NVIDIA RTX Bonsai Diorama

Developer's Guide

Table of Contents

General Overview

NVIDIA RTX Bonsai Diorama is the third NVIDIA RTX Unreal Engine “Showcase” featuring NVIDIA RTX Technologies, focusing on tech such as RTX Mega Geometry, Path Tracing, RTX Dynamic Illumination (RTXDI), DLSS Super Resolution (DLSS-SR), DLSS Ray Reconstruction (DLSS-RR), and DLSS Frame Generation (DLSS-FG).

System Requirements

Operating Systems	Windows 10 64-bit Windows 11 64-bit
Dependencies	NVIDIA RTX GPU – Min: RTX 2080Ti, Rec: RTX 5060 NVIDIA RTX DRIVER - 581.29 or newer
Development Environment (For Source Compilation)	Setting Up Visual Studio Development Environment for C++ Projects in Unreal Engine | Unreal Engine 5.6 Documentation | Epic Developer Community

Prerequisites

Project Source	Direct Download
Project Binary Executable	Direct Download
NVIDIA RTX Branch (NvRTX) 5.6	RTX Branch of Unreal Engine (NvRTX) | NVIDIA Developer

Setup

There are two ways to review the Bonsai Diorama and showcased technologies:

1. Binary executable.
2. Project Source using the NVIDIA RTX Unreal Engine 5.6 editor.

For a quick review, it's recommended to start with the binary executable.

For a deeper dive, it's recommended to download the project and engine source to then view the project in-editor.

Binary Executable

1. Please refer to the **`System Requirements`** and **`Prerequisites`** supplied on the previous page.
2. Download and extract the Binary Package .zip file in the desired location.
3. Launch BonsaiDiorama.exe

Project Source

1. Please refer to the **`System Requirements`** and **`Prerequisites`** supplied on the previous page.
2. Download and extract the Project Source .zip file in desired location.
3. Grab the NVIDIA RTX Unreal Engine branch source code (branch: **`nvrtx-5.6`**) and place it in the desired location.
    a. **Note:** It's important that the engine is somewhere on the root drive. Example: **`C:\`** or **`D:\`**. Otherwise, compilation errors may occur due to long file paths.
4. Compile Engine Source Code by following these instructions: Building Unreal Engine from Source | Unreal Engine 5.6 Documentation | Epic Developer Community
5. Once the engine is compiled, right click over the **`BonsaiDiorama.uproject`** file, extracted from the Project Source .zip. -> **`Switch Unreal Engine Version...`** -> Choose **`nvrtx-5.6`** in the drop down. If you do not see this in the drop down, click on the **`...`** button. In the folder explorer window, click on the location where the engine was cloned and compiled. Accept it with 'Select Folder'.
6. Launch **`BonsaiDiorama.uproject`**.

Start Playing

Game Mode

Game mode refers to either one of the following:

- Launching the **`BonsaiDiorama.exe`**
- Launching the Source Project in **`-game`** mode *or* in Play-In-Editor (PIE) Mode.
    - If launching outside of command line and from within editor, it's recommended to use either Standalone Game Mode or Separate Viewport PIE Mode.

Main Menu

Upon launch, there is a menu where resolution and window settings can be adjusted, prior to launching into the demo level.

These settings can *only* be changed here. There is an ability to come back to this main menu screen to adjust settings if desired.

Recommendations

- Start with Full Screen (default)
- If using an 8GB GPU, start out with 1080p or 1440p depending on display resolution availability.
- If using an 11GB GPU or higher, test the limits with 4k resolution, depending on display resolution availability.

