

---

# **Laigter**

***Release 0.0.1***

**Sep 11, 2020**



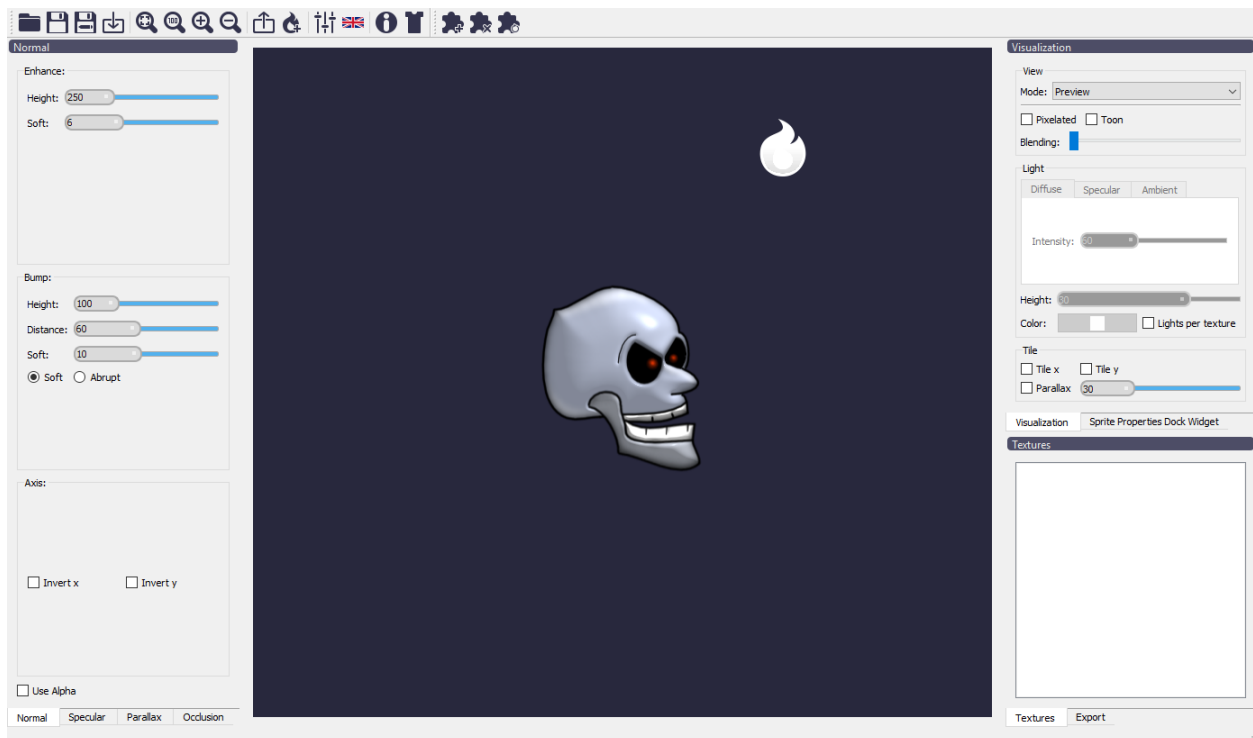
<b>1</b>	<b>General Concepts</b>	<b>1</b>
1.1	What is Laigter for? . . . . .	1
1.2	What are normal maps? . . . . .	2
1.3	Why choosing Laigter? . . . . .	3
1.4	Game Example . . . . .	3
1.5	Support Laigter! . . . . .	4
<b>2</b>	<b>Installing Laigter</b>	<b>5</b>
2.1	Installing on Windows . . . . .	5
2.2	Installing on Linux . . . . .	5
2.3	Running on Mac . . . . .	6
<b>3</b>	<b>Laigter's Main Window</b>	<b>7</b>
3.1	Toolbar buttons . . . . .	8
3.2	Dock Widgets . . . . .	8
3.3	Sliders . . . . .	10
<b>4</b>	<b>Support Laigter's development!</b>	<b>11</b>
4.1	Economic Support . . . . .	11
4.2	Non economic Support . . . . .	11
<b>5</b>	<b>Contact, Community and Development news!</b>	<b>13</b>
<b>6</b>	<b>First steps with Laigter</b>	<b>15</b>
6.1	Importing Sprites . . . . .	15
6.2	Selecting Sprites . . . . .	16
6.3	Moving and Rotating the canvas . . . . .	17
6.4	Sprite Properties Dock Widget . . . . .	17
6.5	Visualization Mode . . . . .	19
<b>7</b>	<b>Normal Map Generation</b>	<b>21</b>
7.1	Enhance controls . . . . .	21
7.2	Bump controls . . . . .	22
7.3	Axis Controls . . . . .	23
<b>8</b>	<b>Specular Map Generation</b>	<b>25</b>
8.1	Introduction . . . . .	25

8.2	Bright Control . . . . .	25
8.3	Contrast Control . . . . .	26
8.4	Threshold Control . . . . .	26
8.5	Soft Control . . . . .	26
8.6	Invert Control . . . . .	26
<b>9</b>	<b>Parallax Map Generation</b>	<b>29</b>
9.1	Binary Mode . . . . .	29
9.2	Height Map Mode . . . . .	30
<b>10</b>	<b>Occlusion Map Generation</b>	<b>33</b>
10.1	Controls . . . . .	33
<b>11</b>	<b>Real-time lighting preview</b>	<b>35</b>
11.1	Preview Settings . . . . .	35
11.2	Adding and removing lights . . . . .	38

# CHAPTER 1

## General Concepts

### 1.1 What is Laigter for?



Laigter is mainly an automatic normal map generator focused on 2D Sprites. You just need to drag and drop your images into Laigter, and a normal map will be generated for you. Then you can play around with controls to adjust the map to get better results for your use case. You can also preview dynamic lighting in real-time, to check how the generated maps will affect lighting in-game.

Laigter also lets you generate and edit other maps, like specular, parallax, and occlusion, although they are a bit less used in 2D games compared to normal maps.

