

# Introduction

## Introduction

pico-Céu is a tiny programming environment for visual and interactive applications such as video games. It is composed of the programming language Céu and minimalist libraries for input, graphics, network, and sound.

## Resources

### Resources

Resource is any external file used by a pico-Céu application, such as images, fonts, and audio files. Every resource should be located in a `res` folder, in the root of the application, and can be used in `GRAPHICS_DRAW_BMP`, `GRAPHICS_SET_FONT`, or `SOUND_PLAY`.

## Graphics

### Graphics

Provides graphics operations, such as for drawing pixels and images on the screen.

TODO: `axis`

### Configuration

#### `GRAPHICS_SET_ANCHOR`

Changes the drawing anchor of all subsequent drawing operations `GRAPHICS_DRAW_BMP`, `GRAPHICS_DRAW_RECT`, and `GRAPHICS_DRAW_TEXT`.

```
output (HAnchor,VAnchor) GRAPHICS_SET_ANCHOR;
```

- Parameters:
  - `HAnchor`: new horizontal anchor
  - `VAnchor`: new vertical anchor

The anchor specifies the part of the shape to appear at the pixel position of the drawing operation.

The possible values for `HAnchor` are `HANCHOR_LEFT`, `HANCHOR_CENTER`, and `HANCHOR_RIGHT`. The initial value is `HANCHOR_CENTER`.

The possible values for `HVnchor` are `VANCHOR_TOP`, `VANCHOR_CENTER`, and `VANCHOR_BOTTOM`. The initial value is `VANCHOR_CENTER`.

### **GRAPHICS\_SET\_BMP\_FRAME**

Changes the drawing frame of all subsequent `GRAPHICS_DRAW_BMP` operations.

output (int?,int?) `GRAPHICS_SET_BMP_FRAME`;

- Parameters:
  - `int?`: new frame index to show (default: 0)
  - `int?`: new number of frames in the image (default: 1)

The initial frame index is 0 and number of frames is 1.

### **GRAPHICS\_SET\_BMP\_SIZE**

Changes the drawing size of all subsequent `GRAPHICS_DRAW_BMP` operations.

output (int?,int?) `GRAPHICS_SET_BMP_SIZE`;

- Parameters:
  - `int?`: new width (default: proportional to new height)
  - `int?`: new height (default: proportional to new width)

If both width and height are set to default, the new size is the original image size.

The initial size is the original image size.

### **GRAPHICS\_SET\_COLOR\_NAME**

Changes the color of all subsequent drawing operations.

output (Color) `GRAPHICS_SET_COLOR_NAME`

- Parameters:
  - `Color`: new color name

The color names are based on the *HTML Web Colors*:

[https://en.wikipedia.org/wiki/Web\\_colors#HTML\\_color\\_names](https://en.wikipedia.org/wiki/Web_colors#HTML_color_names)

The possible values are `COLOR_WHITE`, `COLOR_SILVER`, `COLOR_GRAY`, `COLOR_BLACK`, `COLOR_RED`, `COLOR_MAROON`, `COLOR_YELLOW`, `COLOR_OLIVE`, `COLOR_LIME`, `COLOR_GREEN`, `COLOR_AQUA`, `COLOR_TEAL`, `COLOR_BLUE`, `COLOR_NAVY`, `COLOR_FUCHSIA`, `COLOR_PURPLE`.

The initial color is white.

## **GRAPHICS\_\_SET\_\_COLOR\_\_RGB**

Changes the color in RGB of all subsequent drawing operations.

output (integer, integer, integer) GRAPHICS\_SET\_COLOR\_RGB

- Parameters:
  - **integer**: new red component
  - **integer**: new green component
  - **integer**: new blue component

The initial color is white.