Explore

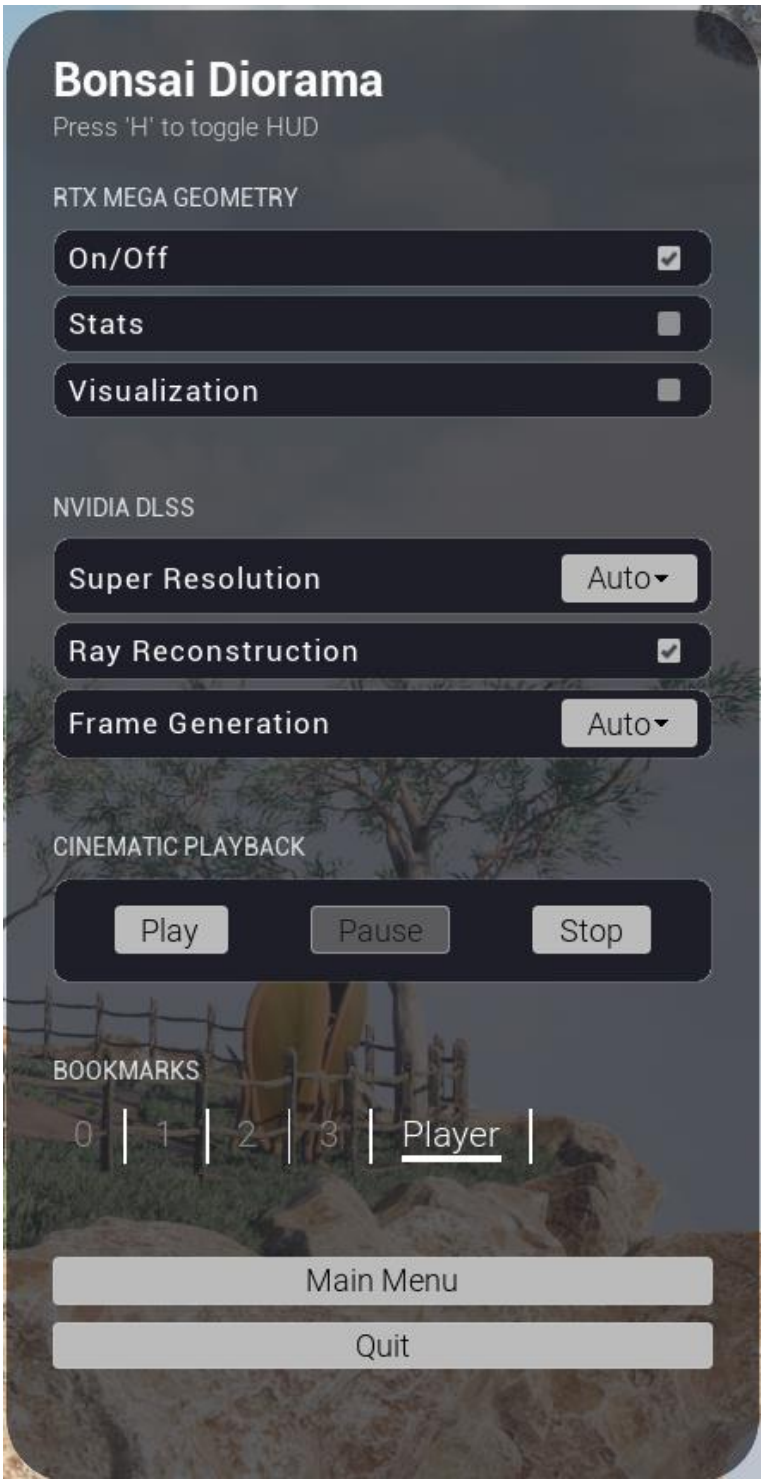
After setting desired resolution and window modes, click the thumbnail to load into the demo level.

Once the level loads, a cinematic sequence is displayed, an FPS counter appears at the top center of the screen, and a notification about enabling the HUD (with 'H' on the keyboard) is shown at the bottom center.

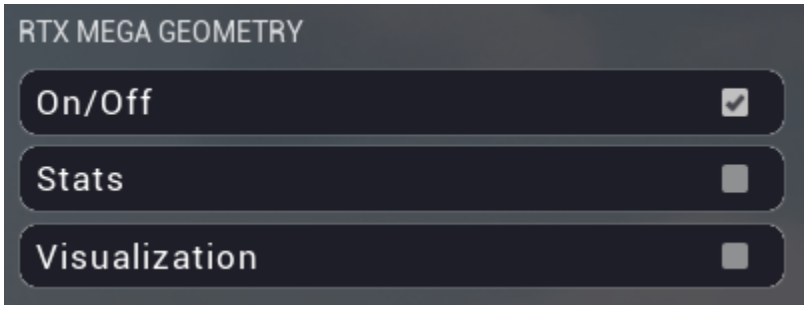
The cinematic sequence highlights dense foliage and complex geometry for RTX Mega Geometry visualization, while also demonstrating dynamic direct and indirect lighting with ReSTIR PT (RTX Dynamic Illumination).

User Interface

After pressing ’H’ on the keyboard, the UI pops up on the right side of the screen, giving a handful of settings to get started.



