Quantyverse Omniverse Tutorials

Welcome to the **Quantyverse Omniverse Tutorials**! This guide is designed to help you master Nvidia Omniverse, a powerful platform for creating and collaborating on real-time 3D simulations and applications.

About This Guide

Whether you're new to Omniverse or an experienced developer, you'll find resources here to help you build and customize applications, create extensions, and design user interfaces.

Guide Structure

Our tutorials are organized into the following categories:

1. Basics
    - Learn fundamental concepts and set up your development environment.
    - Topics include setting up the environment and creating your first extension.
2. User Interface (UI)
    - Explore how to create and customize user interfaces using `omni.ui`.
    - Covers creating UI elements, handling interactions, and advanced layouts.
3. Examples
    - Step-by-step examples demonstrating specific features or use cases.

    - Practical implementations to help you understand various Omniverse features.

Each category contains several tutorials with both instructional content and corresponding code files.

How to Use This Guide

Each tutorial is self-contained with all necessary code and resources. You can follow the tutorials in order or jump to specific topics of interest.

Prerequisites

To follow these tutorials, you should have:

- Nvidia RTX Graphics Card
- Basic understanding of Python programming
- Familiarity with 3D design concepts (helpful but not required)

Getting Started

Begin your journey with the Basics section to set up your environment and create your first Omniverse extension.

Additional Resources

- Nvidia Omniverse Documentation
- Nvidia Developer Forums

Contributing

We welcome contributions! If you have a tutorial to add, a bug to report, or an improvement to suggest, please submit a pull request or open an issue on our GitHub repository.

License

Ready to dive in? Let's start exploring the exciting world of Nvidia Omniverse!

Basics of Nvidia Omniverse

Welcome to the Basics section of our Nvidia Omniverse tutorials. In this section, you'll learn the fundamental concepts and skills needed to start developing with Omniverse.

What You'll Learn

- Setting up your Omniverse development environment
- Creating your first Omniverse extension
- Understanding the core concepts of Omniverse development

Tutorials in this Section

1. Create First App / Setting Up Environment Learn how to install and configure Omniverse for development.
2. Creating Your First Extension Walk through the process of creating a simple Omniverse extension.
3. Core Concepts Explore the fundamental concepts and architecture of Omniverse.

Each tutorial builds upon the previous one, so we recommend following them in order. However, feel free to jump to specific topics if you're already familiar with some concepts.

Ready to get started? Let's begin with Setting Up Your Environment!

Creating your first Nvidia Omniverse App (Setting Up Environment)

This guide will help you set up your development environment for working with the Omniverse Kit App Template by creating your first Omniverse App. Following these steps will ensure that your environment is properly configured for building applications using Nvidia Omniverse.

1. Set Up the Required Tools

- **Install Dependencies**: Ensure your development environment is equipped with the latest version of Python. The Kit App Template is optimized for use with Visual Studio Code, though other IDEs like PyCharm can also be used.
- **NVIDIA RTX GPU**: Make sure your development machine has an NVIDIA RTX GPU with the latest drivers installed. This hardware is essential for running Omniverse applications efficiently.

2. Clone the Kit App Template Repository

Start by cloning the Kit App Template repository from GitHub to your local machine:

```
git clone https://github.com/NVIDIA-Omniverse/kit-app-template.git
```

Navigate into the cloned repository:

```
cd kit-app-template
```

3. Create and Configure New Application From Template

Run the following command to initiate the configuration wizard:

Linux:

```
./repo.sh template new
```

Windows:

```
.\repo.bat template new
```

> **Note:** If this is your first time running the `template new` tool, you'll be prompted to accept the Omniverse Licensing Terms.

Follow the prompt instructions:

- **? Select with arrow keys what you want to create:** Application
- **? Select with arrow keys your desired template:** Kit Base Editor or USD Composer
- **? Enter name of application .kit file [name-spaced, lowercase, alphanumeric]:** [set application name]
- **? Enter application_display_name:** [set application display name]
- **? Enter version:** [set application version]

> **Note:** If you choose for example the Kit Base Editor you need to add into the ./source/apps/your_app_name.kit under dependencies the following "omni.kit.window.extensions" = {}. Otherwise you can not use the Extensions Manager to start new extensions in your app as explained in step 6.