Once you are convinced by the results, you can export each map, and use them in your game to create cool dynamic lighting effects with minimal effort!

## 1.2 What are normal maps?

Normal maps are 2D textures which stores the direction of the normal vector to a surface on its RGB channels. This way, they can be used in a shader to calculate how light should interact.

Normal maps are more often used in 3D games to achieve high-quality lighting at a low cost. A 3D model is composed of tiny triangles; each of one has a normal vector (unit vector perpendicular to the triangle's surface). So light can bounce and reflect according to that normal. However, to achieve high-quality results, the model should have a lot of triangles, which increases the computational cost. With normal maps, you can get a per-fragment normal, only needing to pass a texture with the information of the normal vector on its RGB components. Then, this can be used to achieve a more detailed light interaction with a very cheap GPU cost.

In 2D games, this is more noticeable, as we don't even have surfaces with normal vectors to calculate lighting. We only have a flat Sprite, so all lighting looks flat. Using normal maps, we can give the illusion that the Sprite has some kind of volume when it is affected by light.

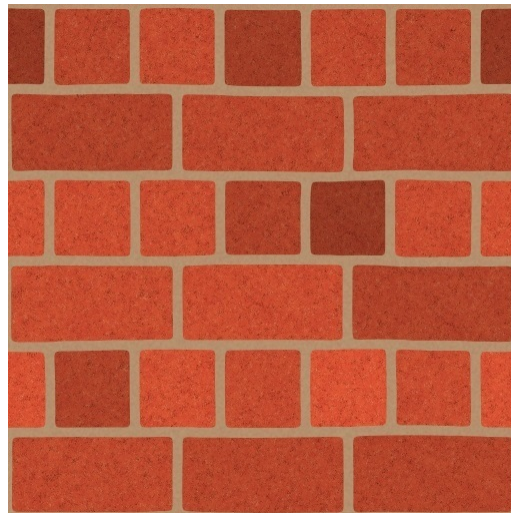


Fig. 1: Bricks Texture Without Normal Map

### 1.2.1 Normal Map Example

The following image shows a normal map example.

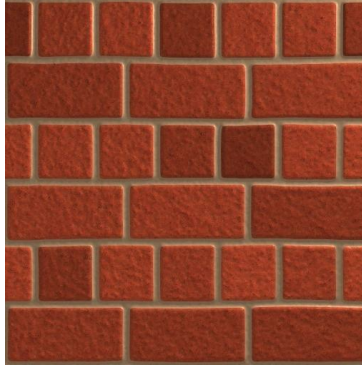
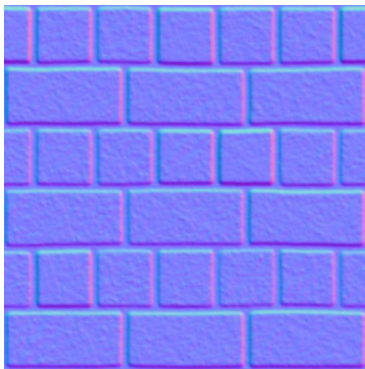


Fig. 2: Bricks Texture With Normal Map



But, how does a normal map store the normal vector on an RGB image? The concept is quite simple. It stores the  $x$ ,  $y$ , and  $z$  components of the vector on the  $R$ ,  $G$ , and  $B$  channels correspondingly, but with a small change.

On the one hand, a normal vector is a unitary vector (its length is equal to 1), and you only need values from -1 to 1 to represent each axis. On the other hand, RGB can store values from 0 to 1, so the normal map has the components of the vector scaled by 0.5, and with a 0.5 offset.

Following this technique, the vector  $(0.0, 0.0, 1.0)$  (a normal vector pointing directly to screen) would transform into  $(0.5, 0.5, 1.0)$ . In a nutshell, this is the reason why normal maps have that blueish color.

## 1.3 Why choosing Laigter?

There are other cool normal maps generators out there, but Laigter is free and open source. This means you can grab the code yourself and implement missing features if you need so, modify it, distribute your modified version to the public, and many other freedoms open source gives.

Also, Laigter is extensible through plugins, so you can make your own extensions and distribute/sell them freely.

Laigter is simple to use, you just need to open your images and tweak some controls. The real-time preview lets you check the result immediately. Laigter's rendering is done in OpenGL 2.x standard, so it can be run on old machines.

## 1.4 Game Example

Here is how my current game looks like, using Laigter to create normal maps:

## 1.5 Support Laigter!

As Laigter is a free and open-source project, it depends on the community to continue improving. So, if you like it, consider contributing to the project in some of these ways:

Become my [Patreon](#)!

Support me on [Kofi](#)!



---

### Installing Laigter

---

Laigter is available for Linux, Windows, and MacOS. Mac version is experimental though, so expect some bugs!

#### 2.1 Installing on Windows

The only way currently to get an installer for Windows, is through [itch.io](https://itch.io).

You can get the binaries directly from the [project page](#). There you can choose to download an installer, or a portable version if you prefer so. The installation process is simple: just run the installer and follow the steps as with most software out there. With the portable version, you just need to extract the zip to a folder of your preference, and run the binary from there.

You can also install Laigter using the [itch.io desktop App](#). Just open the app, and use the search bar to find Laigter, then hit install on the lower right corner.

Note: due to Laigter being a free and open-source tool with zero budget, it is not registered on Windows, so it may raise some alerts on the system. Please trust me, it is totally safe to run Laigter.

#### 2.2 Installing on Linux

For Linux you have two options: *AppImage* or *Flatpak*. The appimage is a self contained executable, with all the needed libraries to run on most linux systems. You can get it from the [project page](#), or from [Github releases page](#). In that last page, you can even get the continuous build to test the latest changes on the development branch!

For running it, you just need to give corresponding permissions and double click on it. If you install [appimaged](#) it can be even integrated with your desktop environment!

