UNREAL
ENGINE

NANITE FOR EDUCATORS AND STUDENTS

TABLE OF CONTENTS

Nanite for Educators and Students

This guide is for educators currently using Unreal Engine 4 or other game engines who are looking to migrate to Unreal Engine 5. We assume you understand how game engines and 3D software render 3D geometry and are familiar with concepts like UVW mapping, polygons, and so on.

Key Concepts for Educators

What's the Big Deal?

Nanite is a new method for rendering 3D graphics that significantly reduces the performance limitations associated with polygon and scene object counts.

Nanite directly and significantly affects the workflows of environmental artists, level designers, technical artists, and layout artists. Students in these fields should begin using Nanite and developing workflows and pipelines around it as soon as possible.

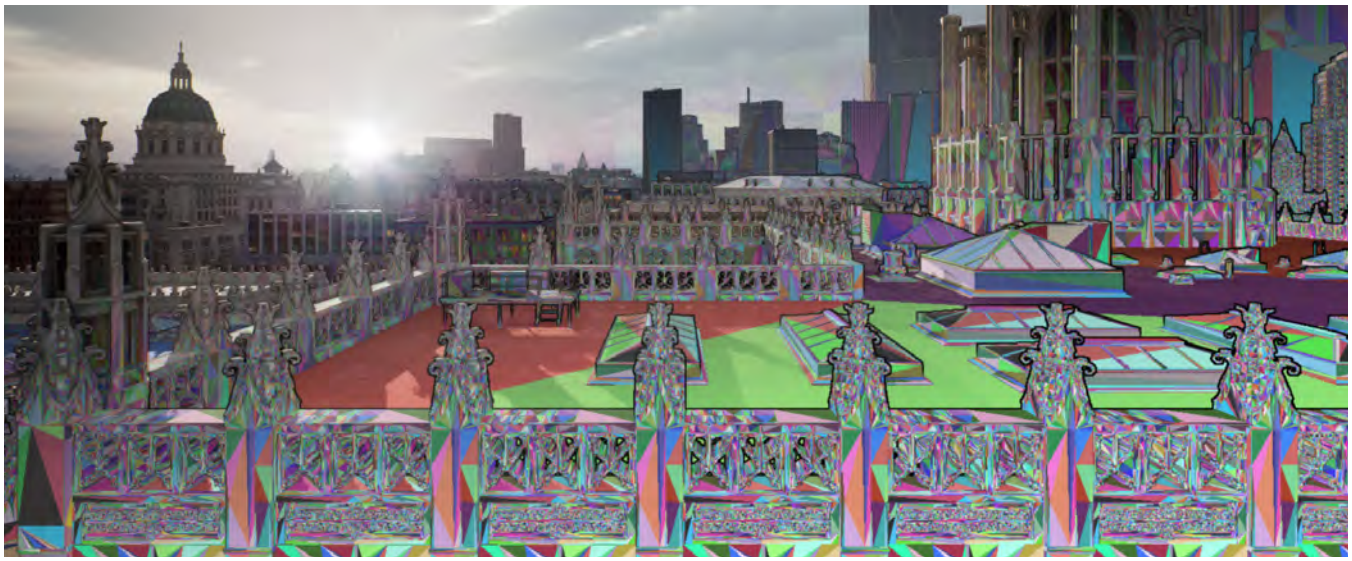
How does Nanite work?

Nanite is a core technology of Unreal Engine 5 that enables rendering extremely complex geometry, with automatic fine-grained streaming and level of detail through a highly optimized custom hybrid SW/HW rasterization pipeline.

Traditionally, game engines render each object in the scene one by one, starting with the most distant objects and working their way toward the camera. This process also requires the engine to render non-visible triangles, including those that aren't facing the player or are obscured by other objects. This introduces several bottlenecks. Specifically, the total number of objects and triangles that can be rendered in a scene.

Nanite uses a novel rendering approach that leverages unique features of modern graphics cards and software APIs to bypass the traditional per-object rendering method. Instead, Nanite renders the entire scene as a single, optimized mesh.

Unreal Engine 5 allows most existing or newly imported Static Meshes to be converted into Nanite, no matter how complex they are. Nanite meshes can then be placed in the world and used like any other Static Mesh in Unreal Engine.



City Sample showing Nanite triangle visualization.

