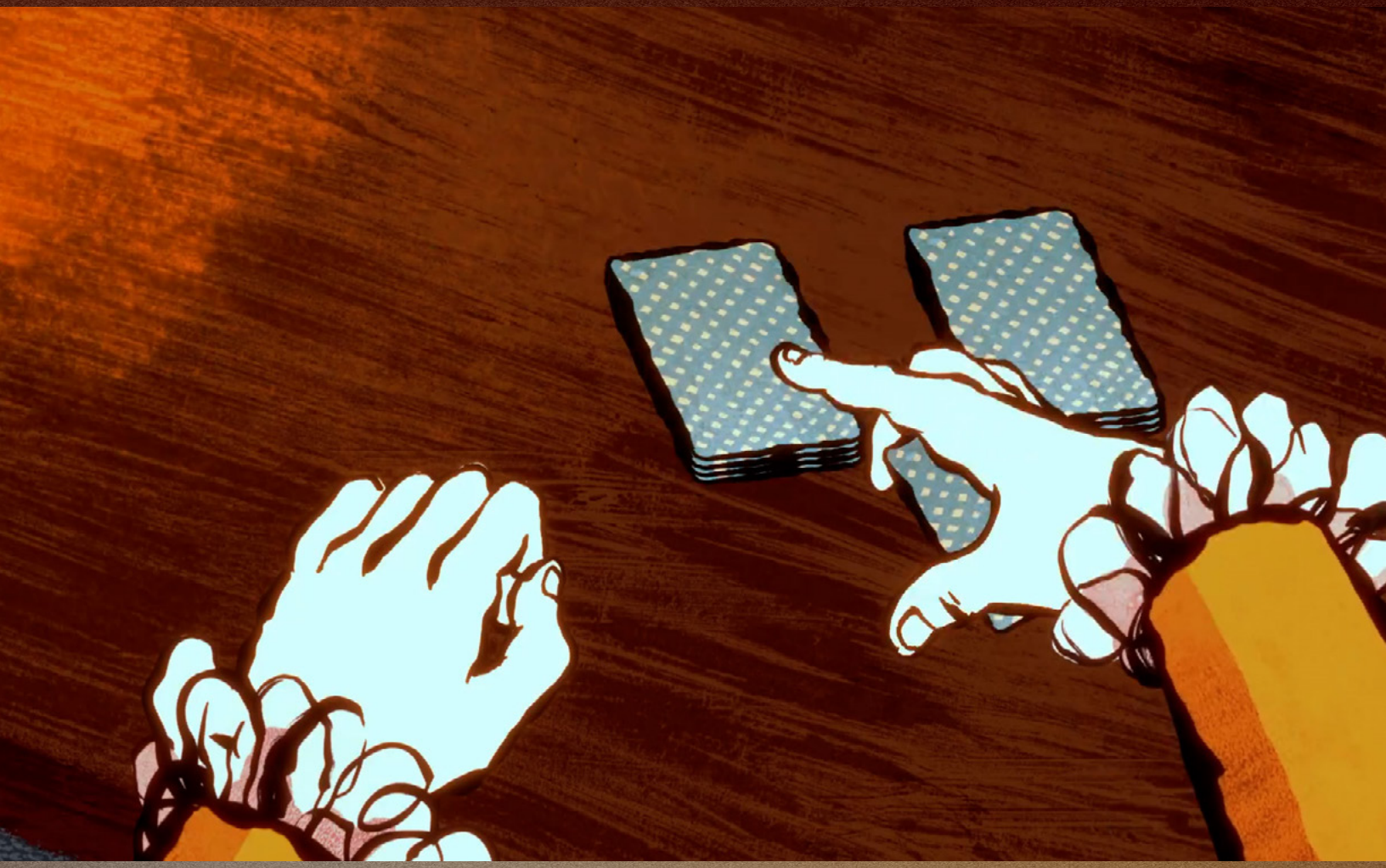


NERIAL'S *CARD SHARK*

USED UNITY'S 2D TOOLS TO BRING AN ARTIST'S VISION TO LIFE

Unity's multiplatform features make porting simpler – without forcing performance compromises.