If you prefer to use *flatpak* instead, you can open your command line, and run:

```
flatpak install flathub io.itch.azagaya.Laigter
```

You can get more information about this way to install on [Flathub](#). After the install, the app should be integrated into your desktop environment.

Thanks a lot to [Calinou](#) for this!

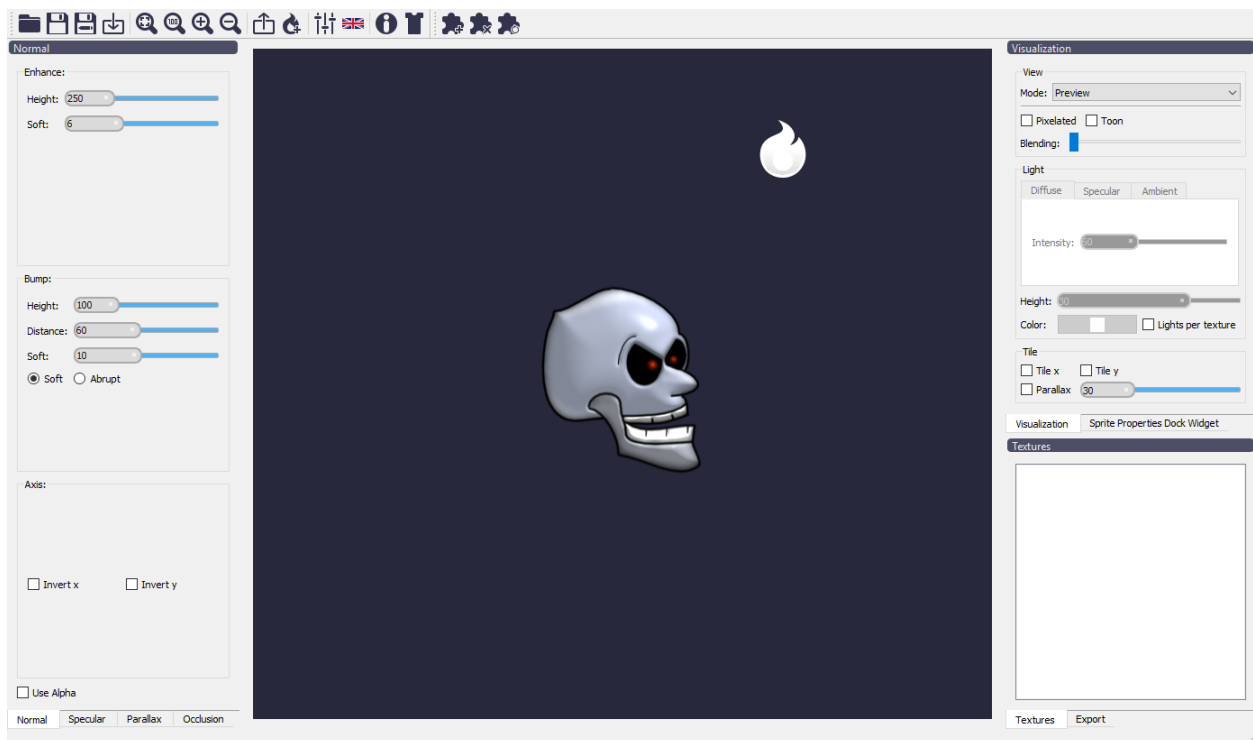
## 2.3 Running on Mac

Laigter is provided as a zipped app in the [project page](#). Just download it and run it as your other apps. If you're not allowed to double click the app to run it, remember to right click on it and click "open". This only needs to be done once, then osx will assume it as a safe application and let you run it by double clicking.

## CHAPTER 3

















### Laigter's Main Window

I try to keep Laigter's GUI as simple as possible, with all needed controls at reach. For this, the main window is made with a central widget, where all the rendering happens, and movable dock widgets, with the controls needed to tweak the maps. Also, there is a toolbar with buttons for the most used features of the tool. When opening Laigter, you should see this window:



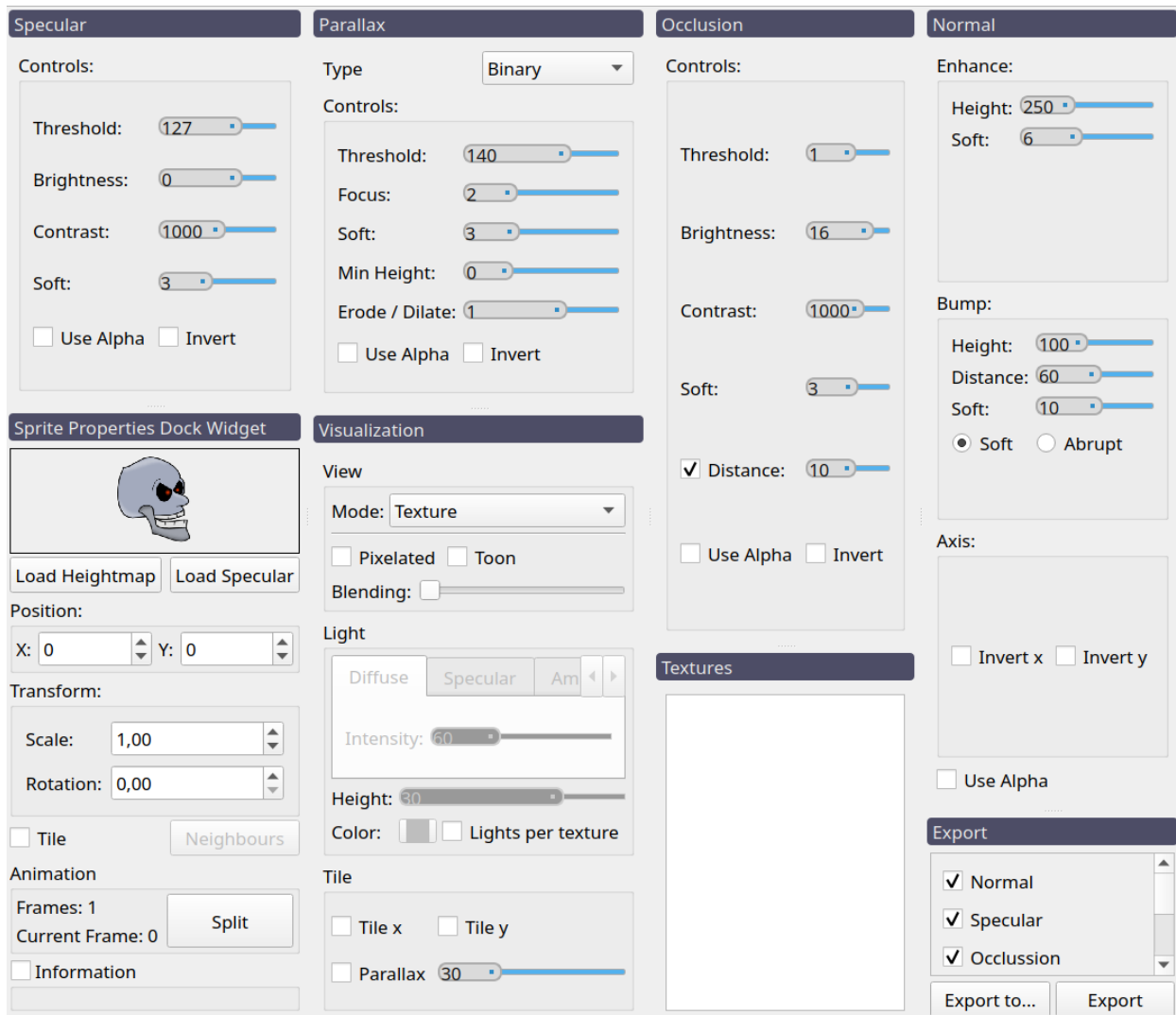
## 3.1 Toolbar buttons

In the toolbar, you will notice a few buttons. This is a brief explanation of each one, in order of appearance in the GUI:

-  Open Project: Open a previously saved Laigter project file.
-  Save Project: Save current work as a Laigter project file. If the project has not been saved previously, it will trigger *Save As* instead.
-  Save As: Saves current work as a new Laigter project. A dialog will pop up and let you choose the location and name of the file.
-  Import: Opens a dialog to choose images to import in Laigter's current project.
-  Fit Zoom: Applies the right amount of zoom and translation of the canvas, in order to fit all textures in the current view.
-  Zoom 100%: Restores the default zoom (1:1 scale of the textures).
-  Zoom +: Zooms in the canvas.
-  Zoom -: Zooms out the canvas.
-  Export: Exports the maps of the currently selected texture. A dialog will let you choose the location and base name for the maps. Selecting which maps to export and batch exporting will be expanded on other sections.
-  Add Light: While on preview mode, it lets you add or remove light sources. Will be covered in other sections.
-  Presets: Lets you save/load/apply presets to the selected textures. Will be covered in other sections.
- Language: Lets you select the language of your preference for the GUI.
-  About: Opens a window with information about Laigter version, contributors, supporters, links to relevant sites, etc.
-  Theme: Lets you select the theme for the GUI.
-  Install Plugin: Opens a dialog to select a plugin from your file system and install it to Laigter.
-  Delete Plugin: Opens a dialog to select and delete a specific plugin.
-  Reload Plugin: Reload all installed plugins.

## 3.2 Dock Widgets

Most of the controls of Laigter are placed in dock widgets that you can move around the GUI. The following image shows all of the dock widgets together:



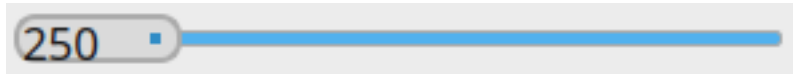
The controls of each individual dock widget will be explained in a section of their own, but here is a brief explanation of the docks.

- **Normal/Specular/Parallax/Occlusion:** this dock contains the controls for tweaking the generation of each map.
- **Sprite Properties Dock Widget:** In this dock, you have some controls and information of the currently selected sprite. You can change its position in the canvas, as well as its rotation and scale. You can select to generate maps in a *tiled* way, and choose how to extend it outside its bounds (select neighbors). Also, here you can make animations out of a sprite sheet, splitting the texture in multiple frames.
- **Visualization:** This dock lets you choose what map you want to see, or select preview to see the result with dynamic lighting. Also, some other controls let you tweak the preview a bit.
- **Textures:** This dock just contains a list of the currently opened textures. You can select multiple of them to show them all at the same time in the central widget.
- **Export:** In this dock, you can select which maps you want to export, and if you want to export directly in the location of the original sprite (Export button) or to a specified folder (Export to.. button). This makes a batch export, exporting maps of all textures.

## 3.3 Sliders

Almost all controls in Laigter's gui are standard, so no explanation is required to use them. However, the slider in Laigter is a custom widget, so a brief usage explanation may be useful.

A slider in Laigter looks like this:



The usage is simple: you can grab it from the right little square indicator, or you can simply write a number, and the slider value will be updated. And that's all!

---

Support Laigter's development!

---

### 4.1 Economic Support

As a free and open-source project, there are no incomes from the tool itself, besides the occasional optional payments from itch.io. That's why, if you like Laigter, I ask you to consider contributing to the development in one of these ways!

- Chose to pay something when downloading the tool from the [project page](#),
- Become a [Patreon!](#).
- Support me on [Kofi!](#).
- If you want another way to support my work, please feel free to contact me!

### 4.2 Non economic Support

Other things you may do to show your support, and help Laigter become more visible to the public, are:

- Star Laigter on [github](#).
- Rate Laigter on [itch.io](#).
- Report Issues on [github](#).





## CHAPTER 5

---

### Contact, Community and Development news!

---

If you would like to suggest features, ask questions related to Laigter, show your current project, etc, you can contact me and other laigter's contributors/users on [Discord](#).