## **GRAPHICS\_\_SET\_\_FONT**

Changes the font for drawing and writing text.

output (text, integer) GRAPHICS\_SET\_FONT

- Parameters:
  - **text**: path for the **.ttf** font filename
  - **integer**: height of the new font in pixels

## **GRAPHICS\_\_SET\_\_SCALE**

Changes the drawing scale of all subsequent drawing operations **GRAPHICS\_DRAW\_BMP**, **GRAPHICS\_DRAW\_RECT**, and **GRAPHICS\_DRAW\_TEXT**.

output (real, real) GRAPHICS\_SET\_SCALE;

- Parameters:
  - **real**: new horizontal scale
  - **real**: new vertical scale

The initial scale is 1.0 x 1.0.

## **GRAPHICS\_\_SET\_\_WRITE\_\_CURSOR**

Changes the cursor starting position for writing text with **GRAPHICS\_WRITE** and **GRAPHICS\_WRITELN**.

output (integer, integer) GRAPHICS\_SET\_WRITE\_CURSOR

- Parameters:
  - **integer**: new position in the x-axis
  - **integer**: new position in the y-axis

The initial starting position is the top-left of the screen.

The current position is reset on every **WINDOW\_CLEAR** operation.

## Drawing

### GRAPHICS\_DRAW\_BMP

Draws a bitmap image on the screen.

output (integer, integer, text) GRAPHICS\_DRAW\_BMP

- Parameters:
  - integer: position in the x-axis
  - integer: position in the y-axis
  - text: path for the .bmp image filename

### GRAPHICS\_DRAW\_PIXEL

Draws a pixel on the screen.

output (integer, integer) GRAPHICS\_DRAW\_PIXEL

- Parameters:
  - integer: position in the x-axis
  - integer: position in the y-axis

The drawing color is specified with GRAPHICS\_SET\_COLOR\_NAME or GRAPHICS\_SET\_COLOR\_RGB.

### GRAPHICS\_DRAW\_LINE

Draws a line on the screen.

output (integer, integer, integer, integer) GRAPHICS\_DRAW\_LINE;

- Parameters:
  - integer: start position in the x-axis
  - integer: start position in the y-axis
  - integer: end position in the x-axis
  - integer: end position in the y-axis

The drawing color is specified with GRAPHICS\_SET\_COLOR\_NAME or GRAPHICS\_SET\_COLOR\_RGB.

### GRAPHICS\_DRAW\_RECT

Draws a rectangle on the screen.

output (integer, integer, integer, integer) GRAPHICS\_DRAW\_RECT

- Parameters:
  - integer: position in the x-axis
  - integer: position in the y-axis
  - integer: rectangle width
  - integer: rectangle height

The drawing color is specified with `GRAPHICS_SET_COLOR_NAME` or `GRAPHICS_SET_COLOR_RGB`.

## **GRAPHICS\_DRAW\_TEXT**

Draws a text on the screen.

output (int,int,text) `GRAPHICS_DRAW_TEXT`;

- Parameters:
  - **integer**: position in the x-axis
  - **integer**: position in the y-axis
  - **text**: text to draw

The drawing font is specified with `GRAPHICS_SET_FONT`. The drawing color is specified with `GRAPHICS_SET_COLOR_NAME` or `GRAPHICS_SET_COLOR_RGB`.

## **Writing**

### **GRAPHICS\_WRITE**

Writes a text on the screen.

output (text) `GRAPHICS_WRITE`;

- Parameters:
  - **text**: text to draw

The drawing position is first specified with `GRAPHICS_SET_WRITE_CURSOR`. The cursor advances automatically for the position after the text. The drawing font is specified with `GRAPHICS_SET_FONT`. The drawing color is specified with `GRAPHICS_SET_COLOR_NAME` or `GRAPHICS_SET_COLOR_RGB`.

### **GRAPHICS\_WRITELN**

Writes a line of text on the screen.

output (text) `GRAPHICS_WRITELN`;

The drawing position is first specified with `GRAPHICS_SET_WRITE_CURSOR`. The cursor advances automatically for the next line after the text, at the same initial position. The drawing font is specified with `GRAPHICS_SET_FONT`. The drawing color is specified with `GRAPHICS_SET_COLOR_NAME` or `GRAPHICS_SET_COLOR_RGB`.

