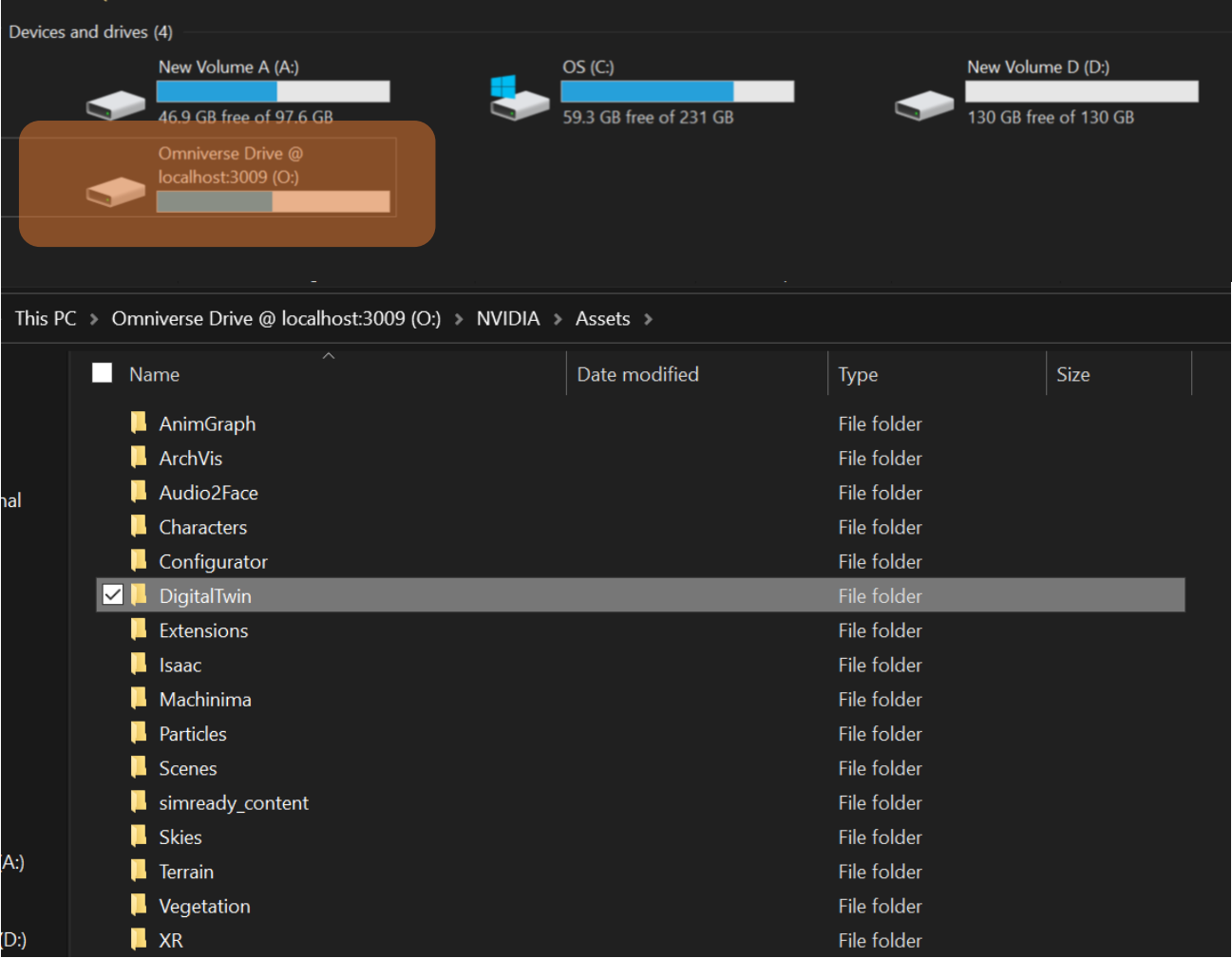


Click on **Exchange** and then select **apps**. Select **Omniverse Drive Beta** and then **install** it and **Launch**



After successful completion we will get a local host with 1TB SSD

Omniverse Drive



Simultaneously, Install

1. Omniverse View
2. Omniverse Kit
3. Omniverse Create

Applications

1. **Virtual Conferencing and Meetings**
2. **Healthcare**
3. **Gaming and Entertainment**
4. **Real Estate and Architecture**
5. **Social Interaction and Networking**
6. **Virtual Workspaces**

For this project we will look into Virtual Conferencing and Meetings application and explore through it.

Virtual Conferencing and Meetings

Business meetings, conferences, and events can take place in a virtual space, allowing participants from around the world to collaborate, share ideas, and attend presentations in a more interactive way.

