Pixelator

A utility software to convert any image into pixel art!

Purpose

The idea of this tool is to generate pixel-art sprites for old-school games, using plain images and other art styles (eg you can import different hi-res sprites and convert them into pixel art).

And of-course it can also be used for fun and to make cool media and art projects.

Licenses

Pixelator can have 3 type of licenses:

Unlicensed

This is the default state after installing Pixelator. Without any license all the images generated from Pixelator are under the CC BY-NC 4.0 license and you cannot use them for commercial projects. In addition, there's a watermark on generated pics.

Indie

This free license removes the watermark from the generated pictures, but they are still under the CC BY-NC 4.0 license. Meaning you cannot use this for commercial projects.

Studio

Paid license that removes the watermark and allow commercial usage (you choose the output license), but does not allow to use Pixelator as a service or embed it as a feature in a bigger app or website.

Usage

In this folder you will find two main files: Pixelator.exe and _pixelator_cmd.exe.

Pixelator.exe is the main Windows app that provides a friendly UI and an option to save and load projects. This is the main file you should normally run.

_pixelator_cmd.exe is the engine itself, a simple program that do all the image processing and designed to be executed from shell. Use this option if you want to automate the process or use values the UI doesn't support.

The _pixelator_cmd.exe program accept the following arguments:

```
  infile                  Input file name.
```

```
  outfile                 Output file name.
  --pixelate              How much to pixelate the image [default=5].
  --colors                How many colors to allow in output palette. Set 0 to
disable palette [default=9].
  --enhance               Enhance colors by this factor. [default=2.0].
  --refine_edges          Every pixel with alpha above this value will turn opaque.
Any pixel with alpha below it will turn transparent. [default=250].
  --stroke_opacity        Stroke opacity. [default=1.0].
  --stroke                Stroke type. [default=outside].
  --stroke_diagonal       If stroke is enabled, will add stroke for diagonal
corners as well [default=False].
  --stroke_on_colors_diff Add stroke on colors diff based on this factor
[default=0.0].
  --smooth                Threshold to remove floating pixels and other undesired
artifacts [default=3].
  --smooth_iterations     How many times to execute the smoothing filter
[default=1].
  --background            Optional background color for transparent pixels
[default=None, format: #rrggbbaa].
  --resize                Resize output image to optimal size [default=False].
  --palette_mode          {adaptive,mixed,web,dither} Colors palette mode
[default=adaptive].
  --fit_size_only         Ignore all settings and only resize the image to a
recommended size [default=False].
  --override              If you want the output file to override the source file,
you must add this flag.
```

For example, the following will convert a picture with all the default params:

```
>python pixelator.py infile.png outfile.png
```

The following is a detailed explanation of all the arguments and possible values.

infile

Path for input image file to process. Recommended format is .png with transparent background (some effects, like stroke, only work with transparent background).

outfile

Path for the output / result file. Always use .png extension as the output image will have opacity and cannot be saved as jpeg.

pixelate

Determine how much to pixelate the picture. Changing this factor have a huge effect on the result. For example, a value of 6 means that every group of 6 pixels will turn into 1 pixel in the result image.

Note that the source image original size remains unless the `--resize` flag is present (explained later).

colors

Determine how many colors the output picture will have. Changing this factor have a huge effect on the result. For example, a value of 8 means that the final picture will only have 8 different colors (not including stroke that might affect it).

enhance

To give the result a more vibrant and game-like colors, this filter will enhance colors and increase saturation. Setting this value to 1.0 or 0.0 will disable it, above 1.0 will increase saturation while values between 0.0 and 1.0 will decrease saturation.

refine_edges

Remove all partly-transparent pixels and force all pixels to either be opaque or transparent. This factor determine the threshold for opacity. If you don't want this to happen, set this argument to 0. But note that strokes may look broken if you have half-transparent pixels.

stroke

Type of stroke to add to the result. Note that stroke only works with transparent pixels, it will not work on a picture with a background.

The following options are valid:

inside

A stroke inside the visible pixels. If the stroke have opacity, it will merge with the pixels below.

outside

A stroke outside the visible pixels.

left

Stroke only on left-side pixels.

right

Stroke only on right-side pixels.

sides_in

Stroke on both sides (left and right), inside the visible pixels.

sides_out

Stroke on both sides (left and right), outside the visible pixels.

none

Will disable stroke completely.

stroke_opacity

Transparency for the stroke effect. If its an inside stroke, this will cause merging with the layer below.

stroke_on_colors_diff

Add internal stroke based on difference between colors, and not just opacity.

stroke_diagonal

If this flag appears, stroke will also take affect on diagonal pixels. This creates a 'thicker' outline, slightly less common with pixel art sprites.

smooth

Sometimes there are random pixels floating around or sticking off edges, and due to the color reduction sometimes artifacts are created on merging points between colors. This factor tries to eliminate those floating pixels by finding individual pixels surrounded by different color / opacity and making them match the surroundings.

The bigger this factor is the more aggressive the cleanup would be.

smooth_iterations

How many times to execute the smoothing filter. Each time it runs it will smooth even more from the previous pass.

background

Optional background color to set instead of transparent pixels.

resize

Since the result is pixelated, the output image uses less pixels than the original image. If this flag appears it will resize the result image to be as small as possible so there would be no duplicated pixels. Its recommended to always set this flag, unless you need to result to be the same size as the input.

palette_mode

How to handle colors palette.

Options are:

adaptive

Adaptive colors palette based on colors occurances.

web

Use the default web colors palette.

mixed

A mix between adaptive and web palettes.

dither

Web palette, with dithering effect.

Tips

To get the best results, its recommended to follow these guidelines:

1. Use png files with transparent background, especially if you want strokes.
2. If the source pic colors are faded, increase contrast before using pixelator.
3. Try to clean "noises" like a point of direct light on someone's face or leftovers in background. Sometimes these things make it to the final result and manifest themselves as rogue pixels you don't want.
4. If you use high pixelate factor for someone with short hair sometimes in the result he will appear bald (because none of the hair made it to the final pixels). To avoid that, you should stretch the person's hair up before processing it in pixelator.
5. Don't use images too big.
6. Try to use pictures with less background details. If the source pic is full of colors and small items all around, it will make a bad colors palette.
7. If you can't get the colors right sometimes it helps to posterize the source image manually before processing it in pixelator.

Support

For support, please email ronenness@gmail.com.

To see more of Ronen Ness' works, please visit

ronenness.com