clipbit

Release 0.1

David Fraser

Sep 27, 2017

Contents

CHAPTER 1

Introduction to ClipBit

This is a book of programs that you can type into a computer. They do some quite cool and fun things; you can also change them to do other interesting things or make your own programs!

Typing in Programs

When you are typing these programs, be careful to write them exactly as they are written. This includes:

- whether you have a small letter `b` or a capital `B`
- the punctuation marks like `+ - * / % < > & | ^ ~ = ! ( ) [ ] { } @ , : . ` ; ' " # \`
- the number of spaces at the start of each line - make sure your line startings line up just like they do in the printed version!

So if you have trouble running your programs, check that you have typed them exactly correctly.

CHAPTER 2

ClipBit Programs

Secret Codes

You can use this program to send secret messages to a friend who also has it, and they can use it to decode the messages so they can understand them.

To get this to work, enter a message and then a number between `0` and `7905853580625` You will then receive a secret code for that message! To decode the message, you need to put in the secret code you got out, with the *same* number as you put in. If you put in a different number, you will get a different answer!

```
ask = raw_input


def make_map(i):
    d = {}
    letters = list("abcdefghijklmnopqrstuvwxyz")
    for p in range(25,0,-2):
        k = letters.pop(0)
        i, r = divmod(i, p)
        v = d[k] = letters.pop(r)
        d[v] = k
        V, K = v.upper(), k.upper()
        d[K] = V
        d[V] = K
    return d


def encode(message, n):
    d = make_map(n)
    return ''.join(d.get(c, c) for c in message)


def main():
    message = ask('Enter your message: ')
    number = int(ask('Enter a number: '))
```

```
    code = encode(message, number)
    print('Now, you can give your friend the encoded message: %s' % code)
    print('All they need to decode it, is the number you entered above')
    print('And this program :)')


if __name__ == '__main__':
    main()
```

Snake

This is a very simple game where you have to move a snake around on the screen If you go into the edge of the screen, you lose! If you eat one of the other blocks, you get longer...

This program requires pygame.

```
import pygame as p,random
I = random.randint
p.init()
d = p.display
s = d.set_mode([64]*2)
B = []
P = 0
L = 5
e = I(0,70)
Y = p.Rect(0,0,8,8)
Q = s.fill
l = lambda z:(8*(z%9),8*(z/9))
m = lambda n:Q(0,Y.move(l(n)))
S = [-1,1,0,0]
N = 1
t = 9
while 1:
    Q(99)
    m(e)
    P = P+t
    B += [P]
    B = B[-L:]
    map(m,B)
    if P == e:
        L += 1
        e = I(0,70)
    if P in B[:-1] or P%9 == 8 or P&256 or P>71:
        break
    d.flip()
    p.time.wait(99)
    o = N
    for v in p.event.get():
        if v.type == 2:
            N = v.key-273
            if -1 < N-2+o < 3:
                t = S[::-1][N]+S[N]*9
```

Stars

This program shows what it would look like if you were in a spaceship travelling at an incredible speed through a galaxy of stars!

You can also click on a part of the screen and the stars will start coming from there.

