

PIXEL VFX FORGE

Pick one of 42 pixel-art effects, remix it with words or the inspector, preview it on a character in a scene and download it as MP4, GIF, sprite sheets or a Godot 4 resource. Free, in the browser, with stage art made through Scenario MCP.

22

layer types

34

forms

18

elements

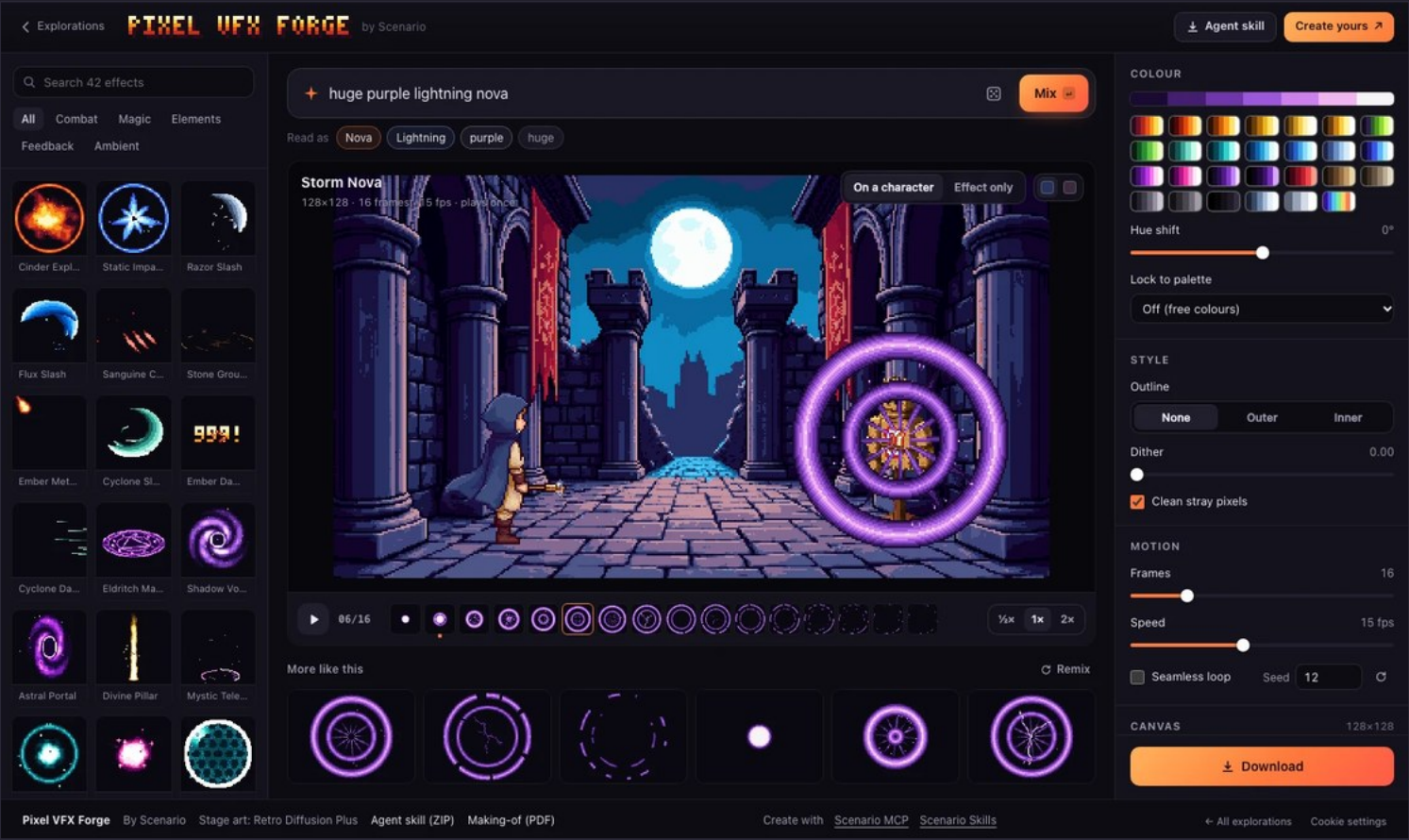
42

library effects

Captured from the page: "big fire explosion" on the training dummy in the Moon hall.

