A 43-SECOND SHORT FILM / MAKING OF

Party Delivery

A cherry-red delivery robot races a balloon through a rainy Tokyo alley, finds the party cancelled and gives the balloon to a lucky cat. Built and rendered in Unreal Engine 5.8 by Claude Code, with every model, texture, sound and note of music generated with Scenario.

43.5 s
final film, 16:9 and 9:16

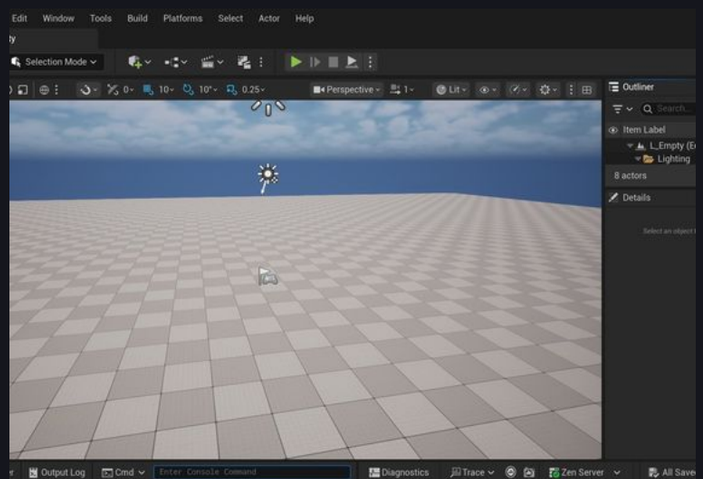
~10 h
of active work over 2.8 days

59
final Scenario generations

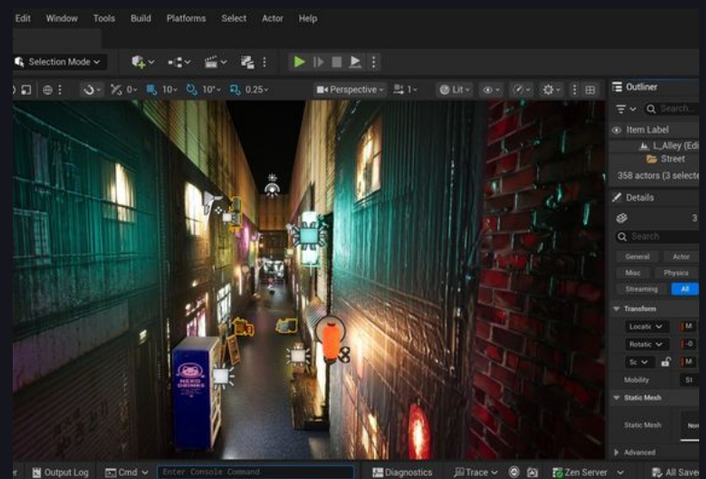
3,019
credits (CU), final 3D and images

An engine, a plugin, one config line.

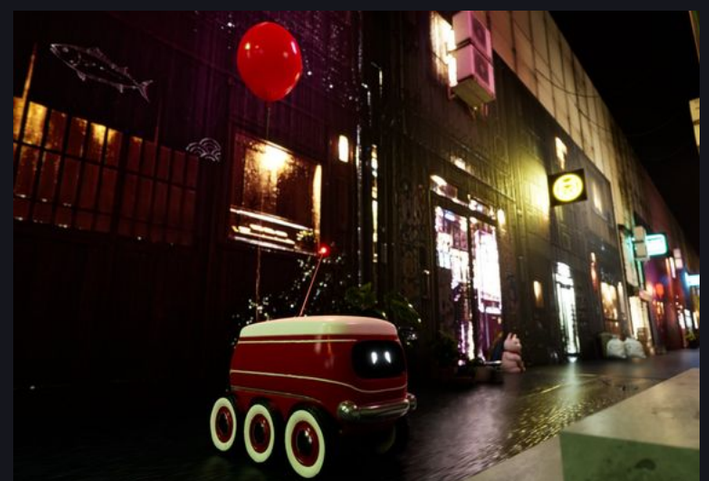
Party Delivery was made with Epic's official Unreal MCP. Claude read the docs, installed the Metal toolchain, enabled the plugin and connected to its server on port 8000. The film starts from an empty level.



