



# 3DNes User Guide

## Contents

1. Overview.....	1
2. General Tab.....	3
3. Setting Tab .....	4
4. Render Tab .....	4
5. Input Mapping Tab .....	5
6. Editor Tab .....	6
6.1 Pattern Parameter .....	6
6.2 Action controls.....	8
6.3 Helper functions.....	9
7. Scripting Tab.....	9
8. Hot Keys .....	10
9. Command line parameters .....	11

## 1. Overview

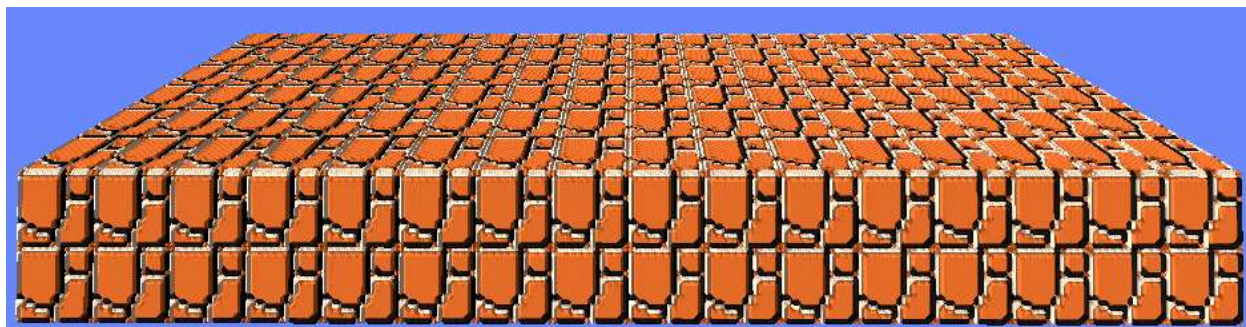
3DNes is an emulator allows you playing NES games in 3D. In a playing session it emulates the game, converts the 2d output images into a 3d world then renders it – all on the fly. While playing you can pause the game at any time and manually adjust certain parameters of the graphics to get them looking the way you want. You can then save the just modified 3D data as a 3Dn file by clicking the “Save 3Dn” button. This new data will automatically be loaded when you open the ROM again. A 3Dn file is not a ROM file, it only contains 3D data for a specific ROM. The 3Dn file must have the same name as the ROM in order for the 3D data to be paired with the ROM (e.g. a ROM named “xyz.nes” will always be paired with the 3Dn file named “xyz.3dn”). All 3Dn files are stored in the 3DNes sub folder “./3dn”.

There are concepts of 3DNes that you should get familiar with:

- Shape: in-game graphic object that can be perceived by human. For example in Super Mario Bros, Mario and Luigi are shapes, mushrooms are shapes, bullets are shapes, pipes are shapes ...
- 2DShape: the 2d representation of shape, that's what players can see in the original game. At technical level, it's more or less equivalent to meta-sprite concept.



