

[Русская версия](#)

[PDF version of this page](#)

# What is Pixilang

Pixilang is a cross-platform programming language for graphics/sound applications. Originally created by Alexander Zolotov (NightRadio) and Mik Razuvaev (Goglus) for non-programmers, demosceners and designers.

Pixilang programs are stored in text files (UTF8) with extensions .txt or .pixi. So you can use your favorite text editor to create/edit these files. Pixilang has no built-in editor.

## Command line options

```
pixilang [options] [source_file_name]
[options]
  -c      generate bytecode; source_file_name.pixicode file will be produced.
          Note: *.pixicode files are architecture-dependent.
```

## Basics

The basis of the Pixilang - are containers and variables.

**Container** is a two-dimensional array of elements (numbers) of the same data type. Different containers may have different data types. Each container has its own unique ID number. ID is always a positive integer or zero: 0,1,2,3,4... Zero ID is ok, but it's the screen container number by default.

Container structure:

- two-dimensional table of container elements - it may be some image, text string, piece of audio, or other data;
- key color, which will be treated as transparent color (with alpha=0);
- reference to a container with alpha-channel;
- additional data:
  - properties (use `get_prop()`, `set_prop()`, `remove_props()`, or a `.` (dot) operator to access them);
  - animation (use Animation functions to access it).

The new container can be created with the `new()` function or returned by some other function. There are two ways to remove the container:

- by user through the `remove()` function;
- automatically after the end of the program.

There is no automatic garbage collector.

Maximum number of containers is 8192. But you can increase it using the parameter `pixi_containers_num` in `pixilang_config.ini`. Also you can change this number in the `user_code_info.cpp` (if you compile Pixilang from source) through the following code: `const char* g_app_options[] = { "pixi_containers_num", 0 }; int g_app_options_val[] = { 16384 };`

**Variable** can contain 32-bit signed integer or 32-bit floating point number.

Numbers can be described in different formats. Examples:

- 33 - decimal;
- 33.55 - decimal floating point;
- 0xA8BC - hexadecimal;
- 0b100101011 - binary;
- #FF9080 - color, as in HTML; base format is #RRGGBB, where RR - red intensity (from 00 to FF), GG - green intensity; BB - blue intensity.

## Examples

```
x = new( 4 ) //Create a new container with 4 pixels. Store its ID into the
variable x.
//Container elements are addressed from zero:
// first element - x[ 0 ];
// second element - x[ 1 ];
// ...
// last element - x[ container_size - 1 ]
x[ 0 ] = #0000FF //Assign blue color to the first pixel of container x.
x[ 2 ] = WHITE //Assign white color to pixel 2.
x[ 3 ] = RED //Assign red color to pixel 3 (last element).
x[ 4 ] = BLUE //Nothing will happen here because index 4 is out of range.
b = x[ 2 ] //Read an element with index 2.
//Here b contains the number corresponding to the white color.
c = x[ 8 ] //Try to read an element with index 8
//Here c contains a null value because index 8 is not available.
remove( x ) //Remove the container
```

```
c = new( 4, 4 ) //Create a new 2D 4x4 pixel container. Store its ID into the
variable x.
c[ 2, 2 ] = WHITE //Assign the white color to a pixel with coordinates x=2
y=2.
remove( c ) //Remove this container
```

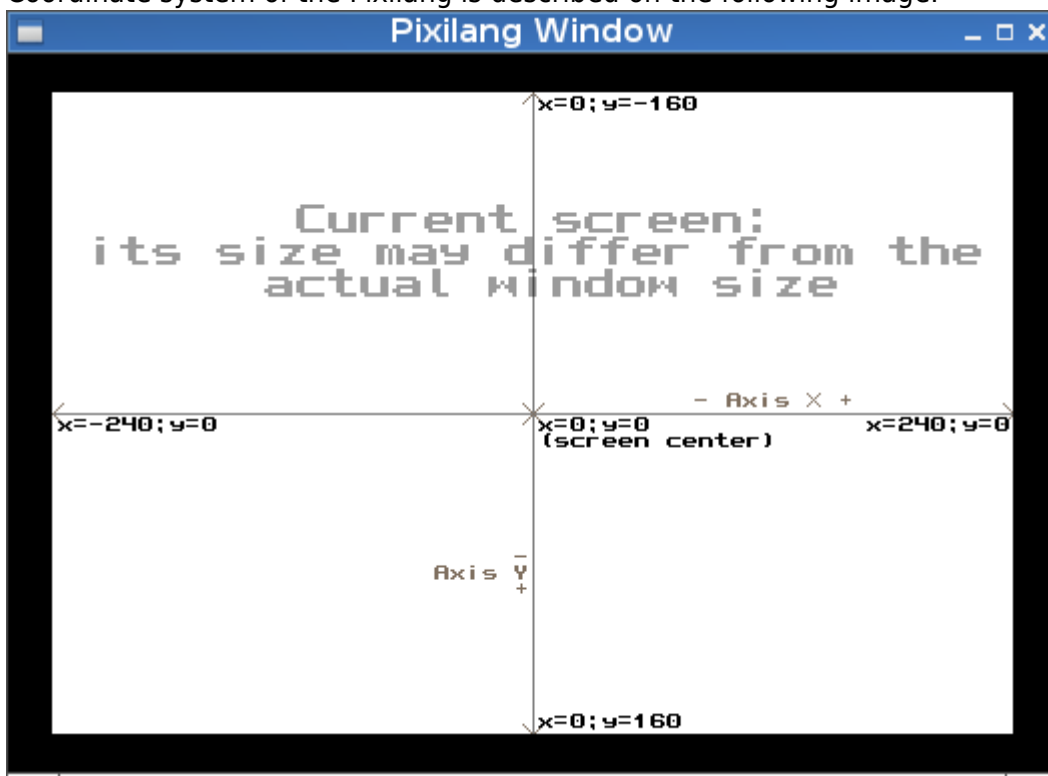
```
str = "Hello" //"Hello" is the string container with five 8-bit characters
(UTF-8 encoding).
//This container is automatically created during the bytecode generation.
//You don't need to delete it manually.
str[ 0 ] = 'h' //Change the first character from 'H' to 'h'.
print( str, 0, 0 ) //will print the word "hello" on the screen
//The string can be terminated with a null character or the end of a
container.
str[ 2 ] = 0 //Replace the char 'l' with a zero
print( str, 0, 8 ) //will print the word "he" on the screen
```

```

a = 4 //Global variable
fn function()
{
    $k = 2 //Local variable
    function2 = {
        //Body of another function
        $x = 899.334 //Local variable
        //$k is not accessible here
    }
    //$x is not accessible here
}
//$k and $x are not accessible here

```

Coordinate system of the Pixilang is described on the following image:



## Paths and file names

```

//File is in the same folder with the pixi-program:
"filename"

//File is in the current working directory:
// Linux/Windows/OSX: in the same folder with Pixilang;
// iOS: documents;
// WinCE: root of local filesystem (/);
"1:/filename"

//File is in the user directory for configs and caches :
// Linux: /home/username/.config/Pixilang;

```

```
// iOS: directory with application support files
(NSApplicationSupportDirectory);
// Android: internal files directory;
"2:/filename"

//File is in the temporary directory:
"3:/filename"
```

## Built-in operators

```
//if, else
if a == b
  { /*This code is executed if a equals b*/ }
else
  { /*This code is executed otherwise*/ }
if x == 4 && y == 2
  { /*This code is executed if x equals 4 and y equals 2*/ }

//while
a = 0
while( a < 3 )
{
  //This code is executed while a less than 3
  a + 3
}

//while + break
a = 0
while( a < 100 )
{
  if a == 10 { break } //If a = 10, break the loop
  //Use breakX operator to break X nested loops. Example: break2
  //Use breakall operator to break all current active loops
  a + 1
}

//while + continue
a = 0
b = 0
while( a < 100 )
{
  if a == 10 { a + 1 continue } //If a = 10, go to the next loop iteration
  // (ignore the next two lines of
code)
  a + 1
  b + 1
}
```

```
//go, goto
m1:
a + 1
goto m1 //Go to the m1 label

//halt, stop
halt //Stop the program here

//include
include "prog2.txt" //Include code from prog2.txt

//fn
fn fff( $x, $y ) //Declare function fff with parameters $x and $y
{
  //Function fff body
  ret //Simple return from the function
  ret( 4 ) //Return with value 4
}
```

The following table lists the mathematical operators. Priority 0 - the highest, such operations will be executed first.

Priority	Operator	Description	Result	Example
0	%	Modulo (remainder)	Integer	a = b % 4
0	/	Division	Floating point	a = b / 4
0	div	Integer division	Integer	a = b div 4
0	*	Multiplication	Depends on the operands	a = b * 4
1	+	Addition	Depends on the operands	a = b + 4
1	-	Subtraction	Depends on the operands	a = b - 4
2	>>	Bitwise right shift	Integer	a = b >> 4
2	<<	Bitwise left shift	Integer	a = b << 4
3	==	Equal to	Integer 1 or 0	if a == b { }
3	!=	Not equal to	Integer 1 or 0	if a != b { }
3	<	Less than	Integer 1 or 0	if a < b { }
3	>	Greater than	Integer 1 or 0	if a > b { }
3	<=	Less than or equal to	Integer 1 or 0	if a <= b { }
3	>=	Greater than or equal to	Integer 1 or 0	if a >= b { }
4		Bitwise OR	Integer	a = b   4
4	^	Bitwise XOR	Integer	a = b ^ 4
4	&	Bitwise AND	Integer	a = b & 4
5		Logical OR	Integer 1 or 0	if a    b { }
5	&&	Logical AND	Integer 1 or 0	if a && b { }

## Built-in constants

## Container types

64-bit types are not supported by default. But they can be enabled in `pixilang.h` for the custom build of Pixilang.

- INT - signed integer (size depends on the version of Pixilang);
- INT8 - signed integer (8 bit); range: -128 to 127;
- INT16 - signed integer (16 bit); range: -32768 to 32767;
- INT32 - signed integer (32 bit); range: -2147483648 to 2147483647;
- INT64 - signed integer (64 bit);
- FLOAT - floating point (size depends on the version of Pixilang);
- FLOAT32 - floating point (32 bit);
- FLOAT64 - floating point (64 bit);
- PIXEL - pixel in native color format; will be converted to type INTx, where x - number of bits per pixel.