The start: an empty level, L_Empty.



The alley taking shape in the editor.



The finished street, rendered with Movie Render Queue.

Getting the server to start

The plugin had to be "All Toolsets", not "MCP Client Toolset". The Auto Start option was missing from Editor Preferences on this build, so it went into EditorPerProjectUserSettings.ini by hand. Quitting the editor dropped that key, so from then on the editor launched with -ModelContextProtocolStartServer. Once connected, 55 toolsets were on offer.

Python, Sequencer and the render queue

Most of the build ran as Unreal Python scripts, and the film is a Level Sequence rendered with Movie Render Queue. Both the Python and the render plugins were switched on in the project file before the first shot.

A concept, a model, six wheels that turn.

Four robot directions, then four mixes of the two favourites, all with GPT Image 2.5. Mix C won: a 1960s cherry-red delivery cart with an LED-screen face. The flag and antenna were edited out, and the screen was blanked so Unreal could draw the eyes.



GPT Image 2.5 Flare: the chosen concept, Mix C.



Meshy 7.1 image-to-3D, the model in the film.



The sticker, raycast onto the body: "I was just a prompt once."

Rigid parts, no skeleton

The robot has no bones. Meshy made the body; Tripo Segmentation split off the six wheels so each one can spin as its own actor. The eyes are an SDF material drawn in HLSL on a screen cut into the body, behind a glass and a gasket written as OBJ meshes.

A balloon that feels like latex

The red balloon is a procedural teardrop with an opaque subsurface material. Clear, glossy and thin-translucent versions all read as glass. Its swing is a small Verlet simulation, solved offline in Python and keyed into Sequencer.

Flat fronts, four real buildings.

The alley is about 60 m of facades. GPT Image 2.5 painted the shop fronts as flat pictures. Four the camera passes closest were redrawn at a 3/4 angle and became real 3D buildings with Hunyuan 3D Pro 3.1, and so did the front door. The props came from Meshy 7.1 and Tripo v3.1.



GPT Image 2.5: an izakaya front used as a flat card.



The kissa front, redrawn in 3/4 view for Hunyuan 3D.



The 20 Scenario 3D models in the film (Blender turntables).

Placed by a script

Every facade, building, prop and light is positioned by Unreal Python (street.py to street5.py, dress.py). The final level has 358 actors. Building depths were measured from the mesh in Blender, because fronts are hard to place flush from estimates alone.

Fixes the eye caught

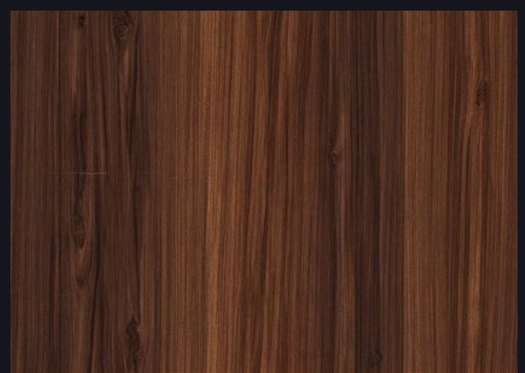
A bin sat inside a wall, and the AC units were turned 90 degrees, because Tripo and Hunyuan meshes don't share one "front" axis. A real drinks brand on the vending machine was replaced with a fictional "NEKO DRINKS". The facade with stairs "looked really bad" and was hidden behind a 3D building.

Brick, wood and paint, all generated.

PATINA made the tiling PBR materials: brick, walnut wood and concrete. GPT Image 2.5 drew the graffiti on transparent backgrounds, the "Party cancelled" sign and the sticker. Normal and roughness maps for the flat facades were derived locally, so wet light catches them.