The metaverse, and specifically Nvidia Omniverse, can enhance the virtual meeting experience:

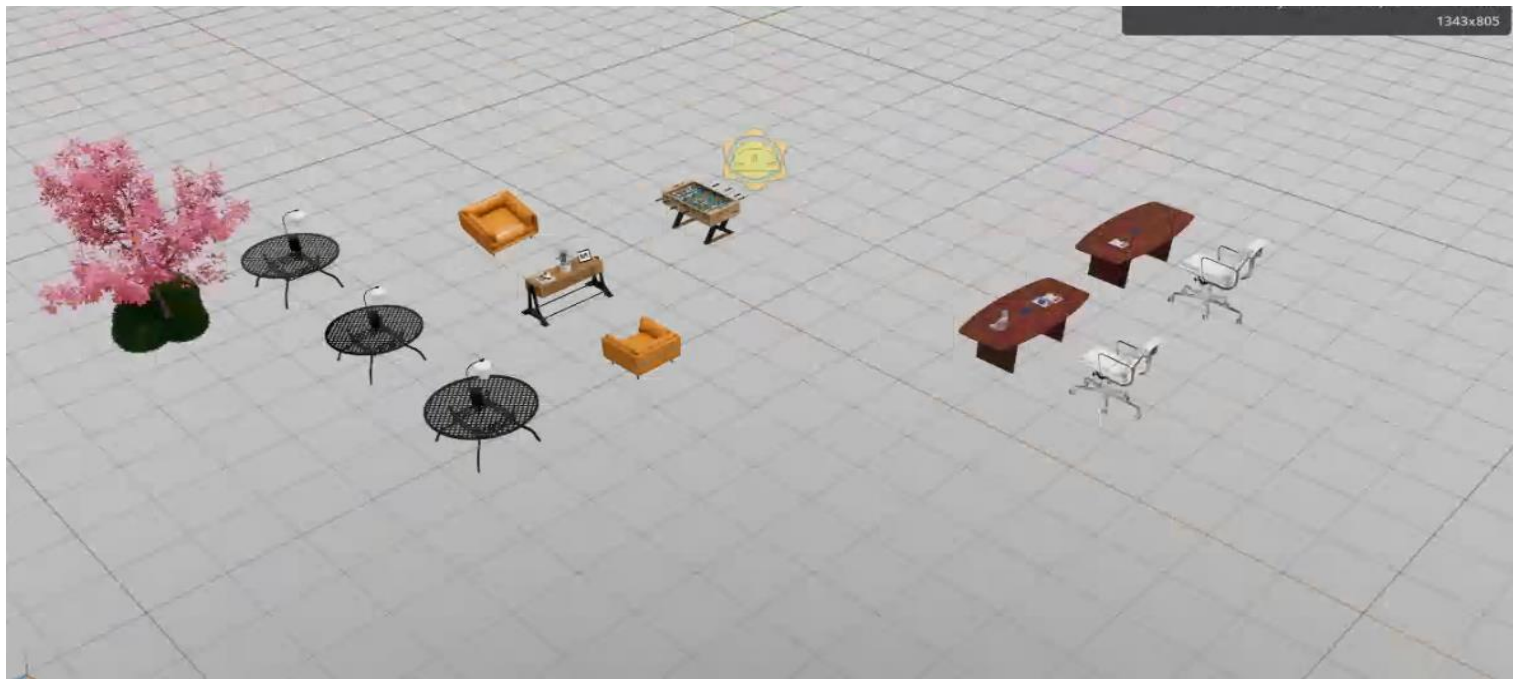
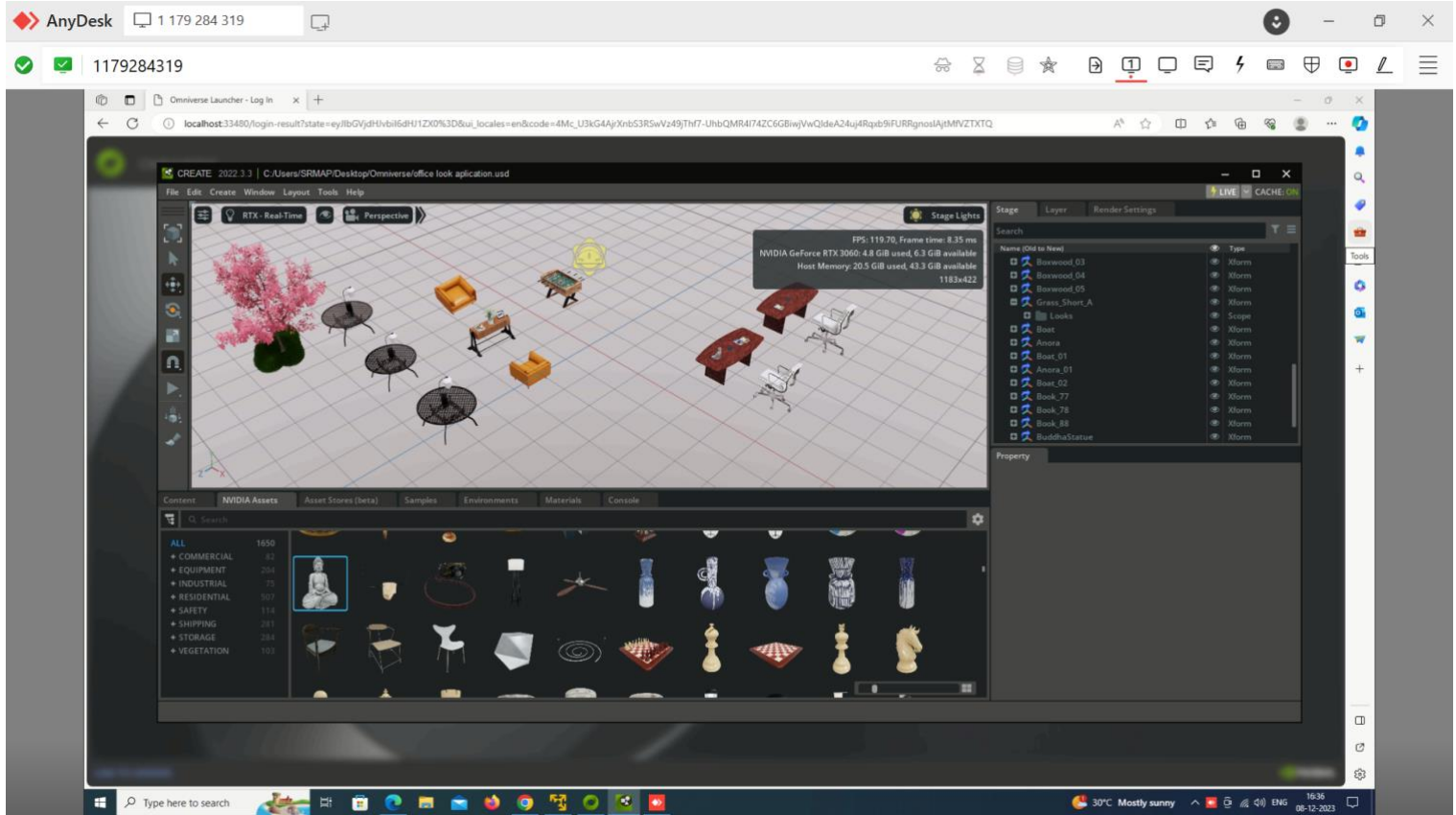
1. **Immersive Environments:** In the metaverse, participants can meet in virtual spaces that simulate real-world environments or entirely new and imaginative settings. This creates a more engaging atmosphere for discussions and collaboration.
2. **Realistic Avatars:** Users can create personalized avatars that represent them in the virtual space. These avatars can mimic facial expressions and body language, adding a human touch to virtual interactions.
3. **Spatial Audio:** Metaverse platforms often incorporate spatial audio technology, allowing users to hear voices and sounds as if they are coming from specific directions. This enhances the sense of presence and makes conversations more natural.

