

[Python \(/archive/?tag=Python\)](#)

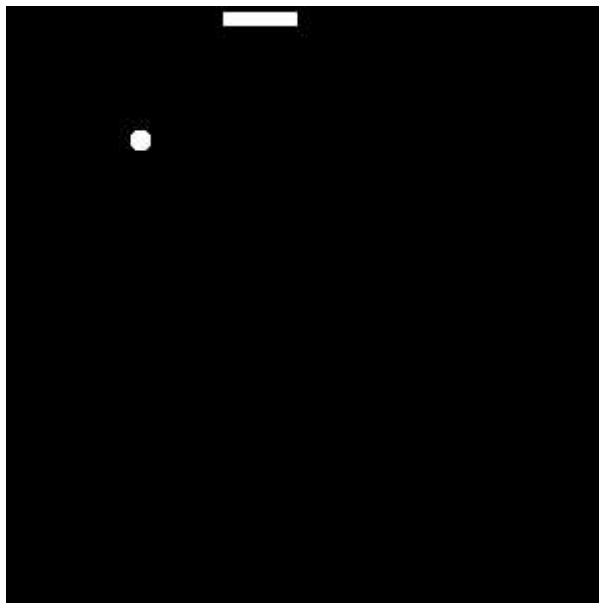
如何用 Python 寫一個貪吃蛇 AI

Hawstein | April 15, 2013

前言

這兩天在網上看到一張讓人漲姿勢的圖片，圖片中展示的是貪吃蛇遊戲，估計大部分人都玩過。但如果僅僅是貪吃蛇遊戲，那麼它就沒有什麼讓人漲姿勢的地方了。問題的關鍵在於，圖片中的貪吃蛇真的很貪吃XD，它把矩形中出現的食物吃了個遍，然後華麗麗地把整個矩形填滿，真心是看得賞心悅目。作為一個CSer，第一個想到的是，這東西是寫程序實現的(因為，一般人幹不出這事。果斷是要讓程序來干的)第二個想到的是，寫程序該如何實現，該用什麼算法？既然開始想了，就開始做。Talk is cheap, show me the code。

開始之前，讓我們再欣賞一下那隻讓人漲姿勢的貪吃蛇吧：(如果下面的動態圖片瀏覽效果不佳的話，可以右鍵保存下來查看)



語言選擇

Life is short, use python! 所以，根本就沒多想，直接上python。

最初版本

先讓你的程序跑起來

首先，我們第一件要做的就是先不要去分析這個問題。你好歹先寫個能運行起來的貪吃蛇遊戲，然後再去想AI部分。這個應該很簡單，c/c++也就百來行代碼(如果我沒記錯的話。不弄複雜界面，直接在控制台下跑)，python就更簡單了，去掉注釋和空行，5、60行代碼就搞定了。而且，最最關鍵的，這個東西網上肯定寫濫了，你沒有必要重復造輪子，去弄一份來按照你的意願改造一下就行了。

簡單版本

我覺得直接寫perfect版本不是什麼好路子。因為perfect版本往往要考慮很多東西，直接上來就寫這個一般是bug百出的。所以，一開始我的目標僅僅是讓程序去控制貪吃蛇運動，讓它去吃食物，僅此而已。現在讓我們來陳述一下最初的問題：

