RubyReference
Complete listing of all official classes and methods in Ruby/Gosu.

Updated Feb 04, 2010 by julianraschke

Ruby API Reference

Please post feedback and additions as comments to this page and visit the boards for questions outside of the scope of a single wiki page. Thank you!
http://www.libgosu.org/cgi-bin/mwf/forum.pl

Also as rdoc

This page will continue to offer an at-a-glance overview over Gosu's Ruby interface. However, http://www.libgosu.org/rdoc/ offers a more detailed reference from now on.

Common arguments

- **Color shortcut:** Whenever a function expects a Gosu::Color value, a 0xAARRGGBB integer can be supplied instead.
- **Additive drawing:** Valid mode parameters for drawing functions are :default and :additive.
- **RMagick users:** Magick::Image instances can be passed instead of filenames to Image.new and Image.load_tiles!

class Gosu::Color

- `initialize(argb)`: argb is a 0xAARRGGBB integer.
- `initialize(a, r, g, b)`
- `alpha`
- `alpha=`
- `red`
- `red=`
- `green`
- `green=`
- `blue`
- `blue=`
- `hue`
- `hue=`
- `saturation`
- `saturation=`
- `value`
- `value=`
- `self.from_hsv(h, s, v)`: converts a HSV triple into a color. Same as from_ahsv with alpha set to 255.
- `self.from_ahsv(a, h, s, v)`: converts a HSV triple into a color, with a given alpha. Ranges: alpha from 0..255, h from 0..360, s from 0..1, v from 0..1.

class Gosu::Font

- `initialize(window, font_name, height)`: font_name can either be the name of a system font, or a filename (must contain '/', does not work on Linux yet). height is the height of the font, in pixels.
- `height`
- `text_width(text, factor_x=1)`: Returns the width in pixels the given text would span.

- `draw(text, x, y, z, factor_x=1, factor_y=1, color=0xffffffff, mode=:default)`
- `draw_rel(text, x, y, z, rel_x, rel_y, factor_x=1, factor_y=1, color=0xffffffff, mode=:default)`: If relX is 0.0, the text will be to the right of x, if it is 1.0, the text will be to the left of x, if it is 0.5, it will be centered on x. Of course, all real numbers are possible values. The same applies to relY.
- `draw_rot(text, x, y, z, angle, factor_x=1, factor_y=1, color=0xffffffff, mode=:default)`: Same as `draw` but rotated at the top left corner.

class Gosu::Image

- `initialize(window, filename_or_rmagick_image, tileable, [srcX, srcY, srcWidth, srcHeight])`
- `width`
- `height`
- `draw(x, y, z, factor_x=1, factor_y=1, color=0xffffffff, mode=:default)`
- `draw_rot(x, y, z, angle, center_x=0.5, center_y=0.5, factor_x=1, factor_y=1, color=0xffffffff, mode=:default)`: center_x Relative horizontal position of the rotation center on the image. 0 is the left border, 1 is the right border, 0.5 is the center (and default)—the same applies to center_y, respectively.
- `draw_as_quad(x1, y1, c1, x2, y2, c2, x3, y3, c3, x4, y4, c4, z, mode=:default)`: Like Window#draw_quad, but with this texture instead of a solid color. Can be used to implement advanced, non-rectangular drawing techniques.
- `self.from_text(window, text, font_name, font_height)`: Creates a line of text. font_name can either be the name of a system font, or a filename (must contain '/').
- `self.from_text(window, text, font_name, font_height, line_spacing, max_width, align)`: Creates a block of text of width max_width. Each line will take font_height + line_spacing pixels of vertical space.`: align must be one of :left, :right, :center oder :justify. : font_name can either be the name of a system font, or a filename (must contain '/', does not work on Linux yet).
- `self.load_tiles(window, filename_or_rmagick_image, tile_width, tile_height, tileable)`: tile_width can either be the width of one tile in pixels or the number of columns multiplied by -1. tile_height is its vertical equivalent
- `gl_tex_info`: See examples/OpenGLIntegration.rb.

