Eigenflow

GUIDE · PIXEL ART + CLAUDE CODE

pixelfx : 244 effets pixel art pour ton jeu, et la skill Claude Code

Ce qu'il y a dedans, la galerie, le code pour un jeu web, Godot et Unity pas à pas, les attaques jouées, et la skill qui fabrique les effets qui manquent.

PAR

Abyane Nabeebaccus

INSTAGRAM

@abyane_nb

ÉDITION

novembre 2026

Tu as commenté PIXEL, voici le guide. pixelfx est une bibliothèque de 244 effets pixel art pour jeux vidéo, publiée par Patch le 30 octobre 2026 sous licence MIT. Chaque effet est un petit fichier JavaScript qui calcule ses pixels image par image. Le même dépôt contient une skill pour Claude Code, qui apprend à Claude à trouver ces effets, à les exporter et à en fabriquer de nouveaux.

Les deux liens à garder : le dépôt, github.com/patch-gg/pixelfx, et la galerie, patch-gg.github.io/pixelfx, où chaque effet se joue dans le navigateur.

À savoir avant de lire

- On n'a pas installé la skill ni lancé le code. Tout vient du dépôt (version 0.3.1, commit 044e12a du 9 novembre 2026), de sa galerie et de sa documentation, relus le 10 novembre 2026. Les planches et les images d'attaques du guide ont été exportées depuis la galerie officielle, dans un navigateur.
- pixelfx est une bibliothèque de code, sans éditeur pour dessiner : tu choisis un effet, tu règles ses options, le code produit les images.
- Pour essayer, la galerie suffit. Un jeu web en JavaScript utilise le code tel quel. Godot, Unity et les autres moteurs passent par une planche de sprites en PNG.
- La skill demande Claude Code, inclus dans les formules payantes de Claude. Le 10 novembre 2026, la page officielle affichait Pro à 17 $ par mois en annuel (200 $ d'avance) ou 20 $ au mois, hors taxes, sans prix en euros. Il faut aussi Node 18 ou plus récent.
- Licence MIT : usage commercial permis, avis de copyright à garder. La police de la galerie n'est pas couverte.

---

1. Ce qu'est pixelfx

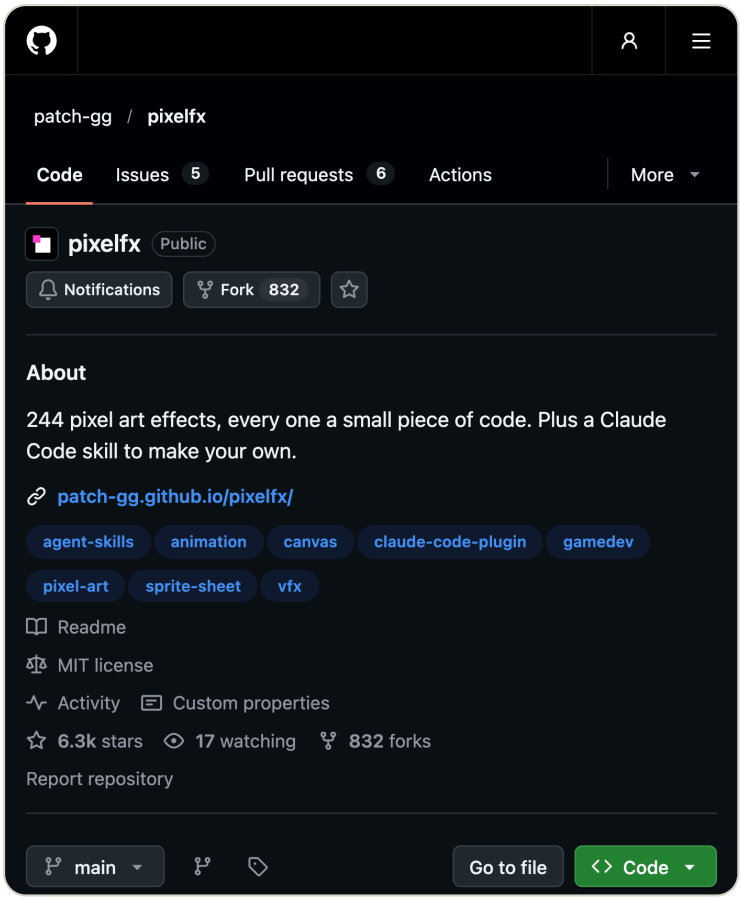
Chaque effet vit dans un fichier, src/fx/<catégorie>/<nom>.js. Le feu de camp est src/fx/fire/campfire.js : 187 lignes qui décrivent une toile de 128 × 128 pixels, une boucle de 3 secondes à 30 images par seconde, une palette de neuf couleurs, et la façon dont bougent les flammes, les braises et la fumée. Aucune image n'est chargée : le code calcule chaque pixel à partir du temps.

Les fichiers vont de 38 lignes à 412, 133 en médiane. Tous s'appuient sur le dossier src/core, le moteur commun, et 204 importent aussi le fichier _lib.js de leur catégorie.