Nanite Advantages

- Nanite meshes render faster and can use less memory and disk space than traditional meshes with baked normal maps. Thanks to the magic of compression, a one-million-triangle Nanite mesh only takes 14 MB of memory, while a single 4K normal map takes about 22 MB
- Nanite features a very steady performance profile that is more affected by rendering resolution than by triangle or scene object count.
- Even lower-polygon content can benefit from Nanite. More intelligent draw calls and automatic LODs can take some of the burdens off of the developer.
- Artists don't have to spend time optimizing and baking normal maps. Instead, they can import raw 3D models, sculpts, and photogrammetry scans directly into Unreal Engine and visualize them directly in the viewport at a very high frame rate.
- Level artists are no longer constrained by object or triangle counts, allowing them to lavish detail onto every surface without worrying about merging objects or unpredictable frame rates.

[Nanite Mesh vs. Baked Maps Mesh]

Nanite

Baked

Technical Limitations

- As of 5.0, Nanite does not support dynamic/deforming meshes, including skeletal meshes and meshes that use World Position Offset (WPO or vertex deformation).
    - NOTE: The primary focus of Nanite for 5.1 is the addition of a programmable rasterization framework, opening the door to features such as masked materials, two-sided foliage, pixel depth offset, and world position offset. **Please note that the exact feature list and expected stability and performance characteristics are still TBD**
- Nanite meshes cannot have materials with opacity masks or translucency, nor can they use two-sided materials
- There is no Mac OS support for Nanite, nor is any likely in the future due to graphics API limitations.
- Nanite meshes can show errors when used with ray-traced shadows.
- The Path Tracer does not support Nanite meshes.

Additionally, Nanite *currently* does not support the following:

- Split-screen and VR
- Forward rendering support
- Multisample anti-aliasing (MSAA)
- Custom stencil and depth buffers

Nanite is in the early stages of development, and while we anticipate solving many of these issues, we cannot provide guidance on timelines.

Other considerations

Nanite offers the promise of unlimited triangles and object counts. Because of this, there are some other practical considerations that you and your students should be aware of in order to plan your workflows and lessons accordingly.

3D applications

Applications like 3ds Max, Maya, and ZBrush can only effectively deal with a certain number of triangles.

For instance, meshes with over 1 million triangles tend to slow these applications down to the point where they are no longer usable. Until these tools evolve, they will often be the limitation to a model's fidelity.

Pipeline

Huge mesh counts also slow down the production pipeline considerably. They increase file sizes, slowing transfer times and compiling/cooking operations. Also, the more triangles a mesh has, the longer it takes to move from application to application and import into Unreal Engine, slowing down iteration.

Finding a balance

One of the most critical skills for artists using Nanite will be the ability to decide on the proper triangle and texture density for assets to balance fidelity with the above considerations.

A suitable method for determining triangle density is to try and aim for one triangle per pixel. Objects the end users interact with or get very close to can be made up of very dense meshes, while background meshes are optimized to improve performance and workflow.

Hardware and Software Requirements

Nanite runs on newer versions of Windows 10 (version 1909.1350 and newer) and Windows 11 with support for DirectX 12 AgilitySDK with the latest graphics drivers installed.

Support for Linux and Vulcan are forthcoming, but no timelines have been announced.

Using Nanite Meshes

Nanite meshes act just like any other Static Mesh in Unreal Engine (aside from the above-mentioned differences). They are typically created by hand or scanned and cleaned in an external application before being imported into Unreal Engine just like most other Static Meshes. You can also download high-polygon models from asset libraries like Quixel, TurboSquid, or Sketchfab.

You can create your own Nanite meshes in most 3D applications such as Blender, 3ds Max, Houdini, and Maya. Unreal imports both FBX and OBJ 3D interchange formats, which are supported by most 3D applications. Nanite supports meshes imported using Datasmith, allowing visualization designers to use high-fidelity design and CAD models in real-time.