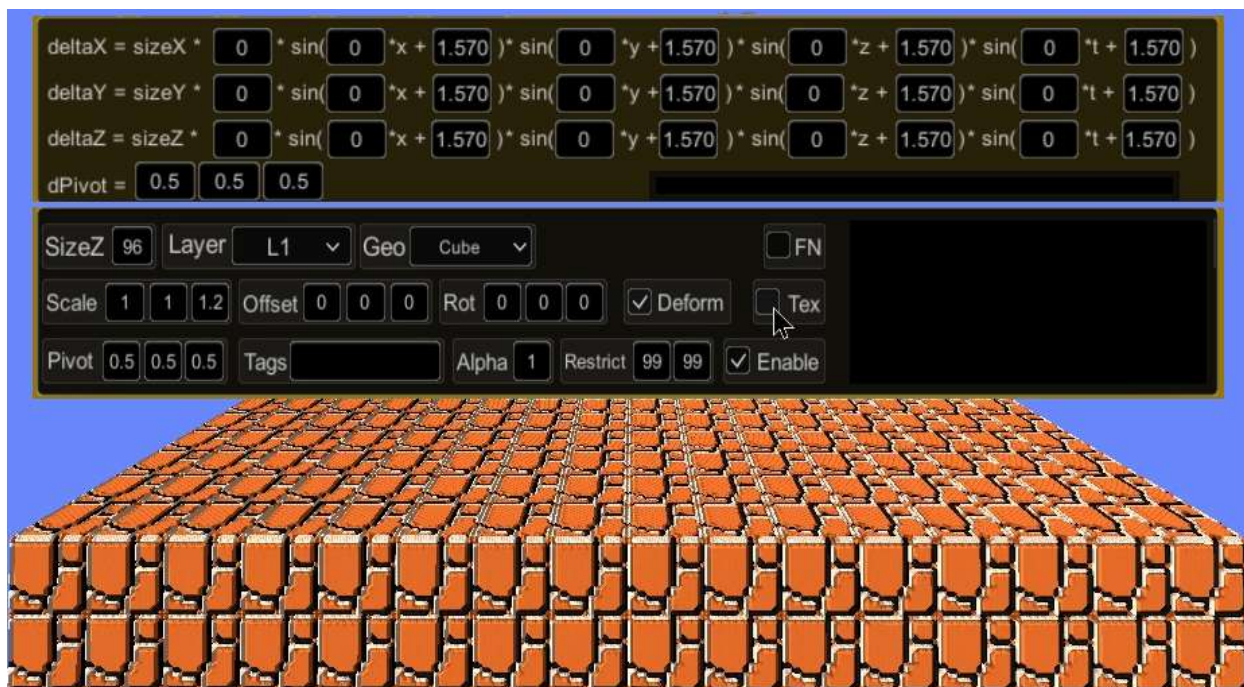
- 3DShape: the real shape in 3d world what 3DNes will create and render while emulating the game.



- Pattern: the most important concept of 3DNes. A pattern is a set of unique tiles, it defines a shape cluster constructed by that set of tile. For example in Super Mario Bros, Mario and Luigi are in the same cluster, they are basically the same with different color palettes, all the vertical pipes with different heights are in the same cluster – they are all created by the same set of unique tiles, all the clouds are also in the same cluster.



- 3DPattern: contains a set of parameter defining how shape cluster linked to the given pattern will be modeled and rendered in 3d space. By default a pattern only has a 3DPattern, it means that a 2DShape will be converted to a 3DShape. But in some cases, a pattern may have more than one 3DPattern and therefore a 2DShape linked to this pattern will be converted into a set of 3DShapes.



## 2. General Tab



- **Load**: open a rom file (nes/zip) or save state file (sav).
- **Save**: Save the current game state to a file.
- **Reset**: Restart the game from the beginning.
- **Save 3DN**: save the current 3d data to a file with the same name with rom file and 3dn as extension.
- **Quick Load**: Load the file **quick.sav** from the current directory.
- **Quick Save**: Save the current game state to the file **quick.sav**.
- **Exit**: quit 3DNes.

### 3. Setting Tab



- **Graphic:** the graphic quality options. There are three values: *High*, *Medium*, and *Low*.
- **VrMode:**
  - o *None* : monitor rendering
  - o *OpenVR*: Vive headset
  - o *Oculus*: Rift headset.
- **Vsync:** sync the graphic update operation with monitor refresh timing. Default is *true*.
- **Auto Layering:** enable/disable the process that automatically assigns layer for shapes. With a completed 3dn file, disable it to have better emulation performance.
- **Speed:** adjust the game speed.
- **Volume:** adjust sound volume.
- **Same Palette:** if *true*, shape will contain same palette tiles only
- **Sprite Tolerance:** by default sprite tiles in a shape will be strictly 8px aligned but we can specify some tolerance for this constraint

### 4. Render Tab



**Render Tab** contains generic parameters that decide how the 3D scene will be rendered.

- **Light Dir:** the direction of light scene – real values.