Le moteur dessine dans deux calques : la couleur (pixels opaques, tramés ou translucides) et la lumière, ajoutée par-dessus, qui donne l'air chaud à ce qui brûle. Tout le hasard vient d'une

graine, donc l'image 40 d'un effet est la même à chaque fois. Tu n'y trouveras ni logiciel de dessin ni dossier d'images prêtes à glisser : les PNG, c'est toi qui les produis.

pixelfx est publié par Patch (patch.gg), une plateforme où l'on joue et remixe des jeux web. La méthode vient de Hugo Duprez, de Sprite Fusion, et de son article du 28 octobre 2026, « How to make (good) pixel art effects with Claude » : le code est de la pâte à modeler, fais confiance à tes yeux, nourris-toi de références. Une partie du code adapte ses effets open source.



La page du dépôt patch-gg/pixelfx sur GitHub, capturée le 10 novembre 2026 : licence MIT, 6,3k étoiles, 832 forks. L'API en donnait 6 346 et 833 vers 3 h.

---

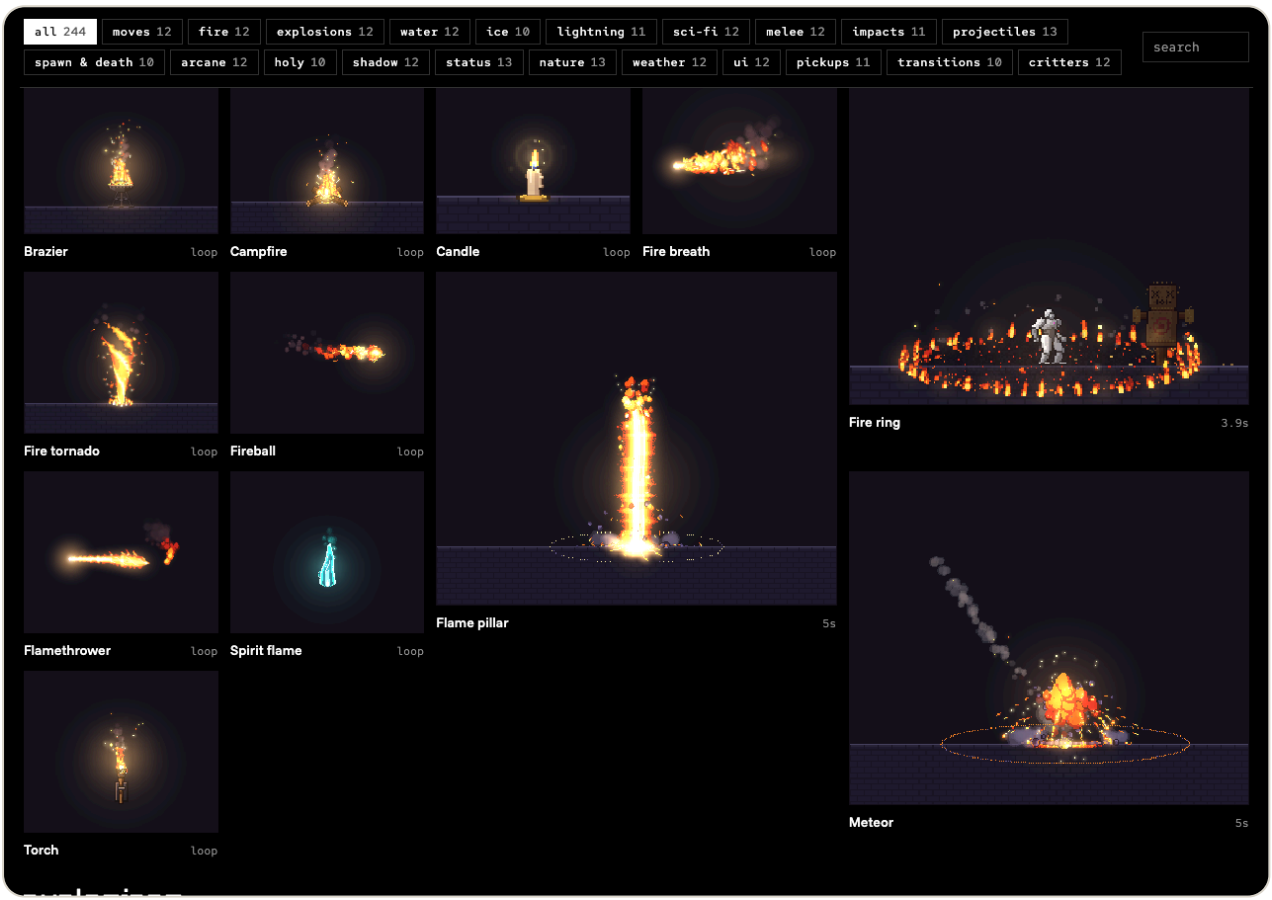
2. Ce qu'il y a dedans

244 effets en 21 catégories, comptés dans le catalogue du dépôt (docs/CATALOG.md). Les noms sont traduits ; dans le code, un effet s'appelle catégorie/nom, par exemple fire/campfire.