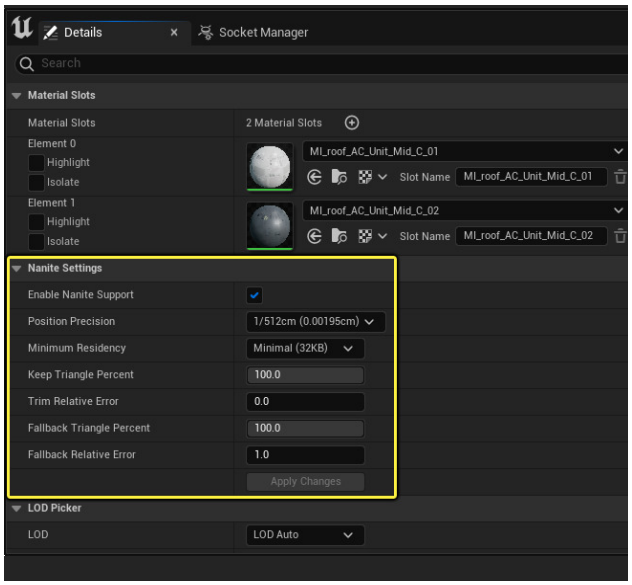
Enabling Nanite

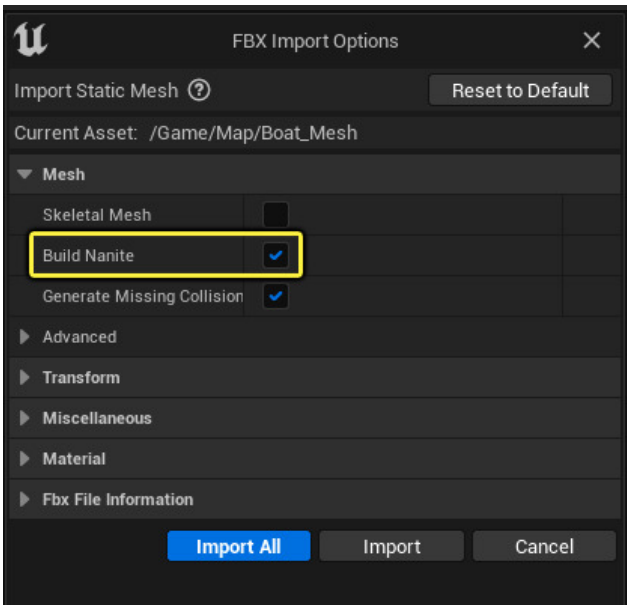
Unreal Engine supports Nanite as a core feature that simply needs to be enabled on individual assets to work. Nanite meshes can be enabled in two ways: converting existing Static Meshes and selecting Nanite when you import a mesh.

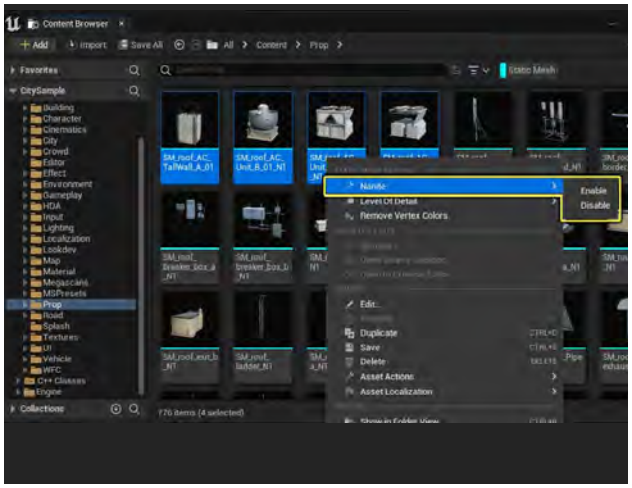
Converting existing static meshes

You can enable or disable Nanite in the static mesh/geometry collection editor by selecting Enable Nanite Support in the Details Panel, or by simply right-clicking one or more static meshes in the content browser.

Note: When you convert existing static meshes to Nanite, they will no longer use existing LOD models and will use the Fallback Mesh for per-polygon collisions, which may change gameplay or visual fidelity.







Importing a new mesh as a nanite mesh

When importing a Static Mesh to the content browser, an option is available to build Nanite. This will enable Nanite on the newly imported mesh.

Collision

Like traditional Static Meshes, Nanite meshes still use low-polygon representations (convex hulls) and analytical shapes (spheres, boxes, and so on) for physics simulations.

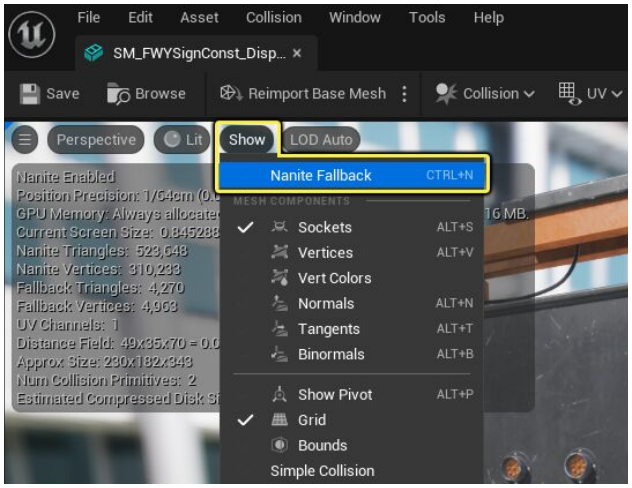
For per-polygon complex collision, UE5 uses the Nanite Fallback Mesh. If this doesn't provide an appropriate result, you can set custom collision LODs if necessary using the traditional LOD method and the *LOD for Collision* property in the general settings of the mesh.