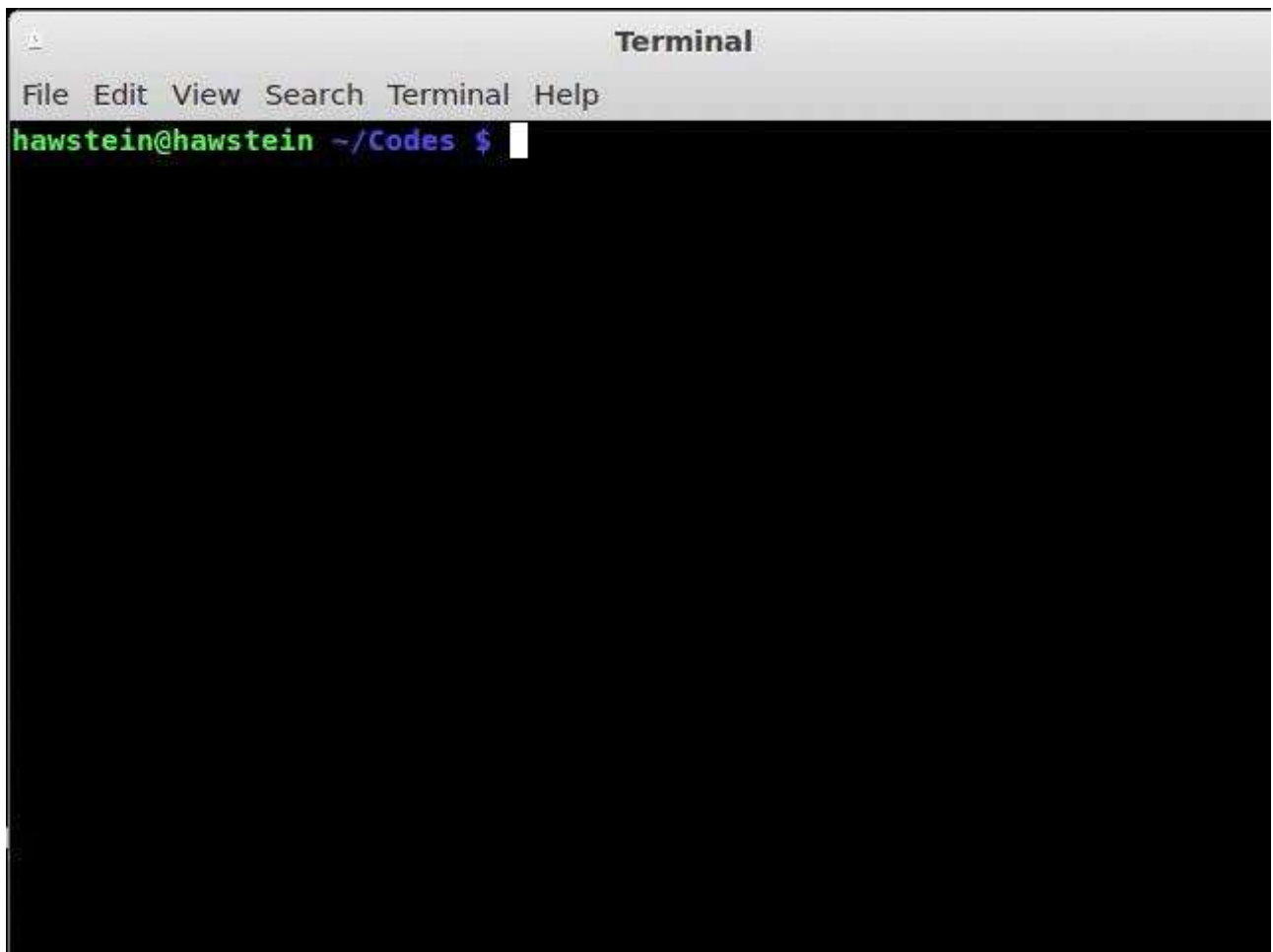
- 1 在一個矩形中，每一時刻有一個食物，貪吃蛇要在不撞到自己的條件下，
- 2 找到一條路(未必要最優)，然後沿著這條路運行，去享用它的美食

我們先不去想蛇會越來越長這個事實，問題基本就是，給你一個起點(蛇頭)和一個終點(食物)，要避開障礙物(蛇身)，從起點找到一條可行路到達終點。我們可以用的方法有：

- BFS
- DFS
- A*

只要有選擇，就先選擇最簡單的方案，我們現在的目標是要讓程序先跑起來，優化是後話。so，從BFS開始。我們最初將蛇頭位置放入隊列，然後只要隊列非空，就將隊頭位置出隊，然後把它四領域內的4個點放入隊列，不斷地循環操作，直到到達食物的位置。這個過程中，我們需要注意幾點：1.訪問過的點不再訪問。2.保存每個點的父結點(即每個位置是從哪個位置走到它的，這樣我們才能把可行路徑找出來)。3.蛇身所在位置和四面牆不可訪問。

通過BFS找到食物後，只需要讓蛇沿著可行路徑運動即可。這個簡單版本寫完後，貪吃蛇就可以很歡快地運行一段時間了。看圖吧：(不流暢的感覺來自錄屏軟件@_@)