RTX Mega Geometry

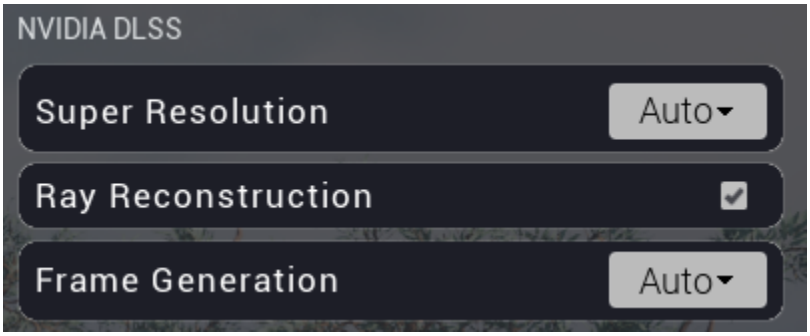


- On/Off: Simple toggle that enables and disables RTX Mega Geometry via the **`r.Nanite.RayTracing.Update`** console variable. More information on this console variable later on in this guide.
- Stats: Enables a debug UI that will help guide budgets along with other metrics for performance and memory usage.
- Visualization: Disables material evaluation, traces primary rays, and outputs the geometry normal.

`Note:` RTX Mega Geometry does not need to be enabled (the first check box in the UI) to visualize the Debug UI.

NVIDIA DLSS

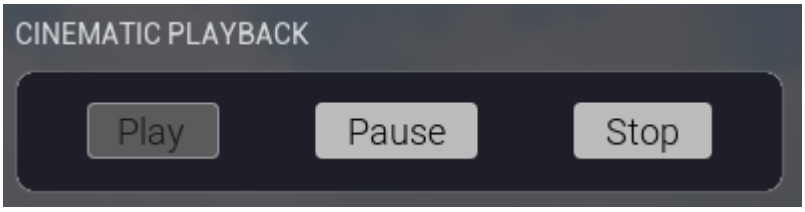
NVIDIA DLSS 4 is available in this demo and depending on the GPU running, it can utilize all technologies specified below.



- Super Resolution: Off, DLAA, Auto, Quality, Balanced, Performance, Ultra Performance.
- Ray Reconstruction: On/Off Toggle.
- Frame Generation: Off, Auto, 2X, 3X, 4X

- 2X is available on 40 and 50 series GPUs, while 3X and 4X is only available on 50 series GPUs.

Cinematic Playback



- Play: Cinematic plays at the load of the level by default.
- Pause: Pause the cinematic sequence at the current frame.
- Stop: End's the cinematic sequence and begins fly through mode.

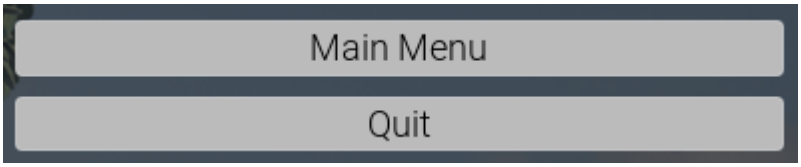
Note: RTX Mega Geometry and NVIDIA DLSS settings can be changed for Comparison shots at any time, regardless of cinematic sequence state.

Bookmarks

Ensure the cinematic sequence is stopped to gain full control of the Bookmarks menu.

- Player: Flythrough Mode
- 0 | 1 | 2 | 3: Designated bookmark locations that pinpoint areas of interest.

Exit Buttons



- Main Menu: Returns to the main menu to adjust resolution and window mode settings.
- Quit: Exits the application.

Using RTX Mega Geometry

RTX Mega Geometry adds full quality Nanite geometry to the ray tracing bounding volume hierarchy (BVH). As a result, ray traced shadows and reflections no longer exhibit visual artifacts typically present with UE5's default behavior.

Below are two examples:

Incorrect Self Occlusion Resolved

High Fidelity Nanite Geometry in Ray Traced Reflections

Stats (Debug UI)

Every project is different. Some projects are more intensive than others due to geometry, material, and lighting complexity. Since the RTX Mega Geometry integration in NvRTX 5.6 is currently experimental, it requires a keen eye to the video memory budgets in place for a given hardware config.

After toggling on the Stats checkbox, a debug UI will present itself on the left side of the screen. This is where all the information on setting up budgets, evaluating memory usage, and streaming data is found.