If you want to be aware of Laigter new releases and features, you should follow me on [itch.io](#).

I also post most frequent updates, features and development news on [twitter](#), so follow me if you are interested!



---

## First steps with Laigter

---

When opening Laigter, you will see a default image, with default settings. All maps were generated for this sprite and you can see each of them and the result with lighting changing the preview mode in the view dock widget.

### 6.1 Importing Sprites


But probably, the first thing you want to do with Laigter, is test with your own sprites, so let's see how to do it.

**Warning:** Since switching processing library image for achieving simpler and more maintainable code, map generation became a bit slower. So you may need to wait a bit after importing a texture, as all maps will be generated at that moment.

If you don't have any sprite to test at the moment, you can use try with this two:





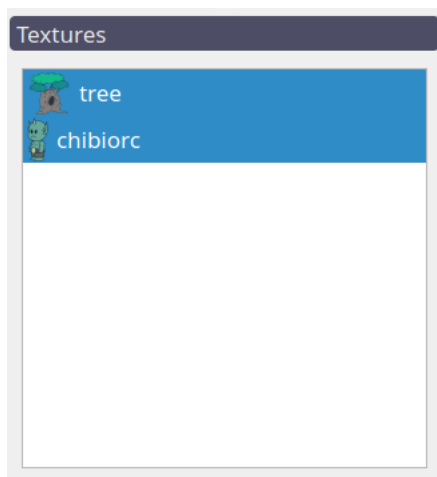
You can import those images into Laigter by simply dragging them to the main window, or using the  (import) button in the toolbar.

## 6.2 Selecting Sprites

If you drag both images at the same time into Laigter, you should see both of them rendered on the preview central widget. Both of them will be selected and positioned in the “World’s” origin (this is, the center of the preview widget when Laigter is opened). Clicking on one of them will only select that sprite. You then can freely move that sprite around. Holding Ctrl while clicking multiple sprites will allow you to select multiple sprites and moving them all at the same time.

### 6.2.1 Textures Dock Widget

Sometimes you don’t want to render all opened sprites, or you want to change the order in which they are rendered, to put a Sprite “in front of” another one. For this purpose, you can use the “Textures” dock widget.



In this widget, you can see a list of the opened images. Selecting one of them will only make that image visible in the canvas. You can select multiple images from the list holding Ctrl and clicking on them, and all selected images will

become visible. You can drag items on that list to change the order of rendering. The images on the top of the list will be rendered first, and thus, appear behind the images on a lower position.

You can right-click on an item of the list to open a context menu with different actions. You can remove an image selecting the “Remove” action from the context menu. The rest of the actions will be covered in other sections.

## **6.3 Moving and Rotating the canvas**

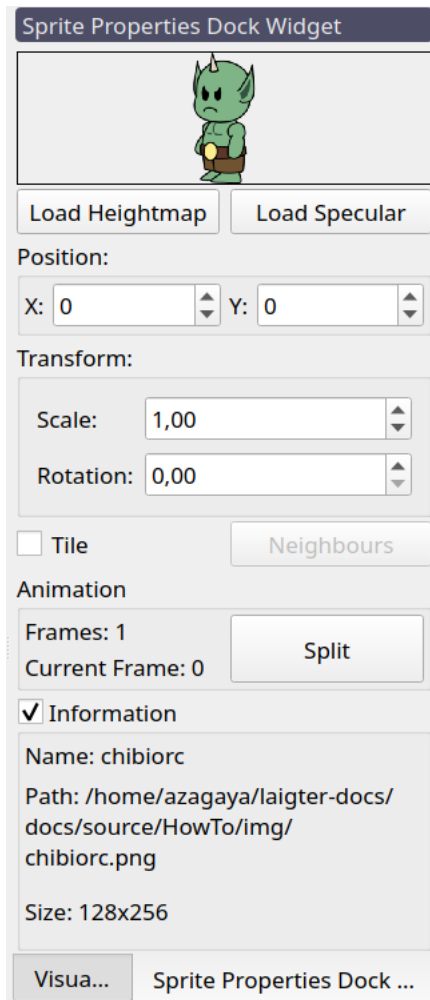
Sometimes it’s useful to move the canvas around, for example, to get a better view of a texture region that is very zoomed in. Sometimes you need to rotate the canvas to make it easier to use brushes.

In Laigter you can move the canvas, which adds an offset to the world’s origin, pressing and dragging right-click on a free space (i.e, without selecting any texture).

You can rotate the full canvas pressing and dragging left-click while holding the Shift key.

## **6.4 Sprite Properties Dock Widget**

This dock widget has some controls to modify the appearance of the currently selected sprite.



### 6.4.1 Transformations

You can apply some simple transformations to the Sprite, to make it easier for you to work on them. This transformations don't affect the map generation, but only change position, rotation, or scale of the sprite.

#### Sprite position

You can move any sprite by simply clicking on it and dragging it around the central canvas.

If you want to set the position in a more precise way, you can also head to the "Sprite properties Dock Widget" and set the desired position in the spin boxes. Have in mind, these positions are relative to the "World", with origin in the center of the widget when you open Laigter.

---

**Note:** The sprite should be selected for the transformation to affect. In the Sprite Properties Dock Widget you can see the currently selected sprite.

---

## Sprite rotation

You can also rotate the sprite. This is useful for hand-painting when using brush plugins (which will be covered on their own sections), or just for previsualization purposes. The only way for rotating a single sprite right now is to set the desired angle in the corresponding spinbox on the “Sprite Properties Dock Widget”.

## Sprite scale

The “Sprite Properties Dock Widget” also includes a spinbox to modify the scale of the sprite. This feature was added for the sole reason of visualization, as some times you may want to make a texture bigger than others for mockup purposes.

For scaling a sprite, you just need to select it and modify the value at the corresponding spinbox.

**Warning:** Some brush operations may work unpredictably with scaled sprites. This feature is currently only for visualization purposes. If you need to use a brush on a closer look at the sprite, use zoom instead.

