ZT Studio

Contents

[About this document 1](#_Toc419462475)

[About the program 1](#_Toc419462476)

[What is ZT Studio? 1](#_Toc419462477)

[Features 1](#_Toc419462478)

[Credits 2](#_Toc419462479)

[Requirements 2](#_Toc419462480)

[Support 2](#_Toc419462481)

[Distribution 2](#_Toc419462482)

[Setting up ZT Studio 3](#_Toc419462483)

[Settings 3](#_Toc419462484)

[Paths 3](#_Toc419462485)

[When writing ZT1 Graphics: 3](#_Toc419462486)

[When writing PNG Graphics: 3](#_Toc419462487)

[Conversions (also see “When writing PNG Graphics”). 4](#_Toc419462488)

[Frame editing 4](#_Toc419462489)

[Command line 5](#_Toc419462490)

[Quick how-to’s 6](#_Toc419462491)

[Basic functionality 6](#_Toc419462492)

[Batch convert files ( ZT1 to PNG and PNG to ZT1 ) 6](#_Toc419462493)

[From ZT1 Graphics to .PNG-files 6](#_Toc419462494)

[From .PNG-files to ZT1 Graphics 6](#_Toc419462495)

[Creating objects of which you can change the color in-game 7](#_Toc419462496)

[Considerations 7](#_Toc419462497)

[Graphics 8](#_Toc419462498)

[Configuration 9](#_Toc419462499)

[File formats 10](#_Toc419462500)

[Overview 10](#_Toc419462501)

[Different types of ZT1 Graphics 10](#_Toc419462502)

[Explanation of the ZT1 Color palettes (.pal) 11](#_Toc419462503)

[Explanation of the ZT1 Graphic file formats 12](#_Toc419462504)

[Terminology 15](#_Toc419462505)

# About this document

This document is work in progress. For now, I’ve ignored the simple stuff (‘this button allows you to open a ZT1 Graphic’). I’m focusing on writing down information which might not be so easy to find out. I’m also trying to write this in an understandable manner.

# About the program

## What is ZT Studio?

ZT Studio is a program created by a long time fan of the original ZT1 game. It development happened during January 2015, so very long after the game was published. The once so vibrant community is a former shade of itself, but some people – including myself – still love playing this game.

ZT Studio was created to allow designers to create ZT1 Graphics a lot faster than with APE or ZOOT. Its feature set really focusses on creating, editing and exporting ZT1 Graphics from and to .PNG-files. It includes features not available in APE nor ZOOT.

If I had programmed it earlier, it would definitely have been a full blown replacement for APE, also featuring configuration file editing (.ai, .uca, .ucs etc). But now, it has not been included in the scope of this project.

## Features

An overview of **most** of the features.  
Also see “File Formats” to get a better idea.

* **ZT1 Graphic file**
  + Open
  + Save
  + Create from a series of .PNG-files
  + Export to a series of .PNG files
  + Change animation speed
* **ZT1 Color palettes**
  + Open
  + [Save] – currently not possible, only as part of creating a ZT1 Graphic
* **Frames**
  + Add
  + Delete
  + Change offset / “rotation fixing” : up, down, left, right
  + Reorder
* **Preview**
  + Change background color
  + Add background graphic
    - E.g. the “used” version of the rope swing (object) as background,   
      when viewing the Orang Utan’s “rpswing” animation (animal)
  + Color replacement
    - E.g. what does the restaurant look like with a blue roof?
* **Tools**
  + Batch conversion
    - Select a folder and convert ZT1/png from/to ZT1/png
* **Command line**
  + For more tech savvy people, there are some command line options.

## Credits

This tool was programmed by **Jeffrey** **Bostoen**. No support will be offered. You can report bugs to [Jeffrey\_bostoen@hotmail.com](mailto:Jeffrey_bostoen@hotmail.com) and hope they’ll get fixed, but there’s no guarantee. Questions about how to use the program, will not be answered. Ask them on a ZT1 community board.

It was mostly based on information provided by **MadScientist** (ZOOT’s creator) and some pointers by **Jay**.

And of course **Blue Fang** created the game, while **Microsoft Game Studios** published it.

## Requirements

You need **Microsoft .NET Framework 4.5**.

## Support

We are not offering any support for this program.

## Distribution

Do not distribute this program without permission from the author.  
Requirement for distribution: you will keep the version as up to date as possible.