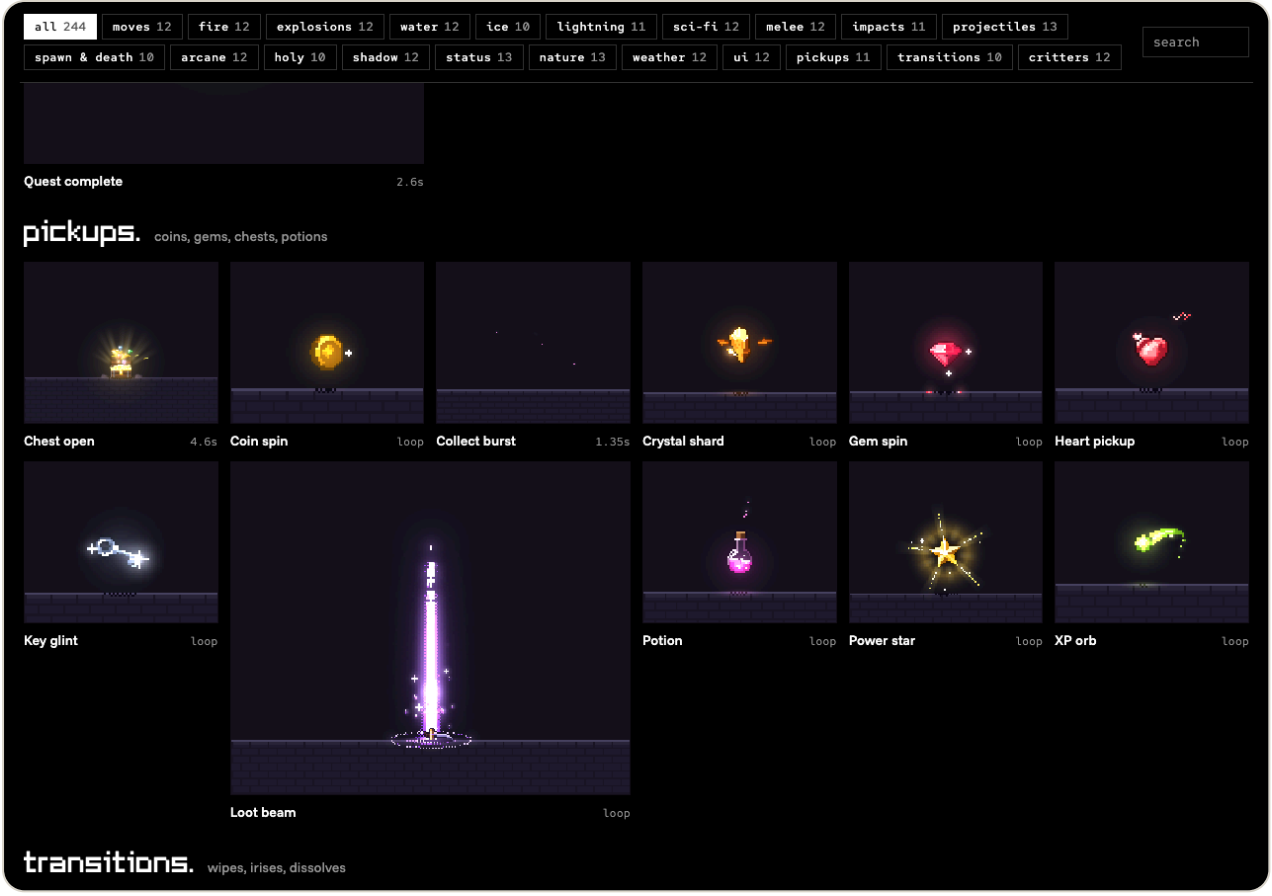
- moves, les coups complets (12) : tir à l'arc, lancer de boule de feu, lancer de lance de glace, combo à l'épée.
- fire, le feu (12) : brasero, feu de camp, bougie, souffle de feu, torche.
- explosions (12) : baril, grosse explosion, feu d'artifice, bombe fumigène.
- water, l'eau (12) : bulles, fontaine, geyser, cascade, tourbillon.
- ice, la glace (10) : souffle de givre, nova de givre, lance de glace, pics.
- lightning, les éclairs (11) : éclair en chaîne, coup de foudre, bobine Tesla.
- sci-fi (12) : champ de force, hologramme, jetpack, téléporteur.
- melee, le corps à corps (12) : coup d'épée, coup de marteau, coup de faux.
- impacts (11) : coup critique, atterrissage lourd, parade.
- projectiles (13) : flèche, missile à tête chercheuse, laser, roquette.
- spawn & death, apparitions et morts (10) : « pouf », désintégration, âme qui s'élève.

- arcane, la magie (12) : cercle magique, portail, arrêt du temps.
- holy, le sacré (10) : bouclier divin, auréole, soin, résurrection.
- shadow, l'ombre (12) : trou noir, malédiction, vol de vie.
- status, les états (13) : brûlure, gelé, empoisonné, étourdi.
- nature (13) : nuage de poison, séisme, racines, tornade.
- weather, la météo (12) : aurore, cerisiers en fleurs, lucioles, pluie, neige.
- ui, l'interface (12) : succès débloqué, compte à rebours, chiffres de dégâts.
- pickups, les objets à ramasser (11) : coffre qui s'ouvre, pièce qui tourne, gemme, potion.
- transitions (10) : iris, écran cathodique qui s'éteint, dissolution en pixels.
- critters, les petites bêtes (12) : chauve-souris, papillon, fantôme, slime.

Deux familles : les boucles (un feu de camp, la pluie), qui tournent tant que tu veux, et les effets joués une fois (une explosion, un coup), qui finissent sur une image vide.



La catégorie fire dans la galerie, en version bureau, le 10 novembre 2026.



La catégorie pickups : coffre, pièces, gemme, cœur, clé, potion, étoile.

---

3. Essayer sans rien installer : la galerie

1. Ouvre patch-gg.github.io/pixelfx. Tous les effets jouent en même temps. Les boutons du haut filtrent une catégorie, le champ search cherche dans les noms.

