Sprite Sheet Cutter

User Guide

Extract individual sprites from sprite-sheet PNGs
with automatic grid detection & background removal

v1.0 | February 2026

Table of Contents

1 Quick Start

Option A: Standalone Executable (recommended)

Double-click SpriteSheetCutter.exe. No Python installation needed. The GUI launches automatically.

Option B: Run from Source

Make sure Python 3.10+ is installed, then:

```
pip install -r requirements.txt
python main.py --gui
```

Basic Workflow

- **1.** Click Browse next to Input Folder and select the folder containing your sprite-sheet PNGs.
- **2.** Click Browse next to Output Folder to choose where extracted sprites should be saved.
- **3.** Adjust settings if needed (defaults work well for most sheets).
- **4.** Click Extract Sprites and watch the progress bar and log output.
- **5.** Find your individual transparent PNGs in the output folder, organized into subfolders.

2 Graphical Interface (GUI)

The GUI uses a dark theme inspired by the Cursor IDE. It is built with customtkinter and provides a clean, modern look.

Title Bar

Shows the application name. The ? button in the top-right corner opens this user guide PDF.

Folder Selectors

Two rows with text fields and Browse buttons. The Input Folder is where your sprite-sheet PNG files are located. The Output Folder is where extracted sprites will be saved. If you leave the output field empty and pick an input folder, it defaults to an 'output' subfolder inside the input folder.

Settings Panel

Four numeric fields for fine-tuning the extraction. See Section 3 for details on each setting.

Extract Sprites Button

Starts the processing pipeline. The button is disabled while processing is in progress. A progress bar and status label below the button show real-time progress.

Output Log

A scrollable text area that shows live logging output from the pipeline: which files are being processed, how many cells were detected, and which sprites were saved.

3 Settings Reference

Setting	Default	Description
Output Size	512	Target square size in pixels. Sprites are resized (aspect-ratio preserved) onto a transparent
Padding	10	Pixels of transparent padding added around the cropped sprite before resizing.
White Threshold	230	RGB value (0-255) above which a pixel is considered white/background. Lower = stricter (les
Flood Tolerance	25	Color tolerance for the flood-fill expansion. Controls how far from pure white the algorithm wi

These settings map directly to the Config dataclass in config.py and can also be set via CLI flags (--size, --padding, --white-threshold, --flood-tolerance).

4 Command Line Interface (CLI)

The tool can also be used entirely from the command line without the GUI:

Process a folder

```
python main.py --input "path/to/sprites/" --output "path/to/output/"
```

Process a single file

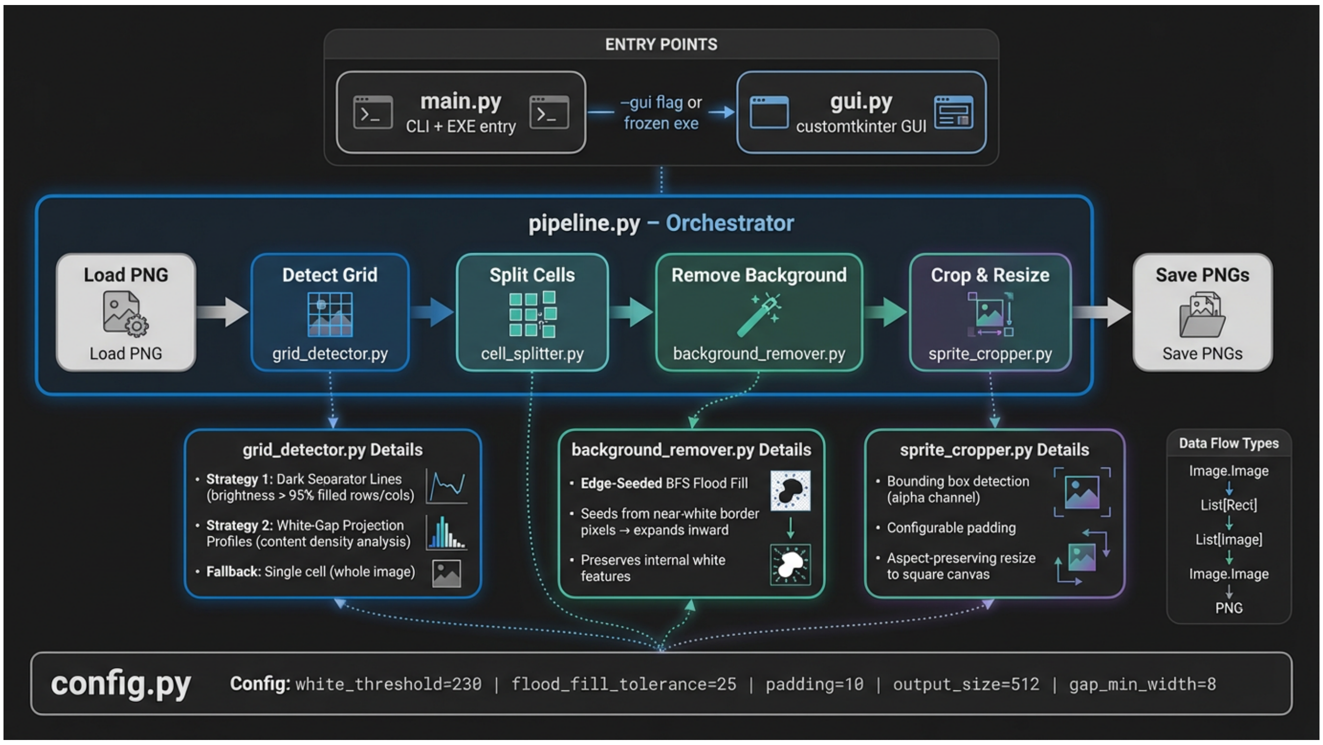
```
python main.py --input "sheet.png" --output "output/"
```