A free pixel VFX library in the browser

The brief: a free, in-browser answer to the desktop pixel-VFX tools circulating on game-dev social media. Pick an effect, remix it, preview it on a character in a scene, then download it for a post or an engine. It first shipped as a generator for any typed effect; review showed it is a library of hand-built effects, and the page now says so. Everything renders locally; nothing is uploaded and no account is needed.



The finished editor with “huge purple lightning nova”, paused on frame 6 of 16.

Start from the library, remix with words

Pick one of 42 curated effects in five groups: Combat, Magic, Elements, Feedback and Ambient, or mix effect, element and style words in the prompt. Library tiles animate while visible, and the *Read as* line shows how the words were understood.

Edit what the words set

The inspector covers colour ramp, hue, palette lock, outline, dither, clean-up, frames, fps, loop, seed, resolution and each layer's colour and parameters. Recipes copy and paste as JSON.

Fast enough to type into

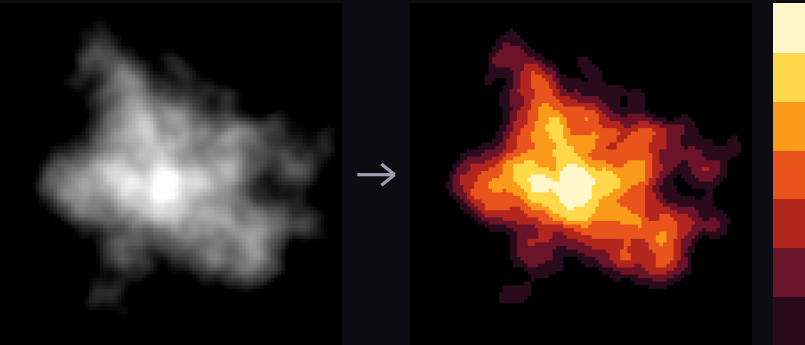
Rendering runs in module workers, with a main-thread fallback. The effect re-renders 260 ms after typing stops, six variations re-roll the seed, and the URL hash keeps prompt, seed and style for sharing.

Three pieces underneath

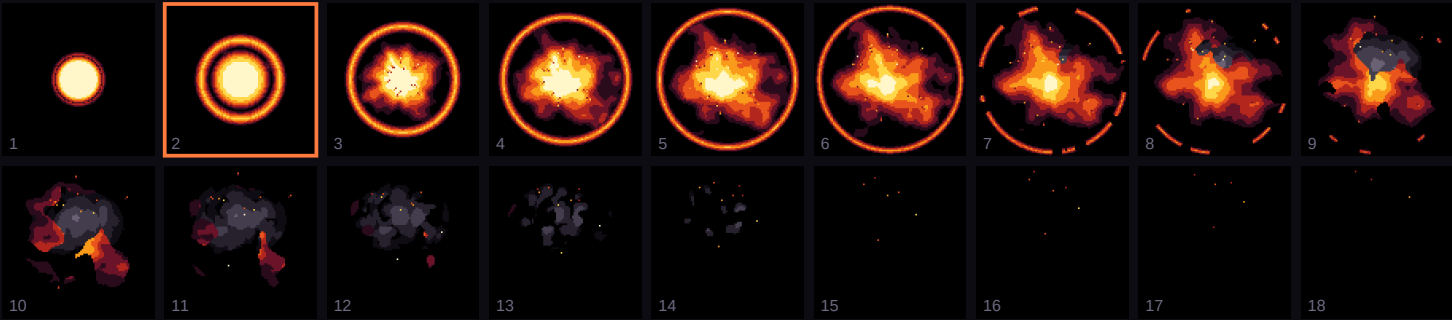
A heat-field renderer (`engine.js`), a rule-based prompt interpreter (`forms.js`) and a stage that times hits to each effect (`stage.js`). The renderer has no DOM, so the same file runs in the page, in a worker and in Node.