This program requires pygame.

```
#!/usr/bin/env python

# License: LGPL

import random, math, pygame
from pygame.locals import *

#constants
WINSIZE = [640, 480]
WINCENTER = [320, 240]
NUMSTARS = 150


def init_star():
    "creates new star values"
    dir = random.randrange(100000)
    velmult = random.random()*.6+.4
    vel = [math.sin(dir) * velmult, math.cos(dir) * velmult]
    return vel, WINCENTER[:]


def initialize_stars():
    "creates a new starfield"
    stars = []
    for x in range(NUMSTARS):
        star = init_star()
        vel, pos = star
        steps = random.randint(0, WINCENTER[0])
        pos[0] = pos[0] + (vel[0] * steps)
        pos[1] = pos[1] + (vel[1] * steps)
        vel[0] = vel[0] * (steps * .09)
        vel[1] = vel[1] * (steps * .09)
        stars.append(star)
    move_stars(stars)
    return stars


def draw_stars(surface, stars, color):
    "used to draw (and clear) the stars"
    for vel, pos in stars:
        pos = (int(pos[0]), int(pos[1]))
        surface.set_at(pos, color)


def move_stars(stars):
    "animate the star values"
    for vel, pos in stars:
        pos[0] = pos[0] + vel[0]
        pos[1] = pos[1] + vel[1]
```

```
        if not 0 <= pos[0] <= WINSIZE[0] or not 0 <= pos[1] <= WINSIZE[1]:
            vel[:], pos[:] = init_star()
        else:
            vel[0] = vel[0] * 1.05
            vel[1] = vel[1] * 1.05


def main():
    "This is the starfield code"
    #create our starfield
    random.seed()
    stars = initialize_stars()
    clock = pygame.time.Clock()
    #initialize and prepare screen
    pygame.init()
    screen = pygame.display.set_mode(WINSIZE)
    pygame.display.set_caption('pygame Stars Example')
    white = 255, 240, 200
    black = 20, 20, 40
    screen.fill(black)

    #main game loop
    done = 0
    while not done:
        draw_stars(screen, stars, black)
        move_stars(stars)
        draw_stars(screen, stars, white)
        pygame.display.update()
        for e in pygame.event.get():
            if e.type == QUIT or (e.type == KEYUP and e.key == K_ESCAPE):
                done = 1
                break
            elif e.type == MOUSEBUTTONDOWN and e.button == 1:
                WINCENTER[:] = list(e.pos)
        clock.tick(50)


# if python says run, then we should run
if __name__ == '__main__':
    main()
```

CHAPTER 3

About ClipBit

Children Learn Intuitive Programming By Interesting Typing

Background

Many programmers today started programming because as children they had access to one of the early home computers. These typically comprised of a keyboard which also contained the CPU, memory etc, and an output that connected to a TV. Significantly, they didn't come loaded with existing programs, but you could get books and magazines with program code listings. By typing these into the computer, you could make it operate. Through the act of typing in these programs, experiencing what they did, and having to correct any mistakes, children were aware of the underlying code and began to form an intuitive understanding of what it meant.

Nowadays, many readily available electronic devices including cell phones (even the basic ones), contain processors more powerful than those computers. However the trend towards ease of use has meant that there is no exposure to code. This means that children are missing out on a potential gold mine of explorative learning.

Goal

The goal of this project is to provide a set of programs for children to type into a computer. Ideally this computer should be made available to them with *only* the ability to type in these programs. The programs should be provided ideally in a book, and they will contain minimal instructions on how to type them in, and what they do, but no traditional instruction in how computer programs work. That can be provided to them as they ask questions or struggle with things that they are trying to create themselves

Finding ClipBit

You can download a printable copy of the latest version from this page, or browse a public version on the web at clipbit.readthedocs.org. To contribute to the project or suggest changes, go to the github page.

Nature of programs

The programs should be:

- As simple to type as possible (i.e. lean towards short variable/function names)
- Fun for children to interact with
- Ideally good examples of logically structured programs
- Possible to change in small, incremental, yet interesting ways

Licensing

CHAPTER 4

License of ClipBit Documentation and Code

Documentation in this file, including code directly included in it (for example, a `.rst` file and the code therein) is under the Creative Commons Attribution-ShareAlike 2.5 South Africa License.

Code included in this project that is *not* inside a documentation file (for example, a `.py` file) is under the MIT license, reproduced below, unless explicitly mentioned in the title. This is usually because the example is sourced from another project, and has been licensed there. Such sources are listed in `sources.txt`.

Creative Commons Attribution-ShareAlike 2.5 South Africa License

To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/2.5/za/deed.en_GB

This is a human-readable summary of the Legal Code (the full licence).

You are free:

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Under the following conditions:

- Attribution — You must give the original author credit.
- Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under a licence identical to this one.

With the understanding that:

- Waiver — Any of the above conditions can be waived if you get permission from the copyright holder.
- Public Domain — Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the licence.

- Other Rights — In no way are any of the following rights affected by the licence:
- Your fair dealing or fair use rights, or other applicable copyright exceptions and limitations;
- The author's moral rights;
- Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.
- Notice — For any reuse or distribution, you must make clear to others the licence terms of this work.

The MIT License (MIT)