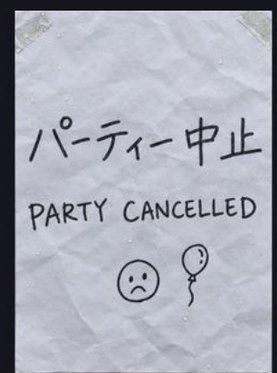
PATINA: brick colour map.



PATINA: walnut for the door.



GPT Image 2.5: transparent graffiti.



GPT Image 2.5: the pinned sign.

Built as material graphs in Python

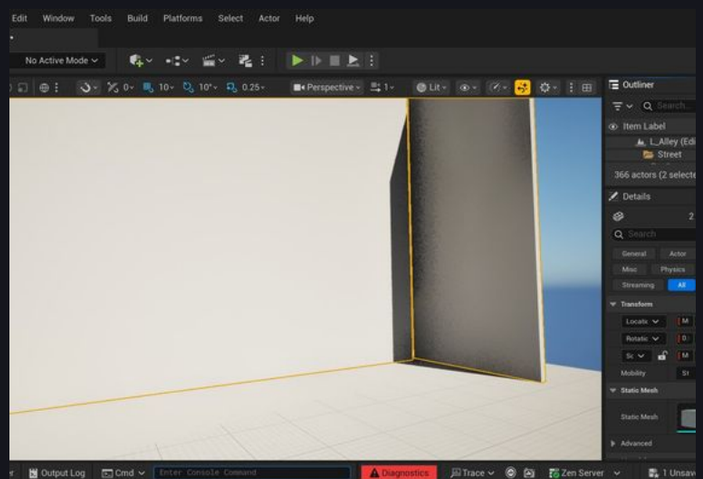
M_FacadePBR, M_NeonMaster, M_LatexSSS, M_LanternGlow and the rest were created node by node in Unreal Python. Two traps: a vector parameter is four floats, so it must be masked to two before scaling UVs, or the material silently falls back to the grid. And a lantern glow above about 0.9 clips to flat yellow.

Neon and wet ground

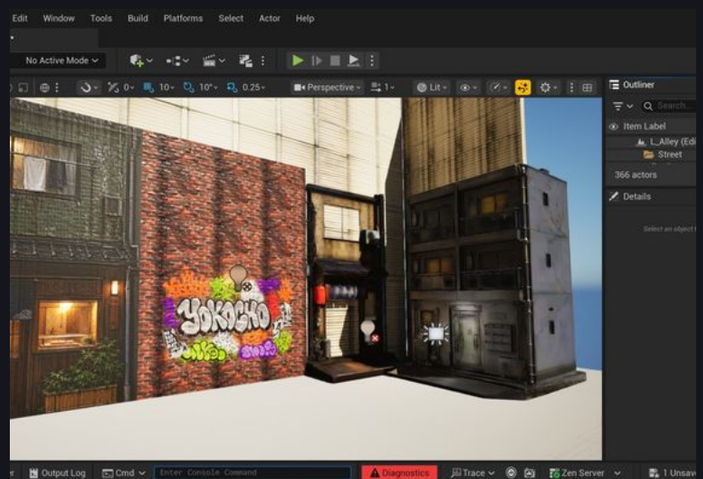
Fourteen neon signs were cut from GPT Image sheets and given emissive materials. The rain look comes from roughness on the asphalt and Lumen reflections; translucent meshes needed Nanite turned off.

Typed into the console, one script at a time.