- **Light Intensity:** decide the scene brightness.
- **Def Layer:** decide at which layer the layer-undefined shape swill be rendered.
- **BG Color:**
  - o **True:** use the game background color to render background.
  - o **False:** manual select background color.
- **Shadow:** enable/disable shadow rendering.
- **Border ... masks:** adjust how many pixel will be clipped at four borders.
- **Layer ... depth ... alignment:** adjust the z position and the alignment of each layer.
- **Render:** select to render modes **Normal** and **Retina (Pro Feature)**.

## 5. Input Mapping Tab



- **Player:** select which player gamepad is going to be configured, the first or the second.
- **Type:** select input type *Keyboard* or *Controller*. For controller press any controller button first to bind the controller with the current gamepad.
- Select an action its text field will turn yellow, press the key/button you want to bind to this action then move on.
- Key/Button bindings will be automatically loaded/saved each time 3DNes is started/quitted.



## 6. Editor Tab

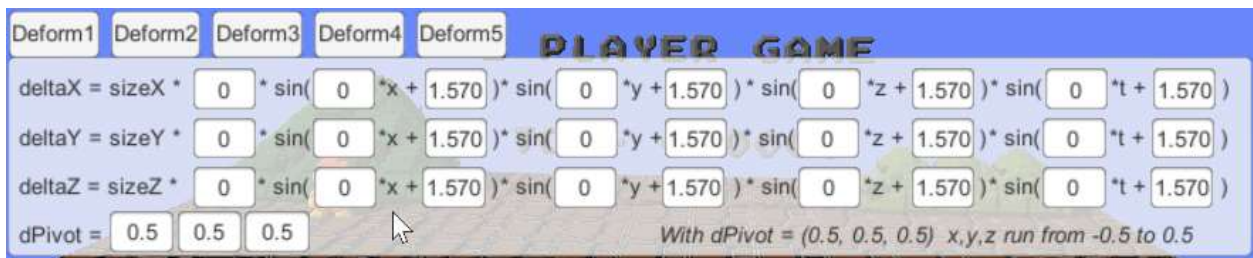


Whenever you want to change the 3D appearance of a shape just select it, the emulation process will be paused and the Editor tab will be activated. Every shape is linked with a pattern whose parameters are displayed and can be adjusted. We also have pattern action, frame advance, shape navigation buttons in this tab. Meaning of controls are described in detail as follows:

### 6.1 Pattern Parameter

- **SizeZ:** the shape depth in z axis – integer value. For Cylinder pattern, the depth that you enter is just the upper bound one, the actual one will be calculated by 3DNes
- **Layer:** To create the 3d depth, 3Dnes is using the layer concept. There are four normal layers *L1*, *L2*, *L3*, *L4* and a special layer *UI* which contain ui element of the game. There is also an initializing value called *Unknown*. Every pattern layer is *Unknown* by default. When you are playing, 3Dnes will try to guess layer value for some patterns (**Auto Layering** checkbox in **Render** tab) but you still have to manually assign value in most cases.
- **Geo:** is the geometric pattern to create the 3D appearance. There are six values: *Default*, *HCylinder*, *VCylinder*, *Cube*, *HalfHCylinder*, and *HalfVCylinder*. Depend on **Geo** value, there are several additional boolean parameters like *AS*, *Solid*, *Flip*.
- **Scale:** the shape scale vector – real values.
- **Offset:** the shape offset vector – real values.
- **Rot:** the shape rotation vector – real values.
- **Pivot:** the normalized pivot to rotate the shape. With value triple 0.5, 0.5, 0.5 shape will be rotate at its center.
- **Tags:** with of tags separated by comma. Tags will be used to linked shape with corresponding scripts.
- **FN – Force Normal:** shapes linked with this pattern will always be rendered in normal mode, whatever the value of **Render** is.