Note: If using this method, set the *Fallback Triangle Percent* to 0 in the Static Mesh Editor.

Nanite Mesh

Nanite Fallback Mesh

Note: You can view the Nanite Fallback Mesh in the Static Mesh Editor by using the Show menu in the Static Mesh Editor viewport.



UV coordinates

Working with UV coordinates on very high polygon meshes becomes burdensome quickly, creating friction in the development pipeline. 3D tools are not yet fully capable of working with dense geometries.

Consider looking at projection-mapping techniques, proceduralism, and other methods that can skip or enhance UVW unwrapping methods.

Virtual Textures

While not specifically a feature of Nanite, Virtual Textures (VTs) allow developers to use huge texture resolutions in Unreal Engine. Without Virtual Textures enabled, Unreal Engine supports a maximum resolution of 8K for textures, while Virtual Textures allows resolutions of hundreds of thousands of pixels on each axis. Additionally, VTs unlock additional Landscape workflows and can reduce overall memory usage versus the traditional texture streaming method.

Even with Virtual Textures enabled, we strongly suggest working with 4K textures for the vast majority of assets. Working with larger images can quickly become burdensome. Large images take up a lot of disk space, are slow to transfer, and can makemost texture editing and painting applications become unresponsive.

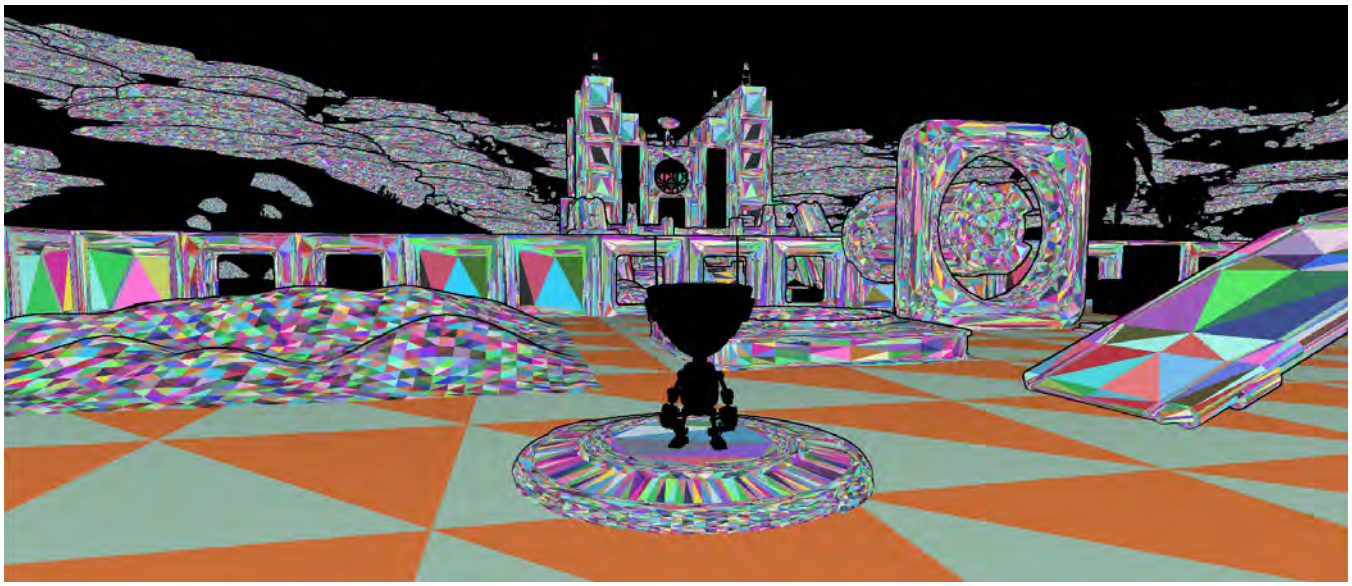
Teaching Guidance

Nanite all the things

We recommend enabling Nanite on any mesh that will support it. That way, your meshes render faster, while using less memory and disk space. For meshes that Nanite does not support, fall back to regular Static and Skeletal Mesh workflows and techniques. Nanite Meshes and non-nanite meshes work together seamlessly within a scene.