## Container flags

- CFLAG\_INTERP - enable software interpolation.

For OpenGL acceleration:

- GL\_MIN\_LINEAR;
- GL\_MAG\_LINEAR;
- GL\_NICEST - use 32bit color, when possible;
- GL\_NO\_XREPEAT;
- GL\_NO\_YREPEAT;
- GL\_NO\_ALPHA.

## Container resizing flags

- RESIZE\_INTERP1 - rough resizing;
- RESIZE\_INTERP2 - resize with linear interpolation;
- RESIZE\_UNSIGNED\_INTERP2;
- RESIZE\_COLOR\_INTERP1;
- RESIZE\_COLOR\_INTERP2.

## Container copying flags

- COPY\_NO\_AUTOROTATE - don't rotate pixels from GL\_SCREEN;
- COPY\_CLIPPING - check and limit the values, if the type of the source container differs from the destination.

## Sizes

- INT\_SIZE - max size (in bytes) of the signed integer;
- FLOAT\_SIZE - max size (in bytes) of the floating point number;
- INT\_MAX - max positive integer;
- COLORBITS - number of bits per pixel.

## ZLib

Compression levels:

- Z\_NO\_COMPRESSION;
- Z\_BEST\_SPEED;
- Z\_BEST\_COMPRESSION;
- Z\_DEFAULT\_COMPRESSION.

## File formats

- FORMAT\_RAW;
- FORMAT\_JPEG;
- FORMAT\_PNG;
- FORMAT\_GIF;
- FORMAT\_WAVE;
- FORMAT\_AIFF (load only);
- FORMAT\_PIXICONTAINER - entire Pixilang container with properties and animation.

## Load/Save options

- LOAD\_FIRST\_FRAME - load first frame only.

GIF saving:

- GIF\_GRAYSCALE;
- GIF\_DITHER.

JPEG saving:

- JPEG\_H1V1 - YCbCr, no subsampling (H1V1, YCbCr 1x1x1, 3 blocks per MCU);
- JPEG\_H2V1 - YCbCr, H2V1 subsampling (YCbCr 2x1x1, 4 blocks per MCU);
- JPEG\_H2V2 - YCbCr, H2V2 subsampling (YCbCr 4x1x1, 6 blocks per MCU); default;
- JPEG\_TWOPASS - number of passes = 2.

## Colors

- ORANGE;

- BLACK;
- WHITE;
- YELLOW;
- RED;
- GREEN;
- BLUE;

## Alignment

- TOP;
- BOTTOM;
- LEFT;
- RIGHT.

## Effects

Effect types for effector() function:

- EFF\_NOISE - noise;
- EFF\_SPREAD\_LEFT - random pixel shift to the left;
- EFF\_SPREAD\_RIGHT - random pixel shift to the right;
- EFF\_SPREAD\_UP - random pixel shift to the top;
- EFF\_SPREAD\_DOWN - random pixel shift to the bottom;
- EFF\_HBLUR - horizontal blur;
- EFF\_VBLUR - vertical blur;
- EFF\_COLOR - color fill.

## OpenGL

gl\_draw\_arrays() (analog of the glDrawArrays()) modes:

- GL\_POINTS;
- GL\_LINE\_STRIP;
- GL\_LINE\_LOOP;
- GL\_LINES;
- GL\_TRIANGLE\_STRIP;
- GL\_TRIANGLE\_FAN;
- GL\_TRIANGLES.

gl\_blend\_func() (analog of the glBlendFunc()) operations:

- GL\_ZERO;
- GL\_ONE;
- GL\_SRC\_COLOR;
- GL\_ONE\_MINUS\_SRC\_COLOR;
- GL\_DST\_COLOR;
- GL\_ONE\_MINUS\_DST\_COLOR;



- GL\_SRC\_ALPHA;
- GL\_ONE\_MINUS\_SRC\_ALPHA;
- GL\_DST\_ALPHA;
- GL\_ONE\_MINUS\_DST\_ALPHA;
- GL\_SRC\_ALPHA\_SATURATE.

State variables (for `gl_get_int()` and `gl_get_float()`):

- GL\_MAX\_TEXTURE\_SIZE;
- GL\_MAX\_VERTEX\_ATTRIBS;
- GL\_MAX\_VERTEX\_UNIFORM\_VECTORS;
- GL\_MAX\_VARYING\_VECTORS;
- GL\_MAX\_VERTEX\_TEXTURE\_IMAGE\_UNITS;
- GL\_MAX\_TEXTURE\_IMAGE\_UNITS;
- GL\_MAX\_FRAGMENT\_UNIFORM\_VECTORS.

Default shader names for the `gl_new_prog()`:

- GL\_SHADER\_SOLID - solid color;
- GL\_SHADER\_GRAD - gradient color;
- GL\_SHADER\_TEX\_ALPHA\_SOLID - solid color + one channel (opacity) texture;
- GL\_SHADER\_TEX\_ALPHA\_GRAD - gradient color + one channel (opacity) texture;
- GL\_SHADER\_TEX\_RGB\_SOLID - solid color + texture;
- GL\_SHADER\_TEX\_RGB\_GRAD - gradient color + texture.

Global OpenGL containers:

- GL\_SCREEN; you can't read pixels from GL\_SCREEN directly, but you can use `copy( destination, GL_SCREEN )` for this;
- GL\_ZBUF.

## Audio

Flags for `set_audio_callback()` function:

- AUDIO\_FLAG\_INTERP2 - linear interpolation (2 points).

## MIDI

Flags for `midi_get_device()`, `midi_open_port()` functions:

- MIDI\_PORT\_READ;
- MIDI\_PORT\_WRITE.

## Events

- EVT - ID of the container with event, which used by `get_event()` function.

EVT field numbers:

- EVT\_TYPE - event type;
- EVT\_FLAGS - event flags;
- EVT\_TIME - event time;
- EVT\_X (0 - center of the screen);
- EVT\_Y (0 - center of the screen);
- EVT\_KEY - ASCII code of pressed key, or one of the KEY\_xxx constants;
- EVT\_SCANCODE - key scancode, or touch number for multitouch devices;
- EVT\_PRESSURE - touch pressure (normal - 1024);
- EVT\_UNICODE.

Event types (for the EVT\_TYPE field):

- EVT\_MOUSEBUTTONDOWN - first touch;
- EVT\_MOUSEBUTTONUP;
- EVT\_MOUSEMOVE;
- EVT\_TOUCHBEGIN - 2th, 3th etc. touch; only for multitouch devices;
- EVT\_TOUCHEND;
- EVT\_TOUCHMOVE;
- EVT\_BUTTONDOWN;
- EVT\_BUTTONUP;
- EVT\_SCREENRESIZE;
- EVT\_QUIT - Virtual Machine will be closed; if EVT[ EVT\_SCANCODE ] = 0, this event can't be ignored, user code must be terminated as soon as possible; if EVT[ EVT\_SCANCODE ] = 1, this event can be ignored by user.

Event flags (for the EVT\_FLAGS field):

- EVT\_FLAG\_SHIFT;
- EVT\_FLAG\_CTRL;
- EVT\_FLAG\_ALT;
- EVT\_FLAG\_MODE;
- EVT\_FLAG\_MODS (mask with all modifiers like Shift, Ctrl, etc.);
- EVT\_FLAG\_DOUBLECLICK.

Event key codes (for the EVT\_KEY field):

- KEY\_MOUSE\_LEFT;
- KEY\_MOUSE\_MIDDLE;
- KEY\_MOUSE\_RIGHT;
- KEY\_MOUSE\_SCROLLUP;
- KEY\_MOUSE\_SCROLLDOWN;
- KEY\_BACKSPACE;
- KEY\_TAB;
- KEY\_ENTER;
- KEY\_ESCAPE;
- KEY\_SPACE;
- KEY\_F1;
- KEY\_F2;
- KEY\_F3;
- KEY\_F4;

- KEY\_F5;
- KEY\_F6;
- KEY\_F7;
- KEY\_F8;
- KEY\_F9;
- KEY\_F10;
- KEY\_F11;
- KEY\_F12;
- KEY\_UP;
- KEY\_DOWN;
- KEY\_LEFT;
- KEY\_RIGHT;
- KEY\_INSERT;
- KEY\_DELETE;
- KEY\_HOME;
- KEY\_END;
- KEY\_PAGEUP;
- KEY\_PAGEDOWN;
- KEY\_CAPS;
- KEY\_SHIFT;
- KEY\_CTRL;
- KEY\_ALT;
- KEY\_MENU;
- KEY\_UNKNOWN (system virtual key code = code - KEY\_UNKNOWN).

Constants for the `set_quit_action()` function:

- QA\_NONE;
- QA\_CLOSE\_VM.

## Threads

Flags for `thread_create()`:

- `THREAD_FLAG_AUTO_DESTROY` - thread with this flag will be destroyed automatically.

## Mathematical constants

- `M_E` -  $e$ .
- `M_LOG2E` -  $\log_2 e$ .
- `M_LOG10E` -  $\log_{10} e$ .
- `M_LN2` -  $\log_e 2$ .
- `M_LN10` -  $\log_e 10$ .
- `M_PI` -  $\pi$ .
- `M_2_SQRTPI` -  $2 / \sqrt{\pi}$ .
- `M_SQRT2` -  $\sqrt{2}$ .
- `M_SQRT1_2` -  $1 / \sqrt{2}$ .