## **Other**

### **GRAPHICS\_SCREENSHOT**

Takes a screen shot.

output (text) GRAPHICS\_SCREENSHOT

- Parameters:
  - **text**: path for the **.bmp** image filename to generate

## Input Devices

### Input Devices

Provides input handling, such as for keyboard and mouse.

#### Keyboard

##### KEY\_PRESS

input (integer) KEY\_PRESS

- Occurrences:
  - whenever a keyboard key is pressed
- Payload:
  - **integer**: numeric key code

TODO: key codes

##### KEY\_UNPRESS

input (integer) KEY\_UNPRESS

- Occurrences:
  - whenever a keyboard key is released
- Payload:
  - **integer**: numeric key code

TODO: key codes

#### Mouse

##### MOUSE\_CLICK

input (integer,integer,integer) MOUSE\_CLICK

- Occurrences:
  - whenever a mouse button is pressed
- Payload:
  - **integer**: numeric button code (TODO: left, middle, right?)
  - **integer**: current mouse position in the x-axis

- **integer**: current mouse position in the y-axis

## **MOUSE\_UNCLICK**

input (integer,integer,integer) MOUSE\_UNCLICK

- Occurrences:
  - whenever a mouse button is released
- Payload:
  - **integer**: numeric button code (TODO: left, middle, right?)
  - **integer**: current mouse position in the x-axis
  - **integer**: current mouse position in the y-axis

## **MOUSE\_MOVE**

input (integer,integer) MOUSE\_MOVE

- Occurrences:
  - whenever the mouse moves
- Payload:
  - **integer**: current mouse position in the x-axis
  - **integer**: current mouse position in the y-axis

# **Sound**

## **Sound**

Provides sound playback.

## **Configuration**

### **SOUND\_SET\_VOLUME**

Changes the volume of all subsequent sound playbacks.

output (integer) SOUND\_SET\_VOUME

- Parameters:
  - **integer**: new sound volume in percentage (from 0 to 100)

## **Playback**

### **SOUND\_PLAY**

Plays a sound file.

output (text) SOUND\_PLAY

- Parameters:
  - **text**: path for the sound filename

The playback volume is specified with SOUND\_SET\_VOLUME.

## Network

### Network

Provides unreliable broadcast communication between peers.

#### Send

##### NET\_\_SEND

Broadcasts a message to all peers.

output (integer,byte&&) NET\_SEND;

- Parameters:
  - **integer**: number of bytes to transmit
  - **byte&&**: stream of bytes

#### Receive

##### NET\_\_RECEIVE

Receives all messages from all peers, including itself.

input (integer,byte&&) NET\_RECEIVE;

- Occurrences:
  - on every received message
- Payload:
  - **integer**: number of received bytes
  - **byte&&**: stream of bytes

## Usart

### Usart

A pico-Céu library to send and receive data using USART (Universal Synchronous and Asynchronous Receiver-Transmitter). Windows-only for now.



## Includes

```
##include "usart.ceu"
```

## Initiate

```
code/await Usart (var int portNumber) -> NEVER do
```

- Parameters:
  - `var int`: Serial port number to use.
- Example: `“‘ceu spawn Usart(3);`

`““` Specify that we’ll use the COM3 port.

## Send

### Usart\_TX

Send a byte vector via serial.

```
code/await Usart_TX (var&[] byte str) -> none do
```

- Parameters:
  - `var&[] byte`: the byte vector to send.
- Example: `“‘ceu spawn Usart(3);`

`var[5] byte str; call String_Append_STR(&str, “send”);`

`await Usart_TX(&str);`

Create a string and send it via serial using `Usart_TX`. Check

`[String_Append_STR]`(#[https://ceu-lang.github.io/ceu/out/manual/v0.30/string/#string\\_append\\_str](https://ceu-lang.github.io/ceu/out/manual/v0.30/string/#string_append_str)) to learn more string manipulation in Céu.

## Receive

### Usart\_RX

```
code/await Usart_RX (var&[] byte str, var int nbChar) -> none do
```

- Payload:
  - `var&[] byte`: byte vector to store the received data
  - `var int`: number of bytes to read
- Example: `“‘ceu spawn Usart(3);`

`var[5] byte buffer; await Usart_RX(&buffer, 5); String_Print(&buffer);`

Receive a string of size 5 from serial port, counting the 0““.