---

4. Build the Project

To build the project, use the provided scripts according to your operating system:

- **Windows**:

  ```
  .\repo.bat build
  ```

- **Linux**:

  ```
  ./repo.sh build
  ```

This command compiles the necessary components and prepares your environment for running the Omniverse applications.

5. Launch the Application

After building the project, you can launch the application:

- **Windows**:

  ```
  .\repo.bat launch
  ```

- **Linux**:

  ```
  ./repo.sh launch
  ```

During the launch process, you'll be prompted to select the `.kit` file associated with your application, typically located in the `_build` directory.

6. Explore and Customize

- **Extension Manager**: After launching, open the Extension Manager by navigating to `Window > Extensions` to explore and enable various extensions. This allows you to extend the functionality of your application.
- **Customization**: Start by exploring and customizing the reference applications included in the template, such as the USD Explorer. This will help you understand how to configure and modify Kit-based applications.

7. Additional Resources

For more detailed instructions and advanced configuration options, refer to the official Omniverse Kit App Template documentation. You can also configure your environment to use community-developed extensions by adjusting the registry settings.

By following these steps, you'll have a fully functional development environment for creating and customizing applications within Nvidia Omniverse using the Kit SDK.

Creating your first extension in Nvidia Omniverse

This guide will help you to create your first extension in the App you just created here Create Frist App / Setting up Environment

1. Create and Configure New Extension From Template

Run the following command to initiate the configuration wizard:

Linux:

```
./repo.sh template new
```

Windows:

```
.\repo.bat template new
```

Follow the prompt instructions:

- **? Select with arrow keys what you want to create:** Extension
- **? Select with arrow keys your desired template:** Basic Python Extension
- **? Enter name of application .kit file [name-spaced, lowercase, alphanumeric]:** [set extension name]
- **? Enter application_display_name:** [set extension display name]
- **? Enter version:** [set extension version]

> **Note:** You can choose between different Extension types like pure Python extension or Python UI extension which will have pre generated content

tailored for the specific type. I recommend to have a look at these extensions.

---

2. Build the Extension

So the extension to be added to your app you have to build the application again, use the provided scripts according to your operating system:

- **Windows**:

```
.\repo.bat build
```

- **Linux**:

```
./repo.sh build
```

This command compiles the necessary components and prepares your environment for running the Omniverse applications with your new extension.

3. Launch the Application

After building the project, you can launch the application and see if your extension was created as expected:

- **Windows**:

```
.\repo.bat launch
```

- **Linux**:

```
./repo.sh launch
```

During the launch process, you'll be prompted to select the `.kit` file associated with your application, typically located in the `_build` directory.

4. Start your Extension

- **Extension Manager**: After launching, open the Extension Manager by navigating to `Window > Extensions` to explore and enable various extensions. Type the name of your extension you just created and start it.

5. Additional Resources

For more detailed instructions and advanced configuration options, refer to the official Omniverse Kit App Template documentation.

By following these steps, you'll have a fully functional extension you can now use to develop. Check out the other chapters on how to fill the extension with content.

Core Concepts: Components of Omniverse Kit SDK

The Omniverse Kit Software Development Kit (SDK) provides a robust framework for building applications within the Omniverse ecosystem. Understanding its core components is crucial for effective development. Let's explore the key elements that make up the Kit SDK.

Kit Kernel Package

At the heart of the SDK lies the Kit Kernel Package, which serves as the central nervous system for Omniverse applications.

Key Features

1. **Extension System**
    - Manages the lifecycle of extensions, including loading, initialization, and shutdown.
    - Provides a plugin architecture for modular application development.
    - Handles dependencies between extensions.
2. **Event System**
    - Facilitates communication between different components of the application.
    - Implements a publish-subscribe model for efficient event handling.
    - Allows for loose coupling between modules, enhancing maintainability.
3. **Update Loop**

- Ensures all components are updated appropriately throughout the application lifecycle.
- Manages frame-by-frame updates for real-time applications.
- Provides hooks for custom update logic in extensions.