## Data processing operations

For `op_cn()` function:

- `OP_MIN` - `op_cn()` return value =  $\min( C1[ i ], C1[ i + 1 ], \dots );$
- `OP_MAX` - `op_cn()` return value =  $\max( C1[ i ], C1[ i + 1 ], \dots );$
- `OP_MAXABS` - `op_cn()` return value =  $\max( | C1[ i ] + N |, | C1[ i + 1 ] + N |, \dots );$
- `OP_SUM` - `op_cn()` return value =  $C1[ i ] + C1[ i + 1 ] + \dots ;$
- `OP_LIMIT_TOP` - if  $C1[ i ] > N$  {  $C1[ i ] = N$  } ;
- `OP_LIMIT_BOTTOM` - if  $C1[ i ] < N$  {  $C1[ i ] = N$  } ;
- `OP_ABS` - absolute value;
- `OP_SUB2` - subtraction with reverse order of the operands ( $N - C1[ i ]$ );
- `OP_COLOR_SUB2` - color subtraction with reverse order of the operands ( $N - C1[ i ]$ );
- `OP_DIV2` - division with reverse order of the operands ( $N / C1[ i ]$ );
- `OP_H_INTEGRAL` - running sum (horizontal);
- `OP_V_INTEGRAL` - running sum (vertical);
- `OP_H_DERIVATIVE` - first difference (horizontal);
- `OP_V_DERIVATIVE` - first difference (vertical);
- `OP_H_FLIP`;
- `OP_V_FLIP`.

For `op_cn()`, `op_cc()` function:

- `OP_ADD` - addition;
- `OP_SADD` - addition with saturation;
- `OP_COLOR_ADD` - color addition;
- `OP_SUB` - subtraction;
- `OP_SSUB` - subtraction with saturation;
- `OP_COLOR_SUB` - color subtraction;
- `OP_MUL` - multiplication;
- `OP_SMUL` - multiplication with saturation;
- `OP_MUL_RSHIFT15` -  $( C1[ i ] * N ) \gg 15$ ;
- `OP_COLOR_MUL` - color multiplication;
- `OP_DIV` - division;
- `OP_COLOR_DIV` - color division;
- `OP_AND` - bitwise AND;
- `OP_OR` - bitwise OR;
- `OP_XOR` - bitwise XOR;
- `OP_LSHIFT` - bitwise left shift;
- `OP_RSHIFT` - bitwise right shift;
- `OP_EQUAL`;
- `OP_LESS`;
- `OP_GREATER`;
- `OP_COPY` - copy;
- `OP_COPY_LESS` - copy only if  $C1[ i ] < C2[ i ]$ ;
- `OP_COPY_GREATER` - copy only if  $C1[ i ] > C2[ i ]$ .

For `op_cc()` function:

- `OP_BMUL` - if  $C2[ i ] == 0$  {  $C1[ i ] = 0$  };

- OP\_EXCHANGE;
- OP\_COMPARE - comparison; return value: 0 indicates that the contents are equal; 1 indicates that the first element that does not match in both containers has a greater value in C1 than in C2; -1 indicates the opposite.

For op\_ccn() function:

- OP\_MUL\_DIV -  $C1[i] * C2[i] / N$ ;
- OP\_MUL\_RSHIFT -  $(C1[i] * C2[i]) \gg N$  (multiplication with bitwise right shift).

For generator() function:

- OP\_SIN - sine;
- OP\_SIN8 - fast, but rough sine (8bit table);
- OP\_RAND - pseudo random (from -amp to +amp).

## Sampler

- SMP\_INFO\_SIZE - size of the container with sample info.

Sample info field numbers:

- SMP\_DEST - destination container;
- SMP\_DEST\_OFF - destination offset;
- SMP\_DEST\_LEN - destination length;
- SMP\_SRC - container with sample;
- SMP\_SRC\_OFF\_H - sample offset (left part of fixed point value);
- SMP\_SRC\_OFF\_L - sample offset (right part of fixed point value from 0 to 65535);
- SMP\_SRC\_SIZE - sample size (or 0 for whole sample);
- SMP\_LOOP - loop start;
- SMP\_LOOP\_LEN - loop length (or 0, if no loop);
- SMP\_VOL1 - start volume (32768 = 1.0);
- SMP\_VOL2 - end volume (32768 = 1.0);
- SMP\_DELTA - delta (playing speed); fixed point ( $real\_value * 65536$ );
- SMP\_FLAGS - flags.

Sample info flags:

- SMP\_FLAG\_INTERP2 - linear interpolation (2 points);
- SMP\_FLAG\_INTERP4 - cubic spline interpolation (4 points);
- SMP\_FLAG\_PINGPONG - ping-pong loop;
- SMP\_FLAG\_REVERSE - reverse direction.

## Native code constants

- CCONV\_DEFAULT;
- CCONV\_CDECL;
- CCONV\_STDCALL;
- CCONV\_UNIX\_AMD64;

- CCONV\_WIN64.

## Constants for POSIX compatibility

- FOPEN\_MAX;
- SEEK\_CUR;
- SEEK\_END;
- SEEK\_SET;
- EOF;
- STDIN;
- STDOUT;
- STDERR.

## Various

- PIXILANG\_VERSION - Pixilang version ((major<<24) + (minor<<16) + (minor2<<16) + minor3); example: PIXILANG\_VERSION = 0x03040700 for v3.4.7;
- OS\_NAME - container with system name; examples: "ios", "win32", "linux";
- ARCH\_NAME - container with architecture name; examples: "x86", "x86\_64", "arm", "mips";
- LANG\_NAME - container with the name of current system language (in POSIX format [language][\_TERRITORY][.CODESET][@modifier]); examples: "en\_US", "ru\_RU.utf8";
- CURRENT\_PATH;
- USER\_PATH;
- TEMP\_PATH;
- OPENGL - 1 if OpenGL available; 0 - otherwise.

## Built-in global variables

- WINDOW\_XSIZE - user's window width (in pixels);
- WINDOW\_YSIZE - user's window height (in pixels);
- WINDOW\_SAFE\_AREA\_X;
- WINDOW\_SAFE\_AREA\_Y;
- WINDOW\_SAFE\_AREA\_W;
- WINDOW\_SAFE\_AREA\_H;
- FPS - frames per second;
- PPI - pixels per inch;
- UI\_SCALE - UI scale (defined by user in the global preferences); use it to scale your UI elements; for example: button\_size = PPI \* UI\_SCALE \* 0.5;
- UI\_FONT\_SCALE - similar to UI\_SCALE, but only for the text size.

## Reserved container properties

These properties can be used during playback, saving and loading of the containers:

- file\_format;
- sample\_rate;
- channels;
- loop\_start - starting point (frame number) of the sample loop;
- loop\_len - sample loop length (number of frames);
- loop\_type - sample loop type: 0-none; 1-normal; 2-bidirectional;
- frames - number of frames;
- frame - current frame number;
- fps - frames per second;
- play - auto-play status (0/1);
- repeat - repeat count (-1 - infinitely);
- start\_time - start time (in system ticks); automatically set by play();
- start\_frame - start frame; automatically set by play().

## Built-in functions

### Containers (memory management)

#### new

Create new data container.

Note: immediately after its creation, the container may contain some random values. You should clean it up or fill it with useful data.

#### Parameters ( xsize, ysize, type )

- xsize - width.
- ysize - height.
- type - type of the atomic element of the container. Valid values:
  - INT - signed integer (size depends on the version of Pixilang);
  - INT8 - signed integer (8 bit);
  - INT16 - signed integer (16 bit);
  - INT32 - signed integer (32 bit);
  - INT64 - signed integer (64 bit);
  - FLOAT - floating point (size depends on the version of Pixilang);
  - FLOAT32 - floating point (32 bit);
  - FLOAT64 - floating point (64 bit);
  - PIXEL - pixel; will be converted to type INTx, where x - number of bits per pixel;

**Return value:** ID of the container, or -1 (error).

#### Examples

```
p = new() //Create 1x1 container. Element type = pixel.  
p = new( 4 ) //Create 4x1 container. Element type = pixel.  
p = new( 4, 4 ) //Create 4x4 container. Element type = pixel.
```

```
p = new( 4, 4, INT32 ) //Create 4x4 container. Element type = INT32.
```

## remove

Remove a container.

### Parameters ( pixi )

- pixi - container ID.

### Examples

```
p = new() //Create new container  
remove( p ) //Remove it
```

## remove\_with\_alpha

Remove the container and its alpha channel (linked container).

### Parameters ( pixi )

- pixi - container ID.

## resize

Resize a container.

### Parameters ( pixi, xsize, ysize, type, flags )

- pixi - container ID;
- xsize - new width (or -1 if width not changed);
- ysize - height (or -1 if width not changed); optional;
- type - type of the atomic element of the container (or -1 if width not changed); optional; valid values:
  - INT8 - signed integer (8 bit);
  - INT16 - signed integer (16 bit);
  - INT32 - signed integer (32 bit);
  - INT64 - signed integer (64 bit);
  - FLOAT32 - floating point (32 bit);
  - FLOAT64 - floating point (64 bit);
  - PIXEL - pixel; will be converted to type INTx, where x - number of bits per pixel;
- flags - resizing flags (RESIZE\_xxx); optional.

**Return value:** 0 - successful; 1 - error.

### Examples

```
p = new( 4, 4 ) //Create new container
```



```
resize( p, 32, 32 ) //Resize it from 4x4 to 32x1  
resize( p, -1, 64 ) //Resize it from 32x32 to 32x64  
remove( p ) //Remove
```

## rotate

Rotate the container by angle\*90 degrees (clockwise).

**Параметры ( pixi, angle )**

## convert\_type

Convert the type of a container.

**Parameters ( pixi, new\_type )**

## clean

Clean a container (fill with zeroes or with selected values).

**Parameters ( dest\_cont, v, offset, count )**

- dest\_cont - container ID;
- v - value; optional; default - 0;
- offset - offset in dest\_cont; optional; default - 0;
- count - number of elements to fill; optional; default - whole container.

## Examples

```
p = new() //Create new container  
clean( p ) //Clean with zero  
clean( p, 3 )  
clean( p, 3, 0, 24 ) //Fill 24 elements with value 3  
remove( p ) //Remove
```

## clone

Make a duplicate of the container.

**Parameters ( pixi )**

- pixi - container ID.

**Return value:** ID of the new container, or -1 (error).

## copy

Copy container elements from src to dest.

### Parameters ( dest, src, dest\_offset, src\_offset, count, dest\_step, src\_step, flags )

- dest - destination container;
- src - source container;
- dest\_offset - offset in destination (first element to copy);
- src\_offset - offset in source (first element to copy);
- count - number of elements to copy;
- dest\_step - destination writing step (default - 1);
- src\_step - source reading step (default - 1);
- flags - COPY\_xxx flags; optional.