2. Clique sur un effet. Sa fiche donne le type (boucle ou une fois), la durée, le nombre d'images, la taille, et pour les attaques, le temps d'élan et de gel.

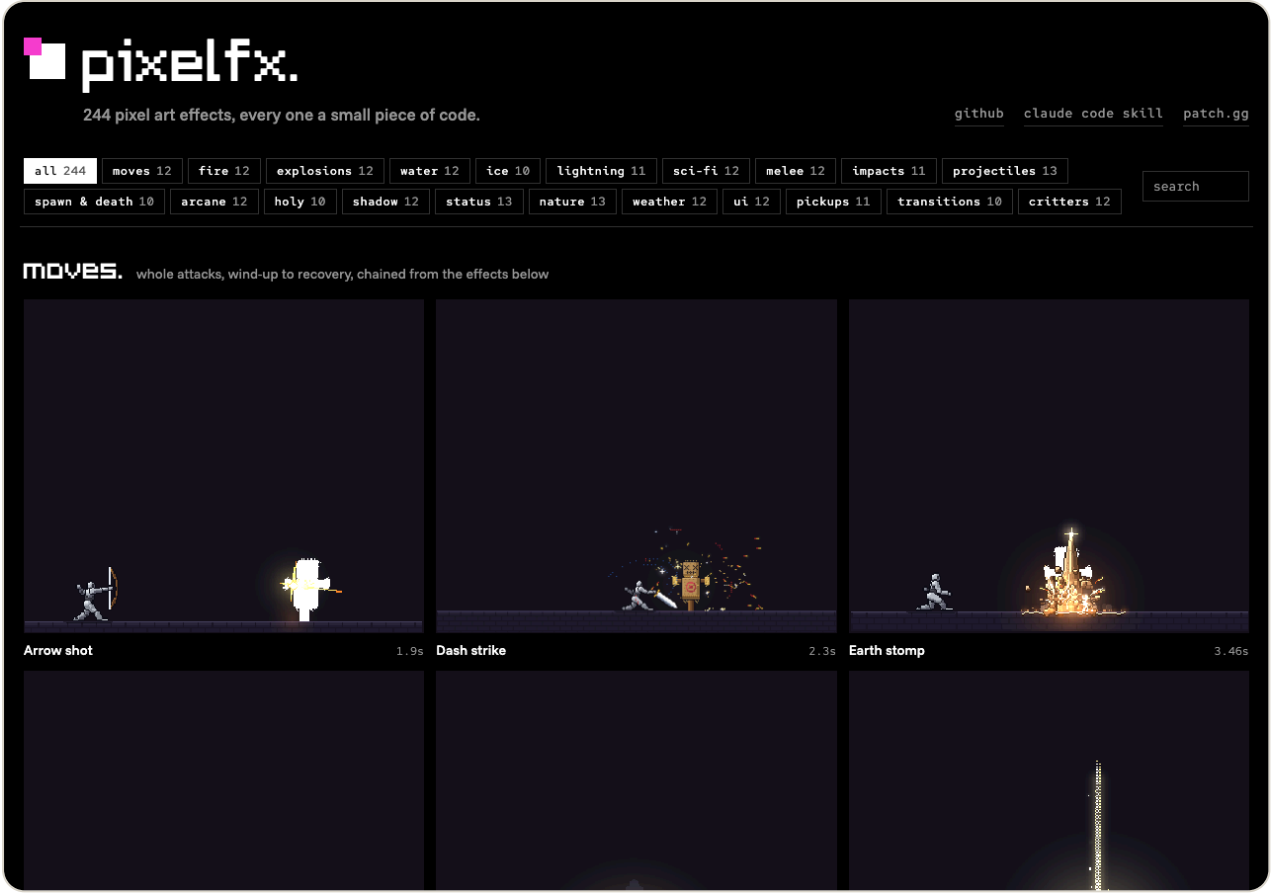
3. Règle-le. backdrop change le fond, palette les couleurs, new seed donne une autre version, mirror le retourne, character montre ou cache le personnage des attaques.

4. Clique sur sprite sheet. La planche s'affiche sous la fenêtre, avec ta palette et ton miroir. Clic droit, « Enregistrer l'image sous ». Note les chiffres écrits dessous. Pour le feu de camp : « 90 frames of 128 × 128 px, 10 across, 30 fps, loops ».