For example, the objects in the scene below that rely on features not supported by Nanite, such as the Landscape and Skeletal Mesh character are rendered using more traditional methods. In the Nanite visualization, they are shown as black silhouettes.

Lit Scene View



Nanite Triangles View

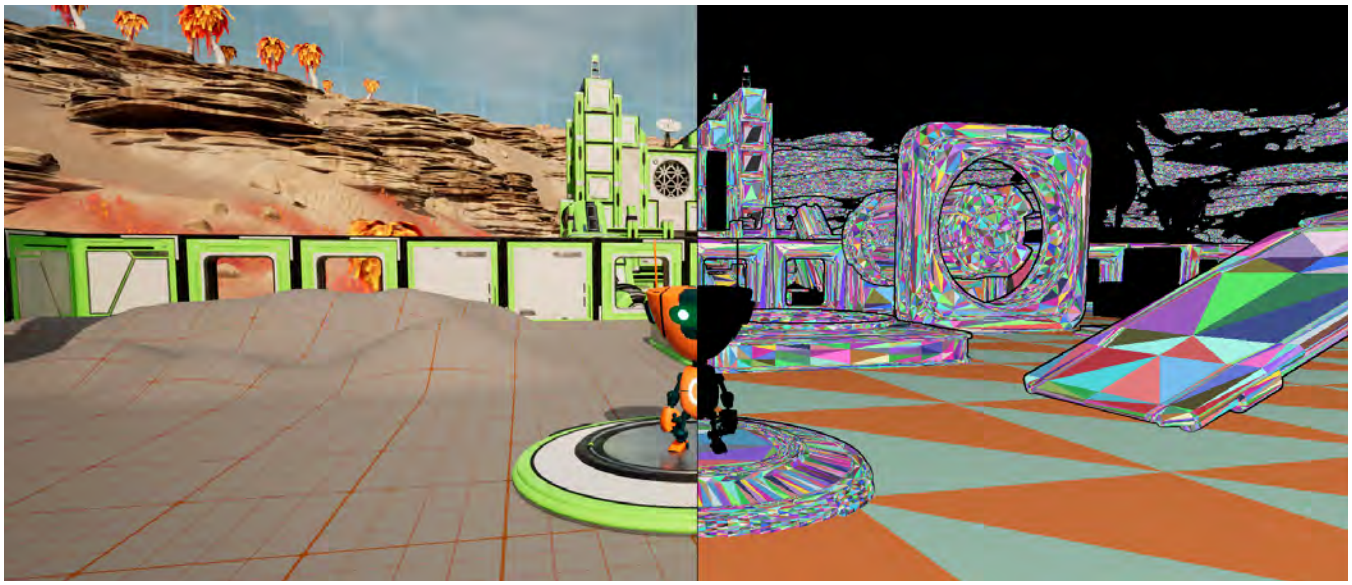


Image of profiler showing nanite and base pass

Keep teaching texture baking

Nanite removes the arduous task of creating low-polygon, baked versions of sculpts, subdivided meshes, or the LOD chains typically required in real-time applications. However, the process of encoding data to textures such as curvature and depth. is still essential to the 3D artist's toolset. For example, the character art pipeline remains largely unchanged with the introduction of Nanite.

While students may not be baking normal maps projected from a high-poly model to an optimized one for their environment meshes, they will still be using tools such as Substance Painter and Quixel Mixer that utilize maps such as ambient occlusion, normal and curvature maps to create surface textures and materials.

Explore new workflows

Nanite meshes have some limitations that require a lot of creative problem solving to overcome. The following are examples of challenges that developers will need to tackle.

- **Nanite does not support per-instance vertex color painting:** Vertex painting to drive material variation has been an important workflow in real-time graphics for over a decade. How do we add artist-driven variation to meshes?
- **Nanite does not support skinned meshes:** However, Nanite meshes can be attached to bones/skeletons. How can we use Nanite meshes on characters and what are the tradeoffs?
- **Nanite makes multi-platform support more complicated:** Developing solid workflows that allow a project to scale from low-end/mobile hardware to next-gen consoles will be a valuable skill for many studios with multi-platform titles.
- **High-poly workflows can take some getting used to:** Developing new habits will be needed to keep projects on schedule. Make sure to unwrap meshes in the low-poly phase to save time and consider different texturing methods like polypaint to use the strengths of these more detailed meshes.