4. **Interactive Presentations:** In virtual meetings, presenters can use 3D models, simulations, and interactive visual aids to convey information. This is more engaging than traditional slide presentations and can improve the understanding of complex concepts.
5. **Collaborative Tools:** Metaverse platforms often include collaborative tools such as virtual whiteboards, 3D models, and shared documents. Participants can work together on projects in real-time, fostering creativity and productivity.
6. **Global Accessibility:** The metaverse breaks down geographical barriers, allowing participants from around the world to attend meetings without the need for physical travel. This can lead to increased inclusivity and participation.
7. **Persistent Spaces:** Some metaverse platforms offer persistent virtual spaces that remain available even when a meeting concludes. This enables ongoing collaboration and a sense of continuity for teams working on long-term projects.



Building Application

- **Realistic Graphics and Simulation:**
 - Nvidia Omniverse leverages advanced graphics capabilities, including real-time ray tracing and AI-driven simulation. This can enhance the visual fidelity of virtual environments and make them more realistic.
- **Physics-Based Rendering:**
 - Omniverse supports physics-based rendering, allowing for the creation of highly realistic materials and lighting effects. This can contribute to a more immersive and visually appealing virtual meeting experience.
- **Interoperability:**
 - Omniverse is designed with interoperability in mind. It supports industry-standard formats, making it easier to import and export 3D models, assets, and scenes from various applications. This can be beneficial for incorporating existing design and simulation data into virtual meetings.
- **AI-Driven Interactions:**
 - Nvidia's AI technologies can be integrated into Omniverse to enhance avatar animations and interactions. This can contribute to more natural and expressive communication in virtual meetings.
- **Simulation and Training:**
 - Omniverse's simulation capabilities can be utilized for training scenarios within virtual meetings. This is particularly valuable for industries such as architecture, engineering, and manufacturing, where simulations can aid in decision-making.



Conclusion

In conclusion, this research contributes to the growing body of knowledge surrounding the metaverse, offering insights into its technological foundations, societal implications, and practical applications. By examining both the promises and challenges associated with the metaverse, this study aims to inform policymakers, businesses, and individuals about the potential future of virtual experiences and their impact on various aspects of our lives.