Heat first, then hard colour bands

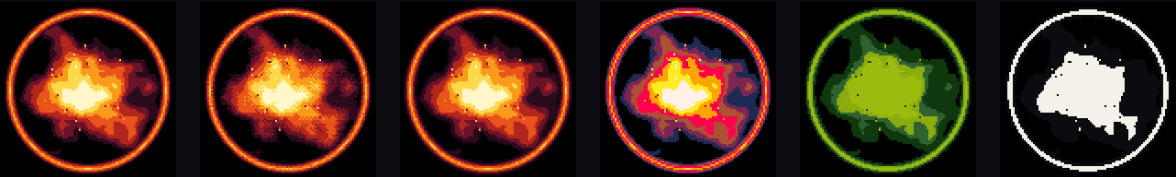
A recipe is a list of layers. Each layer writes a heat field, and the renderer cuts that heat into hard bands from a hue-shifted colour ramp, optionally dithered with a 4x4 Bayer matrix. The hard bands are what make the result read as pixel art rather than a blurred glow.



The burst layer of “big fire explosion” at frame 6: its heat field, then the same field cut into the fire ramp’s 7 colours.



All 18 frames of the same one-shot at 15 fps. Frame 2 is its impact frame.



As shipped Bayer dither Outer outline PICO-8 lock Game Boy lock 1-bit lock
Frame 5 under each post-pass setting.

A post pass on every frame

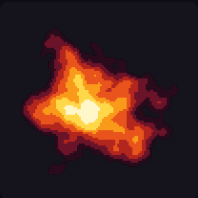
Stray pixels are removed, one-pixel holes filled, and inner or outer outlines drawn from each layer’s darkest colour. A palette lock maps every colour to PICO-8, Sweetie 16, Endesga 32, Game Boy or 1-bit by nearest OKLab distance.

Timing built into the engine

Loops tile because every periodic setting (noise scroll, spin, orbit turns) completes whole cycles over the loop. For a one-shot, the impact frame is the frame with the largest jump in covered pixels in the first 60% of the effect.

22 ways to write heat

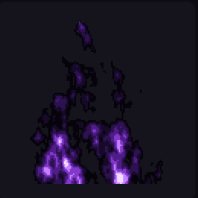
Every effect is assembled from 22 layer types, each a small function that fills a heat field: noise-shaped fireballs, shockwave rings, flames, beams, slash crescents, vortices, pillars, rays, magic circles, smoke puffs, particles in eight emitter modes, lightning, cracks, spikes, impact stars, claw and speed streaks, shield bubbles, flashes, pixel-font text and tornadoes, plus orbs and comets.



burst
fireball



ring
shockwave



flame
flame, aura



beam
straight beam



slash
crescent



orb
energy orb



comet
projectile



vortex
vortex, portal



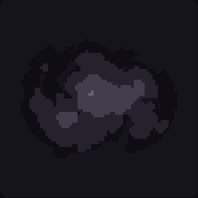
pillar
column



rays
light rays



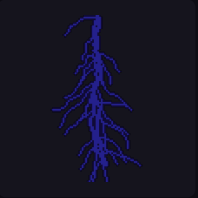
glyph
magic circle



puffs
smoke, gas



particles
8 emitters



bolt
lightning



cracks
ground cracks



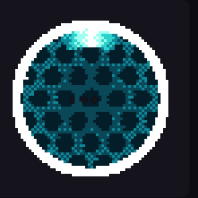
spikes
crystal spikes



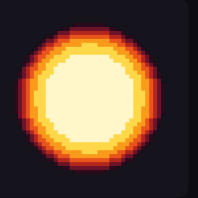
star
impact star



streaks
claws, speed



bubble
shield



flash
white flash



text
pixel text



tornado
funnel

Each type rendered alone by the shipped engine, with the settings it has in the first library effect that uses it, at its fullest frame and cropped to its pixels.