Advice for Students

To reach the broadest audiences, studios are now developing multi-platform hybrid games that run on the current-gen consoles (PlayStation 5 and Xbox Series X), PC, and less powerful platforms like PlayStation 4 and Nintendo Switch.

As the adoption of current-gen consoles increases, studios are beginning to start developing and hiring for their truly next-gen projects that can leverage all the features of Unreal Engine 5.

Recent graduates and seniors

Students planning to enter the workforce in the near future should be aware that Nanite is not yet an industry-standard method for developing 3D assets. Students should still be proficient in traditional high-to-low polygon methods, optimizations, LODs, and so on. However, they should be aware of Nanite and the worldbuilding methodologies of Unreal Engine 5.

New students

Those that are just beginning their studies should assume that in 2–3 years, the adoption of Nanite and the use of Unreal Engine 5 will grow considerably. The need to understand and apply robust high-polygon workflows to art and worldbuilding will become increasingly important.

Topics such as photogrammetry, digital sculpting, and proceduralism will increasingly become part of the 3D artist's toolset. Graduates with robust skills in these areas will have an advantage over their peers.

Current students

Those that are already studying 3D asset production can begin to explore Nanite right away. This is especially true if you are already generating high-fidelity models from sources such as CAD, photogrammetry, or sculpting applications. Students targeting AAA studios should consider working with Nanite and the next-generation features of Unreal Engine 5.

There are, however, many studios that will continue to develop using game engines that do not support a Nanite-style workflow for many years to come. It will be important for you to look at the studios you want to work for and determine if they will be adopting these types of technologies in the near future. Studios specializing in mobile, VR, or Games-as-a-Service (GaaS) will likely be bound to traditional development techniques for some time to come.

Additional software and topics to study

Digital Sculpting

Tools such as Blender, ZBrush, and Mudbox have unleashed an explosion of incredible digital art. Until now, rendering these creations in real time was an arduous task. You needed to convert the high-polygon mesh to an optimized, low-polygon model using a process called texture baking. This process is technical, time consuming, and prone to error. It also has an outsized impact on iteration between a 3D application and a real-time engine, requiring a re-bake of the textures with each iteration.

Artist sculpted 3D Model rendered in Unreal Engine 5 using Nanite.

Nanite allows artists to import their sculpts directly into Unreal Engine without any optimization. The image below shows a statue sculpted in ZBrush containing around 100 million triangles. There are hundreds of this asset duplicated around the level and Nanite renders the scene with ease.

Photogrammetry

Using photographs to scan and capture real-world objects and surfaces as three-dimensional digital meshes is not a terribly new technique. Applications such as RealityCapture and ReCap have been available for some time and artists, historians, researchers, and more are using photogrammetry to capture hundreds of thousands of richly detailed 3D models.

Quixel Photogrammetry expedition to Greenland

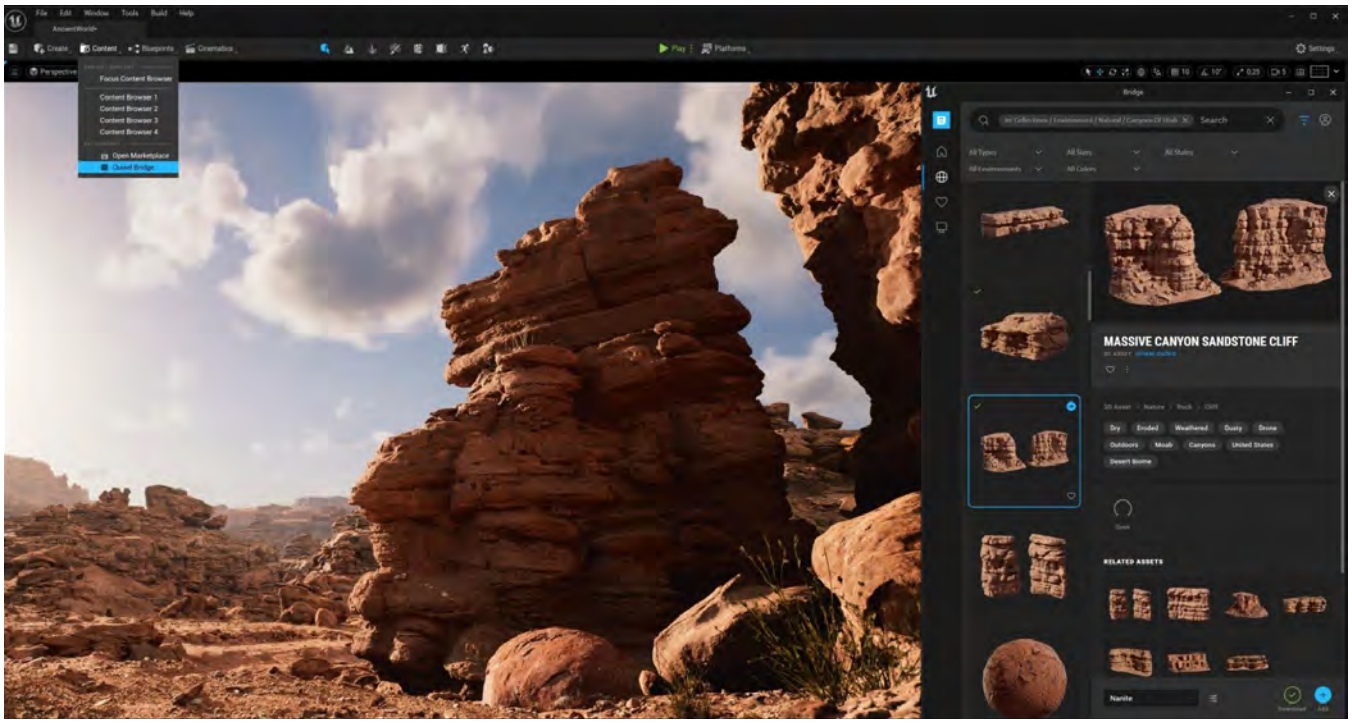
Nanite removes the need to heavily optimize these kinds of models, allowing developers to use hundreds of photogrammetry scanned meshes to create environments with never before seen levels of detail and immersion.