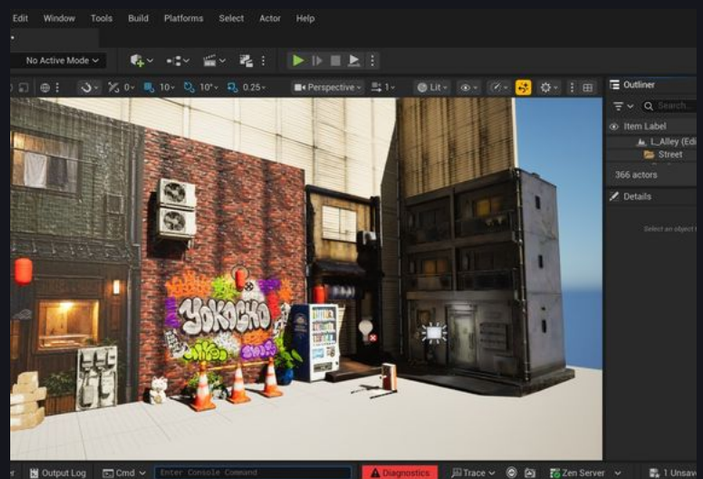
Python remote execution found no Unreal node on macOS. So Claude ran its scripts by typing py "file.py" into the editor's console box, through the MCP's SlateInspector tools. 81 different scripts ran 149 times this way. The time-lapse on the page replays one stretch of the alley, piece by piece, under a work light.



Time-lapse: the bare wall and ground first.



Then the textures, the fronts and the 3D buildings.



Neon, lights and props last.

What the MCP did, and what Python did

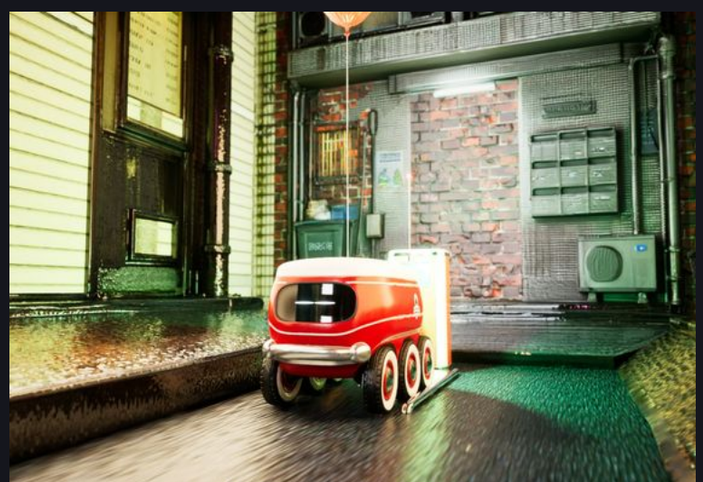
The MCP tools inspected actors, ran ray traces, moved cameras and captured the viewport. Importing GLBs (the MCP import takes FBX or OBJ only), building materials, Sequencer and the render queue all ran in Unreal Python. Even enabling the Python and MRQ plugins meant editing the .uproject, because SetPluginEnabled did not save.

The keypress that emptied the street

To clear the console box, the helper pressed Ctrl+A then BackSpace. Once, the box wasn't focused: the keys reached the viewport and the level dropped from 358 to 149 actors; a modal then froze the MCP server. The level was reloaded from disk without saving. Since then the helper only types, never presses keys.

Every frame planned in a file.

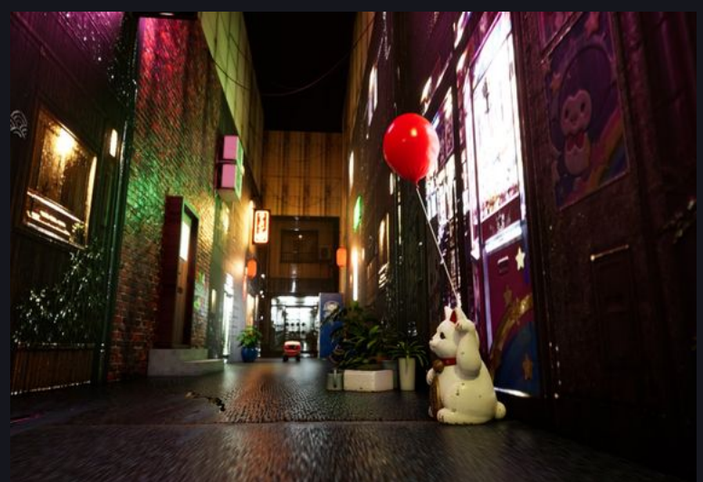
choreo3.py writes the whole film as data: the robot's path and wheel spin, the balloon physics, the eye expressions and fourteen camera shots, 985 frames at 24 fps. seq_build3.py turns it into a Level Sequence with one CineCamera per shot.