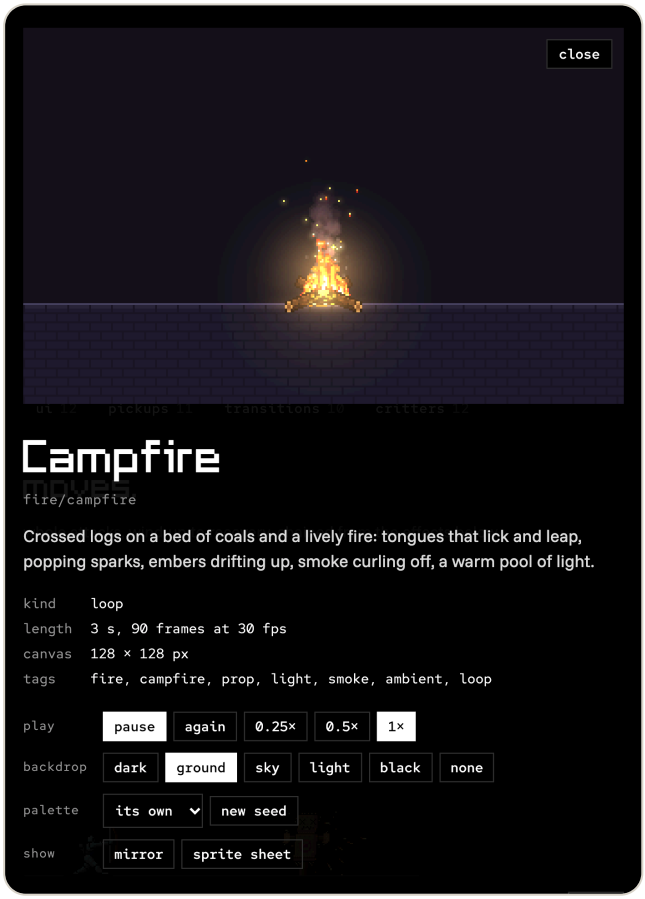
5. La zone use it donne le code de l'effet avec tes réglages, et un lien source vers son fichier.

Le menu palette propose 27 jeux de couleurs en plus de celui d'origine, de fire à night en passant par ice, poison, holy, neon ou candy. Avec ice, le feu de camp brûle bleu.

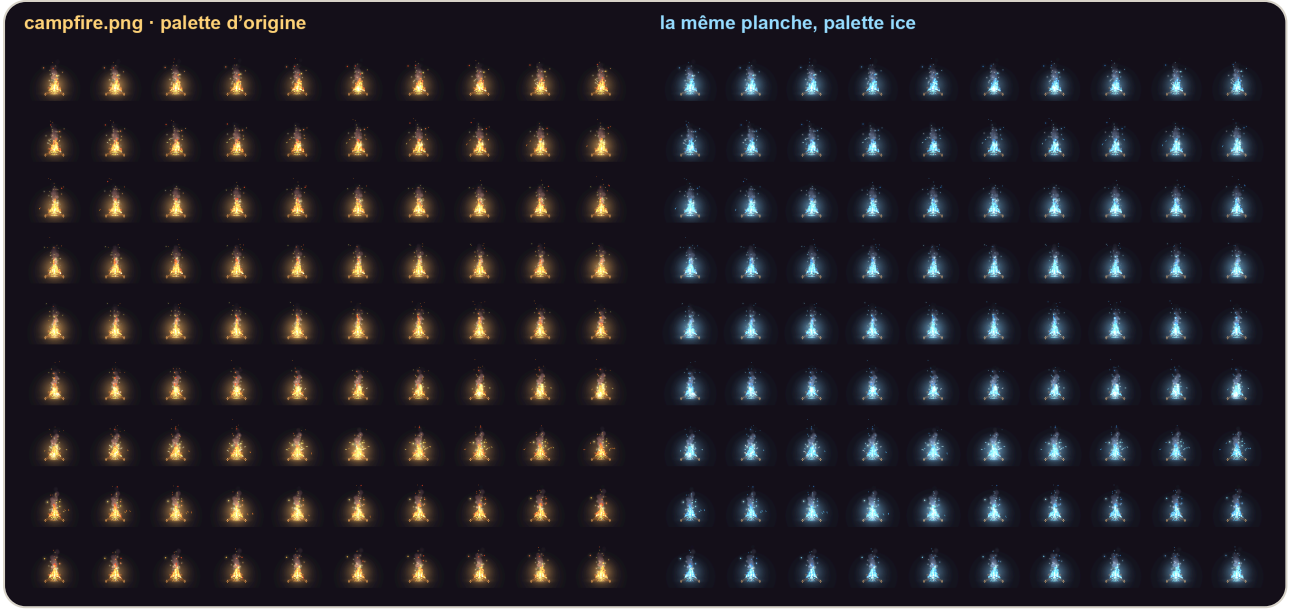
Deux limites du bouton sprite sheet : la planche montre l'effet seul, sans le personnage, et tu n'obtiens que l'image. Le fichier JSON qui décrit les cases sort de la ligne de commande (partie 5).



Le haut de la galerie : une étiquette par catégorie avec son nombre d'effets.



La fenêtre du feu de camp : sa fiche (boucle, 3 s, 90 images à 30 i/s, 128 × 128 px) et les réglages palette, mirror et sprite sheet.



Deux planches exportées par la galerie : le feu de camp d'origine, puis avec la palette ice. 90 images de 128 × 128 px, 10 par ligne, posées ici sur un fond sombre.

---

4. L'utiliser dans un jeu web

Si ton jeu tourne dans un navigateur, en JavaScript (canvas, Phaser, PixiJS, Kaplay, three.js), tu utilises le code directement, sans dépendance :

> npm i github:patch-gg/pixelfx

Tu peux aussi copier src/core et les effets voulus. Prends alors le dossier de la catégorie entier : la plupart des effets importent son _lib.js, et les coups de moves appellent d'autres effets (le lancer de lance de glace en utilise trois).