Sketchfab product image

Asset Libraries

The vast majority of today's game, film, and visualization studios are using asset libraries such as Quixel Bridge, Sketchfab, and the Unreal Engine Marketplace to help them populate their worlds. The level of detail and the vast number of different assets needed to create a single, next-generation level or rendering can be too much for all but the largest teams.

The ability for artists to find high-quality materials from these marketplaces and customize them to fit their studio's particular style or other characteristics is quickly becoming as essential of a skill as raw modeling or sculpting prowess.



Quixel Bridge in Unreal Engine 5

Teaching Resources

While Nanite is brand new, there is already a lot of documentation, official tutorials, and community resources that will help you bring Nanite into the classroom.

Official Documentation

Unreal Engine 5 Documentation

Nanite Documentation

Virtual Shadow Maps Documentation

Stack O Bot

Example Projects

Stack O Bot

This complete UE5 example game project demonstrates many of the next-generation features of Unreal Engine 5 such as Lumen global illumination, Control Rig procedural animation, and of course, Nanite's virtualized geometry. This is a great base for those learners who are looking to get started making less complex experiences but could use a great base to work from. Download the Stack O Bot project from the Unreal Engine Marketplace.

Unreal Learning Kit

Unreal Learning Kit

To help students better understand the entire game development pipeline, we are proud to present the Unreal Learning Kit. This pack of lightweight, whimsical assets - including source 3D and texture files - allows you to explore building content in 3D applications, importing it into Unreal, and iterating on it until it's perfect. Download the project from the Unreal Marketplace here.