Small, composable recipes

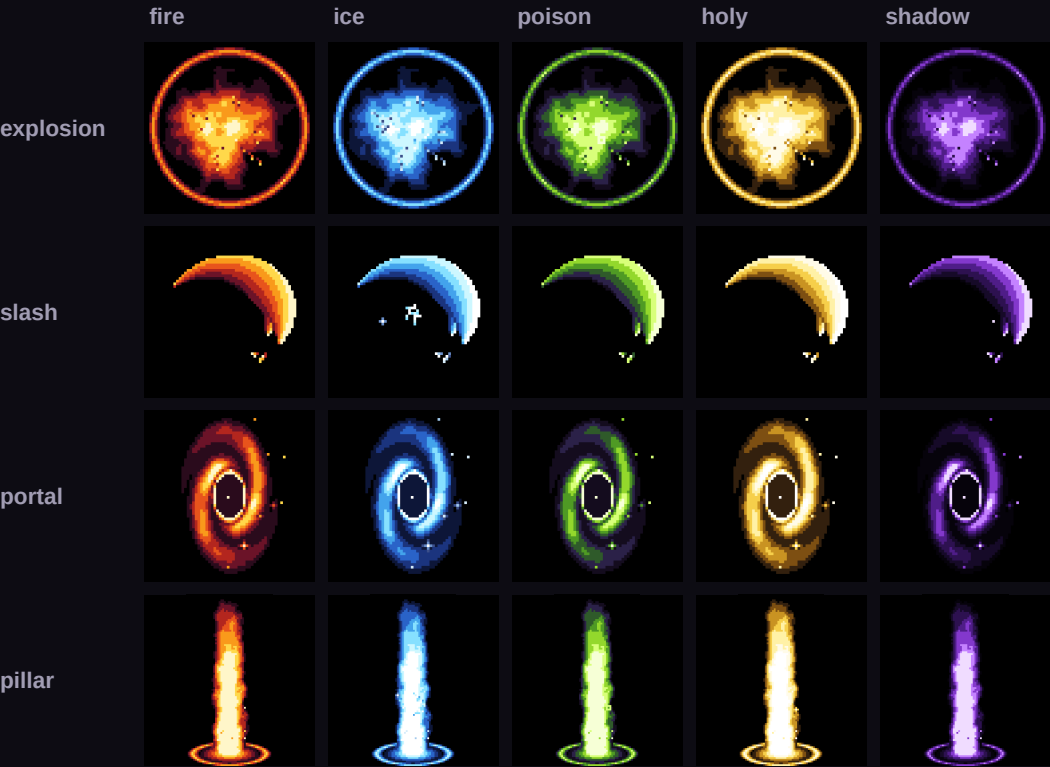
A layer names its type, a colour ramp and a time window inside the effect (t0, t1). The explosion at right stacks late smoke puffs, a burst, a ring, a particle burst and a one-frame flash, and its fire element adds rising embers. The inspector exposes each layer's colour and parameters, and recipes copy and paste as JSON.

Excerpt of the recipe for "big fire explosion", seed 7

```
"w": 96, "h": 96, "frames": 18, "fps": 15, "loop": false, "layers": [
  {"type": "puffs", "ramp": "$smoke", "t0": 0.22, "count": 7, "spread": 0.36, ...}
  {"type": "burst", "ramp": "$main", "r0": 0.1, "r1": 0.74, "grow": 0.42, ...}
  {"type": "ring", "ramp": "$main", "t1": 0.55, "r0": 0.25, "r1": 0.96, "th": 0.12, ...}
  {"type": "particles", "ramp": "$spark", "mode": "burst", "count": 14, ...}
  {"type": "flash", "ramp": "$main", "r": 0.46, "dur": 0.1}
  {"type": "particles", "mode": "rise", "ramp": "$spark", "count": 10, "x": 0, "y": 0, ...}
]
```

Words become a recipe

The prompt interpreter is rule-based, not a language model. It matches a prompt against 34 forms (explosion, slash, beam, portal, strike, heal, level up, weather...) and 18 elements (fire, ice, lightning, poison, holy, shadow, arcane...), plus modifiers for size, speed, density, count, spin, ground placement, loop, outline, dither, palette, direction and colour words, which map to OKLCH hues.