# Setting up ZT Studio

## Settings

Hit the **Settings**-button on the **Misc.** toolbar.

You could also edit the *settings.cfg* file, which has a few additional options, but those are not listed below because they shouldn’t be tampered with at this point.

### Folders

**Root folder:** this is a top-level directory. What do we mean by this? ZT1 has a few folders which have a certain structure. For example, you have folder structures like:  
*<root folder>*\animals\abcd1234\m\walk   
*<root folder>*\objects\abcd1234\idle\idle

So basically, if you download any valid .ZTD-file (ZT1 download), you could unzip it to any location. The path of that location is what we mean by <root folder>.

### Conversions

**Start numbering of .PNG-file series at either 0 (index) or 1 (“frame #1”).**Default: 0  
ZT1 Graphic files contain at least 1 frame, but often multiple ones.   
When using the “batch conversion” feature, a ZT1 Graphic file named “N” will be converted into a series of PNG-files (one PNG file for each frame): N\_0000.png, N\_0001.png, N\_0002.png, N\_000x.png …

In programming, we often start counting at 0. Also, when creating exports from certain 3D programs, by default a series of rendered files might start at 0.

For easy reference, you might however prefer to count starting from 1.

**Delete the source file after conversion   
(e.g. converting from .PNG-files to ZT1: .PNG files will be deleted)**Default: enabled.Performs a clean-up of the original files.

**~~Overwrite files~~**~~Default: enabled.  
ZT Studio will overwrite all existing files. No error messages or warnings will be shown.~~ZT Studio is designed to overwrite all files without warning at this point!

### Rendering frames:

**Render the extra frame in all other frames   
(in case of a ZTAF-file with an extra frame, e.g. restaurant):**Default: enabled.Some ZT1 Graphics have one extra frame. This extra frame contains pixels which are rendered in all previous frames. For example, in the ZT1 Graphic file of the restaurant, the first frames mostly draw only the smoke coming out of the chimney – nothing else. The last/extra frame has the non-variable parts of this graphic, in this case: the building.

**Render background ZT1 Graphic   
(e.g. main graphic = Orang Utan swinging, background: rope swing toy):**Default: enabled.In ZT Studio, you can open a ZT1 Graphic file – let’s say: the swinging animation of the Orang Utan. This is only a part of the complete animation though. It is a trick to overcome a limit in ZT1. Each animation is limited to 255 colors you can freely choose. This way, the game designers found a way to have up to 510 colors for this animation though, by combining 2 animations (2x 255 = 510). Enabling this option will make sure, when you export a ZT1 Graphic to another format, that you have the complete rendering.   
Do keep in mind that you will need to specify the main graphic and the background graphic!

### When writing ZT1 Graphics:

These settings are used when ZT Studio converts graphic files (let’s say .PNG-files) to the ZT1 Graphic file format.

**Add “ZT Animation File”-bytes even if there’s no “extra frame”:**Default value: checked.  
This is a detail in the file format of ZT1 Graphic files. Often, the first bytes in the format spell “FATZ”. But we should read them as “ZTAF”, which probably stands for Zoo Tycoon Animation File. These bytes are present in some official files, but not in all. It’s most commonly used in files where there is an extra frame in the file format, such as in graphics of the original Restaurant. For the explanation about this extra frame, please refer to documentation about the ZT1 file format.

**Create/update .ani-file**:Default value: checked.  
This option will try to generate a new .ani-file in a folder. It will do so based on the names of ZT1 Graphics it detects in the folder. It will look for 4 typical configurations which are most frequent in ZT1 (there are a few exceptions though, e.g. dust cloud):

* N  
  All sorts of icons
* NE/NW/SE/SW  
  All sorts of objects, buildings, foliage etc.
* N/NE/E/SE/S  
  All sorts of moving creatures, such as animals, guests, staff
* 1, 2, 3, … , 20  
  Paths

This is currently considered an experimental feature.

### When writing PNG Graphics:

These settings are used when ZT Studio converts ZT1 Graphic files to a series of .PNG-files.

**Canvas:**Default: Keep canvas size.  
These settings are all related to the image size of the .PNG-files that will be created.  
With “image canvas size”, we mean the width and height of the preview area, in pixels.  
With “relevant pixels”, we mean all pixels which are not transparent (so: colored.)