### Examples

```
//Copy all elements from img1 to img2:
copy( img2, img1 )
//Copy elements 8...200 from img1 to img2:
copy( img2, img1, 8, 8, 200 )
//Copy elements 8...400 from img1 to img2 with step 2:
copy( img2, img1, 8, 8, 200, 2, 2 )
```

**Return value:** number of copied elements.

## get\_size

Get size of a container (number of elements).

### Parameters ( pixi )

- pixi - container ID.

### Examples

```
p = new( 8, 8 ) //Create a new container 8x8
size = get_size( p ) //Save its size to the "size" variable
remove( p )
```

## get\_xsize

Get width of a container.

### Parameters ( pixi )

- pixi - container ID.

### Examples

```
p = new( 8, 8 ) //Create a new container 8x8
xsize = get_xsize( p ) //Save its width to the "xsize" variable
remove( p )
```

## get\_ysize

Get height of a container.

### Parameters ( pixi )

- pixi - container ID.

### Examples

```
p = new( 8, 8 ) //Create a new container 8x8
ysize = get_xsize( p ) //Save its height to the "ysize" variable
remove( p )
```

## get\_esize

Get the size of the element of a container (in bytes).

### Parameters ( pixi )

- pixi - container ID.

### Examples

```
p = new( 8, 8, INT32 ) //Create a new container 8x8; element type = INT32
esize = get_esize( p ) //Save its element's size to the "esize" variable
//Now esize = 4.
remove( p )
```

## get\_type

Get the type of the element of a container

### Parameters ( pixi )

- pixi - container ID.

### Examples

```
p = new( 8, 8, FLOAT32 ) //Create a new container 8x8; element type =
FLOAT32
type = get_type( p ) //Save its element's type to the "type" variable
//Now type = FLOAT32.
```

```
remove( p )
```

## get\_flags

Get container flags. List of supported flags is [here](#).

### Parameters ( pixi )

**Return value:** container flags.

## set\_flags

Set container flags. List of supported flags is [here](#).

### Parameters ( pixi, flags )

### Examples

```
set_flags( img, GL_MIN_LINEAR | GL_MAG_LINEAR )
```

## reset\_flags

Reset container flags.

### Parameters ( pixi, flags )

## get\_prop

Get property value of the container. Each container can have unlimited number of properties.

Another way to get a property - use a . (dot) operator. Example: value = image.fps

### Parameters ( pixi, prop\_name, def\_value )

- pixi - container ID;
- prop\_name - property name;
- def\_vaule - default value, if the property is not exists; optional.

**Return value:** value of selected property.

## set\_prop

Set property value of the container.

Another way to set a property - use a . (dot) operator. Example: image.fps = 30

## Parameters ( pixi, prop\_name, value )

- pixi - container ID;
- prop\_name - property name;
- value.

## Examples

```
p = new( 8, 8, INT32 ) //Create a new container
set_prop( p, "speed", 777 ) //Add "speed" property to this container
v = get_prop( p, "speed" ) //Read the value of "speed" property
//Or more simple way:
p.speed = 777
v = p.speed
```

## remove\_props

Remove all properties of the selected container.

## Parameters ( pixi )

## show\_memory\_debug\_messages

## Parameters ( enable )

## zlib\_pack

## Parameters ( source, level )

- source - container that will be compressed;
- level - Zlib compression level (Z\_NO\_COMPRESSION, Z\_BEST\_SPEED, Z\_BEST\_COMPRESSION, Z\_DEFAULT\_COMPRESSION); optional.

**Return value:** ID of the new compressed container (created from the source) or -1 in case of error.

## zlib\_unpack

## Parameters ( source )

- source - container that will be decompressed.

**Return value:** ID of the new decompressed container (created from the source) or -1 in case of error.

# Strings

## num\_to\_str

Aliases: num2str.

Convert number to string.

### Parameters ( str, num )

- str - container for the string;
- num - numeric value.

### Examples

```
v = 45.64
s = ""
num_to_str( s, v )
fputs( s ) fputs( "\n" )
```

## str\_to\_num

Aliases: str2num.

Convert string to number.

### Parameters ( str )

- str - container with the string.

**Return value:** numeric value.

### Examples

```
a = str_to_num( "-55.44" )
b = a + 4
```

## strcat

Appends a copy of the source string to the destination string. Both strings can be with terminating null character or without it (if the size of the container = number of characters in the string). Size of the source string can be changed after this function executes.

### Parameters ( destination, source )

### Parameters ( dest, dest\_offset, source, source\_offset )

## strcmp

Compares the string str1 to the string str2. Both strings can be with terminating null character or without it (if the size of the container = number of characters in the string).

**Parameters ( str1, str2 )**

**Parameters ( str1, str1\_offset, str2, str2\_offset )**

**Return value:** a zero value indicates that both strings are equal; a value greater than zero indicates that the first character that does not match has a greater value in str1 than in str2; and a value less than zero indicates the opposite.

## strlen

Returns the length of string str. String can be with terminating null character or without it (if the size of the container = number of characters in the string).

**Parameters ( str )**

**Parameters ( str, str\_offset )**

**Return value:** length of string str.

## strstr

Returns the offset of the first occurrence of str2 in str1, or a -1 if str2 is not part of str1.

**Parameters ( str1, str2 )**

**Parameters ( str1, str1\_offset, str2, str2\_offset )**

**Return value:** offset of the first occurrence of str2 in str1, or a -1 if str2 is not part of str1.

## sprintf

Writes into the container str a string consisting on a sequence of data formatted as the format argument specifies. Detailed format description can be found here:

[https://en.wikipedia.org/wiki/Printf\\_format\\_string](https://en.wikipedia.org/wiki/Printf_format_string)

**Parameters ( str, format, ... )**

**Return value:** on success, the total number of characters written is returned; on failure, a negative number is returned.

## Examples

```
sprintf( str, "Some text\n" ) //Just write some text to the str
```

```
printf( str, "Number: %d\n", 12 ) //Write signed decimal integer
```

## printf

Same as sprintf, but the destination is STDOUT.

**Parameters ( format, ... )**

## fprintf

Print formatted output to the stream (opened by fopen() or fopen\_mem()).

**Parameters ( stream, format, ... )**

## Log management

### logf

The same as printf, but the destination is the Pixilang log buffer.

**Parameters ( format, ... )**

### get\_log

Get the Pixilang log buffer.

**Return value:** the new container with the log buffer; (container must be removed manually).

### get\_system\_log

Get the Pixilang system log buffer.

**Return value:** the new container with the system log buffer; (container must be removed manually).

## Working with files

### load

Load container from the file.

**Parameters ( filename, options )**



**Return value:** ID of the new loaded container, or -1 in case of error.

## Examples

```
c = load( "smile.jpg" )
if c >= 0
{
    file_format = c.file_format //FORMAT_RAW / FORMAT_JPEG / ...
    if file_format == FORMAT_WAVE
    {
        printf( "This is audio file\n" )
    }
}
```

## fload

Load container from the stream (opened by fopen() or fopen\_mem()).

### Parameters ( stream, options )

**Return value:** ID of the new loaded container, or -1 in case of error.

## save

Save container to the file.

### Parameters ( pixi, filename, format, options )

- pixi;
- filename;
- format (FORMAT\_RAW, FORMAT\_JPEG, FORMAT\_PNG, FORMAT\_GIF, FORMAT\_WAVE, FORMAT\_PIXICONTAINER);
- options:
  - for JPEG: quality from 0 - worst to 100 - best;
  - for GIF: flags GIF\_GRAYSCALE, GIF\_DITHER.

**Return value:** 0 on success.

## Examples

```
c = load( "smile.jpg" )
save( c, "smile.png", FORMAT_PNG )
save( c, "smile2.jpg", FORMAT_JPEG ) //Quality = 85 (default)
save( c, "smile3.jpg", FORMAT_JPEG, 100 ) //Quality = 100
```

## fsave

Save container to the stream (opened by fopen() or fopen\_mem()).

## Parameters ( pixi, stream, format, options )

**Return value:** 0 on success.

## get\_real\_path

Convert Pixilang-style path (e.g. 1:/img.png) to the real filesystem path.

## Parameters ( path )

**Return value:** ID of the container with real filesystem path; (container must be removed manually).

## Examples

```
filename = "1:/some_file"
realpath = get_real_path( filename )
printf( "File name: %s; Real path: %s\n", filename, realpath )
remove( realpath )
```

## new\_flist

## remove\_flist

## get\_flist\_name

## get\_flist\_type

## flist\_next

Functions for getting a list of files.

## Examples

```
path = CURRENT_PATH
mask = -1 //Examples: "txt/doc", "avi"; or -1 for all files.
fl = new_flist( path, mask )
if fl >= 0
{
    printf( "Some files found in %s\n", path )
    while( 1 )
    {
        file_name = get_flist_name( fl )
        file_type = get_flist_type( fl ) //0 - file; 1 - directory;
        if file_type == 0
        {
            printf( "FILE %s%s\n", path, file_name )
        }
    }
}
```

```
    }
    else
    {
        printf( "DIR  %s%s\n", path, file_name )
    }
    if flist_next( fl ) == 0 //Go to the next file
    {
        //No more files
        break
    }
}
remove_flist( fl )
}
```

## **get\_file\_size**

Get the size of selected file.

**Parameters ( filename )**

## **remove\_file**

**Parameters ( filename )**

## **rename\_file**

**Parameters ( old\_filename, new\_filename )**

## **copy\_file**

**Parameters ( source\_filename, destination\_filename )**

## **create\_directory**

**Parameters ( directory\_name, mode )**

- directory\_name;
- mode - system specific mode; optional.

## **set\_disk0**

Use the `_stream_` (opened by `fopen()` or `fopen_mem()`) as virtual disk 0:/ . `_stream_` must point to the opened TAR archive.

## Parameters ( stream )

- stream - stream, or 0 if you want to disable disk 0:/

## get\_disk0

## fopen

The fopen() function shall open the file whose pathname is the string pointed to by `_filename_`, and associates a stream with it.

### Parameters ( filename, mode )

- filename;
- mode:
  - r or rb - open file for reading;
  - w or wb - truncate to zero length or create file for writing;
  - a or ab - append; open or create file for writing at end-of-file;
  - r+ or rb+ or r+b - open file for update (reading and writing);
  - w+ or wb+ or w+b - truncate to zero length or create file for update;
  - a+ or ab+ or a+b - append; open or create file for update, writing at end-of-file.

