

ONE SCRIPT, THREE LOOKS / MAKING OF

Blender 2D Animation

A playable side-scrolling platformer game, built and animated in Blender: flat cards stacked in depth, a run cycle from a video model and real rigid-body physics. Every picture came from Scenario through the Scenario MCP, and one Python script builds the whole scene.

9

coins per level

3

looks, one script

30

Scenario generations

656

credits (CU)



One frame sets the look.

Each look starts as one in-game screenshot made with GPT Image 2.5 Sunburst: sky, far layer, mid layer, ground, hero and coins. It becomes the reference image for every other asset, so the set stays consistent.



Jungle run: style frame, 1536 × 864, 2 takes, 23 CU.



Halloween night: same recipe, a black cat on a town street.



Snow day: a snow leopard in a white winter valley.

A screenshot of a game that does not exist yet

The style-frame prompt asks for a strictly side-on view, layered for parallax, with a clear flat walkable ground line in the lower quarter. That line becomes the game's floor.

One reference for nine assets

Sky, far layer, mid layer, ground strip, foreground, props, tree and hero poses are each generated with the style frame as the reference image, 13 to 15 CU each.

Each layer on its own.

Every layer is its own generation on a transparent background, so it can sit on its own plane in depth. The props come on one sheet with wide gaps, and the foreground plants on another.



Sky: opaque, 2560 × 1440.



Far layer: sea and islands, transparent, tiled.



Mid layer: jungle and ruins, transparent, tiled.



Props: 8 sprites on one sheet, 14 CU.

Layers that repeat

A script cross-fades each strip's two ends so it tiles without a seam, and rolls the seam to where the silhouette is lowest, so no landmark is cut in half.

Props on one sheet

Pots, an amphora, a jar, an idol, a coconut, a coin, a floating island and a pedestal, cut apart by connected components. Each prop gets a collision shape traced from its outline.

A run, like a treadmill.

The hero is drawn in two poses on flat magenta. The left pose is the first frame of a five-second Minimax H3 clip that runs in place; a script keys the background and keeps the frames that loop most cleanly.



GPT Image 2.5 Sunburst: run start and jump, on magenta.



Minimax H3: image to video, 5 s, 768p, 80 CU.



The kept loop: 9 frames for the panda.

The loop that closes

The script compares every frame with every later one and keeps the window whose last frame leads back into its first: 9 frames for the panda, 12 for the cat and the snow leopard.

When the background drifts

The snow leopard's clip drifted from magenta to green. Each frame was keyed on its own border colour, and the loop was taken from the green part, away from the red scarf.

Flat cards, real depth.

The Scenario MCP makes every asset, and one Python script builds the whole scene. Every image becomes a flat card; from the game camera, the stack reads as a flat 2D picture.



The game camera: orthographic, so the cards line up flat.



Turned: each card at its own depth.



Further out: sky, far, mid, ground and foreground planes.

One script for the scene

The script lays out the cards, the camera, the jumps and the physics, then renders. The same script builds every look; only the pictures change.

Parallax from drivers

Each layer follows the camera by its own factor: sky 0.97, far 0.85, mid 0.6, ground and props fixed, foreground -0.45. Near cards slide faster than far ones, which sells the depth.

Jumps, coins and real physics.

The same script animates it: a camera that follows the hero, jumps keyed with a little squash and stretch, and coins that flip up and out of the frame. The collisions are Blender rigid bodies.



Frame 60: the jump pose, with squash and stretch.



Collision shapes traced from each sprite's outline.



Frame 170: the pots and the idol topple.

Shapes from the sprites

Each prop's collision shape is a convex hull traced from its picture, so a round pot rolls and a jar sits flat. Lane walls keep everything in one plane.

Baked, so it scrubs

The physics is baked before saving, so the file plays the same falls on every scrub and render: EEVEE, 1920 × 1080, 24 fps, 184 frames.

Same script, new pictures.

A new look is only new pictures. A Halloween street with a black cat and a white winter valley with a snow leopard were made with the same recipe and built by the same script: same camera, same jumps, same physics.



Jungle run: red panda, golden hour, 226 CU.



Halloween night: black cat, moonlit street, 215 CU.



Snow day: snow leopard, winter valley, 215 CU.

--set assets_snow

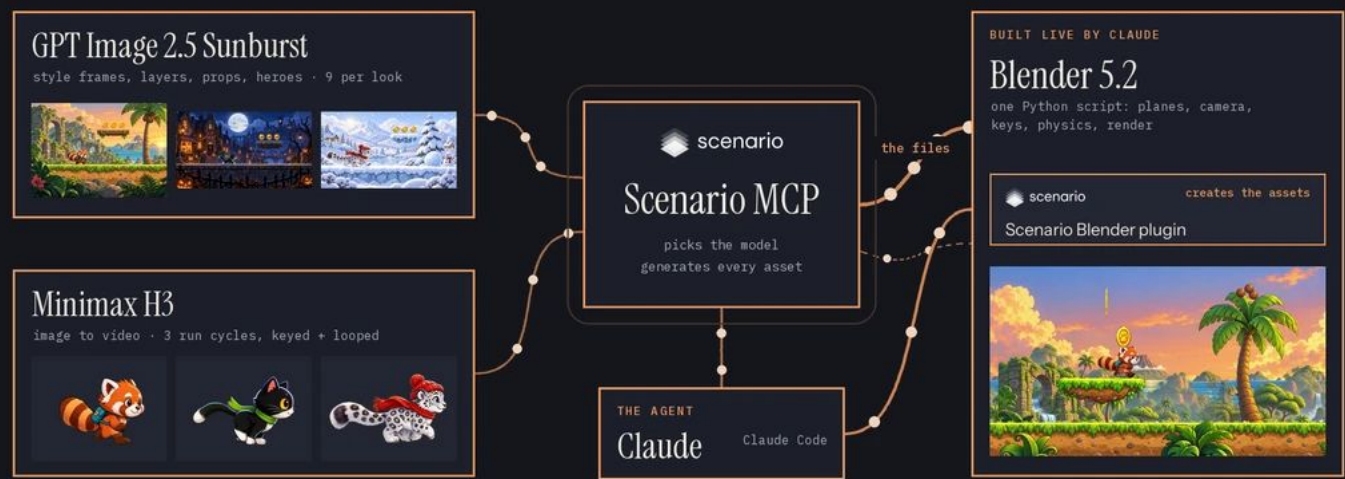
Each look is a folder with the same file names and a small config of numbers measured from its pictures: the horizon, the ground line, the island and pedestal tops, and the hero's size.

Feet on the ground

The ground line is measured where the strip turns solid, then set a little lower, so the hero's feet sink into the grass, the cobbles or the snow instead of hovering above them.

One agent, one script.

The Scenario MCP makes every asset: GPT Image for the art, Minimax for the run. Then Claude runs one Python script in Blender that builds the whole scene.



The connector graph from the film. Here Blender ran the script headless; the Scenario Blender plugin can run it live.

Two models

GPT Image 2.5 Sunburst for every picture, nine per look, and Minimax H3 for the three run cycles. The local scripts key, cut, tile and loop the results at no cost.

The receipt

Jungle look: concept, 5 layers, props, palm, panda poses, run clip	226 CU
Halloween look: concept, 5 layers, props, tree, cat poses, run clip	215 CU
Snow look: concept, 5 layers, props, fir, snow leopard poses, run clip	215 CU
Total, 30 finished Scenario generations	656 CU

From the session's Scenario job records. Final results only: discarded takes, the page's look icons, and the film's narration and music are not counted.