class Gosu::Sample

- `initialize(window, filename)`
- `play(vol=1, speed=1, looping=false)`: See play_pan. Returns a `SampleInstance`.
- `play_pan(pan=0, vol=1, speed=1, looping=false)`: pan can be anything from -1.0 (left) to 1.0 (right). vol can be anything from 0.0 (silence) to 1.0 (full volume). Playback speed is only limited by FMOD's capatibilities and can accept very high or low values. Use 1.0 for normal playback speed. Returns a `SampleInstance`.

class Gosu::SampleInstance

- `stop`: Stops this instance of a sound being played. Calling this twice, or too late, does not do any harm. You can nil out the reference to the instance afterwards as it will be useless.
- `pause`: Pauses this instance to be resumed afterwards. It will still keep a channel filled while paused.
- `paused?`
- `resume`
- `playing?`
- `volume=(vol)`: See Sample#play_pan.
- `pan=(pan)`: See Sample#play_pan.
- `speed=(speed)`: See Sample#play_pan.

class Gosu::Song

- `initialize(window, filename)`
- `play(looping = false)`: Also used to resume paused songs.
- `pause`
- `paused?`
- `stop`
- `playing?`
- `volume`, from 0..1.
- `volume=(vol)`, with vol from 0..1.

class Gosu::TextInput

- `text`
- `text=(str)`
- `caret_pos`: Position of the caret as the index of the character that it's left to.
- `selection_start`: If there is a selection, `selection_start` yields its beginning, using the same indexing scheme as `caret_pos`; otherwise, equal to `caret_pos`.

class Gosu::Window

Note that you should really only use on instance of this class at the same time. This might change later.

- `initialize(width, height, fullscreen, update_interval = 16.666666)`: Note`: Having two or more windows and loading samples or songs on both of them will result in an exception. If you want to re-open your game window, make sure the old one is **really** dead.
- `caption`
- `caption=(str)`
- `show`: Enters a modal loop where the Window is visible on screen and receives calls to `draw`, `update` etc.
- `close`: Tells the window to end the current `show` loop as soon as possible.
- `update`
- `draw`
- `needs_redraw?`: By default, always returns true. Override this to keep the window from redrawing itself at certain times, e.g. to implement frameskip.
- `button_down(id)`: Called when the user presses the button with the given id.
- `button_up(id)`: Called when the user releases the button with the given id.
- `draw_line(x1, y1, c1, x2, y2, c2, z=0, mode=:default)`
- `draw_triangle(x1, y1, c1, x2, y2, c2, x3, y3, c3, z=0, mode=:default)`
- `draw_quad(x1, y1, c1, x2, y2, c2, x3, y3, c3, x4, y4, c4, z=0, mode=:default)`
- `mouse_x`
- `mouse_y`
- `mouse_x=(float)`
- `mouse_y=(float)`
- `set_mouse_position(x, y)`: To avoid the intermediate position of calling `mouse_x=` followed by `mouse_y=`.
- `button_down?(id)`
- `self.button_id_to_char(id)`: Returns the character a given id stands for, or nil.
- `self.char_to_button_id(char)`: Returns the id usually used for a character, or nil.
- `text_input`
- `text_input=`: Sets current text input object that builds an input string
- `width`
- `height`
- `update_interval`
- `gl do ... end`: See examples/OpenGLIntegration.rb.
- `clip_to(x, y, w, h) do ... end`: Limits the drawing area to a given rectangle while evaluating the code inside of the block.

Free functions (in module Gosu)

- `offset_x(angle, dist)`: Returns the horizontal distance between the origin and the point to which you would get if you moved radius pixels in the direction specified by angle.
- `offset_y(angle, dist)`: Returns the vertical distance between the origin and the point to which you would get if you moved radius pixels in the direction specified by angle.
- `angle(x1, y1, x2, y2)`: Returns the angle between two points in degrees, where 0.0 means upwards. Returns 0 if both points are equal.
- `angle_diff(angle1, angle2)`: Returns the smallest angle that can be added to `angle1` to get to `angle2` (can be negative if counter-clockwise movement is shorter).
- `distance(x1, y1, x2, y2)`: Returns the distance between two points.
- `milliseconds()`: Incrementing, possibly wrapping millisecond timer.