## 6.4.2 Properties that affect the map generation

Some properties affect the map generation and will be covered in their corresponding sections for a better understanding of them. Those are:

- Animation controls (Split button)
- Tile options (Tile checkbox and Neighbours button)

You can see those controls on the “Sprite Properties Dock Widget” image at the beginning of this section.

## 6.4.3 Image information

Information is hidden by default, so it doesn’t waste space on low-resolution screens. Checking the “Information” checkbox, a list of data will be shown. This includes:

- The image name.
- The image path on your file system.
- The image size in pixels.

You can see this information on the “Sprite Properties Dock Widget” image at the beginning of this section.

If you would like to have more information there, please request it!

## 6.5 Visualization Mode

You can change the visualization mode in the “Visualization” dock widget, changing the selection in the “Mode” combo box. This will let you see the corresponding generated map for the visible sprites, as well as a real-time lighting preview.

Most of the features included in this widget will be covered in the corresponding section, but it was necessary to show the visualization mode here so you can go ahead and generate your firsts maps with Laigter!



---

## Normal Map Generation

---

Laigter aims to be a simple normal map generator. To tweak the normal map you only to play around with the sliders until you get the desired result. In this section what each of those sliders do will be explained.

All the controls that affect the normal map generation are grouped under the **Normal** dock widget.

### 7.1 Enhance controls

The controls under this group affect the detail, or *inner regions* of the sprite. For a better understanding of what they do, let's put the controls under *Bump* group to zero.

There are just two controls under the *Enhance* group, *Height* and *Soft*.

#### 7.1.1 Height Control

A normal vector can be thought of as the derivative of the height on a surface. If that height almost doesn't change (like on a flat surface) the normal will point upwards (or to the screen, in our case of 2D textures). This will lead to the blueish color we talked about in the [normal map example](#) of the introduction section. So this control lets you choose how high is the difference in height between consecutive pixels. A lower value then, will result in a more flat normal map (more blueish). A higher value would mean normal vector directions point to other directions, and so, more *colorful* normal map (as colors represent components of the vector). The effect of the light will be more noticeable in this last case.

Fig. 1: Effect of *Enhance* height control in the normal map.

Fig. 2: Effect of the *Enhance* height control in the preview.

### 7.1.2 Soft Control

After the normal map due to the “Enhance controls” was generated, a blur is applied to get a better-looking result. The amount of blur applied is controlled with this slider.

Fig. 3: Effect of *Enhance* soft control in the normal map.

Fig. 4: Effect of the *Enhance* soft control in the preview.

## 7.2 Bump controls

Enhance controls are meant for generating normals of inner details of the texture. But for 2D sprite, is useful to give some bump effect from its edges. This means, make the sprite look like it has volume. This is made by making a distance transform with the alpha mask of the texture, and using it as a heightmap for calculating the normal vectors.

For this, we use the controls under the *Bump* group, which has four different controls. for a better understanding of this controls, lets put the *Enhance* controls in zero.

### 7.2.1 Height Control

This control makes the same effect as the one with the same name described in `enhance_controls`, but this time to the normals generated from the edge of the texture.

Fig. 5: Effect of *Bump* height control in the normal map.

### 7.2.2 Distance Control

The previous control helped us to tweak the normal vector direction from the edges of the texture. This control specifies the distance from the edge of the texture to the center of it before the *Bump* normal map flattens.

### 7.2.3 Soft Control

This control also makes the same effect that it's homonymous in the *Enhance* group.

### 7.2.4 Soft/Abrupt radio buttons

These radio buttons are used to specify how the distance should be calculated from the edges of the sprite to the center of it. Selecting *Soft* will make the height generated by *Bump* controls increase in a way that tries to emulate a spheric bump. *Abrupt* option instead, will make the height increase linearly.

Fig. 6: Effect of the *Bump* height control in the preview.

Fig. 7: Effect of *Bump* distance control in the normal map.

## 7.3 Axis Controls

This group contains only two checkboxes, used to invert the  $x$  or  $y$  component of the normal vector respectively. This is used in case you want the final effect looks like a *bump* or a *sink*.

---

**Note:** Some engines or tools (like Godot) use down direction as positive  $y$ . For normal maps to work correctly on them, you need to check *Invert y* control. Right now is the only way to achieve it, although shortly there will be an on-export option for this.

---

Fig. 8: Effect of *Bump* soft control in the normal map.

Fig. 9: Effect of the *Bump* soft control in the preview.

Fig. 10: Effect of *Bump* soft/abrupt radio control in the normal map.

Fig. 11: Effect of the *Bump* soft/abrupt radio control in the preview.

Fig. 12: Effect of inverting axis in the normal map.

Fig. 13: Effect of inverting axis in the preview.

---

## Specular Map Generation

---

Although Laigter is focused on generating normal maps for 2D sprites, as those are the most used maps for dynamic lighting in 2D, with the time, other map generations where added. One of them is Specular Map Generation.

### 8.1 Introduction

Specular maps are a lot simpler to understand than normal maps. Specular lighting is based on reflective properties on material, and the angle of observation. When light hits a surface, if it's reflective, the light will bounce in the opposite angle from the normal vector. Some amount of this reflected light will reach the observer's eye. But for specular lighting to work in a shader or engine, we need to tell it how much reflective a surface is.

So, similarly than with normal maps, we store the reflectiveness in a texture. In this case is easier, as we only need to store one parameter. So we can make a grayscale image, where lighter colors mean more reflective, and darker colors mean less reflective.

Laigter automatically generates a normal map based on the sprite, and lets you tweak it with some simple controls. Most of them are the usual effects of image processing software.

---

**Note:** For testing the following explanations, please change the view mode to *Specular Map*. Also note, you have to change to the *Specular* dock widget to access the controls being explained in this document.