Additional Kernel Features

4. **Configuration System**
    - Manages application and extension settings.
    - Supports runtime configuration changes.

Kit SDK Extensions

The Kit SDK includes a set of foundational extensions that provide essential functionality for Omniverse applications.

Highlighted Extensions

1. **omni.usd**

    - Provides synchronous and asynchronous interfaces to manage USD (Universal Scene Description) contexts.
    - Facilitates querying and modification of scene state.
    - Handles USD stage loading, saving, and live updates.

2. **omni.kit.commands**

    - Used to register and execute Commands within the application.
    - Implements undo/redo functionality for user actions.
    - Allows for the creation of custom commands for specific application needs.

3. **omni.kit.viewport.window**

    - Manages the Hydra-based viewport for 3D scene rendering.

    - Handles user interactions within the viewport, such as selection and camera controls.
    - Provides APIs for custom viewport manipulations and overlays.

4. **omni.ui**

    - The Omniverse UI framework for creating user interfaces.
    - Offers a wide range of UI elements and layouts.
    - Supports both 2D overlay UIs and 3D in-viewport interfaces.

Additional Important Extensions

5. **omni.kit.menu**

    - Manages application menus and context menus.
    - Allows for dynamic menu creation and modification.

6. **omni.kit.property.usd**

    - Provides property inspection and modification for USD objects.
    - Facilitates the creation of custom property editors.

7. **omni.kit.window.extensions**

    - Manages the extension browser and loader interface.
    - Allows users to view, enable, and disable extensions at runtime.

Relationships and Interactions

Understanding how these components interact is crucial for effective Omniverse development:

- The Kit Kernel acts as the foundation, managing the overall application lifecycle.
- Extensions build upon the Kernel, providing specific functionalities.

- The Event System allows extensions to communicate without direct dependencies.
- The Update Loop ensures all active extensions are updated each frame.
- USD serves as the common language for 3D data, with many extensions interacting with USD data.

Practical Application

To illustrate how these components work together, let's consider a simple scenario:

1. A user clicks a button in the UI (`omni.ui`) to add a cube to the scene.
2. This triggers a command (`omni.kit.commands`) to create a cube.
3. The command interacts with the USD stage (`omni.usd`) to add the cube geometry.
4. The viewport (`omni.kit.viewport.window`) is updated to display the new cube.
5. The property panel (`omni.kit.property.usd`) is refreshed to show the cube's attributes.
6. All of this is orchestrated by the Kit Kernel, which manages the extension lifecycles and update loop.

Conclusion

The Omniverse Kit SDK provides a powerful and flexible framework for building 3D applications. By understanding its core components and how they interact, developers can create sophisticated, performant, and extensible applications within the Omniverse ecosystem.

As you delve deeper into Omniverse development, you'll discover how these components can be leveraged and extended to create unique and powerful applications.

Nvidia Omniverse Basic UI Elements Tutorials

Introduction

Welcome to the **Basic UI Elements** section of our Nvidia Omniverse tutorials. Here, you will learn how to create and customize fundamental UI elements using the `omni.ui` library. These tutorials are designed to help you build a strong foundation in user interface development within the Omniverse environment.

Whether you're new to Omniverse or an experienced developer looking to refine your skills, these tutorials provide step-by-step guidance and practical examples to get you started.

What You'll Learn

In this section, we will cover the following basic UI elements:

1. **ui.Label**: Learn how to create and customize labels to display text information in your user interfaces. Labels are simple yet essential components that provide context and clarity to users.

2. **ui.Button**: Discover how to create interactive buttons. Buttons are crucial for enabling user interactions, triggering actions, and navigating through different parts of your application.

3. **UI Models**: Basic understanding of how models of omni.ui work to set and get the values for the view of ui elemnts. With view we mean what the user gets presented on the screen.

4. **ui.StringField, FloatField, and IntField**: Master the use of input fields to capture user input. Learn how to implement and handle different types of inputs, including text, floating-point numbers, and integers.