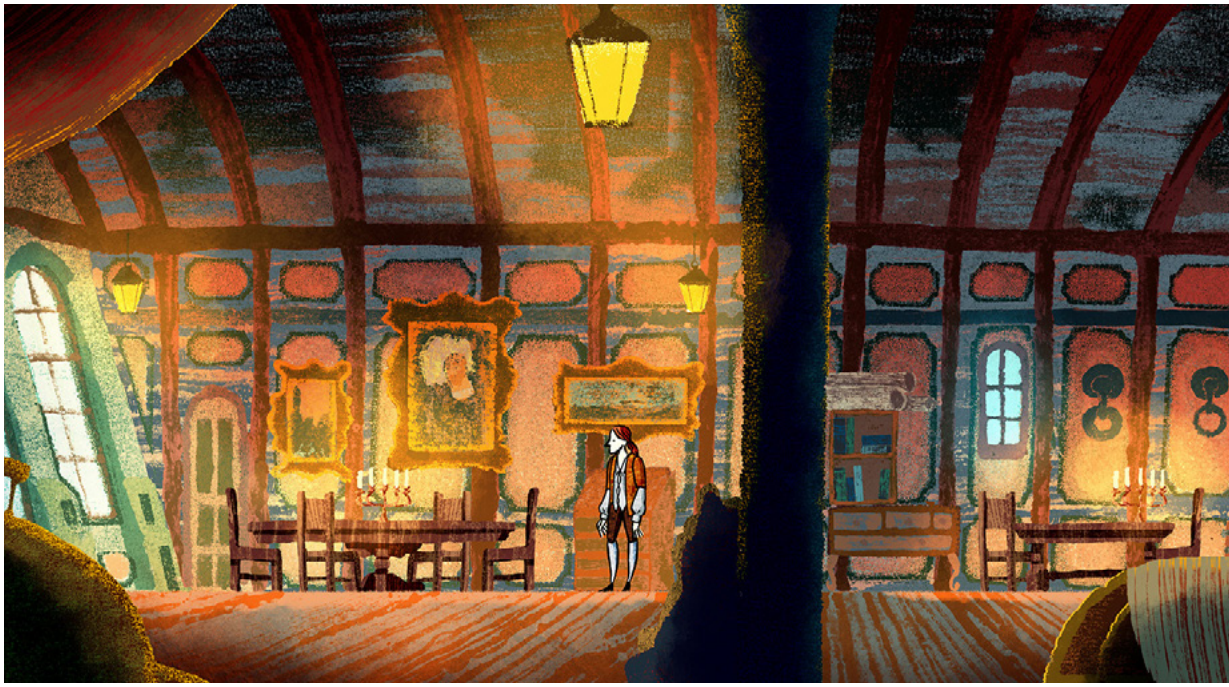
The challenge
Inputting high-quality hand-drawn art into a 2D game quickly

Platforms
PC, Switch

Project Staff
10

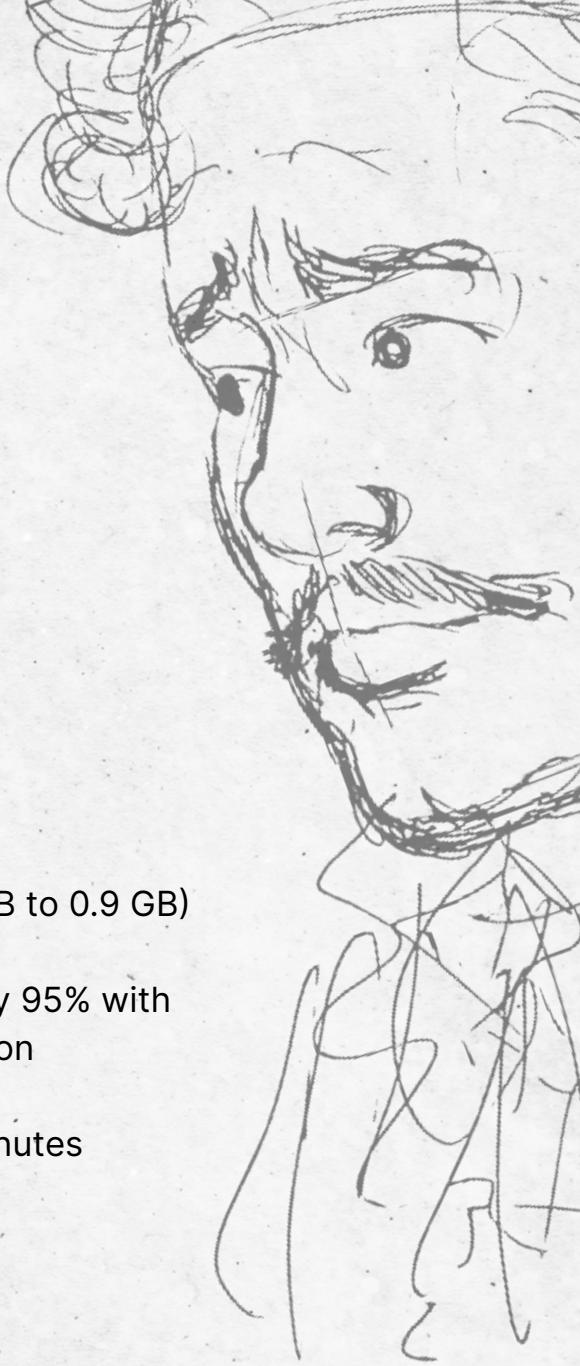
Location
Paris, France
London, U.K.
Madrid, Spain
Bangkok, Thailand