"<element> <form>" at seed 3: four forms read with five elements, rendered by the shipped engine.

+

tiny fast ice slash with outline

⊞

Mix

Read as

Slash

Ice

tiny

fast

outline

+

holy explosion then healing

⊞

Mix

Read as

Explosion

Heal

Holy

The *Read as* line on the page shows how each prompt was understood.

Forms build, elements colour

Forms build layered recipes. Elements colour them and add their own accent layers: embers, shards, arcs, bubbles, motes, wisps, leaves, droplets, debris or gusts.

Two forms, one effect

A second form adds its layers time-offset, so “holy explosion then healing” plays a burst and then a heal.

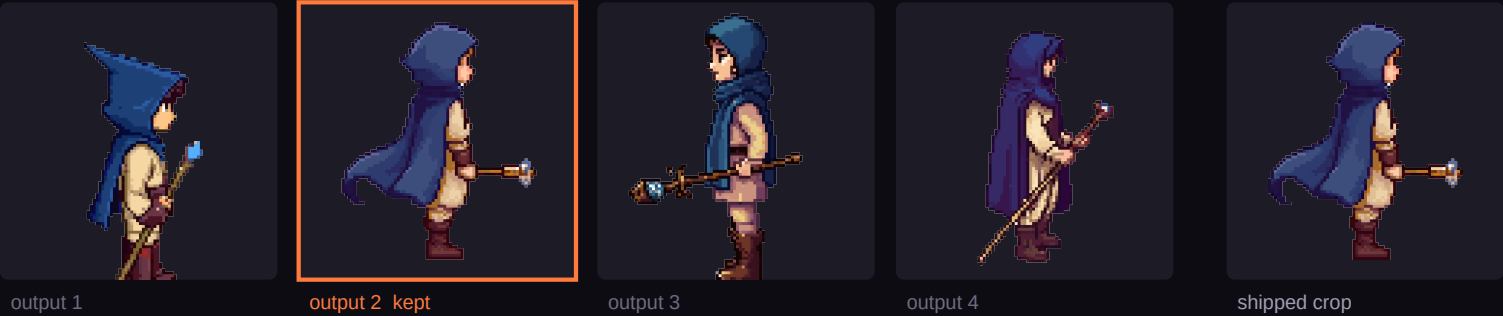
Stable results, honest misses

Unknown words beside a known one still change the seed, so each such prompt gives a stable, distinct result. A prompt with no form or element word keeps the current effect and offers words to add. Effects outside the vocabulary can be written as recipes and loaded with Paste JSON.

A caster and a target on a true pixel grid

The stage art was generated through [Scenario MCP](#) with Retro Diffusion Plus, which renders on a true pixel grid at native resolution (16 to 384 px), so the art and the effects share one pixel density. Retro Diffusion Plus returned four candidates for each sprite with the background removed.

CASTER



style: default · 96x96 · 4 outputs
removeBg: true · seed 4107

“16-bit JRPG side-view battle sprite of a young battle mage facing right, full body standing, short indigo hooded cloak, cream tunic, brown boots, brass-tipped wooden staff with a small blue crystal held forward, readable silhouette, limited palette”

Output 2 kept: its staff is held forward, so beams start at the staff tip.

TARGET



style: default · 96x96 · 4 outputs
removeBg: true · seed 4108

“16-bit side-view sprite of a straw training dummy on a wooden post, burlap sack head with stitched X eye, rope bindings, painted red and white target on the chest, small square wooden base, facing left, limited palette”

Output 3 kept.

All eight candidates as Retro Diffusion Plus returned them, at the same scale. The last tile in each row is the shipped sprite, cropped to its content.

Two scenes, prepared without losing a pixel

The scene prompt ran with style: environment at 384x216, two outputs, seed 4109. Both outputs were kept and became the Moon hall and the Archway.



