
GameboyCore Python Documentation

Release 0.6.0

Natesh Narain

Aug 23, 2019

Contents

1	Build and Installation	3
1.1	The easy way	3
1.2	From source	3
2	Documentation for GameboyCore Python	5
3	Examples	7
3.1	Simple PyGame Example	7

GameboyCore Python is the Python binding for the GameboyCore library, enabling an easily embeddable Gameboy emulator that can interacted with via Python scripts!

1.1 The easy way

Windows:

```
pip install gameboycore
```

Linux:

Install from source

Note: Install binary packages on Linux is broken. Install from source instead.

1.2 From source

```
git clone https://github.com/nnarain/gameboycore-python
```

```
cd gameboycore-python
```

```
pip install -r requirements.txt
```

```
python setup.py install
```


CHAPTER 2

Documentation for GameboyCore Python

3.1 Simple PyGame Example

Note: This is not a very optimized example.

```
import gameboycore
import pygame
import sys

WINDOW_SIZE = (640, 480)

class Example(object):
    def __init__(self, dim):
        pygame.init()
        pygame.display.set_caption('GameboyCore Python Example')

        self.dim = dim
        self.screen = pygame.display.set_mode(WINDOW_SIZE)
        # 160x144 is the Gameboy window size
        self.lcd = pygame.Surface((160, 144))
        self.done = False

        self.core = gameboycore.GameboyCore()

        self.action_map = {
            pygame.KEYDOWN: gameboycore.KeyAction.ACTION_PRESS,
            pygame.KEYUP: gameboycore.KeyAction.ACTION_RELEASE
        }
        self.key_map = {
            pygame.K_w: gameboycore.JoypadKey.KEY_UP,
            pygame.K_a: gameboycore.JoypadKey.KEY_LEFT,
            pygame.K_d: gameboycore.JoypadKey.KEY_RIGHT,
            pygame.K_s: gameboycore.JoypadKey.KEY_DOWN,
            pygame.K_j: gameboycore.JoypadKey.KEY_A,
```

(continues on next page)

(continued from previous page)

```

        pygame.K_k:gameboycore.JoypadKey.KEY_B,
        pygame.K_RETURN:gameboycore.JoypadKey.KEY_START,
        pygame.K_LSHIFT:gameboycore.JoypadKey.KEY_SELECT
    }

    def run(self, filename):
        self.core.open(filename)
        self.core.register_scanline_callback(self.scanline_callback)

        while not self.done:
            self.core.emulate_frame()

            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    self.done = True
                elif event.type == pygame.KEYDOWN or event.type == pygame.KEYUP:
                    self.processInput(event.key, event.type)

            pygame.display.flip()

    def processInput(self, key, action):
        gbaction = self.action_map[action]
        gbkey = self.key_map[key]

        self.core.input(gbkey, gbaction)

    def scanline_callback(self, scanline, line):
        for i in range(len(scanline)):
            pixel = scanline[i]
            self.lcd.set_at((i,line), (pixel.r,pixel.g,pixel.b))
        self.updateScreen()

    def updateScreen(self):
        pygame.transform.scale(self.lcd, self.dim, self.screen)

def main():
    ex = Example(WINDOW_SIZE)
    ex.run(sys.argv[1])

if __name__ == '__main__':
    main()

```