Why These Elements Matter

Basic UI elements are the building blocks of any application. By mastering these components, you can create intuitive and user-friendly interfaces that enhance the user experience. Understanding how to use these elements effectively is crucial for:

- Building interactive and responsive applications.
- Ensuring data is captured and handled correctly.
- Creating a consistent and accessible user experience.
- Laying the groundwork for more advanced UI components and interactions.

Getting Started

Each tutorial includes:

- **Explanations**: We explain the purpose and functionality of each UI element, providing a clear understanding of how and when to use them.
- **Step-by-Step Instructions**: Follow along with easy-to-understand steps to create and customize each element.
- **Code Examples**: See practical examples of code implementations that you can use and modify for your projects.

Prerequisites

Before diving into these tutorials, make sure you have the following:

- A basic understanding of Python programming.
- Nvidia Omniverse installed and set up. If you haven't done this yet, please refer to our Create Omniverse App / Setting up environment.
- Familiarity with basic UI concepts will be helpful but is not required.

Start Learning

Choose a tutorial from the list below to get started:

- ui.Label
- ui.Button
- UI Models
- ui.StringField, FloatField, and IntField

We hope these tutorials help you become proficient in using `omni.ui` to create compelling and effective user interfaces within the Nvidia Omniverse. Happy learning!

ui.Label Tutorial

Introduction

The `ui.Label` is one of the simplest and most commonly used UI elements in `omni.ui`. It is used to display text within your UI. In this tutorial, we will learn how to create and customize labels.

Prerequisites

- Basic understanding of Python.
- Nvidia Omniverse environment setup.

Creating a Basic Label

To create a label, you use the `ui.Label` class. Below is an example of how to create a basic label in a window.

```
import omni.ui as ui

# Create a window
window = ui.Window("Label Example", width=300, height=200)

# Create a label inside the window
with window.frame:
    ui.Label("Hello, Omniverse!")
```

Change the text of a label

You can change the text of a label by using the `text` property.

```
import omni.ui as ui

# Create a window
window = ui.Window("Label Example", width=300, height=200)

# Create a label inside the window
with window.frame:
    label = ui.Label("Hello, Omniverse!")
    label.text = "New Text"
```

Customizing a Label

You can customize a label by setting various properties. For example, you can change the text color, font size, and alignment.

```
import omni.ui as ui
from omni.ui import color as cl

# Create a window
window = ui.Window("Customized Label Example", width=300,
height=200)

# Create a label inside the window
with window.frame:
    ui.Label("Customized Label", style={"font_size": 24, "color":
cl("#097eff")})
```

Summary

In this tutorial, we learned how to create and customize labels in `omni.ui`. We covered the basics of creating a label, changing its text, and customizing its appearance. Labels are essential for displaying text in your UI applications, and

mastering them will help you create more sophisticated and informative user interfaces.

ui.Button Tutorial

Introduction

The `ui.Button` is used to create interactive buttons in `omni.ui`. This tutorial will show you how to create buttons and handle button click events.

Prerequisites

- Basic understanding of Python.
- Nvidia Omniverse environment setup.

Creating a Basic Button

To create a button, you use the `ui.Button` class. Below is an example of how to create a basic button and handle a click event.

```
import omni.ui as ui

# Create a window
window = ui.Window("Button Example", width=300, height=200)

# Create a button inside the window
with window.frame:
    def on_button_click():
        print("Button clicked!")

    ui.Button("Click Me", clicked_fn=on_button_click)
```

Customizing a Button

You can customize a button by setting its text, icon, and other properties. Below is an example of how to create a button with a custom icon and text.

```
import omni.ui as ui
from omni.ui import color as cl

window = ui.Window("Customized Button", width=300, height=200)
with window.frame:
    def on_button_click():
        print("Styled button clicked!")

    ui.Button(
        "Styled Button",
        clicked_fn=on_button_click,
        style={"background_color": cl.('#097eff')}
    )
```

Summary

In this tutorial, we learned how to create and handle button click events in `omni.ui`. We covered the basics of creating a button and handling click events. Buttons are essential for creating interactive UI applications in Nvidia Omniverse, and mastering them will help you create more sophisticated and user-friendly interfaces.

UI Models

ui.StringField, FloatField, and IntField