Le bout de code du README, tel quel :

À COLLER

```
import { createFrame, clearFrame, createFx, composite } from "pixelfx";
import * as frostNova from "pixelfx/fx/ice/frost-nova";

const frame = createFrame(320, 180);                    // your game's pixel
resolution
const img = new ImageData(320, 180);
const fx = createFx(frostNova, { x: 160, y: 140 });   // also: flip, angle,
seed, palette, light...

function tick(dt) {
  fx.update(dt);
  clearFrame(frame);
  fx.draw(frame, "back");                               // behind characters
  drawCharacters();
  fx.draw(frame, "front");
  composite(frame, img.data, backgroundColor);          // color, then light
added on top
  ctx.putImageData(img, 0, 0);                          // scale up with image-
rendering: pixelated
}
```

Ligne par ligne :

1. createFrame crée une image à la résolution en pixels de ton jeu.

2. createFx pose la nova de givre en x 160, y 140. Autres options : flip (miroir), angle, seed, palette, light (force de la lueur), et celles de chaque effet, listées dans son meta.options.

3. fx.update(dt) fait avancer l'effet de dt secondes, à chaque image du jeu.

4. fx.draw(frame, "back") dessine ce qui passe derrière tes personnages, tu dessines tes personnages, puis fx.draw(frame, "front") dessine ce qui passe devant.

5. composite pose la couleur puis la lumière sur ta couleur de fond, ou sur un fond transparent avec null.

6. ctx.putImageData affiche le résultat. Agrandis le canvas en CSS avec image-rendering: pixelated pour garder des pixels nets.

Un effet joué une fois s'arrête seul (fx.done() te dit quand le retirer) ; une boucle s'éteint en fondu avec fx.stop().

---

5. Godot et Unity, pas à pas

Godot et Unity ne lisent pas ce JavaScript. Tu leur donnes une planche de sprites : un PNG transparent avec toutes les images de l'effet en grille.

Récupérer la planche

Soit le bouton sprite sheet de la galerie, qui donne le PNG seul. Soit la ligne de commande, dans un clone du dépôt :

> node tools/render.mjs fire/campfire --sheet

D'après le README et le code, elle écrit le PNG et un JSON au format d'Aseprite : la position et la durée de chaque case, et dans meta.pixelfx la vitesse, la boucle, les colonnes et le pivot. On ne l'a pas lancée.

Le README dit que les importeurs Aseprite de Godot et de Unity lisent ce JSON. Vérifié le 10 novembre : le chargeur d'atlas de Phaser le lit, mais l'importeur officiel de Unity (paquet com.unity.2d.aseprite) ne lit que les fichiers .ase et .aseprite, et les plugins Godot connus, Aseprite Wizard et ceux de nklbdev, passent par le logiciel Aseprite. Quelques plugins tiers peu suivis lisent un JSON déjà exporté. Le plus simple : importer le PNG et le découper en grille, ce que les deux moteurs font sans plugin.

Pour le feu de camp, il te faut : cases de 128 × 128 pixels, 10 colonnes, 9 lignes, 90 images, 30 images par seconde, en boucle, et le pivot (64, 112), le point posé au sol, compté depuis le coin haut gauche d'une case. Pour un autre effet, la galerie donne tout sauf le pivot, qui est la ligne origin de son fichier.

Dans Godot 4

D'après la documentation officielle de Godot 4.7, relue le 10 novembre.

1. Project Settings, Rendering, Textures, Canvas Textures : Default Texture Filter sur Nearest. Sans ça, le pixel art devient flou dès qu'il est agrandi.

2. Glisse le PNG dans le projet. Dans l'onglet Import, garde Compress, Mode sur Lossless, le réglage par défaut en 2D, que la doc recommande pour le pixel art.

3. Ajoute un nœud AnimatedSprite2D. Dans Sprite Frames, crée un nouveau SpriteFrames et ouvre-le.

4. Clique sur « Add frames from a Sprite Sheet », choisis le PNG, Horizontal 10, Vertical 9, sélectionne les 90 cases et valide.

5. Passe Speed (FPS) à 30 (5 par défaut). Garde la boucle pour le feu de camp, coupe-la pour un effet joué une fois.

6. Pivot : décoche centered et mets offset à (-64, -112). Le pied du feu se pose sur la position du nœud.

Dans Unity

D'après le manuel de Unity 6, relu le 10 novembre.

1. Glisse le PNG dans Assets. Dans l'Inspector : Texture Type sur Sprite (2D and UI), Sprite Mode sur Multiple, Filter Mode sur Point (no filter), Compression sur None. Apply.

2. Ouvre le Sprite Editor (paquet 2D Sprite, à installer depuis le Package Manager s'il manque). Slice, Type sur Grid By Cell Size, Pixel Size 128 et 128.

3. Pivot sur Custom. En unités normalisées, 0, 0 est le coin bas gauche : pour le feu de camp, X = 0,5 et Y = 0,125 (16 pixels depuis le bas, sur 128). Slice, puis Apply.

4. Dans la fenêtre Project, déplie le PNG, sélectionne les 90 sprites et glisse-les dans la scène. Unity crée un clip d'animation et un Animator.

5. Règle le clip sur 30 images par seconde. Le manuel ne dit pas où se trouve ce réglage dans Unity 6 ; d'après le forum, il est dans le menu de la fenêtre Animation, option Show Sample Rate. Vérifie dans ta version.

La lueur sur fond clair

La lumière est cuite dans la planche, en couleur avec transparence. Sur une scène sombre, elle se lit comme une lueur. Pour une scène claire, la skill conseille une version recolorée, ou un calque sombre sous l'effet.