- `default_font_name()`: Returns a font name that will work on any system.
- `screen_width()`, `screen_height()`: Return the dimensions of the system's primary screen. Can be used to choose the size of your windowed resolution.

Button IDs used by the input system (in module Gosu)

For clarity: Kb = Keyboard, Ms = Mouse, Gp = Gamepad.

```
Kb0
Kb1
Kb2
Kb3
Kb4
Kb5
Kb6
Kb7
Kb8
Kb9
KbA
KbB
KbC
KbD
KbE
KbF
KbG
KbH
KbI
KbJ
KbK
KbL
KbM
KbN
KbO
KbP
KbQ
KbR
KbS
KbT
KbU
KbV
KbW
KbX
KbY
KbZ
KbBackspace
KbDelete
KbDown
KbEnd
KbEnter (Numpad)
KbEscape
KbF1
KbF10
KbF11
KbF12
KbF2
KbF3
KbF4
KbF5
KbF6
KbF7
KbF8
KbF9
KbHome
KbInsert
KbLeft
```

```
KbLeftAlt
KbLeftControl
KbLeftShift
KbNumpad0
KbNumpad1
KbNumpad2
KbNumpad3
KbNumpad4
KbNumpad5
KbNumpad6
KbNumpad7
KbNumpad8
KbNumpad9
KbNumpadAdd
KbNumpadDivide
KbNumpadMultiply
KbNumpadSubtract
KbPageDown
KbPageUp
KbPause
KbReturn (center of keyboard)
KbRight
KbRightAlt
KbRightControl
KbRightShift
KbSpace
KbTab
KbUp
MsLeft
MsMiddle
MsRight
MsWheelDown
MsWheelUp
GpButton0
GpButton1
GpButton10
GpButton11
GpButton12
GpButton13
GpButton14
GpButton15
GpButton2
GpButton3
GpButton4
GpButton5
GpButton6
GpButton7
GpButton8
GpButton9
GpDown
GpLeft
GpRight
GpUp
```

Extensions to class Numeric

- `Numeric#gosu_to_radians`: Converts from an Gosu angle as shown in BasicConcepts to an angle in radians.
- `Numeric#radians_to_gosu`: Converts from an angle given in radians to a Gosu angle as shown in BasicConcepts.

Comment by jrmair, Jun 26, 2008

how do i draw things like circles and lines?

Comment by project member julianraschke, Jun 26, 2008

jrmair, for lines there is the Window#draw_line method. You can either draw circles by calling draw_line in a loop and using offset_x/offset_y or sin/cos, or by using an image that stores a circle. Gosu only provides what is available on the hardware here.

Comment by jrmair, Jul 03, 2008

hey, sorry to bug you again. but i assume there's no way to get access to the image pixel buffer? i need that data to figure out per-pixel collisions for my game. Or if you could suggest another way to do it? (if that functionality is not in gosu). i dont really want to use RMagick, as what im doing is quite simple and i dont want all that bloat and extra functionality for such a simple function.... thx!!

Comment by project member julianraschke, Jul 04, 2008

Only way would be to use OpenGL calls, as Image has the tex_info stuff. That will only work for images that fit on a single OpenGL texture though (which is guaranteed for up to 500x500 pixels). Also you wlll have to figure out how to do that :) I will think about adding it in the next version though, since it's been requested a few times in the last weeks.

Comment by jrmair, Jul 07, 2008

hey in your ref you say "See examples/OpenGLIntegration.rb. " where can i find OpenGlIntegration?.rb ?