How does a boutique indie showcase a noted artist's hand-drawn style within an immersive 2D multiplatform game? Nerial's *Card Shark* explores card tricks and chicanery among the royalty of eighteenth-century Europe through carefully crafted monoprint artwork, all imported, layered, optimized, and brought to life in Unity. Creating with handmade graphics required some trickery from Nerial's small team, like loading highly complex 2D assets built with 4K textures and bringing technically disinclined artists into the production process. Fortunately, Nerial had an ace up their sleeve: Unity.



Choosing the right platform for a gorgeous 2D game

Sophisticated game development platforms make it easy to achieve a creative vision by taking simple ideas and transforming them into engaging graphics. This ability is priceless for developers who are strong on coding and gameplay but less so on design. Nerial, on the other hand, had a rich source of extraordinary art, and they chose Unity Pro to bring it to life in a 2D game that looks great on PC and Nintendo Switch.



The results

— Cut build size by over 60% (from 3 GB to 0.9 GB)

— Reduced narrative production time by 95% with a sophisticated collaboration extension

— Built stunning transition scenes in minutes

Betting big on an artist's vision

François and Tamara Alliot founded Nerial in 2013 and, after releasing a number of innovative 2D games primarily for Steam on PC, won big with *Reigns* in 2016. They quickly followed up with three *Reigns* sequels and a *Reigns*-themed board game. For *Card Shark*, Nerial expanded the team, collaborating with artist Nicolai Troshinsky, who would serve as the game's lead artist, concept designer, and animator.

Troshinsky is a well-known illustrator and animator based in Madrid, whose work is heavily influenced by '80s-style Eastern European and Russian animation. He's also a serious hobbyist – in addition to being one of Spain's top pinball players, he's been studying the history of card tricks. Troshinsky had long dreamed of creating a game exploring this history by teaching players card tricks that they could practice in real life.

The challenge was to create a fluid, tactile experience that faithfully represented his artistic vision and style without compromising on visuals or gameplay. Troshinsky knew he'd need to work with a studio to execute his idea – and, as fate would have it, his neighbor in Madrid happened to be Nerial's creative director, Arnaud de Bock. During their conversations, de Bock learned more about Troshinsky's vision, and the fit for Nerial's next game became obvious. In 2018, they began work on *Card Shark*.

Adding movement to bring static images to life

Troshinsky made all of *Card Shark*'s characters and backgrounds by creating paper cutouts and monoprints for animation frames. For most studios, producing every raw graphic element by hand would be considered grossly inefficient and costly. For Troshinsky, it was a labor of love. After he built and scanned every visual element in the game, his image catalog had over a thousand individual components.