Shot 1: waking in the charging bay.



Rack focus on the sign, keyed in Sequencer.



The ending: the balloon tied to the lucky cat.

Movie Render Queue

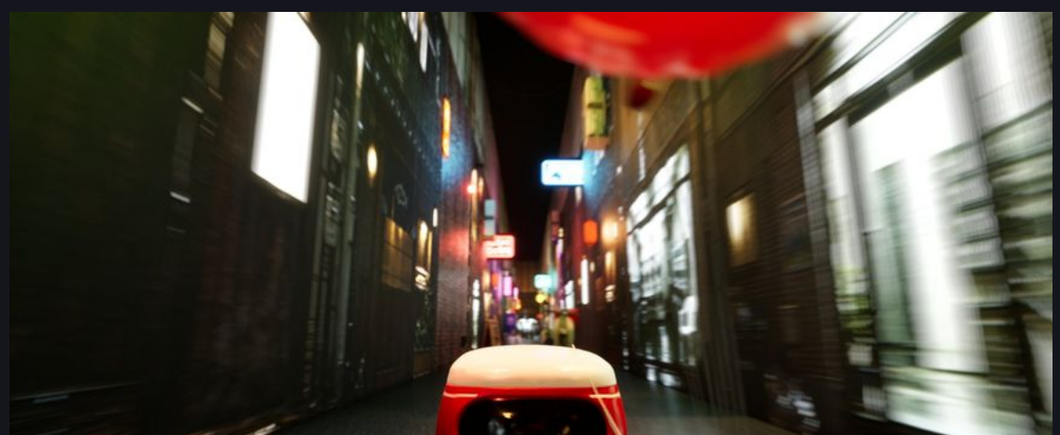
1920 x 1080 PNG frames, 8 temporal samples, 24 warm-up frames, motion blur and depth of field at quality 4. A full render took 20 to 40 minutes, about 1.6 s a frame. Material keys added the documented way collapsed to frame 0, so the channel keys were written directly.

Renders and re-renders

Five full passes over the edit, plus partial re-renders of only the frames that changed after feedback, such as the eye screen and the balloon.

Big-band jazz, spliced, not regenerated.

The score is ElevenLabs Music Advanced v2.5, with a soft middle section from Music Extend. Extend re-synthesises the parts it keeps, so the approved opening and finale were spliced back in with ffmpeg and checked by cross-correlation. The sound effects are ElevenLabs Sound Effects v2.



The chase: the balloon string pulled taut.



Arrival: the robot reaches the cat.

Made with

Claude Code driving the Scenario MCP and Epic's Unreal MCP, Unreal Engine 5.8 on an Apple M5 Mac, Blender for checks and turntables, ffmpeg for the edit.

The receipt

Robot: GPT Image 2.5, Meshy 7.1, Tripo Segmentation v2 (3)	233 CU
Street: GPT Image 2.5 fronts, Hunyuan 3D Pro 3.1 buildings (22)	810 CU
Props: GPT Image 2.5, Meshy 7.1, Tripo v3.1 (27)	1,846 CU
Textures and flat art: GPT Image 2.5, PATINA (7)	130 CU
Total, 59 final Scenario generations, 6 models	3,019 CU

From the session's Scenario job records. Final 3D models and images only, each counted once with the one image it was made from.