* **Keep canvas size (<variable image canvas format>)**This will create .PNG files which have the same size as in the preview window.   
  If you have a very small ZT1 Graphic, the image canvas will still be the same as for any other ZT1 Graphic you might export with this setting. No need to worry about file sizes, PNG is a very good format and the size remains quite small. This option is recommended. If you export a PNG this way and reimport it, you won’t need to re-adjust any offsets.  
  🡺 If you have a ZT1 Graphic of 5x5 pixels, your PNG will be <variable image canvas size>
* **Crop to largest relevant width / height in this graphic**This setting will crop each .PNG file to only the relevant canvas size, considering all the frames in the ZT1 Graphic.  
  🡺 This takes all frames in consideration. Based on the most top left relevant pixel of all frames, and the most bottom right one, all PNG files will be cropped to this size.
* **Crop to relevant width / height of this frame**This setting will crop each .PNG file to only the relevant canvas size of the particular frame.  
  🡺 If the graphic in your frame consists of 5x5 pixels, your PNG will be 5x5 pixels.

### Editing graphics

**Adjust offsets of a single frame (instead of all frames in the graphic) when "rotation fixing"**Default: disabled.Rotation fixing is a term which comes down to adjusting offsets. In simple words: most new graphics you import in ZT1, are not in the right position. Adjusting is necessary: move the graphic up/down, left/right.

This can be done separately for each frame, if you enable this.  
Most of the times however, you’ll want to adjust all frames simultaneously.

## Recommendations

ZT Studio was primarily designed to convert graphics from one format to the other in batch:   
ZT1 Graphic files ⬄ series of .PNG-files. It’s performing best in that way.

For this to work, ZT Studio expects the **root folder** to be set (see *Settings*).  
Under the root folder, you should have something like this for a basic non-animated object:

|  |  |
| --- | --- |
| **Example of a ZT1 Graphic:** objects\abcd1234\idle\NE objects\abcd1234\idle\NW  objects\abcd1234\idle\SE objects\abcd1234\idle\SW  objects\abcd1234\idle\idle.ani objects\abcd1234\abcd1234.pal | **Example of .PNG-files (only 1 frame / view ):** objects\abcd1234\idle\NE\_0000.PNG objects\abcd1234\idle\NW\_0000.PNG objects\abcd1234\idle\SE\_0000.PNG objects\abcd1234\idle\SW\_0000.PNG |

Using the batch conversion feature, you can convert a ZT1 Graphic with all its views (NE, NW, SE, SW) to .PNG-files, or the other way around.

## Command line

Untested. This was what we had intended.

|  |  |  |
| --- | --- | --- |
| /crop | 0 1 2 | Keep canvas size. Crop to relevant pixels of frame. Crop to relevant pixels of graphic. |
| /cleanup | 0  1 | Delete original file |
| /convertFile | “<filename>” | File to convert |
| /convertFolder | “<folder>” | Folder to convert. Expects a root folder. |
| /startIndex | 0 1 | Start numbering of frames at 0 or 1 in the PNG filenames ( SE\_0000.png or SE\_0001.png ) |
| /folderRoot | “<folder>” | Root folder. |

# Quick how-to’s

## Basic functionality

To be described. Will be done at a later stage, since you can figure out most of this quite easily.   
This will simply document what buttons do: save/open files etc.

It will also show slightly more advanced stuff, like seeing how the Orang Utan swings on the ropes.

## Batch convert files ( ZT1 to PNG and PNG to ZT1 )

This is by far one of the most useful features of ZT Studio. It allows you to save quite some time.

Currently, ZT Studio features two types of batch conversions:

* From ZT1 Graphics to .PNG-files
* From .PNG-files to ZT1 Graphics

A PNG file supports transparency, but no animation.  
A ZT1 Graphic also supports transparency. In some cases, there’s only 1 frame, in other situations there are multiple frames.

If you choose to perform a batch conversion, there are a few different options which you can adjust to your wishes. For more information, please read the chapter “Setting up ZT Studio”.

Most of you can simply choose what kind of conversion you want to perform.

### From ZT1 Graphics to .PNG-files

For each ZT1 graphic in the root folder, ZT Studio will generate a .PNG-file for each frame.

