USD Time Controller Documentation

Overview

The TimeController class is a component that manages time in the USD Stage within NVIDIA Omniverse environment. This controller maps real dates/times (datetime) to USD's timecode system, allowing users to intuitively explore time-based data.

USD Timecode System

USD (Universal Scene Description) manages all time-based data around a single timeline:

- **Timecode**: The basic unit for representing time in USD, expressed as a floating-point value.
- **Unified Timeline**: All animation properties (position, color, scale, etc.) are mapped onto the same timeline.
- **Property Independence**: Each property independently stores values at timecodes.

TimeController Prerequisites

For TimeController to work correctly, the USD Stage must have a `/World/TimeManager` prim with a `baseTime` value set. This is the essential reference point for mapping between real time and USD timecodes.

python 복사

```
def "TimeManager" (
    customData = {
        string baseTime = "2025-01-01T00:00:00Z"
        string currentTime = "2025-01-01T00:00:00Z"
        double timeScale = 1.0
    }
)
```

- If `baseTime` is not set, TimeController should find or set an appropriate default value. This can be done by finding the earliest time in the data or applying a user-specified default value. (This is not implemented.)

Key Features of TimeController

TimeController provides the following key features:

1. **Real Time and Timecode Mapping**: Converting real dates/times (datetime) to USD timecodes
2. **Time Range Setting**: Defining a time range of interest (start/end times)

3. **Time Control**: Setting the current time, play/pause, adjusting playback speed
4. **Progress Control**: Intuitive time navigation using relative progress (0.0-1.0)

Time Format Considerations

When implementing TimeController, there are important considerations regarding time formats:

Time Format Consistency

The baseTime in TimeManager and the time parsing function in the code must match. By default, ISO 8601 format (`"%Y-%m-%dT%H:%M:%SZ"`) is used, but other formats can be used as well:

python 복사

```
# ISO 8601 format (default)
base_dt = datetime.datetime.strptime(base_time_str, "%Y-%m-%dT%H:%M:%SZ")
# Another format example (MM/DD/YYYY HH:MM:SS)
base_dt = datetime.datetime.strptime(base_time_str, "%m/%d/%Y %H:%M:%S")
```

- The important thing is to use a consistent format throughout the system. If you change the format, you must update both the parsing function (`strptime`) and the string generation function (`strftime`) to use the same format.

Time Mapping Mechanism

1. Setting the Reference Time

TimeController uses the "baseTime" stored in the `/World/TimeManager` USDA as a reference point:

python 복사

```
time_manager.SetCustomDataByKey("baseTime", "2025-01-01T00:00:00Z")
```

2. Converting Date/Time to Timecode

What is a Timecode?

- In USD, a timecode is a simple numerical value that represents a position in time. It is typically measured in seconds and indicates how much time has elapsed from a specific reference point. This timecode is used to determine the state of animations or time-varying data at a specific point in time.

The process of converting real time to USD timecode is as follows:

python 복사

```python
def _datetime_to_timecode_value(self, dt):
    # 1. Get the reference time
    base_time_str = time_prim.GetCustomDataByKey("baseTime")
    base_dt = datetime.datetime.strptime(base_time_str, "%Y-%m-%dT%H:%M:%SZ")

    # 2. Calculate elapsed time (seconds)
    delta_seconds = (dt - base_dt).total_seconds()

    # 3. Use elapsed time in seconds as timecode
    return delta_seconds
```

3. Applying Timecode to Stage

The process of setting the USD Stage's time (timecode) to the user-selected time point:

python 복사

```python
# Set USD's time (current_time) to the time the user wants to view
def _update_stage_time(self):
    # 1. Convert the user-selected time point (self._current_time) to timecode
    timecode_value = self._datetime_to_timecode_value(self._current_time)

    # 2. Apply to Omniverse timeline interface
    self._timeline.set_current_time(timecode_value)

    # 3. Update metadata (optional)
    time_prim.SetCustomDataByKey("currentTime", self._current_time.strftime("%Y-%m-%dT%H:%M:%SZ
```

Principles of Timecode Control

How TimeController controls the USD system's timecode:

1. **Centralized Control**: The controller centrally controls the timecode for the entire Stage.
2. **Property Independence**: The controller does not directly control individual properties (position, color, etc.).
3. **Automatic Updates**: When the timecode changes, the USD system automatically updates all time-based properties.

This approach is similar to adjusting the playback position in a video player. The player does not need to know the content of the video (color, sound, etc.), it only changes the timeline position.

Code Example: Time Control

python 복사

```python
# 1. Move to a specific time point
controller.set_current_time(datetime.datetime(2025, 1, 1, 0, 0, 30))  # Move to 30-second point

# 2. Move using relative progress
controller.set_progress(0.5)  # Move to the midpoint of the entire time range

# 3. Start/Stop playback
controller.toggle_playback()  # Toggle playback state

# 4. Adjust playback speed
controller.set_playback_speed(2.0)  # 2x speed playback
```

Compatibility with Various Data Types

TimeController can work with various types of time-based data:

- **Position Data**: Object movement (translate_op)
- **Color Data**: Visual changes (display_color_attr)
- **Shape Data**: Shape changes (points, normals)
- **Visibility Data**: Show/hide status (visibility)
- **Other Properties**: All animatable properties in USD

Since the controller only controls the timecode without directly handling individual properties, there is no need to modify it when new types of time-based data are added.

Usage Scenarios

1. **Simulation Data Exploration**: Checking the state of simulations at various points in time
2. **Sensor Data Visualization**: Observing changes in sensor data over time
3. **Time Travel Interface**: Intuitively exploring past/future data

Implementation Considerations

- **Performance**: In large datasets, updating all properties when the timecode changes can affect performance.
- **Accuracy**: Precision loss may occur during time conversion, so caution is needed in high-precision applications.
- **Scalability**: Timecode values can become very large for very wide time ranges (e.g., multiple years).

Conclusion

The TimeController class leverages USD's powerful timecode system to allow users to explore time-based data in an intuitive way. It provides a consistent interface for all types of time-based data through single timeline control and is extensible when new data types are added.