**Return value:** upon successful completion, fopen() shall return ID of the object controlling the stream; otherwise, 0 shall be returned.

### Examples

```
f = fopen( "/tmp/data.txt", "rb" ) //Open file data.txt for reading
fclose( f ) //...and close it.
```

## fopen\_mem

Open the `_data_` container as file.

### Parameters ( data )

**Return value:** upon successful completion, fopen\_mem() shall return ID of the object controlling the stream; otherwise, 0 shall be returned.

## fclose

The fclose() function shall cause the stream to be flushed and the associated file to be closed.

### Parameters ( stream )

**Return value:** upon successful completion, fclose() shall return 0.

### Examples

```
f = fopen( "/tmp/data.txt", "rb" ) //Open file data.txt for reading.
c = fgetc( f ) //Get a byte from this file.
fclose( f ) //Close the stream.
```

## fputc

Put a byte on a stream.

**Parameters ( c, stream )**

### Examples

```
f = fopen( "/tmp/data.txt", "wb" ) //Open file data.txt for writing.
fputc( 0x12, f ) //Put a byte 0x12 to this file.
fclose( f ) //Close the stream.
```

## fputs

Put a string on a stream.

**Parameters ( s, stream )**

### Examples

```
f = fopen( "/tmp/data.txt", "wb" ) //Open file data.txt for writing.
str = "Hello!"
fputc( str, f ) //Put a string "Hello!" to this file.
fclose( f ) //Close the stream.
```

## fwrite

The fwrite() function shall write, from the container **data**, up to **size** bytes, to the stream pointed to by **stream**.

**Parameters ( data, size, stream, data\_offset\_optional )**

**Return value:** the number of bytes successfully written, which may be less than **size** if a write error is encountered.

### Examples

```
f = fopen( "/tmp/data.txt", "wb" ) //Open file data.txt for writing.
str = "Hello!"
fwrite( str, 2, f ) //Put first two bytes from the string "Hello!" to this
file.
fclose( f ) //Close the stream.
```

## fgetc

Get a byte from a stream.

### Parameters ( stream )

**Return value:** upon successful completion, fgetc() shall return the next byte from the input stream pointed to by **stream**.

## fgets

Get a string from a stream.

### Parameters ( s, n, stream, offset )

**Return value:** length of the string.

### Examples

```
string = new( 256, 1, INT8 )
f = fopen( "/tmp/data.txt", "rb" ) //Open file data.txt for reading.
fgets( string, 256, f ) //Get a string from this file.
fclose( f ) //Close the stream.
```

## fread

The fread() function shall read into the container pointed to by **data** up to **size** bytes, from the stream pointed to by **stream**.

### Parameters ( data, size, stream, data\_offset\_optional )

**Return value:** the number of bytes successfully read which is less than **size** only if a read error or end-of-file is encountered.

## feof

Test end-of-file indicator on a stream.

### Parameters ( stream )

**Return value:** non-zero if the end-of-file indicator is set for **stream**.

## fflush

Flush a stream.

## Parameters ( stream )

### fseek

Reposition a file-position indicator in a stream.

#### Parameters ( stream, offset, origin )

- stream;
- offset;
- origin:
  - SEEK\_SET - beginning of file;
  - SEEK\_CUR - current position of the file pointer;
  - SEEK\_END - end of file.

### ftell

Return a file offset in a stream.

#### Parameters ( stream )

**Return value:** the current value of the file-position indicator for the stream measured in bytes from the beginning of the file.

#### Examples

```
//One of the ways to get the file size:  
f = fopen( "/tmp/data.txt", "rb" )  
fseek( f, 0, SEEK_END )  
size_of_file = ftell( f )  
fclose( f )
```

### setxattr

Set an extended attribute value of the file.

#### Parameters ( path, attr\_name, data, data\_size\_in\_bytes, flags )

**Return value:** 0 on success, or -1 in case of error.

## Graphics

### frame

Draw the current screen on the display and sleep for selected number of milliseconds.

## Parameters ( delay, x, y, xsize, ysize )

- delay - pause length in milliseconds;
- x, y, xsize, ysize - region of the screen; optional parameters.

### Return value:

- 0 - successful;
- -1 - no screen;
- -2 - timeout (graphics engine is suspended?).

## vsync

Enable/disable vertical synchronization (when possible). vsync(1) - enable. vsync(0) - disable.

## Parameters ( enable )

## set\_pixel\_size

Change the size of the screen pixels. Default size = 1.

## Parameters ( size )

## get\_pixel\_size

Get the size of the screen pixel.

## set\_screen

Set current screen. ID of the default screen container is 0.

## Parameters ( pixi )

- pixi - ID of the container with image which will be used as the screen.

## get\_screen

Get current screen.

**Return value:** screen container ID.

## set\_zbuf

## Parameters ( zbuf\_container )



Set container (INT32 elements) with Z-buffer.

## **get\_zbuf**

## **clear\_zbuf**

## **get\_color**

Get color by r,g,b (red,green,blue).

### **Parameters ( red, green, blue )**

- red - red intensity (0..255);
- green - green intensity (0..255);
- blue - blue intensity (0..255).

**Return value:** color value.

## **get\_red**

Get red component intensity in a selected color.

### **Parameters ( color )**

**Return value:** red component intensity; from 0 to 255.

## **get\_green**

Get green component intensity in a selected color.

### **Parameters ( color )**

**Return value:** green component intensity; from 0 to 255.

## **get\_blue**

Get blue component intensity in a selected color.

### **Parameters ( color )**

**Return value:** blue component intensity; from 0 to 255.

## **get\_blend**

Get an intermediate color value between two selected colors.

### **Parameters ( c1, c2, v )**

- c1 - first color;
- c2 - second color;
- v - position between c1 and c2; 0 - closer to c1; 255 - closer to c2.

**Return value:** intermediate color value.

## **transp**

Set transparency.

### **Parameters ( t )**

- t - transparency from 0 to 255.

## **get\_transp**

## **clear**

Clear current screen.

### **Parameters ( color )**

## **dot**

Draw a dot.

### **Parameters ( x, y, color )**

## **dot3d**

Draw a dot in 3D.

### **Parameters ( x, y, z, color )**

## **get\_dot**

Get a dot's color.

### **Parameters ( x, y )**

**Return value:** color value.

## **get\_dot3d**

Get a dot's color in 3D.

**Parameters ( x, y, z )**

**Return value:** color value.

## **line**

Draw a line.

**Parameters ( x1, y1, x2, y2, color )**

## **line3d**

Draw a line in 3D.

**Parameters ( x1, y1, z1, x2, y2, z2, color )**

## **box**

Draw a rectangle.

**Parameters ( x, y, xsize, ysize, color )**

## **fbox**

Draw a filled rectangle.

**Parameters ( x, y, xsize, ysize, color )**

## **pixi**

Display the container with image.

**Parameters ( pixi\_cont, x, y, color, xscale, yscale, src\_x, src\_y, src\_xsize, src\_ysize )**

- pixi\_cont - container ID (source);
- x;
- y;
- color - color of the filter; optional; default value is WHITE;
- xscale - scaling factor (x axis); optional; default value is 1;
- yscale - scaling factor (y axis); optional; default value is 1;
- src\_x - x offset in pixi\_cont; default - 0;

- `src_y` - y offset in `pixi_cont`; default - 0;
- `src_xsize` - width of drawable `pixi_cont` region; default - width of `pixi_cont`;
- `src_ysize` - height of drawable `pixi_cont` region; default - height of `pixi_cont`.

## Examples

```
pixi( image )
pixi( image, 10, 20 )
pixi( image, 30, 40, GREEN )
pixi( image, 90, 20, GREEN, 0.5, 0.5 )
```

## triangles3d

Draw array of triangles.

### Parameters ( `vertices`, `triangles`, `tnum` )

- `vertices` - container of vertices; width = 8; height = number of vertices; vertex data format: X, Y, Z, TextureX (0..Width), TextureY (0..Height), Unused, Unused, Unused;
- `triangles` - container of triangles; width = 8; height = number of triangles; triangle data format: NumberOfVertex1, NumberOfVertex2, NumberOfVertex3, Color, Texture (or -1), Opacity (0..255), Unused, Order (triangles with lower Order value will be drawn first);
- `tnum` - number of triangles; optional.

## sort\_triangles3d

Sort triangles by Z value.

### Parameters ( `vertices`, `triangles`, `tnum` )

- `vertices` - container of vertices; format is the same as in `triangles3d()`;
- `triangles` - container of triangles; format is the same as in `triangles3d()`;
- `tnum` - number of triangles; optional.

## set\_key\_color

Set / reset the color of transparency in the container.

### Parameters ( `pixi`, `color` )

- `pixi` - container ID;
- `color` - color to be transparent; ignore this parameter if you want to disable transparent color for this container.

## get\_key\_color

## set\_alpha

Attach a container with alpha channel to another container. Alpha channel should be of type INT8.

### Parameters ( pixi, alpha )

- pixi - container ID;
- alpha - container with alpha channel; ignore this parameter if you want to disable alpha-channel for the pixi container.

## get\_alpha

Get linked container with the alpha channel.

### Parameters ( cont )

**Return value:** container ID or -1 (if no alpha channel).

## print

Show text on the screen.

### Parameters ( text, x, y, color, align, max\_xsize )

- text - container with text;
- x, y - point of the alignment;
- color;
- align - alignment; optional;
- max\_xsize; optional.

### Examples

```
print( "Hello Pixi!", 0, 0 ) //color = WHITE; centered;
print( "line1\nline2", 50, 50, RED ) //centered;
print( "line1\nline2", -50, 50, RED, TOP | LEFT ) //alignment = top left;
```

## get\_text\_xsize

### Parameters ( text, align, max\_xsize )

- text;
- align; optional;
- max\_xsize; optional.

**Return value:** text width in pixels.

## get\_text\_ysize

### Parameters ( text, align, max\_xsize )

- text;
- align; optional;
- max\_xsize; optional.