Comment by project member julianraschke, Jul 07, 2008

If you installed Gosu via Gem, then it's in <rubypath>/lib/ruby/gems/1.8/gems/gosu.../examples. Don't know how I can make that more accessible :(

Comment by jrmair, Jul 19, 2008

hey, im trying to read in the pixel buffer from an image im using in gosu. you suggested i use the opengl functions, so im doing the following:

```
@image=Gosu::Image.new(self,"test3.png");

info=@image.gl_tex_info;

glEnable(GL_TEXTURE_2D);

glBindTexture(GL_TEXTURE_2D, info.tex_name);

data=glGetTexImage(GL_TEXTURE_2D, 0, GL_RGB, GL_FLOAT);

texture=data.unpack("f");
```

... the image im using is only 4x4 pixels...and every pixel is coloured (none black). However when i display the contents of the texture array every element appears to be equal to 0.0, also the array seems to be much larger than necessary for a 4x4 image. This is clearly not right.

How do i get the correct image data back an array using opengl? how are you creating the textures using glTextImage2d() ? what parameters do i send to glGetTexImage() to get the image back? or what am i doing wrong?

thanks in advance...i really need to get this pixel-level stuff working for my gosu game..

john

Comment by jrmair, Jul 19, 2008

sorry the above texture=data.. should read:

```
texture=data.unpack("f*")
```

thanks

Comment by project member julianraschke, Jul 20, 2008

Gosu always creates large quadratic textures and allocates the image data for as many Images as possible on those. Even if it wouldn't do this, it might have to round up the image size to the next power of two (on older chipsets). This is why the GLInfo type has the left, top, right and bottom fields. These are values in the range of 0 to 1 that specify where the relevant image data is located. You can calculate these back into pixel values by assuming that the texture is always quadratic.

Comment by jrmair, Jul 20, 2008

so you're saying that glGetTexImage returns the data for the large 'quadratic'(whatever that means) texture, and not just the texture associated with my @image? so...somewhere in all that data is the data specifically for the the texture i want?

sorry to bug you, but any more hints how i'd get just the data for my specific texture out from all that returned by glGetTexImage?

Comment by project member julianraschke, Jul 20, 2008

Knowing it's quadratic, you can calculate the width and height of the texture by taking the square root of the number of pixels. Then just multiply the left/top/width/height fields of the GLInfo type with the texture size and then you know where, in pixels, the relevant image data is stored. Then you have to use some array operations to select those.

Comment by project member julianraschke, Jul 23, 2008

Same as Window#draw_quad, you supply the corners in "reading order", that is top left, top right, bottom left, bottom right. If you don't see anything, check that the colors you hand in don't have an alpha channel of zero.

Comment by mechacrash@hotmail.com, Jul 23, 2008

got it XD

also is there a way to select the transparency of an image on draw... as far as i remember, alpha is just from total black (0) to full colour (255) right?

Comment by joseignacio.fernandez, Jul 28, 2008

Hi! Congrats for this great API :) I'm using it for a game in which the full screen requires drawing on it. So, as many people here, I need to access texture data. I've managed to do it using gl_tex_info, glGetTexImage and all that. But Gosu limits texture size to 512x512.

If using, e.g., a screen res of 1024x768, I have to manually create & manage a 1024x1024 texture because Gosu, instead, uses many smaller textures to build up a 1024x768 image and doesn't allow me to use gl_tex_info.

Why this 512-size limitation? Thx!!

Comment by project member julianraschke, Jul 28, 2008

When the code was first written, drivers reported all sorts of things, but nothing close to the truth, so in fact it had been locked to 256x256 until recently. I'll put it on my To Do to bump it up to 1024 for the next version. I have a tendency to be careful about Gosu working with old hardware/drivers, though, because the requirement of a 3D accelerator alone has caused deployment problems even last year :(

Comment by rremedio, Aug 07, 2008

Hi there!

Thank you for your great library. I'm not even a programmer and within one hour I could make a working test "game" for what I wanted!

I have a question, I think it's a simple one. Can you give an example or any clue on how to create a textbox that can be edited by the user and have it's content stored in a variable? I'd like to have it in both Ruby and C++, but I guess I can translate it to C++ if you show it in Ruby.