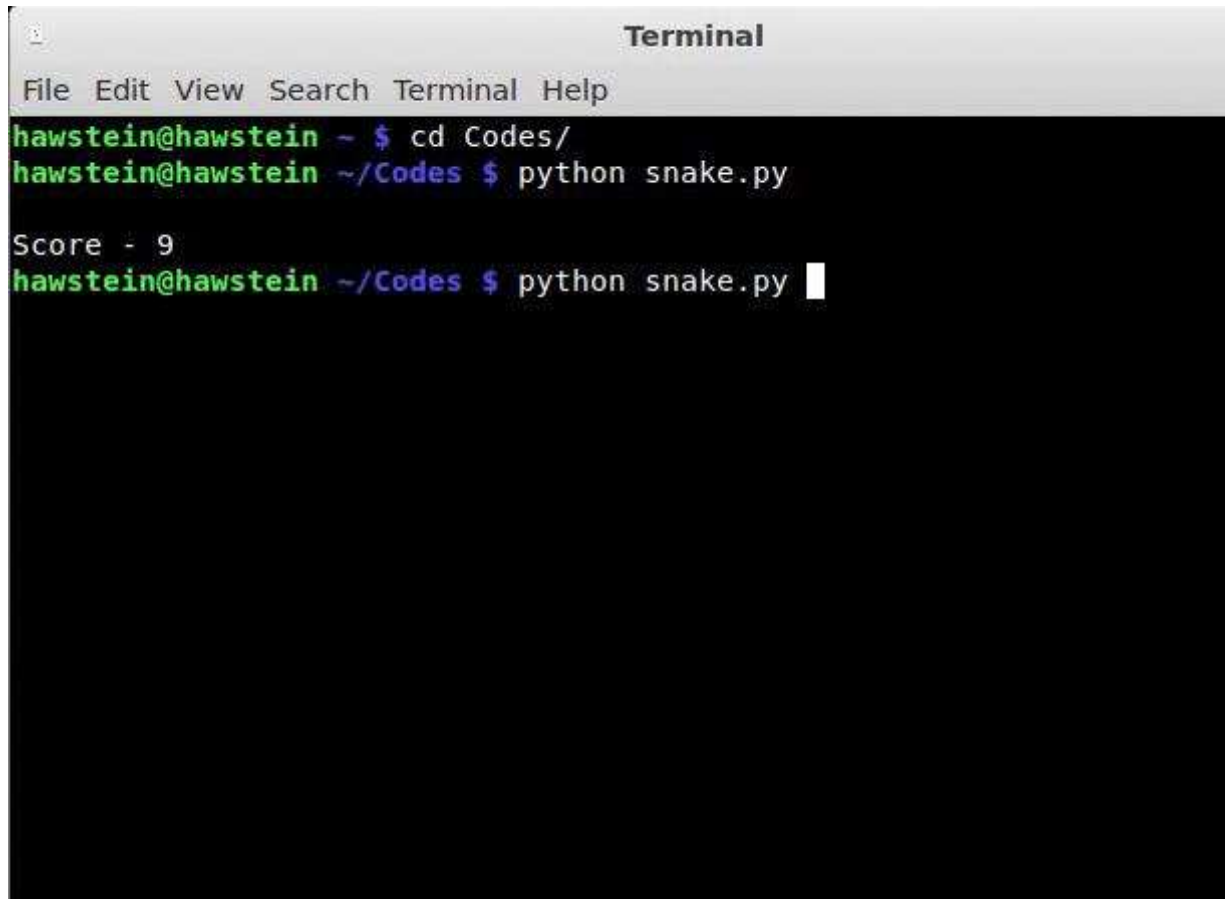
為了盡量保持簡單，我用的是curses模塊，直接在終端進行繪圖。從上面的動態圖片可以看出，每次都單純地使用BFS，最終有一天，貪吃蛇會因為這種不顧後果的短視行為而陷入困境。而且，即使到了那個時候，它也只會BFS一種策略，導致因為當前看不到目標(食物)，認為自己這輩子就這樣了，破罐子破摔，最終停在它人生中的某一個點，不再前進。(我好愛講哲理XD)

BFS+Wander

上一節的簡單版本跑起來後，我們認識到，只教貪吃蛇一種策略是不行的。它這麼笨一條蛇，你不多教它一點，它分分鐘就會掛掉的。所以，我寫了個Wander函數，顧名思義，當貪吃蛇陷入困境後，就別讓它再BFS了，而是讓它隨便四處走走，散散心，思考一下人生什麼的。這個就好比當你困惑迷茫的時候還去工作，效率不佳不說，還可能阻礙你走出困境；相反，這時候你如果放下手中的工作，停下來，出去旅個游什麼的。回來時，說不定就豁然開朗，土地平曠，屋舍儼然了。