All CLI flags

Flag	**Default**	**Description**
--gui	off	Launch the graphical interface instead of CLI mode.
-i, --input	--	Path to a single PNG file or a folder of PNGs.
-o, --output	--	Output directory for extracted sprites.
--size	512	Target square size in pixels (0 = keep original).
--padding	10	Transparent padding around cropped sprite.
--white-threshold	230	RGB value above which a pixel counts as white.
--flood-tolerance	25	Color tolerance for flood-fill expansion.
-v, --verbose	off	Enable debug-level logging.

5 Architecture Overview

The tool follows a modular pipeline architecture. Each processing stage is implemented in its own Python module, making it easy to test, debug, and extend.


ENTRY POINTS
main.py
CLI + EXE entry
–gui flag or frozen exe
gui.py
customtkinter GUI
pipeline.py – Orchestrator
Load PNG
Load PNG
Detect Grid
grid_detector.py
Split Cells
cell_splitter.py
Remove Background
background_remover.py
Crop & Resize
sprite_cropper.py
Save PNGs
Save PNGs
grid_detector.py Details
Strategy 1: Dark Separator Lines (brightness > 95% filled rows/cols)
Strategy 2: White-Gap Projection Profiles (content density analysis)
Fallback: Single cell (whole image)
background_remover.py Details
Edge-Seeded BFS Flood Fill
Seeds from near-white border pixels → expands inward
Preserves internal white features
sprite_cropper.py Details
Bounding box detection (aipha channel)
Configurable padding
Aspect-preserving resize to square canvas
Data Flow Types
Image.Image
List[Rect]
List[Image]
Image.Image
PNG
config.py
Config: white_threshold=230 | flood_fill_tolerance=25 | padding=10 | output_size=512 | gap_min_width=8

Module Summary

Module	Responsibility
main.py	Entry point: parses CLI args, dispatches to GUI or CLI pipeline.
gui.py	customtkinter GUI with dark Cursor-style theme.
pipeline.py	Orchestrates the full extraction workflow for images/folders.
grid_detector.py	Detects grid layout using two detection strategies + fallback.
cell_splitter.py	Crops an image into sub-images based on detected grid cells.
background_remover.py	Edge-seeded BFS flood fill to make white backgrounds transparent.
sprite_cropper.py	Tight-crop to content bounding box + aspect-preserving resize.
config.py	Shared Config dataclass with default thresholds and settings.

6 Processing Pipeline

For each input PNG, the pipeline runs these stages sequentially:

- **Load:** Open the PNG file and convert to RGBA mode using Pillow.
- **Detect Grid:** Analyze the image to find the grid layout (3x3, 2x2, or single). Returns a list of (x, y, w, h) rectangles.
- **Split Cells:** Crop the original image into individual cell images based on detected rectangles.
- **Remove Background:** For each cell, run edge-seeded flood fill to make the white background transparent. White areas inside the sprite (not connected to borders) are preserved.
- **Crop & Resize:** Tight-crop each sprite to its non-transparent bounding box with configurable padding, then resize to a uniform square canvas with preserved aspect ratio.
- **Save:** Write each sprite as an individual transparent PNG into a subfolder named after the source file.