Regards!

Comment by project member julianraschke, Aug 10, 2008

rremedio, thanks! Just take what's inside the file `examples/TextInput.rb` and copy TextInput?#text to your variable after pushing enter, or in a timer, or so... If you want a text field with clipping, that can easily be done in the next version of Gosu (which I've been to get around to release for a while now). The example will be updated for that too.

Comment by jrmair, Aug 25, 2008

if anyone is interested in trying out or bugtesting TexPlay? (a c extension for gosu for manipulating images) go here: http://banisterfiend.wordpress.com/

it currently supports get_pixel and the ability to manipulate gosu images at runtime (the drawing of circles, lines boxes etc)

Comment by jrmair, Aug 26, 2008

it shouldn't be too difficult to compile on windows, should just be a matter of taking out the c99 specific stuff. ill look up the 'blob' function and check how difficult it would be to implement

Comment by patrickli.2001, Aug 27, 2008

Hi julian, Awesome work on this library. It's so easy to use. I have one question: Is there any way to do a "global" translation or rotation? It'd be nice to be able to be able to pan and rotate the entire screen for a camera system. May I ask how you're doing it?

Comment by project member julianraschke, Aug 28, 2008

patrick, I came up with an approach that a friend used for a similar project (among others). It works by monkey-patching the Gosu library: http://www.raschke.de/julian/temp/gosu-rot.rb

It's not complete, and you can only extend it to cover the Font class with the latest SVN codebase, which features Font#draw_rot. If you need a complete solution, you can file a feature request, it might coincide with current proposals to change the drawing syntax ;)

Comment by patrickli.2001, Aug 28, 2008

Thanks julian for the help. That approach should work just fine.

Java's 2D graphics package, offers a translate(x,y,z) and a rotate(angle) function that let's you change the coordinate system.

Those are the only two basic functions that seem to differ between Gosu and other libraries.

Other than that, I really like the drawing syntax. You can anything you want, and it's simple to read.

Comment by himatako, Sep 17, 2008

@mecha

> Oh that's ok I'm not in a hurry or anything. Take your time and thank you for doing this. :D

By the way, I've tried drawing the text in Thai(my native language), but it doesn't work even though I've changed to font to Thai compatible font. Do you have any suggestion on how should this be fixed? Or should I just use Bitmap font?

P.S. julian, where's the donation jar? This great library could use some support :)

Comment by himatako, Sep 18, 2008

I hope there's a way to fix the font thingy :(I really want to make my game to be multilanguages.

Comment by project member julianraschke, Sep 19, 2008

himatako: Regarding the support for Thai, it works for me—to a point. If you want to use any special characters in Ruby/Gosu apps you will have to save the file in UTF-8 format, but without a leading byte order mark. Unfortunately, Notepad has no option for that, I'm not sure about SciTE. I created this file on my Mac and also tested it successfully on Windows: http://www.raschke.de/julian/temp/ThaiTest.rb & ….png The gotcha right now is that surrogates that are put over/under the previous character have a negative width in theory … which causes `Gosu::Font` to crash. So you can also use the "real" characters in Font, but use full Thai test in `Gosu::Image.from_text`. Hope that clears things up :) And no, I don't have a donation box yet. Might think about that when I get to restructure all this, most notably set up the forum finally. Thanks for your feedback!

Comment by bestguigui, Mar 07, 2009

Hey Julian ! I really want to THANK YOU for your excellent work here ! I'm writing a huge french tutorial about Ruby Gosu (http://www.relite.org/forum/ruby-gosu-cours-complet-t4879.html) and I keep telling people how amazing your lib is on Youtube.