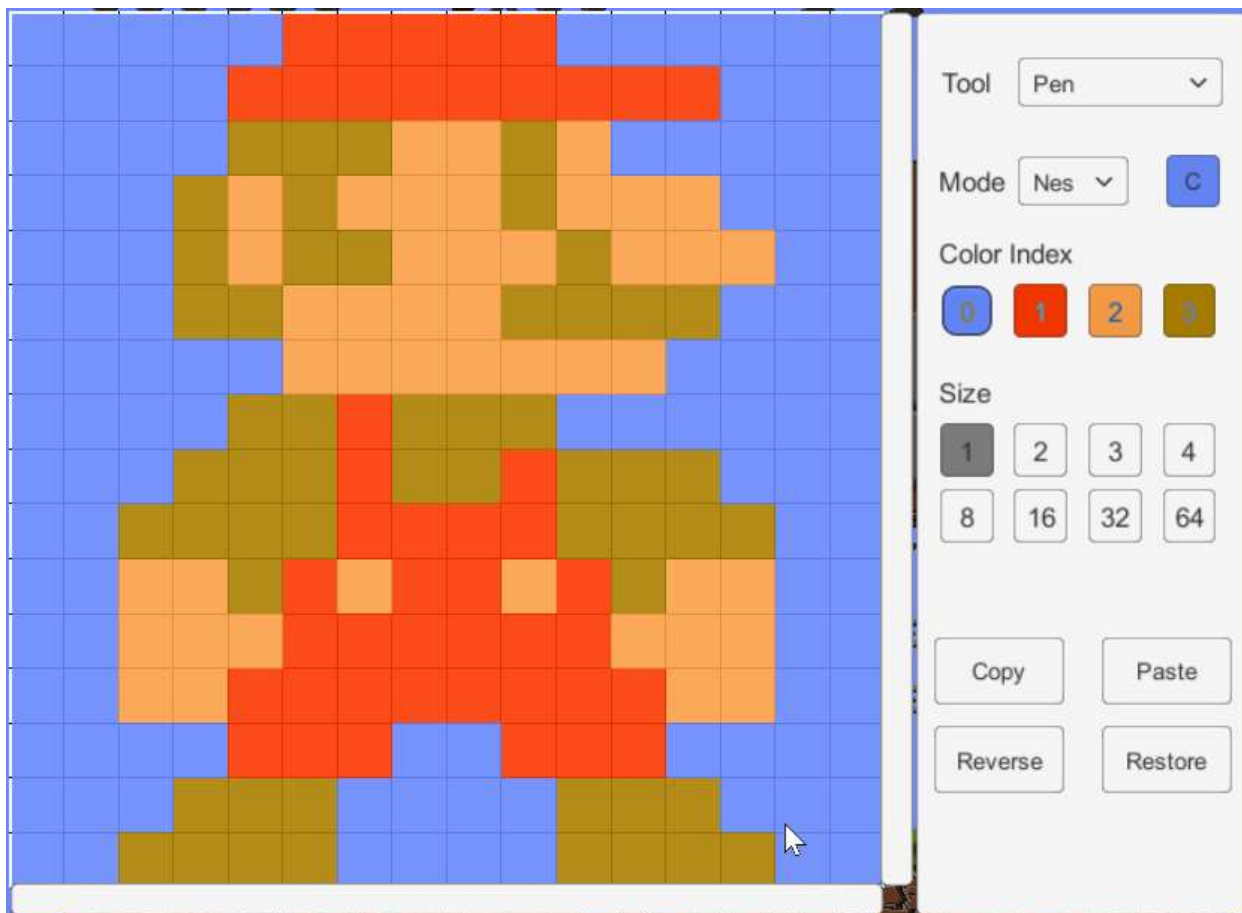
- **Alpha:** shape transparency, from 1: solid to 0: invisible – real value.
- **Restrict:** the maximum width and height (in tile unit) of shape linked with this pattern.
- **Enable:** shape linked with this pattern will be visible or not. Normally this parameter is always set to *true*. It could be set to *true/false* and toggled if some conditions are met via script.
- **Five Slots:** located at bottom left (Water, Rock, Config1 ...). They are popular sets of parameter that we use often in the game. Steps to use those **Slots**:
  - **Shift + Mouse Click:** a slot to rename it.
  - **Ctrl + Mouse Click:** a slot to save pattern parameters in **Editor Tab** and **Deform Tab** to it.
  - **Mouse Click:** a slot to load its store d parameters.
- **Deform:** show/ hide the sub-window **Deform**.
- **Tex:** show/hide the sub-window **Texture Editor**.
- **Deform window**