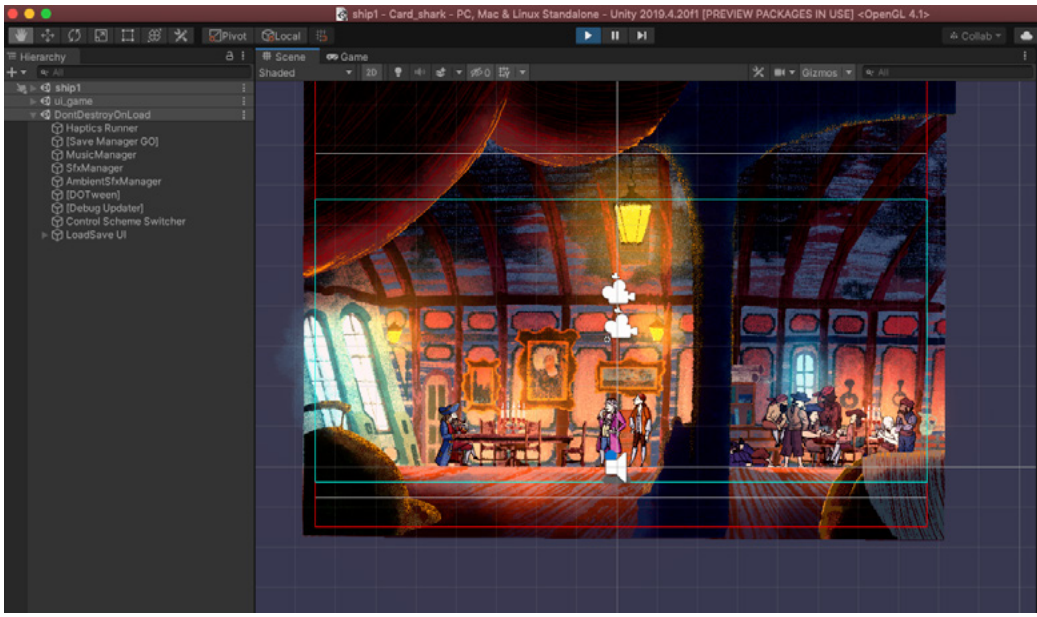
One of Troshinsky's most arresting backgrounds is a harbor with the setting sun shimmering off a rippling sea. To make the transition scene, he wanted to add movement to his original painting – beyond NPCs walking around in the background – so he added waves as a tiled texture and used particle effects to reflect light. "The Unity 2D Particle System rendered my little globs of paint, and it only took a few minutes to set up the entire water texture," he says. "It was very easy."

Handling multiple draw calls with 2D tools

Considering the number of layers appearing in *Card Shark*, Sprite Atlas was an invaluable resource for Ben Obikoya, lead programmer at Nerial. In any game with many textures, multiple draw calls can become resource intensive and reduce performance. Sprite Atlas enables a single draw call that accesses packed assets all at once to free up memory.

Card Shark's background scenes are far more complex and detailed than they might first appear. Each is composed of several layers of stylized high-resolution art. Because it's all hand-drawn, the source artwork is extremely large and awkwardly sized – often long, thin, and several thousand pixels along each axis. Even after reducing the size of the textures, the aspect ratio prevented Nerial from using texture compression without a lot of rework.

The playing cards are also incredibly detailed, and they presented the same challenge as the backgrounds in that they are hand-drawn and don't conform to the requirements to enable texture compression. Nerial sometimes manipulated the textures programmatically, which can take a few seconds at the start of a scene when loading the raw textures in their native size.



"It's great to have the confidence that Unity is going to do my artwork justice."