But one thing is still bothering me : it's your opengl texture coordinates system. Is there a way to specify a value different from 1.0 for texture coordinates ? I need this to be able to repeat a texture or specify for example 0.125 ! I made a .obj loader for Gosu which works perfectly, but I need to use SDL to have a good texturing job done. I really want to use Gosu only, please help me :)

Comment by project member julianraschke, Mar 08, 2009

bestguigui: Thanks for the forum link, I added it to the DocsOverview. Looks good, even though I didn't have time for the French course this spring and understand little of it :)

How are you drawing the .obj triangles right now, via Image#draw_as_quad? If you use ruby-opengl and Image#gl_info, you can just take values between the reported 'left' and 'right' coordinates.

Comment by bestguigui, Mar 08, 2009

Thanks for publishing my tutorial, I will update it for sure :)

I'm using ruby-opengl to texture an .obj 3D model. I understood that you provide "left", "right", "top", "bottom" to simplify, which it does, but I can't figure how to, for example, duplicate the texture in width.

In OpenGL, it will go like : glTexCoord2d(2.0, 0.0), but I can't do like glTexCoord2d(tex.right + tex.right, 0.0) or even multiply, which cause a textures mixin...

Comment by project member julianraschke, Mar 08, 2009

bestguigui: Ah, I see what you mean. Gosu used to have the optimization where images with a square, power-of-two

size and hard borders would use a fresh, single texture. I'll try to bring it back and document/guarantee it—that should fix your problem! (Assuming your textures happen to fufill these requirements)

Comment by bestguigui, Mar 08, 2009

My texture is a 512*512 .png file. When I ask for "left" and "right" value, I get 0.6279296875 and 0.8779296875. I'm able to ask for a texture crop, by lowering the right value (say about 0.7), but I can't ask for more to repeat it. If I understood right, It's just impossible for now ?

Comment by project member julianraschke, Mar 08, 2009

Right. But with the optimization I mentioned above you could be sure that the PNG file would become a single 512x512-sized texture, and that left/top would be 0 and right/bottom would be 1. The optimization actually existed a while ago, but only in the old Direct3D port (I think).

If you can tell me what operating system you are on I can give you a preview build by tonight.

Also, are you going to post your .obj loader to the Extending Gosu board or is it closed sourced? Sounds like something that will be useful to quite some people (myself included)! :)

Comment by bestguigui, Mar 08, 2009

I'm working on Windows XP, but I work with a Mac OS X user. So I'll be able to test both plateforms. I always share what I do, so there is no problem post it when it's done. Thanks a lot !

Comment by project member julianraschke, Mar 08, 2009

bestguigui: I uploaded preview bundles at http://www.raschke.de/julian/temp/gosu.bundle and http://www.raschke.de/julian/temp/gosu.so. I'll document the change after I release 0.7.13 publicly. If you would like to post your library before that date, these bundles will still be there.

Hope that helps! :)

Comment by bestguigui, Mar 09, 2009

Thanks a lot ! That's sure progressing : It's working perfectly since your modification. Because I don't want to extend this page (which is not its purpose), I'm gonna post a thread on your forum :)

Comment by christian.tietze, Mar 22, 2009

draw_line 0,0,0xFF000000,0,240,0xFF0000 should draw a 1px line at the left edge of my window. It doesn't. If I set x2=1 though it works. Using x1=1 and x2=3 the line moves to the right but instead of spanning 3px width it's only 2px wide ... is this the kind of "depending on OpenGL" you mentioned above?

BTW I'm not able to register at your forum because the reCAPTCHA expired or so (!!!)

Comment by project member julianraschke, Mar 22, 2009

christian.tietze: Basically, the lower right coordinate of a every drawing operation is exclusive. It seems to be consistent across drivers now that all ports use OpenGL, so I should docuent it. It is a bit weird for draw_line. I think the proper coordinates would be 1, 1—1, 240 in your case. This seems so weird to me that I think it should be changed, I'm just not sure if I already break anyone's design by that. :)

When did you last check the boards? There was a problem with recaptcha that I've fixed a while ago, and it seems to work today/right now. Maybe there was a short hiccup at the recaptcha servers, or you still have an old bookmark?

Enter a comment:

Wiki markup help show

Submit