NE, frame 1 🡺 NE\_0000.PNG   
NE, frame 2 🡺 NE\_0001.PNG

( You can also go to ZT Studio’s settings to start counting at 1 rather than at 0 )

### From .PNG-files to ZT1 Graphics

You have seen the principle above. This works just the other way.   
It is important to properly name your .PNG-files: *name\_0000.PNG, name\_0001.PNG*  
So: <desired name of the graphic>\_<frame number>.PNG

NE\_0000.PNG 🡺 NE, frame 1   
NE\_0001.PNG 🡺 NE, frame 2

( You can also go to ZT Studio’s settings to start counting at 1 rather than at 0 )

## Creating objects of which you can change the color in-game

Here I’ll document steps on how to create an object of which the user can select the color.

Keep in mind that for some steps, you will need proper software for image manipulation. We are using **GIMP** in this example. GIMP is a free and open source alternative to Photoshop.

The concept is not that difficult, but I do recommend that you’ve got a good understanding of ZT1 buildings.

Example: Jeff’s Umbrella table.

### Considerations

Before I share the basic instructions, there’s a few things to take into account.  
If you are creating an item where the user will be able to alter the colors in game, there are a few restrictions to keep in mind.

* ZT1 has a maximum of 255 (+1 transparent) colors per graphic.
* There’s a maximum of 2 sets of colors which can be replaced.  
  A typical graphic of this kind will offer you this:
  + The first color range / palette consists of 16 colors
  + The second color range / palette consists of 8 colors
  + If you have 2 palettes, your range of colors which can’t be changed, is limited to (256 – 16 – 8 =) 231 colors (+ 1 transparent color).

It is recommended to use two colors which are very easy to spot. For example: very fluorescent yellow and pink. These colors will help you to create a color palette in a very specific way. In the game, you’ll never see those two colors anyway – they will be replaced by default colors.

You can also choose to only use one color palette. I’m not 100% sure if you can choose whether you use only the large or only the small version. You might also be able to use 2 color palettes of 16 colors.

And last but not least, very important: all your 4 views will have to share one .pal file.

To sum up:

* A graphic can contain maximum 2 color ranges of which the color can be replaced.
* A custom color palette consists of either 8 or 16 colors. This is the number of colors you will need to define in a color range in your graphic’s color palette. However, you are not required to use all of them.
* All your views need to use the same .pal file

### Graphics

So, what does this lead us to?

* Create the images you’d like to import – in your favorite software.   
    
  For this example, we’ve combined the picnic table with an umbrella we’ve extracted from a Gift Cart. In this example, we will work with two colors the user will be able to determine in the game.   
    
  Since this was a graphic to experiment, it was nice to have a part of a graphic of which users can already pick the colors in game. The key point here is to use 2 colors which aren’t used in any other parts of the graphic (e.g. our umbrella is green and yellow; while the bench is mostly brown. If you design from scratch, pick e.g. pink and bright yellow. You’ll never see these colors in game.).
* Use ZT Studio to create a new view of your object, based on the .PNG image you created in the previous step.
* ZT Studio should have generated a color palette for you.   
  Now, make sure you group each of the two colors at the bottom of this newly generated color palette. In ZT Studio’s main window, you can right click on any color in the color palette displayed at the right. This will show you a small menu, which will allow you to move the color you just right clicked on, all the way to the bottom.
* Save this graphic once again. This step is necessary to save your adjusted color palette and to have ZT Studio write out the graphic correctly, since the order of your colors has changed.
* Bonus tip: Specifically for this example: we have 4 views. We only designed one though, and there’s a flipped version. This means: they all use the same colors. So using ZT Studio, we imported the second graphic as well. ZT Studio again generated a color palette. But then we opened the color palette of our first view. From there, we used a feature to replace all colors in the automatically generated color palette. We saved this graphic. We did the same for the two other views. With some clever HEX-editing (or in future versions: using ZT Studio), you can then simply have all graphic files point to the same color palette and remove the 3 other color palette files.

### Configuration

I’ll restrict this to the minimum. Your object will need to be configured as a building.