While all this information is useful for memory management, the most useful stats are under the Summary category.

Nanite:

- **Unique Meshes:** Displays the number of *unique* Nanite meshes in the scene.
- **Primitives:** Displays the number of Nanite primitives (static meshes) in the scene.
- **Primitive Instances:** Displays the number of Nanite primitive *instances* in the scene. Accounts for the instances of all Nanite primitives (placed either via the foliage tool *or* via blueprint actors).

Budgets:

Each line in this section displays the value that needs to be set in the project's DefaultEngine.ini for the given console variable.

- **r.Nanite.RayTracing.BLAS.MaxToBuild:** Displays the maximum number of Clustered BLAS that need to be built per frame.
- **r.Nanite.RayTracing.BLAS.CLASAddresses.Max:** Displays the maximum number of BLAS CLAS Addresses used across all Clustered BLASes.
- **r.Nanite.RayTracing.CLAS.Memory.Root.MaxSizeInMB:** Displays the maximum amount of memory used for root CLAS, in megabytes. This value can also be used for **r.Nanite.RayTracing.CLAS.Memory.Root.FixedSizeInMB**.
- **r.Nanite.RayTracing.CLAS.Memory.Streaming.MaxSizeinMB:** Displays the maximum amount of memory used for streaming CLAS, in megabytes. This value can also be used for **r.Nanite.RayTracing.CLAS.Memory.Streaming.FixedSizeInMB**.

Memory Used: Displays memory used vs. allocated (in megabytes)

Start Developing

Setting up RTX Mega Geometry

RTX Mega Geometry uses Nanite geometry and puts the full resolution nanite mesh into the BVH. All that's needed is to enable RTX Mega Geometry and ensure static meshes have Nanite enabled.

To enable RTX Mega Geometry, ensure **`r.Nanite.RayTracing=1`** and **`r.Nanite.RayTracing.Update=1`** are set in **DefaultEngine.ini**.

For the sake of this Bonsai Diorama, this is already set. There are also other console variable tunings set in the DefaultEngine.ini under the Render Settings section. We have commented out a sub-section title to provide the list of RTX Mega Geometry related settings for convenience.

`Note:` The *only* settings required are the first two (also mentioned above).

```
;; RTX Mega Geometry
r.Nanite.RayTracing=1
r.Nanite.RayTracing.Update=1
r.Nanite.RayTracing.Visualize=1
r.Nanite.RayTracing.Stats=1
r.Nanite.RayTracing.Stats.Show=0
r.Nanite.RayTracing.Shaders.Debug=0
r.Nanite.RayTracing.Culling.InsideFrustumRadius=1500000
r.Nanite.RayTracing.Culling.OutsideFrustumRadius=500000
r.Nanite.RayTracing.BLAS.MaxToBuild=10000
r.Nanite.RayTracing.BLAS.CLASAddresses.Max=1500000
r.Nanite.RayTracing.CLAS.Memory.Root.FixedSizeInMB=2
r.Nanite.RayTracing.CLAS.Memory.Streaming.FixedSizeInMB=768
r.Nanite.Streaming.StreamingPoolSize=768
r.AllowStaticLighting=False
```

Navigating Bonsai Diorama in Editor

Bonsai_Diorama.umap

This is the main demo level.

Key info

There are in-editor bookmarks spanning from keys **0** through **7** on the keyboard. These are separate bookmarks from the in-game bookmarks available in the UI.

Nanite Assets in the level:

- Bonsai Tree
- Eucalyptus Trees
- Rocks
- Fence
- Ivy
- Mirrors

Since RTX Mega Geometry is currently an experimental feature, mesh-types such as Skeletal Meshes are not yet supported.

Instancing

Instancing is a powerful tool for managing video memory. This is generally the case even outside of RTX Mega Geometry.

Items such as the Rocks, Bonsai Tree, Mirrors, and Eucalyptus Trees are placed as Blueprint Actors. The Ivy in the level is not instanced.

Inside the Blueprint Actors, we use Instanced Static Mesh components to place duplicates of meshes that were either generated within the editor (such as the mirrors), or custom-made assets imported in as FBXs.

For information on how the Bonsai Tree and Eucalyptus Trees were assembled, navigate to **`Levels/Assembly.umap`** and follow the on-screen messages.

Questions?

Feel free to reach out to us with any questions at NvRTX-Support@nvidia.com