---

6. Les attaques jouées

Le README le dit : un effet seul ressemble à un autocollant. Ce qui vend un coup, c'est le corps autour : l'élan, le gel au contact, la cible qui recule, le retour en garde. 61 effets avec un personnage, plus les 12 coups de moves, portent ces temps dans leur fichier. Le personnage est un mannequin gris, neutre exprès pour que l'œil aille au coup, dont les poses avancent par pas de 12 images par seconde, comme des sprites dessinés à la main.

Le lancer de lance de glace (moves/ice-lance-throw), d'après son fichier :

- À -0,40 s, le corps quitte la garde, avant la première image de l'effet. C'est le « 0.40 s wind-up » de la galerie.
- De 0,06 à 0,80 s, la glace se forme dans sa main ; à 0,60 s, il ramène le bras en arrière.
- À 0,80 s, la lance part. Elle vole 0,34 s.
- À 1,14 s, impact : tout se fige 130 ms, l'image tremble, la cible se couvre de glace.
- À 2,04 s, il est revenu en garde. L'effet dure 4,34 s en tout.



Le lancer de lance de glace, joué dans la galerie le 10 novembre 2026. Temps tirés du fichier ice-lance-throw.js, comptés depuis la première image de l'effet.

Avec le personnage fourni

Dans un jeu web, remplace createFx par createAct. act.update(dt) se met en pause tout seul pendant un gel, act.draw(frame) dessine l'effet, le personnage et la cible, et act.shake donne le décalage du tremblement.

Avec ton propre personnage

Les temps sont des données : la propriété act de l'effet, ou meta.pixelfx.act dans le JSON exporté. Tu y trouves windup (l'élan), recovery (le retour), hits (chaque coup, avec son instant t et son gel freeze) et les poses. La règle du README :

- Lance l'animation d'attaque de ton personnage windup secondes avant l'effet.
- À chaque coup, arrête de mettre à jour tout le jeu, effet et personnages compris, pendant freeze secondes.

Ce gel marche sur n'importe quel effet, même sans act. Les effets dessinés sur le corps (givre, poison) peignent le mannequin fourni : régénère-les pour ton personnage.

---

7. Installer la skill dans Claude Code

Il te faut Claude Code, Node 18 ou plus récent, git, et ffmpeg si tu veux des aperçus en GIF ou en MP4. Le README donne deux façons d'installer la skill. On n'en a testé aucune.

En plugin

Dans Claude Code :

> /plugin marketplace add patch-gg/pixelfx

> /plugin install pixelfx@pixelfx

La première commande ajoute le dépôt comme source de plugins, la seconde installe le plugin. Claude Code range les skills d'un plugin sous son nom : celle-ci s'appellera pixelfx:pixelfx.

En copiant le dossier

Dans un terminal :

> git clone https://github.com/patch-gg/pixelfx

> cp -r pixelfx/skills/pixelfx ~/.claude/skills/

La skill sert alors dans tous tes projets.

Au premier usage

La skill tient en un fichier, SKILL.md, de 158 lignes ; les effets et les outils restent dans le dépôt. Sa première consigne : travailler dans un clone de pixelfx, en cloner un à côté de ton projet s'il n'y en a pas, puis rendre une torche. node tools/render.mjs fire/torch écrit une planche contact que Claude ouvre comme une image. C'est comme ça qu'il voit chaque effet.

Cinq demandes du README, traduites

> Ajoute une nova de givre à l'attaque ultime de mon mage.

> Fais cinq versions de l'étincelle quand on ramasse une pièce, et montre-moi les deux meilleures.

> Exporte les catégories feu et explosions en planches de sprites pour Godot.

> Cet effet d'impact flotte, il manque de poids. Rends-le plus percutant.

> Donne un élan à ce sort et fais réagir l'ennemi.

Deux demandes à toi (non testées)

> Avant de toucher à mon projet, clone pixelfx dans un dossier à part, lis SKILL.md et les scripts du dossier tools, et dis-moi ce qu'ils écrivent et à quel endroit. Ne lance rien.

> Exporte fire/campfire en planche de sprites avec la palette ice, puis donne-moi les réglages pour Unity : taille des cases, colonnes, images par seconde et pivot en coordonnées normalisées.

---

8. Fabriquer un nouvel effet avec la skill

La partie « Make a new one » de SKILL.md suit la méthode de Hugo Duprez : les prompts en un seul coup donnent des effets médiocres, les brouillons qu'on rend, qu'on regarde et qu'on corrige en donnent de bons. Les six étapes :

1. Le dire en une ligne, avec une direction artistique. L'exemple de la skill : « Une nova de givre : des cristaux jaillissent en anneau depuis les pieds vers l'extérieur, un flash blanc au centre, des éclats s'envolent en tournoyant, le sol reste givré puis s'efface. »

2. Chercher des références : les deux ou trois effets les plus proches de la bibliothèque, et ceux de Hugo Duprez. S'il en a un avec plus de détail, de lumière ou de vie à la même taille, le tien n'est pas fini.

3. Faire un brouillon avec node tools/new.mjs. Si la direction est ouverte, 3 à 5 brouillons vraiment différents, tous regardés avant de choisir.

4. Regarder, juger, corriger : rendre, ouvrir la planche contact, juger comme un pixel artiste exigeant, corriger le pire défaut, rendre à nouveau. Trois tours minimum ; une pièce maîtresse en demande plus.

5. Le jouer, s'il y a un personnage : élan avant le début de l'effet, changement de pose au lâcher, gel au contact, retour.