– Nicolai Troshinsky, Lead Artist, Concept Designer, Animator, *Card Shark*

Reshuffling textures to cut down on build size

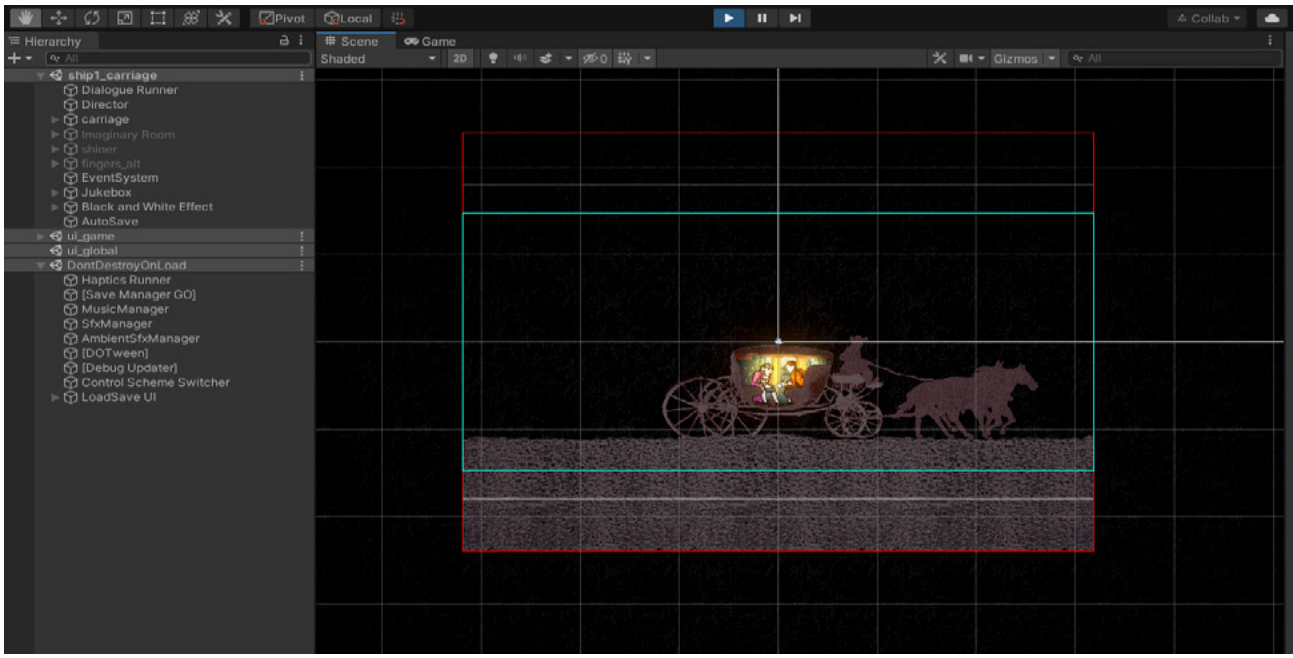
By tweaking texture import settings, Nerial was able to easily balance each texture for memory size and quality before using Sprite Atlas to combine all the background assets for one scene into a few textures. The resulting Sprite Atlases conform to compression requirements, allowing Nerial to bring the size down dramatically.

“We put all the assets, such as 52 card sprites for deck scenes, into a folder, and Sprite Atlas packed and compressed them into a manageable volume,” says Obikoya. “Our build size was starting to approach 3 GB when we started taking a serious look at our import settings and Sprite Atlas. Afterwards, it went down to around 0.9 GB.”

The Unity Profiler and Memory Profiler also helped Nerial optimize builds by displaying, hierarchically, any elements that were hogging memory. Another optimization aid was the Build Report Inspector, currently available in Preview. “Build Report helped us track how much time was going into our builds and how much storage we used per build,” says Obikoya. In both cases, the tools helped Nerial gauge, per target platform, which assets needed to be compressed and by how much.

“Unity makes it really easy for artists to get more deeply involved in development, and I extended the Editor for the exact functions we needed for Troshinsky’s art.”