**Return value:** text height in pixels.

## set\_font

### Parameters ( first\_char\_utf32, font\_image, xchars, ychars )

## get\_font

### Parameters ( char\_utf32 )

**Return value:** container ID with font for selected character.

## effector

Apply an effect to selected screen area. Coordinates of this function can't be changed by t\_xxx transformation functions.

### Parameters ( type, power, color, x, y, xsize, ysize, x\_step, y\_step )

## color\_gradient

Draw a color gradient.

### Parameters ( color1, transp1, color2, transp2, color3, transp3, color4, transp4, x, y, xsize, ysize, x\_step, y\_step )

## split\_rgb

Split a container (with pixels) by color channels (red, green, blue) and vice versa.

### Parameters ( direction, image, red\_channel, green\_channel, blue\_channel, image\_offset, channel\_offset, size )

- direction: 0 - from image to RGB; 1 - from RGB to image;
- image - source/destination container of type PIXEL;
- red\_channel - container with red components of the image; optional; can be -1;
- green\_channel - container with green components of the image; optional; can be -1;

- `blue_channel` - container with blue components of the image; optional; can be -1;
- `image_offset` - offset in the image container; optional;
- `channel_offset` - offset in the channel containers; optional;
- `size` - number of pixels to split; optional.

## Examples

```
img = load( "some_image.jpg" )
xsize = get_xsize( img )
ysize = get_ysize( img )
r = new( xsize, ysize, INT16 )
g = new( xsize, ysize, INT16 )
b = new( xsize, ysize, INT16 )
split_rgb( 0, img, r, g, b ) //Convert image to RGB
//Get red value (from 0 to 255) of the first pixel:
value = r[ 0 ]
```

## split\_ycbcr

Same as `split_rgb()` but for YCbCr conversion.

## OpenGL base

OpenGL-version of Pixilang is based on the [OpenGL ES 2.0](#) with [OpenGL ES Shading Language 1.0](#) (GLSL ES).

Here is very good [OpenGL ES 2.0 Quick Reference Card](#).

## set\_gl\_callback

Set OpenGL frame drawing callback.

### Parameters ( `gl_callback`, `user_data` )

- `gl_callback` - callback function with parameters (`$user_data`); almost all graphics functions will be redirected to OpenGL within this callback;
- `user_data` - some user-defined data that will be sent to `gl_callback()` during the frame redrawing.

## Examples

```
fn gl_callback( $user_data )
{
    set_screen( GL_SCREEN ) //Enable OpenGL drawing mode
    clear( YELLOW )
    set_screen( 0 ) //Back to the default screen
}

set_gl_callback(
```

```
gl_callback, //OpenGL frame drawing function
0 ) //Some user-defined data

while( 1 )
{
  while( get_event() ) { if EVT[ EVT_TYPE ] == EVT_QUIT { break2 } }
  frame()
}

set_gl_callback( -1 ) //Remove OpenGL callback
```

## remove\_gl\_data

Remove all GL-specific data from the container. This data will be created again inside the OpenGL drawing callback.

### Parameters ( container )

## gl\_draw\_arrays

Hybrid of the OpenGL functions glColor4ub(), glBindTexture(), glVertexPointer(), glColorPointer(), glTexCoordPointer(), glDrawArrays().

Can only be used inside the OpenGL drawing callback.

### Parameters ( mode, first, count, color\_r, color\_g, color\_b, color\_a, texture, vertex\_array, color\_array, texcoord\_array )

- mode - what kind of primitives to render:
  - GL\_POINTS;
  - GL\_LINE\_STRIP;
  - GL\_LINE\_LOOP;
  - GL\_LINES;
  - GL\_TRIANGLE\_STRIP;
  - GL\_TRIANGLE\_FAN;
  - GL\_TRIANGLES;
- first - starting index in the enabled arrays;
- count - number of indices to be rendered;
- color\_r, color\_g, color\_b, color\_a - RGBA color (only if color\_array is not specified);
- texture - container with texture, or -1;
- vertex\_array - INT8, INT16 or FLOAT32 container with vertices;
- color\_array - INT8 or FLOAT32 container with RGBA colors, or -1; optional;
- texcoord\_array - INT8, INT16 or FLOAT32 container with texture coordinates; optional.

## gl\_blend\_func

Full analog of the OpenGL function glBlendFunc() (specify pixel arithmetic) or glBlendFuncSeparate (if sfactor\_alpha and dfactor\_alpha are specified).



Can only be used inside the OpenGL drawing callback.

Call this function without parameters, if you want to reset the pixel arithmetic to the default values.

### Parameters ( **sfactor**, **dfactor**, **sfactor\_alpha**, **dfactor\_alpha** )

- **sfactor** - specifies how the red, green, and blue blending factors are computed;
- **dfactor** - specifies how the red, green, and blue destination blending factors are computed;
- **sfactor\_alpha** - specified how the alpha source blending factor is computed; optional;
- **dfactor\_alpha** - specified how the alpha destination blending factor is computed; optional.

## gl\_bind\_framebuffer

Convert specified pixel container (cnum) to the OpenGL framebuffer (with attached texture) and bind it. All rendering operations will be redirected to this framebuffer. To unbind - just call this function without parameters. Pixel size does not affect the bound framebuffer.

Can only be used inside the OpenGL drawing callback.

### Parameters ( **cnum** )

## gl\_bind\_texture

Bind selected pixel container (cnum) to the specified texture image unit.

Can only be used inside the OpenGL drawing callback.

### Parameters ( **cnum**, **texture\_unit** )

- **cnum** - pixel container;
- **texture\_unit** - texture image unit number: 1, 2, 3, ... ; 0 is the main texture unit used by pixi().

### Example

```
fn gl_callback( $userdata )
{
    set_screen( GL_SCREEN )
    gl_bind_texture( some_image2, 1 ) //bind some_image2 to texture unit 1
    gl_bind_texture( some_image3, 2 ) //bind some_image3 to texture unit 2
    gl_use_prog( gl_prog ) //Use user-defined GLSL program
    gl_uniform( gl_prog.g_texture2, 1 ) //set shader variable: g_texture2 =
texture image unit 1
    gl_uniform( gl_prog.g_texture3, 2 ) //set shader variable: g_texture3 =
texture image unit 2
    pixi( some_image )
    gl_use_prog() //Back to default GLSL program
    set_screen( 0 ) //Back to the default screen
}
gl_vshader = GL_SHADER_TEX_RGB_SOLID //Vertex shader = default shader for
solid primitives drawing
gl_fshader = //Fragment shader
```

```
"uniform sampler2D g_texture; //main texture image unit 0 (set by pixi())
uniform sampler2D g_texture2; //texture image unit 1
uniform sampler2D g_texture3; //texture image unit 2
uniform vec4 g_color;
IN vec2 tex_coord_var;
void main()
{
    vec4 c1 = texture2D( g_texture, tex_coord_var );
    vec4 c2 = texture2D( g_texture2, tex_coord_var );
    vec4 c3 = texture2D( g_texture3, tex_coord_var );
    gl_FragColor = ( c1 + c2 + c3 ) * g_color;
}
"
gl_prog = gl_new_prog( gl_vshader, gl_fshader )
```

## gl\_get\_int

Get the value of the simple GL state variable. Full analog of the OpenGL function `glGetIntegerv()`. Can only be used inside the OpenGL drawing callback.

### Parameters ( pname )

- pname - symbolic constant indicating the state variable to be returned.

## gl\_get\_float

Get the value of the simple GL state variable. Full analog of the OpenGL function `glGetFloatv()`. Can only be used inside the OpenGL drawing callback.

### Parameters ( pname )

- pname - symbolic constant indicating the state variable to be returned.

## OpenGL shaders

OpenGL-version of Pixilang uses [OpenGL ES Shading Language 1.0](#) (GLSL ES) for the vertex and the fragment shaders.

Notes about the vertex shader syntax in Pixilang:

- use **IN** instead of **attribute** and **in** qualifiers;
- use **OUT** instead of **varying** and **out** qualifiers;
- use **LOWP**, **MEDIUMP** and **HIGHP** instead of the **lowp**, **mediump** and **highp** qualifiers;
- use **PRECISION( P, T )** instead of **precision P, T**.

Notes about the fragment shader syntax in Pixilang:

- use **IN** instead of **varying** and **in** qualifiers;

- use **LOWP**, **MEDIUMP** and **HIGHP** instead of the **lowp**, **mediump** and **highp** qualifiers;
- use **PRECISION( P, T )** instead of **precision P, T**.

## gl\_new\_prog

Create a new GLSL program - container with a vertex shader and a fragment shader. You can use this function anywhere in your code. Shader code will be compiled, and the program will be linked later - during the `gl_use_prog()` function call.

### Parameters ( **vertex\_shader, fragment\_shader** )

- `vertex_shader` - container with the source code of the vertex shader, or one of the `GL_SHADER_XXX` constants (predefined shaders);
- `fragment_shader` - container with the source code of the fragment shader, or one of the `GL_SHADER_XXX` constants (predefined shaders).

**Return value:** ID of the new container with GLSL program, or negative value in case of some error; (container must be removed manually).

## gl\_use\_prog

Use the GLSL program previously created by the `gl_new_prog()` function. If you use this program first time - it will be compiled and linked.

Can only be used inside the OpenGL drawing callback.

### Parameters ( **prog** )

## gl\_uniform

Change the value of the uniform variable within the GLSL program.

Can only be used inside the OpenGL drawing callback.

`gl_uniform()` can also change the contents of arrays if you use this function as follows: `gl_uniform( var_location, src_container, vector_size, first_vector, count )`, where `count` is the number of vectors to write to the array.

### Parameters ( **var\_location, v0, v1, v2, v3** )

- `var_location` - location of the uniform variable to be modified; you can get it from the GLSL program container easily: `PROGRAM.UNIFORM_NAME`;
- `v0, v1, v2, v3` - new values to be used for the specified uniform variable; `v1, v2` and `v3` are optional; number of values depends the number of components in the data type of the specified uniform variable.

## Examples

```
gl_use_prog( gl_prog ) //Use GLSL program gl_prog (vertex shader + fragment shader)
```

```
gl_uniform( gl_prog.g_time, get_timer( 0 ) ) //Set value of uniform variable g_time
```

## gl\_uniform\_matrix

Change the value of the uniform matrix within the GLSL program.