*[Characteristics/Integers]  
cIsColorReplaced = 1*

Add the bottom, add:

[colorrep]  
*; cr\_color is listed below  
color = cr\_color  
; cr\_part1 is listed in building.ai  
replace = cr\_part1  
title = 2300  
defaultpal = scenery/building/pals/brwn16.pal*

*; cr\_part2 is listed in building.ai  
replace = cr\_part2  
title = 2301  
defaultpal = scenery/building/pals/gold8.pal*

*[cr\_color]  
ncolors = 210  
fullpal = objects/umbtable/idle/SE.pal  
colorpal = objects/umbtable/idle/SE.pal*

So basically, in the *[colorrep]* part you will see a reference to the *[cr\_color]-*block. That’s where we simply refer to our color palette. In the original ZT1 files, there is a slight difference.   
*Fullpal* will refer to a .pal file containing all colors.   
*Colorpal* will only contain the colors which you can’t replace in the game.  
*ncolors* is the total number of colors which you can’t replace in the game. Yes, this means we do not even use 255 (+1 transparent) colors in this graphic, otherwise *ncolors* would have been 232.

As for the other configuration: you can specify the default in game colors, for example: we want the large custom color range to be brown, and the smaller one to be gold. “Title” refers to a string ID.

For more examples, refer to the graphics and configuration of original buildings.

# File formats

## Overview

The tool can read and write the following formats:

* **ZT1 Graphic**. These files have no file extension.
* **ZT1 Color Palette** **(.pal)**.
* **PNG**. An image format which has a very good compression level and supports transparency. It only contains a single image = 1 frame.

## Different types of ZT1 Graphics

There are different kind of ZT1 Graphics. This is the list of versions we’ve found so far.   
It is possible that there are more.

* **Basic** objects/bamboo/idle/SE
* **ZTAF, no background frame**
* **ZTAF, background frame** objects/restrant/used/SE
* **ZTAF, compressed** objects/dolphin/m/ssurfswi

The first two types are identical, except for 9 bytes.

**Read***APE* crashes as soon as there’s an extra frame.  
*ZOOT* only renders the first frames, but it does not render the “background frame”.  
*ZT Studio* can render all of them.

**Write**  
APE and ZOOT can only write basic files.  
ZT Studio can write all file types, except for the compressed format (at this time). The compressed format is of very little use to designers, so it was decided not to invest time in allowing people to create this type of graphic at this point. It can be easily implemented though.

## Explanation of the ZT1 Color palettes (.pal)

The structure of the color palettes is quite easy.

Simply put: a **color palette** is an **overview**, which **lists all unique colors used in a graphic**.

The **first 4 bytes will tell you how many colors are defined in the palette**. The format of these files is in Little Endian though, which means you’ll have to reverse the bytes.

Then, you will have **4 bytes for each color**. These are the **RGBA** (red, green, blue, alpha) values of each color. The alpha level is typically ‘FF’. You will see that only the first color might have a transparency byte of ‘00’.

Tools like APE and ZOOT (?) typically fill a palette with 255 colors (+1 transparent color) in any case, even if your graphic doesn’t have 255 distinct colors.

## Explanation of the ZT1 Graphic file formats

For questions, I’d refer to <http://www.ztcdd.org/DD/index.php?topic=7558> .

**Optional: 9 bytes, which define the file type**

The different types can be identified quite simply.

If the first 4 bytes spell “FATZ” (or reversed: ZTAF), then the game will consider this as an animation file. The next 4 bytes will all be 00. The 9th byte will either be 0 (no extra frame) or 1 (extra frame = background frame).

What is this extra frame? To render some graphics faster and to compress the files, things which are always exactly the same in an animation, are stored in a background file. The restaurant is an example of this: all the animation frames simply contain the smoke coming out of the chimney, the extra frame is the non-animated part of the building.

* **ZTAF, no background frame** example: <example needed>
* **ZTAF, background frame** example: objects/restrant/used/SE
* **ZTAF, compressed** example: objects/dolphin/m/ssurfswi

If no such header is present, it is a very basic ZT1 Graphic.

* **Basic** example: objects/bamboo/idle/SE

**Required: 4 bytes, animation speed**

A number which defines how much milliseconds pass before the next frame is shown.  
Even if your

I’ll give one example on how to read this value and every other value which is mentioned.   
Values are stored in Little Endian format. In human language: they’re stored in a reversed way.

Let’s say these bytes read "7D 00 00 00". In the above example becomes "00 00 00 7D”.   
If you convert this from hex to decimal, you get 125 milliseconds.

**Required: 4 bytes, length of relative path and filename of color palette**A value.

**Required: X bytes, one for each character in the relative path and filename of color palette.**You also need to add 00 at the end.