Data flow between stages

```
Image.Image  -->  List[Rect]  -->  List[Image]  -->  Image.Image  -->  PNG file
 (load)         (detect_cells)   (split_cells)    (remove_bg,       (save)
                                                   crop, resize)
```

7 Grid Detection Strategies

The grid detector tries two strategies in order, falling back to treating the whole image as a single sprite if neither succeeds.

Strategy 1: Dark Separator Lines

Many sprite sheets (especially 3x3 grids) have thin dark lines separating each cell. The detector calculates per-row and per-column brightness profiles. Rows/columns where more than 95% of pixels are non-white (below the white threshold) are flagged as separator bands. Bands thicker than 20px are rejected (likely part of a sprite, not a separator).

Cell boundaries are defined as the gaps between consecutive separator bands, ensuring that no dark separator pixels end up inside the extracted cells. This is critical for the background removal stage to work correctly.

Strategy 2: White-Gap Projection Profiles

For sheets without explicit separator lines (e.g. 2x2 layouts), the detector analyzes content density along each axis. It computes the fraction of non-white pixels per row and per column, then looks for wide bands (>= gap_min_width pixels) where content density drops below 2%.

The widest 1-2 gaps are selected, and a balance check ensures the resulting segments are roughly even (each at least 25% of the total dimension). This prevents false splits caused by narrow white gaps within a single sprite (e.g. between a pair of legs).

Fallback: Single Cell

If neither strategy finds a multi-cell grid, the entire image is treated as a single sprite cell.

8 Background Removal

The background remover uses an edge-seeded BFS (breadth-first search) flood fill algorithm. This approach was chosen specifically to preserve white features inside sprites (e.g. ice horns, pearl textures, white fur) while removing only the outer white background.

Algorithm Steps

- **Seed:** Scan all four borders of the cell image for pixels whose RGB values are all above the white threshold.
- **Expand:** From each seed pixel, expand outward using BFS. A neighbor pixel is included if all its RGB channels are within (white_threshold - flood_tolerance).
- **Mask:** All pixels reached by the flood fill get their alpha channel set to 0 (fully transparent).
- **Preserve:** Any white region not connected to the border is left untouched.

The tolerance parameter controls how aggressively the fill expands. A higher value removes more off-white pixels (useful for JPEG-like artifacts), while a lower value is more conservative and preserves light-colored sprite edges.

9 Output Structure

Each source image gets its own subfolder. Sprites are numbered left-to-right, top-to-bottom:

```
output/
  Gemini_Generated_Image_abc123/
    0.png
    1.png
    ...
    8.png
  single_sprite/
    0.png
```

Each output PNG is a transparent-background sprite at the configured output size (default 512x512), centered on a transparent canvas with preserved aspect ratio.

10 Tuning Tips

Too much of a light-colored sprite gets removed

- Lower White Threshold (e.g. 210).
- Lower Flood Tolerance (e.g. 15).

Background remnants remain around edges

- Raise White Threshold (e.g. 240).
- Raise Flood Tolerance (e.g. 35).

Unusual grid layouts

The tool falls back gracefully to treating the whole image as a single sprite. For non-standard grids, you may get better results processing individual cells manually.

Very large sprite sheets

Processing time scales with pixel count. For very large images (>4000px), expect a few seconds per sheet. The GUI shows real-time progress.

11 Building the Executable

To rebuild the standalone Windows .exe from source:

```
pip install pyinstaller
python -m PyInstaller --noconfirm --onefile --windowed \
  --name SpriteSheetCutter \
  --collect-data customtkinter \
  --hidden-import PIL --hidden-import numpy \
  --hidden-import scipy --hidden-import scipy.ndimage \
  main.py
```

The result lands in dist/SpriteSheetCutter.exe (~56 MB). The exe bundles Python, all dependencies, and the customtkinter theme data. First launch may take a few seconds as the onefile archive unpacks to a temp directory.

12 Troubleshooting

Exe won't start / shows no window

The onefile exe needs a few seconds on first launch to unpack. Wait 5-10 seconds. If it still doesn't appear, try running it from a terminal to see error output.

'No PNG files found'

Make sure the input folder directly contains .png files (the tool does not search subfolders recursively).

Sprites still have white background

This can happen if the cell borders contain dark separator pixels instead of white. The v1.0 fix ensures separator-line cells are bounded by the gaps between dark bands. If you still see this, try raising the Flood Tolerance.

Grid detected incorrectly (too many / too few cells)

The grid detector is tuned for common Gemini-generated sprite sheet layouts. For unusual layouts, you can process images individually or pre-crop them manually.