Can only be used inside the OpenGL drawing callback.

### Parameters ( size, matrix\_location, transpose, matrix )

- size - dimensionality of the matrix: 2 = 2×2 matrix; 3 = 3×3 matrix; 4 = 4×4 matrix;
- var\_location - location of the matrix to be modified; you can get it from the GLSL program container easily: PROGRAM.MATRIX\_NAME;
- transpose - transpose the matrix (0 - no; 1 - yes);
- matrix - ID of the container with the matrix.

### Examples

```
gl_use_prog( gl_prog ) //Use GLSL program gl_prog (vertex shader + fragment shader)
gl_uniform( 4, gl_prog.g_mat, 0, source_matrix ) //Set value of uniform matrix g_mat
```

## Animation

### pack\_frame

Pack current frame (from container data to hidden storage with frames). Frame number must be stored in the "frame" container property.

### Parameters ( pixi )

### unpack\_frame

Unpack current frame (from hidden storage with frames to container data). Frame number must be stored in the "frame" container property.

### Parameters ( pixi )

### create\_anim

Create the hidden storage with frames (animation) in the selected container.

### Parameters ( pixi )

## **remove\_anim**

Remove the hidden storage with frames (animation) from the selected container.

**Parameters ( pixi )**

## **clone\_frame**

Clone current frame. Frame number must be stored in the "frame" container property.

**Parameters ( pixi )**

## **remove\_frame**

Remove current frame. Frame number must be stored in the "frame" container property.

**Parameters ( pixi )**

## **play**

Enable auto-play mode. Frame will be changed automatically during the pixi() function call.

**Parameters ( pixi )**

## **stop**

Disable auto-play mode.

**Parameters ( pixi )**

## **Transformation**

Coordinate system transformation.

## **t\_reset**

Reset transformation.

## **t\_rotate**

**Parameters ( angle, x, y, z )**

- angle - specifies the angle of rotation, in degrees;
- x, y, z - specify the x, y, and z coordinates of a vector, respectively.

## **t\_translate**

**Parameters ( x, y, z )**

## **t\_scale**

**Parameters ( x, y, z )**

## **t\_push\_matrix**

Save current transformation matrix to the stack.

## **t\_pop\_matrix**

Restore previous saved transformation matrix from the stack.

## **t\_get\_matrix**

Get transformation matrix (4×4 FLOAT).

**Parameters ( matrix\_container )**

## **t\_set\_matrix**

Set transformation matrix (4×4 FLOAT).

**Parameters ( matrix\_container )**

## **t\_mul\_matrix**

Multiply transformation matrix (4×4 FLOAT). Current matrix = current matrix \* matrix\_container.

**Parameters ( matrix\_container )**

## **t\_point**

Transform point (container with 3 elements of type FLOAT).

## Parameters ( point\_coordinates )

# Audio

## set\_audio\_callback

### Parameters ( callback, userdata, sample\_rate, format, channels, flags )

- callback - audio callback (function address);
- userdata - user-defined data for the callback;
- sample\_rate - sample rate; set it to 0, if you want to use the sample rate from the global Pixilang preferences;
- format - sample format;
- channels - number of channels;
- flags - flags AUDIO\_FLAG\_xxx.

### Examples

```
fn audio_callback( $stream, $userdata, $channels, $frames, $time )
{
    generator( OP_SIN, $channels[ 0 ], 0, 32767 / 2, 0.1, 0 ) //Left channel
    generator( OP_SIN, $channels[ 1 ], 0, 32767 / 2, 0.1, 0 ) //Right channel
    ret(1)
    //return values:
    // 0 - silence, output channels are not filled;
    // 1 - output channels are filled;
    // 2 - silence, output channels are filled with zeros (or values close to
    zero).
}
//Start audio:
set_audio_callback( audio_callback, 0, 22050, INT16, 2, AUDIO_FLAG_INTERP2 )

//Stop audio:
set_audio_callback( -1 )
```

## get\_audio\_sample\_rate

### Parameters ( source )

- source: 0 - local sample rate; 1 - global (from Preferences) sample rate.

**Return value:** sample rate in Hz.

## enable\_audio\_input

### Parameters ( disable\_enable )

## get\_note\_freq

### Parameters ( note, finetune )

- note - note number; 0 = C-0; 1 = C#0; 2 = D-0 ...
- finetune - from -64 (previous note) to 64 (next note).

**Return value:** note frequency in Hz.

## MIDI

### midi\_open\_client

#### Parameters ( client\_name )

**Return value:** client ID or negative value in case of some error.

### midi\_close\_client

#### Parameters ( client\_id )

### midi\_get\_device

#### Parameters ( client\_id, device\_num, flags )

**Return value:** selected device name, or -1 if not exists.

### midi\_open\_port

#### Parameters ( client\_id, port\_name, device\_name, flags )

**Return value:** port ID or negative value in case of some error.

### midi\_reopen\_port

#### Parameters ( client\_id, port\_id )

### midi\_close\_port

#### Parameters ( client\_id, port\_id )



## **midi\_get\_event**

**Parameters ( client\_id, port\_id, data\_cont )**

**Return value:** size of the current MIDI event (in bytes).

## **midi\_get\_event\_time**

**Parameters ( client\_id, port\_id )**

**Return value:** time of the current event (in system ticks).

## **midi\_next\_event**

Go to the next event.

**Parameters ( client\_id, port\_id )**

## **midi\_send\_event**

**Parameters ( client\_id, port\_id, data\_cont, data\_size, t )**

# **Time**

## **start\_timer**

Start selected timer.

**Parameters ( timer\_num )**

## **get\_timer**

Get value (in milliseconds) of selected timer.

**Parameters ( timer\_num )**

**Return value:** 32bit value of selected timer (in milliseconds).

## **get\_year**

## **get\_month**

## **get\_day**

## **get\_hours**

## **get\_minutes**

## **get\_seconds**

## **get\_ticks**

Get current system tick counter (32bit).

## **get\_tps**

Get number of system ticks per second.

## **sleep**

### **Parameters ( delay )**

- delay - delay in milliseconds.

## **Events**

### **get\_event**

Get a new event from the system.

**Return value:** 0 - no events; 1 - event is received and placed into the container EVT.

### **set\_quit\_action**

Set the program's behavior when receiving event EVT\_QUIT.

#### **Parameters ( action )**

- action - action number.

Possible values for the action parameter:

- QA\_NONE - do nothing;
- QA\_CLOSE\_VM (default) - close the current virtual machine, but don't quit from Pixilang.

## Threads

### thread\_create

**Parameters ( thread\_function, user\_data, flags\_optional )**

**Return value:** thread ID or -1 if error occurred.

#### Examples

```
fn thread_body( $thread_id, $user_data )
{
    printf( "Thread code\n" )
}
thread_id = thread_create( thread_body, 0 )
err = thread_destroy( thread_id, 1000 ) //Wait for the thread to terminate
if err == 0 { printf( "Thread closed successful\n" ) }
if err == 1 { printf( "Time-out. Thread is not closed\n" ) }
if err == 2 { printf( "Some error occurred in thread_destroy() function\n" ) }
}
```

### thread\_destroy

**Parameters ( thread\_id, timeout\_ms )**

- thread\_id;
- timeout\_ms - time-out in milliseconds; negative values - don't try to kill the thread after timeout; INT\_MAX - infinite.

**Return value:**

- 0 - thread closed successfully;
- 1 - time-out;
- 2 - some error.

### mutex\_create

#### Examples

```
new_mutex = mutex_create()
mutex_lock( new_mutex )
mutex_unlock( new_mutex )
mutex_destroy( new_mutex )
```

## mutex\_destroy

## mutex\_lock

## mutex\_trylock

## mutex\_unlock

# Mathematical

acos( x ); acosh( x ); asin( x ); asinh( x ); atan( x ); atanh( x ); ceil( x ); cos( x ); cosh( x ); exp( x );  
exp2( x ); expm1( x ); abs( x ); floor( x ); mod( x, y ); log( x ); log2( x ); log10( x ); pow( base, exp );  
sin( x ); sinh( x ); sqrt( x ); tan( x ); tanh( x );

rand() - get random number from 0 to 32767; rand\_seed( seed ) - set random seed.

# Data processing

## op\_cn

Execute data processing operation. Operands:

1. container C1 (destination);
2. numerical value N.

Operation expression:

```
for each element of C1: C1[ i ] = C1[ i ] OP N,  
Where i - element number; OP - selected operation.
```

**Parameters ( opcode, C1, N ) - for whole container C1**

**Parameters ( opcode, C1, N, x, xsize ) - for 1D region**

**Parameters ( opcode, C1, N, x, y, xsize, ysize ) - for 2D region**

- opcode - operation;
- C1 - destination container;
- N - numerical value;
- x,y,xsize,ysize - region.

## Examples

```
//Add 32 to the whole container img:  
op_cn( OP_ADD, img, 32 )
```

```
//Add 32 to 128..256th elements of the container img:
op_cn( OP_ADD, img, 32, 128, 128 )

//Add 32 to two-dimensional region (8,8,32,32) of elements:
op_cn( OP_ADD, img, 32, 8, 8, 32, 32 )
```

## op\_cc

Execute data processing operation. Operands:

1. container C1 (destination);
2. container C2 (source).

Operation expression:

```
for each element of C1: C1[ i ] = C1[ i ] OP C2[ i ],
Where i - element number; OP - selected operation.
```

**Parameters ( opcode, C1, C2 ) - for whole container C1**

**Parameters ( opcode, C1, C2, dest\_x, src\_x, xsize ) - for 1D region**

**Parameters ( opcode, C1, C2, dest\_x, dest\_y, src\_x, src\_y, xsize, ysize ) - for 2D region**

- opcode - operation;
- C1 - destination container;
- C2 - source container;
- x,y,xsize,ysize - region.

## op\_ccn

Execute data processing operation. Operands:

1. container C1 (destination);
2. container C2 (source);
3. numerical value N.