Moon hall (output 2)

“side-view 2D game battle background, moonlit stone courtyard of a ruined castle at night, flat flagstone floor across the bottom third, low brick wall, distant broken arches and two crumbling pillars, torn hanging banners, pale moon, deep blue and violet palette, low contrast and calm, no characters”



Archway (output 1)

Preparation

Alpha was thresholded at 128, leaving no partial alpha, and the sprites were cropped to their content. All four images were converted to lossless WebP with sharp (lossless: true, exact: true) and verified pixel-identical.

Why lossless WebP

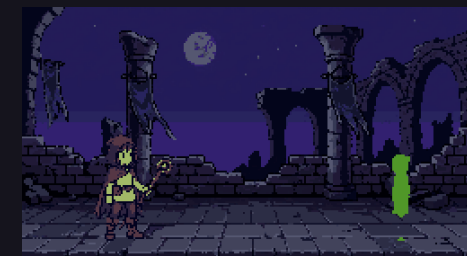
The site's importer re-encodes loose PNGs lossily, which would smear pixel art. Lossless WebP ships every pixel as generated.

An earlier attempt, dropped

Gemini 3.1 produced good-looking art first, but its implied pixel grid was about 5.9 px with soft edges. Snapped to a grid it lost detail, and its density did not match the effects.



A detail of the original, enlarged



Snapped to a grid and placed on the stage

Every effect knows where it lands

The scene is 384x216 at native resolution, scaled by whole numbers. Effects are placed by their form and pivot: on the target, at its feet, at the caster, from the staff tip, travelling, or across the full screen. One-shots trigger a two-frame white hit flash and knockback on the target and, for heavy effects, screen shake, all timed to the impact frame.



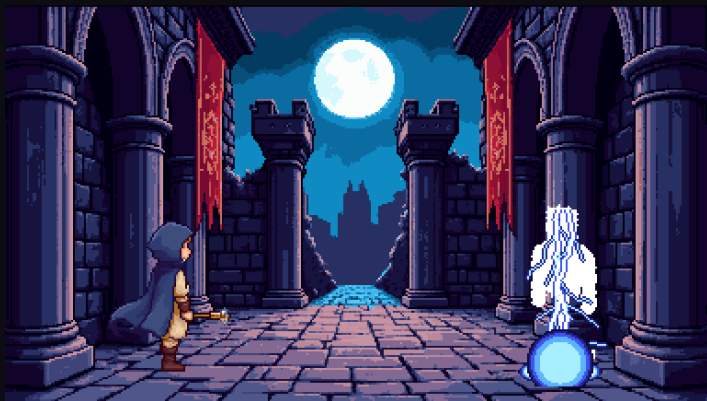
From the staff tip: "blue energy beam".



At the caster, drawn behind: "dark shadow aura".



At the caster's feet: "healing with rising crosses".



White hit flash and knockback, one frame after impact: "lightning strike from the sky".

Frames read from the page's preview canvas at native resolution. The staff tip is found as the rightmost opaque column of the caster sprite.



At the target's feet: "fire tornado".



Across the full screen: "snow falling".

Contact sheets caught what the code did not

During the build, a dev script rendered sampled frames of many effects onto one dark grid, and each effect was judged by eye. The upper rows below come from those sheets; the lower rows are the shipped engine, laid out the same way.

Impact star

Before

Shipped

Eight thin symmetric spikes read as a snowflake. Seven chunkier, jittered spikes at a random rotation read as a hit.

Sword slash

Before

Shipped

The crescent profile peaked too late, so only half the arc showed. A fuller profile plus a thin inner arc fixed it.

Tornado

Before

Shipped

Diagonal stripes looked like a banana. Bands wrapped around the cylinder (asin of the cross-section) and a filled body read as a funnel.

“ice spikes erupting”

Before

Shipped

“erupting” matched the pillar form before “spikes” was read. A second form now has to come from a non-overlapping phrase.

“hex shield barrier”

Before

Shipped

“hex” pulled in a magic circle. The phrase lists were adjusted, and the shield now stands alone.

