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`.

Data Types

Data Types

pico-Céu already provides some data types

Color

```
data Color with
    var integer r;
    var integer g;
    var integer b;
end
```

- Parameters:
  - `integer`: red component
  - `integer`: green component
  - `integer`: blue component

Example:

```
var Color color = val Color(255,1,1);
emit GRAPHICS_SET_COLOR_NAME(color);
emit GRAPHICS_DRAW_PIXEL(0,0);
emit GRAPHICS_SET_COLOR_RGB(color.r, color.g, color.b);
```

```
emit GRAPHICS_DRAW_PIXEL(1,1);
```

Point

```
data Point with
    var integer x;
    var integer y;
end
```

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

Example:

```
var Point pt = val Point(0,0);
emit GRAPHICS_DRAW_PIXEL(pt.x, pt.y);
```

Rect

```
data Rect with
    var integer x;
    var integer y;
    var integer w;
    var integer h;
end
```

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

Example:

```
var Rect rect = val Rect(0,0,4,5);
emit GRAPHICS_DRAW_RECT(rect.x, rect.y, rect.w, rect.h);
```

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 `VAnchor` 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

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`, `GRAPHICS_DRAW_TEXT`, `GRAPHICS_DRAW_INT`, and `GRAPHICS_DRAW_REAL`.

`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`.

GRAPHICS_DRAW_INT

Similar to `GRAPHICS_DRAW_TEXT`, but draws a integer on the screen.

```
output (int,int,int) GRAPHICS_DRAW_INT;
```

- Parameters:
  - `integer`: position in the `x-axis`
  - `integer`: position in the `y-axis`
  - `int`: int to draw

GRAPHICS_DRAW_REAL

Similar to `GRAPHICS_DRAW_REAL`, but draws a real on the screen.

```
output (int,int,real) GRAPHICS_DRAW_REAL;
```

- Parameters:
  - `integer`: position in the `x-axis`
  - `integer`: position in the `y-axis`
  - `real`: real to draw

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

Examples:

```
var int c = await KEY_PRESS;

_printf("%c\n", c);
```

```
var int c = await KEY_PRESS until c==KEY_a;

_printf("%c\n", c);
```

`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
    - MOUSE_LEFT
    - MOUSE_MIDDLE
    - MOUSE_RIGHT
    - MOUSE_X1
    - MOUSE_X2
  - `integer`: current mouse position in the `x-axis`
  - `integer`: current mouse position in the `y-axis`

Example:

```
var int c;
var int x;
var int y;

(c,x,y) = await MOUSE_CLICK until c==MOUSE_LEFT;

_printf("(%d,%d)\n", x,y);
```

MOUSE_UNCLICK

`input (integer,integer,integer) MOUSE_UNCLICK`

- Occurrences:
  - whenever a mouse button is released
- Payload:
  - `integer`: numeric button code (same as `MOUSE_CLICK`)
  - `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
```

- Parameters:
  - `var int`: Serial port number to use.
- Example:

```
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
```

- Parameters:
  - `var&[] byte`: the byte vector to send.
- Example:

```
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` to learn more string manipulation in Céu.

Receive

Usart_RX

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

- Payload:
  - `var&[] byte`: byte vector to store the received data
  - `var int`: number of bytes to read
- Example:

```
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

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 then 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.