---

### 8.2 Bright Control

This slider just adds a constant amount to each pixel value. If the slider value is 0, the image will remain the same as the original. Is the value is positive, the image will become lighter, and if it is negative, it will become darker.

Fig. 1: Effect of *Bright* control on specular map.

---

**Note:** The effect of the specular lighting is that brighter spots you can see in the above figure, at the top right of the head. Specular lighting depends on view direction, which in 2D is usually fixed from the center of the screen.

---

## 8.3 Contrast Control

This slider multiplies the color of each pixel by a value, to increase or decrease the difference between light and dark colors in the image. This way, rising the value will make lighter pixels even lighter, and darker pixels even darker. Lowering it will make them more similar, until getting a full gray image at 0 contrast.

Fig. 2: Effect of *Contrast* control on specular map.

## 8.4 Threshold Control

*Contrast* control reduce or increase the difference between light and dark pixels. But how do we decide which pixels are considered dark and which considered light, depends on the *Threshold* value. Usually, the threshold is 127 (the middle between 0 and 255 approximately). In Laigter you can change this value, to apply the formula from other reference values.

Let's assume in the previous example, you just want the teeth of the head to have specular bright. Modifying *Bright* control won't work, because it makes the whole texture reflectiveness change. Modifying *Contrast* alone won't work either, as teeth and head colors are too similar. You can change then the threshold until you see teeth remain in a lighter color, while the rest gets darker.

Fig. 3: Effect of *Threshold* control on specular map.

**Warning:** This technique is quite limited and will only give good results in some cases. If you need to customize the specular map, it's better to use a brush plugin, or loading a custom specular base from start (to be discussed later on this document).

## 8.5 Soft Control

As in *Normal Map*, this control just adds some blur to the image. The higher the value, the more blurry the specular map will become.

Fig. 4: Effect of *Soft* control on specular map.

## 8.6 Invert Control

Invert checkbox is self-explanatory. It just inverts the colors of the specular map.

Fig. 5: Effect of *Invert* control on specular map.





---

### Parallax Map Generation

---

This one is probably the less used map in 2D games, but you can generate and preview it with Laigter anyways! Parallax mapping is similar to displacement techniques, in the sense that both techniques use grayscale textures to give information about height/depth/displacement. In displacement techniques, however, we use that information to move the vertices of the 3D object to achieve better surface detail. This makes it necessary for the object to have many vertices to look nice. Parallax mapping, instead, makes the calculation on fragments, just picking the color the user would see if the object had that displacement. This way, we can just apply the displacement with as little as four vertices (one quad or two triangles).

This is an example of how a 2D texture looks with parallax mapping technique.

Fig. 1: Parallax effect example.

Parallax generation in Laigter has two modes, *Binary* and *Heightmap*. The first makes a black and white version of the current texture and lets you adjust some parameters to tweak the parallax map. The resulting parallax map is still a grayscale image, as the controls will affect the resulting color. The latter will take the generated heightmap for normal map calculation, and use it directly as a parallax map, again, allowing users to tweak it through controls.

---

**Note:** In Laigter, the parallax map is analog to a depth map. Lighter colors mean that fragment should be at a deeper z position, and darker colors mean that the fragment should remain closer to the original position. Of course, the z position is just an effect, as no 3D processing is done here.

---

### 9.1 Binary Mode

Binary mode is useful for simple patrons, like brick walls. It will binarize an image given a threshold, and generate the parallax map from that image.

---

**Note:** For those who doesn't know, binarizing an image means taking the value of a pixel, and making it white if it's

---

above a threshold, or black if it's below.

---

### 9.1.1 Threshold Control

This control allows the user to chose the threshold for the binarization.

Fig. 2: Effect of *Threshold* control on parallax map.

### 9.1.2 Focus Control

This control applies a blur previous to the binarization. That way, is the image is too noisy, you can get rid of small “islands” when making the binary image.

Fig. 3: Effect of *Focus* control on parallax map.

### 9.1.3 Soft Control

As in all maps, this control just applies a blur to the resulting map.

Fig. 4: Effect of *Soft* control on parallax map.

### 9.1.4 Min Height

This slider lets you adjust the minimum height of the map. As in this mode, the image is binarized, this means all pixels will turn black (0) or white (255) depending on the threshold, and after that, other effects will be applied. With this control, you can adjust the minimum to be higher than 0.

### 9.1.5 Erode/Dilate

The binarized image will generally result in black islands on a white background or the other way round. With this, you can make those islands smaller (erode) or bigger (dilate). Positive value will dilate the white regions, and negative will erode them.

### 9.1.6 Invert

This checkbox just lets you invert the parallax map.

## 9.2 Height Map Mode

Instead of binarizing the texture, this mode uses the generated or loaded heightmap used for normal map generation, as the parallax map. Of course, it also provides a set of controls to tweak this, without affecting the normal map.

Fig. 5: Effect of *Min Height* control on parallax map.

Fig. 6: Effect of *Erode/Dilate* control on parallax map.

### 9.2.1 Soft Control

Same as *soft* slider explained in binary mode. It just applies a blur to the result.

### 9.2.2 Brightness Control

With this control, you can add a constant value (positive or negative) to the map, to make it lighter or darker.

### 9.2.3 Contrast Control

This slider is used to augment or reduce the contrast of the resulting map.

### 9.2.4 Invert Control

Same effect as *invert* control explained in binary mode.

Fig. 7: Effect of *Invert* control on parallax map.

Fig. 8: Effect of *Brightness* control on parallax map.

Fig. 9: Effect of *Contrast* control on parallax map.

---

## Occlusion Map Generation

---

An occlusion map is a 2D grayscale texture. It just specifies how much amount of ambient light should affect each pixel. Darker pixels on the occlusion map means less light should affect the corresponding pixel in the original texture, and white pixels mean that the corresponding pixels in the original texture should be affected by the light normally.

Fig. 1: Example of the effect of an *occlusion map*.