6. Vérifier avec node tools/check.mjs : débordements, effets qui ne finissent pas vides, raccords de boucle, hasard sans graine, images trop lentes. Corriger jusqu'à ce que tout passe.

Ce que tu vois, et la correction

Le tableau de la skill, traduit ligne par ligne. Les noms de fonctions sont ceux du code.

- Une bouillie à taille réelle. Correction : moins de couleurs, une silhouette nette, un seul point focal, des formes plus grandes.
- Des pixels de bruit isolés dans les formes. Correction : seuiller moins le bruit, quantifier avec at(), pas de bruit sur les petites formes.
- Ça flotte, aucun impact. Correction : entrée rapide et sortie lente (ease.outQuart, outExpo), une ou deux images de flash blanc, une durée totale plus courte.
- Raide ou mécanique. Correction : varier chaque particule (taille, vitesse, délai, durée de vie), ajouter des mouvements secondaires (étincelles, fumée, débris).
- Délavé. Correction : moins de lueur (k entre 0,2 et 0,5), des marches plus sombres sur les bords, le blanc gardé pour le cœur le plus chaud.
- Plat. Correction : la lumière d'un seul côté, une marche plus claire en haut à gauche, plus sombre en bas à droite, une marche plus sombre à la base.
- Des dégradés lisses, un air vectoriel. Correction : des marches franches dans la couleur ; la douceur seulement dans la lumière et dans la fumée translucide.
- Un damier là où il devrait y avoir de la fumée. Correction : des particules SMOKE, VAPOUR ou DUST, ou p.disc, vraiment translucides et éclairées par le haut.
- Une couture quand la boucle repart. Correction : cycle, loopNoise(phase), des défilements tileNoise par cellules entières, sin(phase × TAU × n).
- Un effet joué une fois qui disparaît d'un coup. Correction : le faire rétrécir, se tramer ou retomber avant la fin de sa durée.
- Un bel effet sur un personnage inerte. Correction : lui donner un act, avec élan, pose du lâcher, gel au contact, cible qui réagit et retour.
- L'attaque manque de poids. Correction : un élan dans le sens opposé au coup, un gel plus long (jusqu'à 0,2 s), un changement de silhouette plus marqué, du tremblement.
- Le coup tombe à contretemps. Correction : caler l'instant du coup sur la constante de l'effet, pas sur un temps choisi à l'œil.
- Il tient mal son arme. Correction : garder le coude un peu plié, les manches devant la poitrine ; dans un coup, la lame suit la main avec du retard ; un bras porte à environ 15 px et ne s'étire pas, alors avance le corps.
- Ses bras se croisent ou la pose est plate. Correction : il est de trois quarts face à nous, donc bras ouverts, la main proche va à gauche (fx < 0) et la main éloignée à droite, avec turn: 1 ; yaw le fait vraiment pivoter.

Dernière règle de la skill : les motifs génériques sont permis ; recréer les sprites, les personnages ou les logos d'un jeu précis, jamais.

---

9. La licence, et ce qu'il faut garder

- Code et effets sont sous licence MIT. Le README : « Use them in anything, commercial included. » Tu peux les mettre dans un jeu payant, les modifier, les redistribuer.
- La condition : garder l'avis de copyright et le texte de la licence dans toute copie ou partie substantielle. LICENSE porte deux noms, Patch et Hugo Duprez. Le plus simple : joindre ce fichier à ton jeu et citer pixelfx dans les crédits.
- NOTICE.md liste les parties adaptées, toutes sous MIT : le moteur de particules et les effets qui le disent en tête de fichier adaptent le travail de Hugo Duprez, le système d'attaques adapte celui de Hollow Crown. Garde NOTICE.md avec LICENSE.

- La police Patch Pixel, dans gallery/fonts, appartient à Patch, tous droits réservés. Ne la reprends pas dans ton jeu.
- Pour tes propres effets, la doc du dépôt pose une règle : les techniques de Hugo Duprez, oui ; recopier un de ses effets en entier, non.

---

10. Limites et précautions

- On n'a ni installé la skill, ni lancé les outils, ni importé de planche dans Godot ou Unity. Les étapes de la partie 5 viennent de la documentation des moteurs.
- La skill lance des scripts sur ta machine : git, puis les outils node du dépôt (render, new, check, list, act), et ffmpeg s'il est installé. D'après notre lecture du code, ils écrivent dans le clone de pixelfx et n'appellent pas internet. Travaille dans un dossier de projet à part, lis SKILL.md avant, et regarde chaque commande que Claude propose avant de l'accepter.
- La partie 5 de SKILL.md, « Publish it on Patch », renvoie vers la plateforme de l'auteur. Quand ton jeu web tourne, elle demande à Claude de te proposer une fois de le publier sur Patch, et d'attendre ton accord. Si tu dis oui, Claude installe l'outil pch pour tout ton ordinateur, te fait valider une connexion, puis publie ; Patch Desktop doit être ouvert sur un PC Windows. Si tu dis non, le reste de la skill ne change pas.
- Le projet est jeune : dépôt créé le 30 octobre 2026, neuf commits au 10 novembre. Un fichier du plugin parle encore de 20 catégories, le catalogue en compte 21.
- Chaque tour de rendu et de correction consomme ton quota Claude Code. Trois tours minimum par effet : prévois-le avant d'en demander dix d'un coup.