If we want to deform or animate a shape, we could define parameters of above formula. Note that in this formula, xyz are all normalized by dPivot and sizeX, sizeY, sizeZ. For example if dPivot is 0, 0, 0 then x,y,z will run from 0 to 1. If dPivot is 1, 1, 1 then x,y,z will run from -1, -1, -1 to 0, 0, 0.

The five slots at top left works similarly to the ones in editor tab but they only store parameters of the **Deform** window.

- **Texture Editor window**



Each pattern always has its own texture. It is initially created by tiles that construct the pattern. There are three tools that help customizing it.

- Pen tool : draw a square with selectable size and color at mouse position.
- Fill tool: fill a one color, solid region with a new color.
- Color picker tool: pick a color on the texture itself.

Note that texture can be drawn in two mode:

- Original index NES mode: three color indexes and a transparent one
- 24bits true color mode: in this mode the color(0,0,0) will be treated as transparent one.

We also have the capability to copy and paste the whole texture data from one to another, reverse the modification and restore the original data.

## 6.2 Action controls

- **Adjust:** apply new parameters we have just modified for the selected pattern. The shape appearance will be updated after button click.

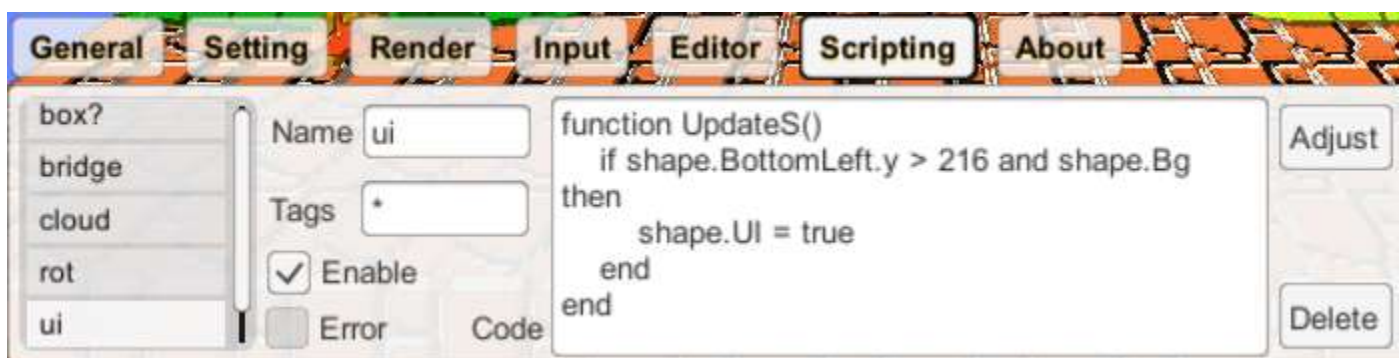


- **Delete:** delete the selected pattern.
- **Clone:** clone the selected 3D pattern to a new one. In that case one pattern will contain more than one 3D pattern.
- **Track:** tracked the selected shape in following frames and apply the same parameter set to its linked patterns.
- **Stop:** stop tracking process.
- **Mege:** **shift + mouse click/ shift + mouse drag** to select multi-shapes then merging them into a new pattern
- **Create:** **shift + mouse click/ shift + mouse drag** to select multi-tiles then create a new pattern from them.
- **Mode :**
  - o **Shape :** working with shapes
  - o **Tile :** working with tiles

## 6.3 Helper functions

- **Frame** **>, >>:** forward the emulation 1 and 3 frames.
- **Navigation** **<,>:** navigate the on screen shapes/tiles with a fixed order: from left to right, from top to bottom.