“critical hit damage 999”

Before

Shipped

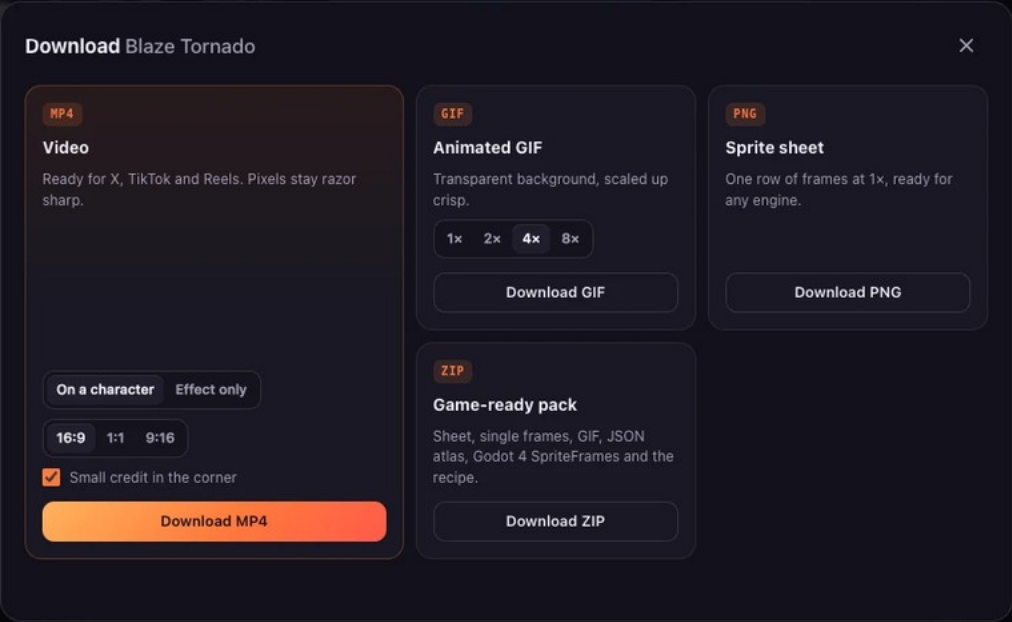
“critical hit” matched the impact form. It now reads as a damage number.

Aura. The hollow centre left dark embers at the base. A stronger, wider hollow and a gentler taper keep the flames on the sides.

Ambient embers and bubbles. They spawned below the canvas and were invisible. The spawn moved to the bottom edge with a faster rise.

Out of the browser, into a feed or an engine

The export module loads when the download sheet first opens and encodes everything in the browser.



The download sheet.



A frame of the 1920x1080 MP4 exported for "fire tornado", with the optional corner credit.

MP4 for posting

WebCodecs H.264 with an in-house MP4 muxer, Constrained Baseline first so there are no B-frames, with a MediaRecorder fallback where WebCodecs is missing. Video is 1920x1080, 1080x1080 or 1080x1920 at 30 fps, at least four seconds, with an optional corner credit. The "fire tornado" export probes as H.264, 1920x1080, 30 fps, 120 frames.

GIF, sheets and a game-ready ZIP

GIF uses an in-house LZW encoder. The sprite sheet is one row up to 4096 px, then a grid. The ZIP holds the sheet, single frames, GIFs at 1x and 4x, a TexturePacker-style atlas with Aseprite frame tags, a Godot 4 SpriteFrames .tres, the recipe and a README: 23 files for "fire tornado".

The agent skill

The skill download carries byte-identical copies of the engine, colour and interpreter files plus a render script. An agent writes a JSON recipe, renders it locally, checks the frames and iterates, and can stage the effect on a pixel-art character made or uploaded through Scenario MCP. Recipes move between the skill and the page through Paste JSON.

```
node scripts/render.mjs --prompt "huge purple lightning nova" --out vfx
node scripts/render.mjs effect.recipe.json --backdrop hero.png --at 120,150
```

Open Pixel VFX Forge

Scenario MCP

Scenario Skills