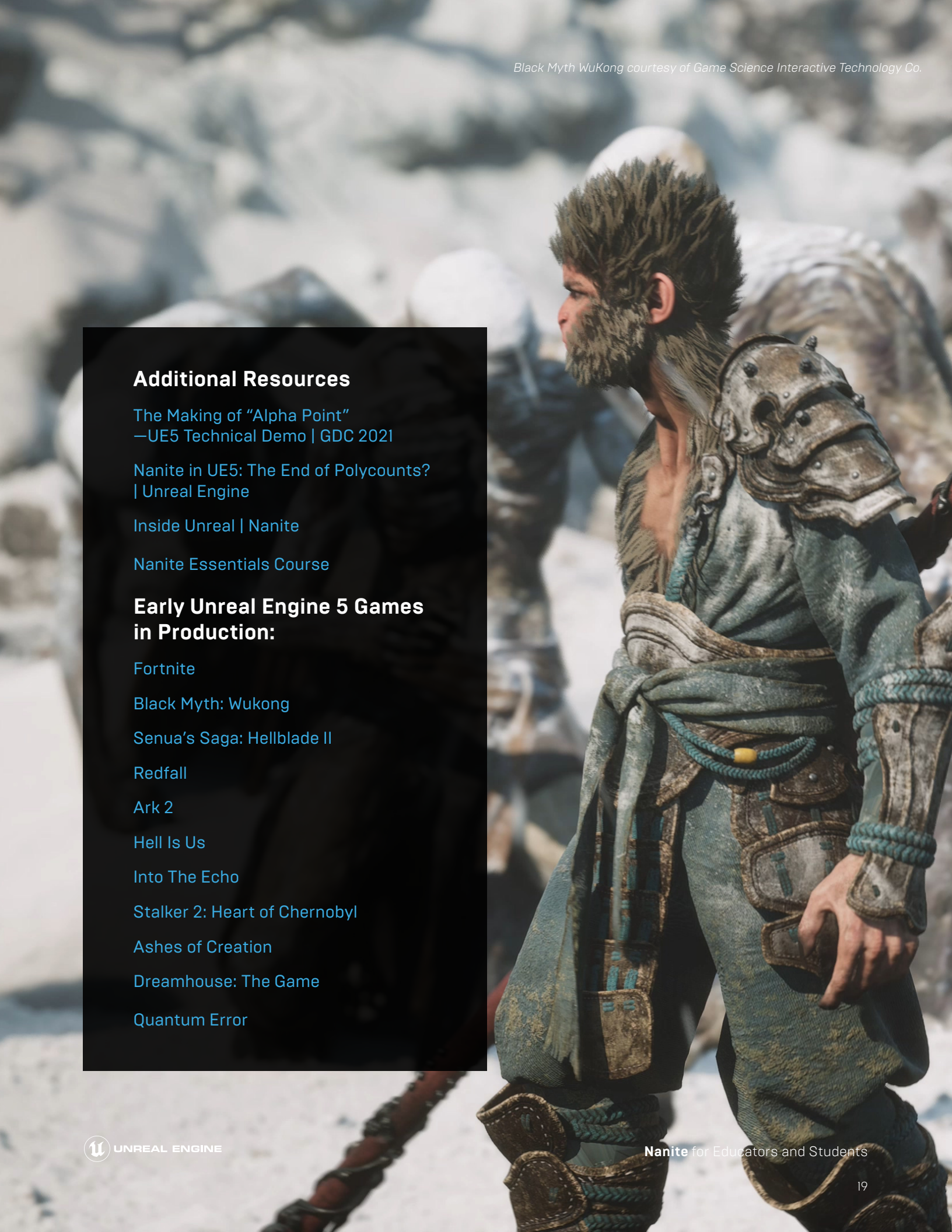
We've also developed a series of lesson plans and other learning content to help you bring Unreal into the classroom find out more here!

City Sample

City Sample

For an example of a next-generation open-world gaming experience, download the City Sample project. This project demonstrates what the top creative and technological minds in gaming and feature films can make with Unreal Engine 5. Then, learn how to build massive, hyper-detailed worlds with unparalleled realism and detail with the tutorial series!

If the full City Sample is too much to download or run, consider downloading the City Sample Vehicles or Buildings packs for more specific examples of how Nanite was used in the demo.



Black Myth WuKong courtesy of Game Science Interactive Technology Co.

Additional Resources

The Making of "Alpha Point" —UE5 Technical Demo | GDC 2021

Nanite in UE5: The End of Polycounts? | Unreal Engine

Inside Unreal | Nanite

Nanite Essentials Course

Early Unreal Engine 5 Games in Production:

Fortnite

Black Myth: Wukong

Senua's Saga: Hellblade II

Redfall

Ark 2

Hell Is Us

Into The Echo

Stalker 2: Heart of Chernobyl

Ashes of Creation

Dreamhouse: The Game

Quantum Error

In Conclusion

Nanite represents a major improvement in rendering fidelity and performance. It removes some of the most serious limitations that have separated interactive games and visualizations from offline rendered scenes, allowing artists and designers to achieve their vision without having to tackle complex technical limitations.

In the short term, students will need to learn to work with both traditional low-polygon rendering, and high triangle meshes. There will be platforms and games in development that will need those skills immediately.

In the long term, however, if students are targeting AAA or feature films, they should begin utilizing Nanite and all the other next-gen features of UE5 as soon as they can. Students in Architectural, Automotive, advertising, and other programs can also begin exploring Nanite. As of Unreal Engine 5.0, these workflows have not been fully vetted but are still showing very promising results.

Finally, this is a brand new technology. Expect significant improvements in performance, compatibility, and workflows in future versions of Unreal Engine 5.

More information about Nanite and Unreal Engine 5 for Educators can be found here:
https://unrealengine.com/educators

EPIC GAMES

UNREAL ENGINE