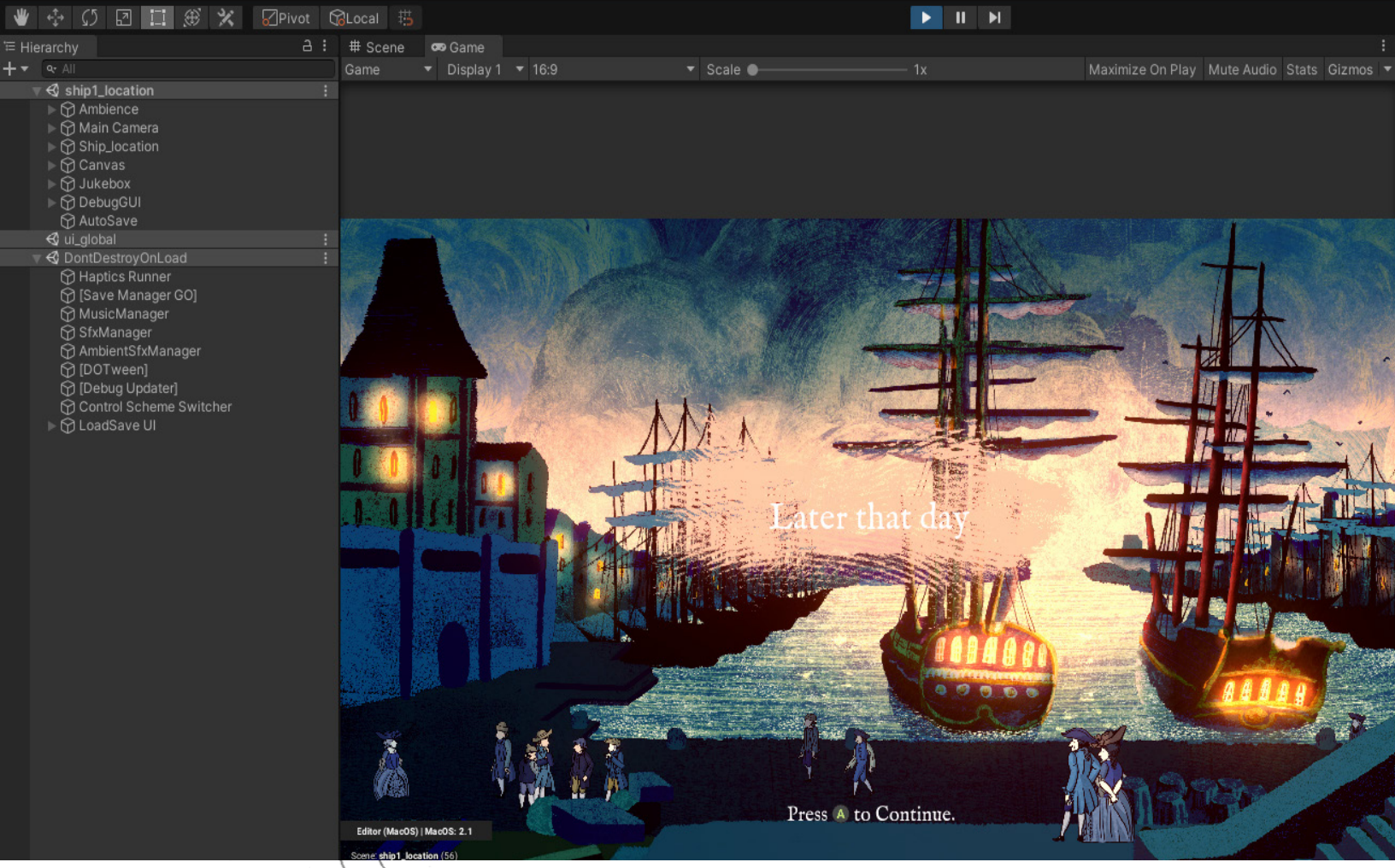
– Ben Obikoya, Lead Programmer, Nerial



Extending the Unity Editor for the studio’s unique needs

To help Troshinsky closely orchestrate animations in Unity with the story narrative and work independently, Obikoya extended the Unity Editor for Editor Windows and Property Drawers. “I wanted Nicolai to be able to make substantial changes himself without always having to explain exactly what he wanted to an engineer,” says Obikoya.

“Unity makes it really easy for artists to get more deeply involved in development, and I extended the Editor for the exact functions we needed for Troshinsky’s art.” Obikoya’s extension was a node graph that empowered Troshinsky to modify and drag-and-drop partial or entire sequences. Working with Alliot for stage direction, the process saved Nerial an astonishing 95% of production time, more than making up for the extra work from layering so many hand-drawn images. Troshinsky adds, “It’s great knowing that Unity is going to do my artwork justice.”



Ensuring a consistent control UI

Card Shark teaches tricks via mini-games that provide specific instructions on which button to push, when. To speed up the process of generating different instructions for different platform controls, Obikoya used Unity TextMeshPro, a text styling and texturing tool. He says, “TextMeshPro has great systems for showing different glyphs and characters, I could easily add sprites for all the control systems we support.” Nerial also used DoTween, an open-source tweening tool available in the Unity Asset Store, to enhance transitions with the card tricks and create a fluid gameplay experience.

Going all-in on collaboration

Troshinsky's art and animations are labor intensive, but the Nerial team was able to use Unity to define workflows that eased and accelerated the production process. The lead artist could guide *Card Shark*'s look and feel to realize his vision – without overly impacting the technical team.

This level and quality of collaboration is what made *Card Shark* possible. Any extra time it took for the artist to physically draw and paint an elaborate image was more than made up for by the ease-of-use of the Unity platform and Nerial engineers' expertise. Plus, magic and tricks are fun – and fun may be the strongest driver of collaboration out there.

Obikoya and Troshinsky certainly felt that way. "I learned card tricks, I learned how an artist thinks and feels, and together with Unity we made a really fun game," says Obikoya. According to Troshinsky, "It's a blast. I get to imagine things, paint and draw them, and the developers magically turn it into something that others can experience. It's beautiful."


Unity®