**Required: 4 bytes, number of frames**A value. A graphic contains minimum 1 frame.

**Required: 2 bytes, number of frames**A value. A graphic contains minimum 1 frame.

**For each frame:**

* 4 bytes: these list how many bytes will follow, defining the frame.
* 2 bytes: value for the height of the image.
  + Special case: if the second byte is 80, we’re dealing with the compressed format.   
    This compressed format is used for certain shadows.
* 2 bytes: value for the width of the image.
* 2 bytes: defining the vertical offset
* 2 bytes: defining the horizontal offset
* 2 bytes: these are the only two bytes we haven’t been able to identify yet.  
  You can use a default value of 01 00. We’ll refer to these two bytes as mystery bytes.  
  We’ve considered a few things, but it is …
  + Not a signature
  + Not a length of the number of bytes in the frame
  + Not a length of the number of color blocks in the frame
  + Not a length of the number of colors in the frame
  + Not a length of the number of colored pixels in the frame
  + …
* **For each horizontal line of pixels:**
  + 1 byte: lists the number of color instruction blocks.  
    There are a few types of instruction blocks.  
    - **Typical block:**Example: 07 02 15 13   1 byte, determining an offset (‘move 7 pixels to the right, because the first pixels are transparent’).  
      1 byte, determining the number of colored pixels (2) to follow.  
      1 byte for each color which needs to be drawn. This byte (15, 13) refers to the index of the color in the ZT1 Color palette.
    - **End block:**  
      Example: 08 00  
      The last block in a line is usually an offset (move 8 pixels to the right – not drawing any colored pixels), followed by 00 (no more colors to draw).
    - **Shadow block:**  
      Example: 40 02  
      This block is only found in the compressed graphic format. Only one color is needed for this kind of graphic: black. It’s used for some shadows.   
      1 byte refers to the offset (‘move 40 pixels to the right’).  
      1 byte determines the number of black pixels to draw.

This series of bytes should be repeated for each frame.  
So next would be the value of the number of bytes which define frame #2, then you’d get the height and width again, and so on.

By now, you have defined your graphic. You might stumble upon files where there are still more bytes at the end of the file.

**In files generated by APE:**  
Sometimes, there seem to be random bytes which are useless. They are ignored by the game, because the actual relevant pixels were defined earlier on.

## Explanation of the ZT1 .ani file format

.Ani most likely stands for animation. It is a file which you can open with any text editor.   
You will notice an *[Animation]*-header, *dir*-lines (dir0, dir1, dir2, … dirN) which represent the path structure to this file, starting from the root directory, *animation-*lines for each view in a graphic,X and Y offsets (x0, y0) and coordinates for the bottom right pixel ( x1 = x0 + image width; y1 = y0 + image width). Basically you are defining the canvas.

For icons, plaques, sold items… it’s quite simple. There’s only one ‘view’, “N”.  
So the coordinates only refer to that one view. The game seems to take them into account.

For objects and moving creatures (animals, guests, staff), it’s more complicated. It’s not exactly clear which view (or perhaps the combination of them all) define the canvas. The game might completely ignore the .ani-files.

*[animation]  
dir0 = objects  
dir1 = restrant  
dir2 = used  
animation = NE  
animation = SE  
animation = SW  
animation = NW  
x0 = -138  
y0 = -176  
x1 = 138  
y1 = 72*

As for possible *animation-*lines, we have seen a few types:

* **1** ZT1 Graphic: **‘N’ ,**   
  e.g. list icon, plaque, sold item, purchase menu icon, small/medium research icons …
* **4** ZT1 Graphics: **‘NE’, ‘SE’, SW’, ‘NW’**.  
  e.g. objects
* **5** ZT1 Graphics: **‘N’, ‘NE’, ‘E’, ‘SE’, ‘S’.**e.g. animals, guests, staff
* **20** ZT1 Graphics: **‘1’, ‘2’, ‘3’, ... , ‘20’**  
  e.g. paths