Operation expression:

```
for each element of C1: C1[ i ] = C1[ i ] OP C2[ i ] OP2 N,
Where i - element number; OP - selected operation; OP2 - additional
operation associated with OP.
```

**Parameters ( opcode, C1, C2, N ) - for whole container C1**

**Parameters ( opcode, C1, C2, N, dest\_x, src\_x, xsize ) - for 1D region**

**Parameters ( opcode, C1, C2, N, dest\_x, dest\_y, src\_x, src\_y, xsize, ysize ) - for 2D region**

- opcode - operation;

- C1 - destination container;
- C2 - source container;
- N - numerical value;
- x,y,xsize,ysize - region.

## generator

Generate a signal.

### Parameters ( opcode, pixi, phase, amplitude, delta\_x, delta\_y, x, y, xsize, ysize )

- opcode - operation;
- pixi - container;
- phase;
- amplitude;
- delta\_x;
- delta\_y;
- x,y,xsize,ysize - region.

### Examples

```
//Generate a sine wave to the whole container img:
generator( OP_SIN, img, 0, 1, 0.1, 0.1 )

//Generate a rough sine wave to the whole container img:
generator( OP_SIN8, img, 0, 1, 0.1, 0.1 )

//Generate a sine wave to 8...128 elements of the container img:
generator( OP_SIN, img, 0, 1, 0.1, 0.1, 8, 120 )

//Generate a sine wave to region (8,8,32,32) of the container img:
generator( OP_SIN, img, 0, 1, 0.1, 0.1, 8, 8, 32, 32 )
```

## wavetable\_generator

Very fast multichannel sampler, where the sample (table) is always looped and has a fixed size (32768).

### Parameters ( dest, dest\_offset, dest\_length, table, amp, amp\_delta, pos, pos\_delta, gen\_offset, gen\_step, gen\_count )

- dest - audio destination (INT16 or FLOAT32 container);
- dest\_offset;
- dest\_length;
- table - table with waveform (supported formats: 32768 x INT16, 32768 x FLOAT32);
- amp - INT32 array of amplitudes (fixed point 16.16); after each sample, these values will increase by the values from amp\_delta;
- amp\_delta - INT32 array of amplitude delta values (fixed point 16.16);
- pos - INT32 array of wavetable positions (fixed point 16.16); after each sample, these values will

increase by the values from `pos_delta`;

- `pos_delta` - INT32 array of wavetable position delta values (fixed point 16.16);
- `gen_offset` - number of the first generator;
- `gen_step` - play every `gen_step` generator;
- `gen_count` - total number of generators to play.

**Return value:** 0 if success.

## sampler

### Parameters ( `sample_info` )

### Examples

```
sample_data = new( 256, 1, INT16 ) //16bit sample
sample_info = new( SMP_INFO_SIZE, 1, INT32 )
clean( sample_info )
sample_info[ SMP_DEST ] = buffer //Destination container
sample_info[ SMP_DEST_OFF ] = 0 //Destination offset
sample_info[ SMP_DEST_LEN ] = 256 //Destination length
sample_info[ SMP_SRC ] = sample_data
sample_info[ SMP_SRC_OFF_H ] = 0 //Sample offset (left part of fixed point
value)
sample_info[ SMP_SRC_OFF_L ] = 0 //Sample offset (right part of fixed point
value from 0 to 65535)
sample_info[ SMP_SRC_SIZE ] = 0 //Sample size (0 - whole sample)
sample_info[ SMP_LOOP ] = 0 //Loop start
sample_info[ SMP_LOOP_LEN ] = 128 //Loop length (or 0, if no loop)
sample_info[ SMP_VOL1 ] = 0 //Start volume
sample_info[ SMP_VOL2 ] = 32768 //End volume (32768 = 1.0)
sample_info[ SMP_DELTA ] = ( 1 << 16 ) //Delta; fixed point (real_value *
65536)
sample_info[ SMP_FLAGS ] = SMP_FLAG_INTERP4 | SMP_FLAG_PINGPONG //Cubic
spline interpolation and ping-pong loop
sampler( sample_info ) //Go!
```

## envelope2p

Apply gain and DC-offset two-points envelope to selected container area. Without clipping.

### Parameters ( `data_cont`, `v1`, `v2`, `offset`, `size`, `dc_offset1`, `dc_offset2` )

- `data_cont` - container with data (audio waveform, for example);
- `v1` - initial gain (0 - no signal; 32768 - original unchanged amplitude);
- `v2` - final gain (0 - no signal; 32768 - original unchanged amplitude);
- `offset` - offset in the `data_cont`; optional; default - 0;
- `size` - size of area for envelope; optional; default - whole container;
- `dc_offset1` - initial DC offset; optional; default - 0;
- `dc_offset2` - final DC offset; optional; default - 0.

## gradient

**Parameters ( container, val1, val2, val3, val4, x, y, xsize, ysize, x\_step, y\_step )**

## fft

Perform a fast fourier transform.

**Parameters ( inverse, im, re, size )**

## new\_filter

Create a new filter with the following function:

```
output[ n ] = ( a[ 0 ] * input[ n ] + a[ 1 ] * input[ n - 1 ] + ... + a[
a_count - 1 ] * input[ n - a_count - 1 ]
                + b[ 0 ] * output[ n - 1 ] + ... + b[
b_count - 1 ] * output[ n - b_count - 1 ] ) >> rshift;
```

**Parameters ( flags\_for\_future\_use )**

**Return value:** ID of the new container with the filter.

## remove\_filter

**Parameters ( filter )**

## reset\_filter

**Parameters ( filter )**

## init\_filter

**Parameters ( filter, a, b, rshift, flags )**

- filter;
- a - container with the feedforward filter coefficients;
- b - container with the feedback filter coefficients; optional; can be -1 (for FIR filters);
- rshift - bitwise right shift for fixed point computations; optional;
- flags - for future use; optional.

**Return value:** 0 if success.



## apply\_filter

### Parameters ( filter, output, input, flags, offset, size )

- filter;
- output - output container;
- input - input container;
- flags - for future use; optional;
- offset - output and input offset; optional;
- size - size of the processed block; optional.

**Return value:** 0 if success.

## replace\_values

For each element of dest container:  $dest[i] = values[(unsigned)src[i]]$ . The dest container must have the same type as the values.

### Parameters ( dest, src, values, dest\_offset, src\_offset, size )

- dest - destination;
- src - source;
- values - substitution values;
- dest\_offset - offset in the destination container; optional;
- src\_offset - offset in the source container; optional;
- size - number of elements to replace; optional.

### Examples

```
//Convert 8-bit image with palette to the screen pixel format:  
replace_values( scr, img8, palette )
```

## copy\_and\_resize

### Parameters ( dest, src, flags, dest\_x, dest\_y, dest\_rect\_xsize, dest\_rect\_ysize, src\_x, src\_y, src\_rect\_xsize, src\_rect\_ysize )

- dest - destination container;
- src - source container;
- flags - flags RESIZE\_xxx; optional; default = RESIZE\_INTERP1;
- dest\_x, dest\_y, dest\_rect\_xsize, dest\_rect\_ysize, src\_x, src\_y, src\_rect\_xsize, src\_rect\_ysize - optional parameters.

## Dialogs

## file\_dialog

Open file dialog.

### Parameters ( dialog\_name, mask, id, default\_name )

- dialog\_name;
- mask - file type mask (examples: "gif/jpg" for .gif and .jpg files; "" for all files);
- id - name of the file for saving the current dialog state;
- default\_name - default file name; optional.

**Return value:** ID of the container with selected file name; or -1 if file not selected; (container must be removed manually).

## prefs\_dialog

Open window with global Pixilang preferences.

## Network

### open\_url

Open web browser window with selected URL.

### Parameters ( url\_string )

## Native code

### dlopen

Open dynamic library (for example - .DLL file in Windows, or .SO file in Linux).

### Parameters ( lib\_file\_name )

**Return value:** library ID or -1 if error occurred.

### Examples

```
//For example we have some C function int show_info( int x, int y, void*
data )
//in mylib.dll library.
//Let's call it from Pixilang:
dl = dlopen( "mylib.dll" ) //Open the library
if dl >= 0
{
```

```
f = dlsym( dl, "show_info", "iip" ) //Get the function show_info() from
dynamic library
// "iip" - int int pointer
if f >= 0
{
    retval = dlcall( dl, f, 1, 2, "blahblah" ) //Call the function
show_info() with parameters 1, 2, "blahblah"
}
dlclose( dl ) //Close the library
}
```

## **dlclose**

Close dynamic library.

### **Parameters ( lib\_id )**

## **dlsym**

Get symbol (function or variable) from dynamic library.

### **Parameters ( lib\_id, symbol\_name, format, calling\_convention )**

- lib\_id;
- symbol\_name - function/variable name;
- format - function format; optional; text string with the following structure: "R(P)", where the R - return value type (one ascii character), P - parameter types (multiple ascii characters or empty); below is a list of characters that are used for the construction of this string:
  - v - void;
  - c - signed int8;
  - C - unsigned int8
  - s - signed int16;
  - S - unsigned int16;
  - i - signed int32;
  - I - unsigned int32;
  - l - signed int64;
  - L - unsigned int64;
  - f - float32;
  - d - double64;
  - p - pointer;
- calling\_convention - one of the CCONV\_xxx constants; optional; default - CCONV\_DEFAULT.

**Return value:** symbol ID or -1 if error occurred.

## **dlcall**

Call the function from dynamic library.

**Parameters ( lib\_id, symbol\_id, optional\_function\_parameters )**

## System functions

### system

Issue a OS command.

**Parameters ( command )**

**Return value:** termination status of the command.

#### Examples

```
//Remove some file:  
system( "rm /tmp/data.txt" )
```

### argc

Returns the number of arguments.

### argv

Returns the container with selected argument.

**Parameters ( n )**

#### Examples

```
if argc() >= 4  
{  
    a = argv( 3 )  
    remove( a )  
}
```

### exit

Quit from Pixilang.

**Parameters ( exit\_code )**

#### Examples

```
exit( 4 ) //Exit with code 4
```

From:

<https://www.warmplace.ru/wiki/> - **WarmPlace Wiki**

Permanent link:

[https://www.warmplace.ru/wiki/doku.php?id=pixilang:manual\\_en](https://www.warmplace.ru/wiki/doku.php?id=pixilang:manual_en)

Last update: **2019/09/09 10:08**