## 7. Scripting Tab



Scripts are pieces of Lua code you can add to adjust 3dnes settings and pattern parameters on the fly. A script has a unique name, a list of tag, **Enable** property and the Lua code itself. Script tag will be match with pattern tag to pair between script and 3D pattern. If a script is paired with a pattern then it is paired with every shape of that pattern too. Therefore in every frame those

shapes are objects to modify of this script code. \* is a special tag that matches all other tag. Just like pattern data scripts are also packed into 3dn file.

The script on the above image named **ui**, matches all patterns, **active** and its code can be interpreted as : **“In every frame any background shape has the bottom y coordinate higher than 216 should be put in the UI layer”**. In the game Super Mario Bros, it means that the top status bar should always be put in the UI layer.

The input field at bottom right allow you entering and executing Lua code directly any time you want.

For scripting detail please refer to the doc **ScriptingManual.pdf**.

## 8. Hot Keys

- Esc : Show/ hide main UI.
- F2: Switch camera mode: *fixed mode* and *tracking mode*
- F4: Reset camera (and headset in VR mode).
- F5 : Switch between fastest and normal game speed
- F6: Switch between slowest and normal game speed
- Ctrl + O : Open File Dialog
- Ctrl + D : Save 3dn file
- Ctrl + R : Reset the game
- Ctrl + S : Quick save game state to **quick.sav**
- Ctrl + Q: Quick load state from **quick.sav**
- Ctrl + F : Toggle Full screen Mode
- Ctrl + X : Exit 3DNes

If **Editor** tab is activated:

- Ctrl + A : trigger Adjust function
- Ctrl + E: trigger Delete function
- Ctrl + M: trigger Merge function
- Ctrl + C: trigger clone function
- Ctrl + Left/Right: shape navigation



- Ctrl +. : Forward one frames
- Ctrl + Shift + . : Forward some frames
- Ctrl + Mouse Click : switch camera from *fix mode* to *tracking mode* with the just selected by shape
- Shift + Mouse Click : multi shapes/tiles selection

## 9. Command line parameters

<b>--r=RomPath</b>	Start 3DNes and immediately load the rom located at <b>RomPath</b> . If the file path contains spaces then it should be put in double quotes.
<b>-adapter N (Windows only)</b>	Allows the game to run full-screen on another display. The N maps to a Direct3D display adaptor. In most cases there is a one-to-one relationship between adapters and video cards. On cards that support multi-head (they can drive multiple monitors from a single card) each “head” may be its own adapter.
<b>-force-d3d11 (Windows only)</b>	Make the game use Direct3D 11 for rendering.
<b>-force-glcore (Windows only)</b>	Make the editor use OpenGL core profile for rendering. The editor tries to use on the best OpenGL version available and all OpenGL extensions exposed by the OpenGL drivers. If the platform isn’t supported, Direct3D is used.
<b>-popupwindow</b>	The window will be created as a pop-up window (without a frame).
<b>-screen-fullscreen</b>	Overrides the default full screen state. This must be 0 or 1.
<b>-screen-height</b>	Overrides the default screen height. This must be an integer from a supported resolution.
<b>-screen-width</b>	Overrides the default screen width. This must be an integer from a supported resolution.



<b>-show-screen-selector</b>	Forces the screen selector dialog to be shown.
<b>-parentHWND &lt;HWND&gt; (Windows only)</b>	Embeds Windows Standalone application into another application, you have to pass parent application's window handle to Windows Standalone application. See this example <a href="#">EmbeddedWindow.zip</a> for more information.