# Terminology

* **Animation:** an animation is like a video: it consists of a series of frames.   
  You can compare a frame with a photo. If you play it fast enough, it will look like a small movie.
* **APE:** Animal Project Editor. A basic editor which was released by Blue Fang. It helped users to create lots of new content, based on existing game content. You could for example pick an in-game statue, and you can configure a few things (e.g. price) and change its graphics. This way, you’d create new content for Zoo Tycoon.
* **Color palette:** a collection of distinct colors. By distinct, we mean that we only have one of each color in the collection.
* **Frame:** an animation consists of multiple frames. You can compare a frame with a photo.
* **GIMP:** Graphic Image Manipulation Program.   
  A free and open source alternative to PhotoShop.
* **PNG**: a common file format (Portable Network Graphic) which supports transparency.
* **Rotation fixing:** when you have created a new graphic of e.g. a statue for ZT1, you’ll often find that it is not correctly positioned in the game. You will have to use a tool to move it to the left/right/top/bottom. Also called offsets, offsetting.
* **View:** ZT1 has a few different views in-game: S, SE, SW, N, NE, NW, E.   
  NE means North East, and you also see S for South, W for West. You might be wondering why some directions are missing. This is because ZT1 cleverly mirrors images. When a view for “West” is needed, ZT1 simply uses the “E” (East) view and mirrors it in-game.
* **ZIP file:** this is a file format, which you can compare with a folder. Each .ZIP-file is basically a folder you can open. It is mainly used to compress data and to easily share files on the internet.
* **ZOOT:** a tool developed by MadScientist, a fan. It focuses solely on ZT1 Graphics. You can export graphics to .PNG-files. It also supports “rotation fixing”.
* **ZT Studio:** the name of this tool.
* **ZT1 Color palette**: each ZT1 Graphic file comes with a color palette. It’s a file with a .pal file extension. A color palette theoretically contains a maximum of 256 colors – not all color palettes have this many colors, but the first one is reserved and determines the transparent color. So you can have up to 255 different colors per graphic which is linked to this color palette. A color palette can be used by multiple ZT1 Graphic files.
* **ZT1 Graphic file**: a graphic file in a format which is only used for Zoo Tycoon. This file represents a view in the game. Strangely enough, this file format has no file extension at all.
* **ZTD file:** this is simply a ZIP-archive. The extension of .ZIP has been changed to .ZTD, which stands for Zoo Tycoon Download.

## Changelog

Currently, the tool is at version **1.0**. This reflects that it is the first main version intended for public release. The last part translates in the year, month and day of its compilation.

E.g. 1.0.2015.0511 = version 1.0, last updated 11th of May 2015

### Version 1.0

**Build 2016.08.27:**

From this build onward, the code is released/updated on GitHub as well.

Fixes:

* Previous / next frame buttons didn’t work (the slider did)
* Setting was stored incorrectly   
  - delete files vs. overwrite files during conversion
* Last used directory per ZT1 Graphic and per PNG-file is what will be opened when opening/saving another one of those files.
* .PNG files could not be deleted automatically since apparently there was a file lock after reading them in ZT Studio
* The transparent color was not correctly rendered in the color palette   
  (despite being properly written to a .pal file and internally being read correctly)

Improvements:

* Minor improvements in GUI
* The transparent color is no longer by default pink.   
  It is now automatically the background color you prefer.

**Build 2015.0815:**

Fixes:

* Some performance improvements. Less triggers for ZT Studio to update certain information.  
  Definitely helpful when the .ani-files are generated by ZT Studio.

**Build 2015.0624 :**

New features:

* Create .ani-files. This option will automatically generate a .ani-file in the folder, if it detects one of the following combinations of file names:
  + N
  + NE, NW, SE, SW
  + N, NE, E, SE, S
  + 1, 2, 3, 4, 5, … , 20

Fixes and improvements:

* Exporting PNGs:
  + The button to export a frame as a .PNG image is now disabled,  
    when no existing frame has been loaded or when the frame is still empty.
  + The setting to remember which option you prefer when saving PNGs wasn’t properly remembered. (Canvas size, relevant size of frame, relevant size of graphic).
  + Cropping didn’t work, all .PNG images were rendered on the full canvas size.
* Color palettes:
  + An issue with a slow performing datagridview control has been resolved.   
    Color palettes are displayed much faster.
* Changed some labels
* Changed lay-out of *Settings*-window
* Review and changes in *settings.cfg* and in the internal naming of the settings.
* Several minor enhancements.

**Build 2015.0511:**

* No changelog available. It supports all initial features which are present in the application but which are not mentioned in newer release notes.