### 10.1 Controls

Threshold, Brightness, Contrast, Soft and Invert controls have the exact same effects on the occlusion map as those explained in *Specular Map Generation* for a specular map, so we won't go in detail here. There is only one control that is worth explaining in this section.

#### 10.1.1 Distance Control

This control consists of a checkbox and a slider. If enabled, instead of a regular grayscale version of the original texture, you will end up having a distance map. It will be assumed that pixel values below the threshold value should be darker, and the distance control lets you adjust how much that darkness should spread to pixels nearby. If disabled, all the other controls work just like in *Specular Map Generation*.



---

## Real-time lighting preview

---

Besides generating the maps for dynamic illumination, Laigter also has a real-time preview mode, in which you can add light sources, configure them, and move them around, to see how the lighting effect would look in-game.

---

**Note:** All settings in the *Preview Dock* are only for preview purposes. They will not affect the generated maps, and if there is any effect you want to reproduce in your game, you will have to code the shader yourself if it is not supported out of the box in your game engine.

---

### 11.1 Preview Settings

Similar to the docks that contains the controls for the maps generation, the preview dock has settings to tweak what is rendered. This settings are divided in *View*, *Light*, and *Tile* groups.

**Warning:** Different versions of the tool may have some controls in a different group compared to what is written here.

#### 11.1.1 View

This group contains the basic preview settings.

##### Mode

This drop-down menu lets you choose the current view mode. There is an option for previewing each map, with its respective name, and an option for real-time lighting preview.

Fig. 1: Changing view mode in the *Preview Dock* settings.

## Pixelated Option

Checking this option will make all textures to be rendered without filter, so you can see the pixels of the original texture, and resulting maps. This option is useful when generating maps for pixel-art sprites.

Fig. 2: Testing *Pixelated* option in real-time preview.

## Toon Option

This options lets you set lighting in a simple toon mode. It uses the simplest toon shader, so dont expect anything fancy. When you are making a game with such effect, it is useful to be able to preview at least something similar to the final effect before exporting.

---

**Note:** This option only affects real-time preview, as it needs light to work.

---

Fig. 3: Testing *Toon* option.

## Blending Control

When using brushes, for painting a normal map for example, it is useful to be able to see the original texture for reference. This slider allows you to blend the original sprite with the currently selected map in the preview mode drop-down menu.

---

**Note:** This option only works while previewing a genearted map (normal, parallax, specular or occlusion), as blending the original texture in texture or preview mode won't have any effect.

---

Fig. 4: Blending original sprite with normal map.

## Parallax Controls

For previewing parallax effect, you need to check the check-button. The slider allows you to choose the depth of the parallax map. This is disabled by default, as it uses a considerable amount of resources.

### 11.1.2 Light

This group of controls are used to tweak properties of illumination.

---

**Note:** You have to select a light for this group of controls to be neabled.

---

This group is divided in three tabs: *Diffuse*, *Specular*, and *Ambient*. All light sources in Laigter have a diffuse and a specular component. Also, there is an ambient light present, independent of light sources. The settings of each type of light are under the corresponding tab in the *Light* group.



Fig. 5: Example of parallax effect preview.

Also, after this tabs, there are controls for setting height and color of the light source. These are not located in the tabs, because color and height affect specular and diffuse light components the same way.

Last option is to be able to have individual light sources per texture loaded.

### Diffuse Light Controls

The diffuse component of the light sources has only one setting, the *Intensity* of the light. The higher this value is set, the brighter the light source becomes.

Fig. 6: Effect of changing intensity of diffuse light.

### Specular Light Controls

The specular component of the light source has two controls. *Intensity* specifies, as in diffuse component, the energy of the specular component (the higher, the brighter), and *Scatter* makes the specular component be less focused on one point, and scatter in the surface.

Fig. 7: Effect of changing settings of specular component of the light source.

### Ambient Light Controls

In this group of settings, you can change the intensity and color of the ambient light, and the color of the background of the preview. Think of ambient light as a uniform light that is present in all the scene.

#### Light Color

This button just opens a color-picker, so you can choose a new color for the selected light source.

#### Light Height

These settings affect where in the z-axis should the light be placed. If we imagine the canvas as a plane where all the textures are rendered, the height of the light is how far is the light from this plane.

#### Lights per texture

Usually, having light sources that affect all sprites is enough. But sometimes, for mockup purposes, it is useful that each texture retains its own light sources and settings.

If you have this option checked, selecting a texture will also recover the light settings that were configured the last time that texture was selected. This allows you a flexible way to have different lights on each sprite.

Selecting multiple textures with this option enabled will make all of the light sources for each texture to appear in the screen, and affect all sprites.

Fig. 8: Effect of ambient light settings.

Fig. 9: Changing color of selected light source.

**Warning:** When unselecting *lights per texture* option, Laigter will use default lights. This lights are those created and modified when the option was unselected. So changes you make to lights with *lights per view* enabled will not be present in the default lights when disabling that option.

## 11.2 Adding and removing lights

Laigter also lets you add multiple lights to the same scene. This is done with the *Add Light* toolbar button.

The *Add Light* button is a toggle button. When pressed, you enter in *add-light* mode. Left-click on the canvas will add a light in the position of the click, and right-click on a light, will remove it from the scene. You can exit the *add-light* mode by unpressing the *Add Light* button, or by right-clicking in the canvas (i.e right-click anywhere in the canvas where there isn't a light source placed).

Every new light will use the same settings of the last light selected. You can change the settings afterwards by selecting the light you want to tweak, and modifying the controls explained above in this document.

---

**Note:** *Add Light* button will only be enabled if you are in preview mode, as you cannot see lights in other modes.

---

**Warning:** Adding a light with *lights per texture* will only add that light to the currently selected sprite, and thus, it will not be visible when selecting other sprite.

Fig. 10: Changing the height of a light source.

Fig. 11: Changing textures with *Lights per texture* option enabled.

Fig. 12: Selecting multiple sprites whit *Lights per texture* enabled.

Fig. 13: Adding light sources.