# Frame Management

## Frame Management

Manages the game frames, such as for updating animations and redrawing the screen.

### Configuration

#### FRAMES\_SET

Enables or disables the generation of periodic `FRAMES_UPDATE` and `FRAMES_REDRAW` inputs to the application.

output (yes/no) `FRAMES_SET`

- Parameters:
  - `yes/no`: new state
    - \* `yes`: enables the generation of frames
    - \* `no`: disables the generation of frames

### Inputs

#### FRAMES\_UPDATE

input (integer) `FRAMES_UPDATE`

- Occurrences:
  - on every frame, before `FRAMES_REDRAW`
- Payload:
  - `integer`: the number of milliseconds elapsed since the previous frame

#### FRAMES\_REDRAW

input (none) `FRAMES_REDRAW`

- Occurrences:
  - on every frame, after `FRAMES_UPDATE`
- Payload:
  - `none`: no payload

Before the input occurs, the screen is automatically cleared with `WINDOW_CLEAR`.

# Window Management

## Window Management

Manages the application window.

### Configuration

#### WINDOW\_SET\_CLEAR\_COLOR\_NAME

Changes the background color of WINDOW\_CLEAR.

output (Color) WINDOW\_SET\_CLEAR\_COLOR\_NAME

- Parameters:
  - **Color**: new color name

The color names are based on the *HTML Web Colors*:

[https://en.wikipedia.org/wiki/Web\\_colors#HTML\\_color\\_names](https://en.wikipedia.org/wiki/Web_colors#HTML_color_names)

The possible values are COLOR\_WHITE, COLOR\_SILVER, COLOR\_GRAY, COLOR\_BLACK, COLOR\_RED, COLOR\_MAROON, COLOR\_YELLOW, COLOR\_OLIVE, COLOR\_LIME, COLOR\_GREEN, COLOR\_AQUA, COLOR\_TEAL, COLOR\_BLUE, COLOR\_NAVY, COLOR\_FUCHSIA, COLOR\_PURPLE.

The default color is black.

#### WINDOW\_SET\_CLEAR\_COLOR\_RGB

Changes the background color of WINDOW\_CLEAR in RGB.

output (integer, integer, integer) WINDOW\_SET\_CLEAR\_COLOR\_RGB

- Parameters:
  - **integer**: new red component
  - **integer**: new green component
  - **integer**: new blue component

The default color is black.

#### WINDOW\_SET\_GRID

Enables or disables a visual grid delimiting the screen pixels.

output (yes/no) WINDOW\_SET\_GRID

- Parameters:
  - **yes/no**: new state
    - \* **yes**: enables the grid

\* **no**: disables the grid

The ratio between the real and logical dimensions set with `WINDOW_SET_SIZE` must be greater than one.

The window is automatically cleared with `WINDOW_CLEAR`.

## **WINDOW\_SET\_SIZE**

Changes the real and logical sizes of the window.

output (integer, integer, integer, integer) `WINDOW_SET_SIZE`

- Parameters:
  - **integer**: new real width
  - **integer**: new real height
  - **integer**: new logical width
  - **integer**: new logical height

The window is automatically cleared with `WINDOW_CLEAR`.

The arithmetic division between the real and logical dimensions must be exact.

## **WINDOW\_SET\_TITLE**

Changes the title of the window.

output (text) `WINDOW_SET_TITLE`

- Parameters:
  - **text**: new window title

## **Clear**

## **WINDOW\_CLEAR**

Clears the window screen.

output (none) `WINDOW_CLEAR`

- Parameters:
  - **none**: no parameters

The clear color is specified with `WINDOW_SET_CLEAR_COLOR_NAME` or `WINDOW_SET_CLEAR_COLOR_RGB`.

The default color is black.

## License

## License

pico-Céu is distributed under the MIT license reproduced below:

Copyright (C) 2012-2017 Francisco Sant'Anna

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.