Wander函數怎麼寫都行，但是肯定有優劣之分。我寫了兩個版本，一個是在可行的範圍內，朝隨機方向走隨機步。也就是說，蛇每次運動的方向是隨機出來的，總共運動的步數也是隨機的。Wander完之後，再去BFS一下，看能否吃到食物，如果可以那就皆大歡喜了。如果不行，說明思考人生的時間還不夠，再Wander一下。這樣過程不斷地循環進行。可是就像“隨機過程隨機過”一

樣，你“隨機Wander就隨機掛”。會Wander的蛇確實能多走好多步。可是有一天，它就會把自己給隨機到一條死路上了。陷入困境還可以Wander，進入死胡同，那可沒有回滾機制。所以，第二個版本的Wander函數，我就讓貪吃蛇貪到底。在BFS無解後，告訴蛇一個步數step(隨機產生step)，讓它在空白區域以S形運動step步。這回運動方向就不隨機了，而是有組織有紀律地運動。先看圖，然後再說說它的問題：

A screenshot of a terminal window titled "Terminal". The terminal has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The prompt is "hawstein@hawstein ~". The user enters "cd Codes/" and the prompt changes to "hawstein@hawstein ~/Codes". Then the user enters "python snake.py". The output shows "Score - 9" followed by the prompt "hawstein@hawstein ~/Codes \$ python snake.py" with a cursor.

沒錯，最終還是掛掉了。S形運動也是無法讓貪吃蛇避免死亡的命運。貪吃蛇可以靠S形運動多存活一段時間，可是由於它的策略是：

```
1 while 沒有按下ESC鍵：
2     if 蛇與食物間有路徑：
3         走起，吃食物去
4     else：
5         Wander一段時間
```

問題就出在蛇發現它自己和食物間有路徑，就二話不說跑去吃食物了。它沒有考慮到，你這一去把食物給吃了後形成的局勢(蛇身佈局)，完全就可能讓你掛掉。(比如進入了一個自己蛇身圍起來的封閉小空間)

so，為了能讓蛇活得久一些，它還要更高瞻遠矚才行。

高瞻遠矚版本

我們現在已經有了一個比較低端的版本，而且對問題的認識也稍微深入了一些。現在可以進行一些比較慎密和嚴謹的分析了。首先，讓我們羅列一些問題：（像頭腦風暴那樣，想到什麼就寫下來即可）

- 蛇和食物間有路徑直接就去吃，不可取。那該怎麼辦？
- 如果蛇去吃食物後，佈局是安全的，是否就直接去吃？（這樣最優嗎？）
- 怎樣定義佈局是否安全？
- 蛇和食物之間如果沒有路徑，怎麼辦？
- 最短路徑是否最優？（這個明顯不是了）
- 那麼，如果佈局安全的情況下，最短路徑是否最優？
- 除了最短路徑，我們還可以怎麼走？S形？最長？
- 怎麼應對蛇身越來越長這個問題？
- 食物是隨機出現的，有沒可能出現無解的佈局？
- 暴力法(brute force)能否得到最優序列？（讓貪吃蛇盡可能地多吃食物）

只要去想，問題還挺多的。這時讓我們以面向過程的思想，帶著上面的問題，把思路理一理。一開始，蛇很短(初始化長度為1)，它看到了一個食物，使用BFS得到矩形中每個位置到達食物的最短路徑長度。在沒有蛇身阻擋下，就是曼哈頓距離。然後，我要先判斷一下，貪吃蛇這一去是否安全。所以我需要一條虛擬的蛇，它每次負責去探路。如果安全，才讓真正的蛇去跑。當然，虛擬的蛇是不會繪製出來的，它只負責模擬探路。那麼，怎麼定義一個佈局是安全的呢？如果你把文章開頭那張動態圖片中蛇的銷魂走位好好的看一下，會發現即使到最後蛇身已經很長了，它仍然沒事一般地走出了一條路。而且，是跟著蛇尾走的！嗯，這個其實不難解釋，蛇在運動的過程中，消耗蛇身，蛇尾後面總是不斷地出現新的空間。蛇短的時候還無所謂，當蛇一長，就會發現，要想活下來，基本就只能追著蛇尾跑了。在追著蛇尾跑的過程中，再去考慮能否安全地吃到食物。（下圖是某次BFS後，得到的一個佈局，0代表食物，數字代表該位置到達食物的距離，+號代表蛇頭，*號代表蛇身，-號代表蛇尾，#號代表空格，外面的一圈#號代表圍牆）

```

1  # # # # # # #
2  # 0 1 2 3 4 #
3  # 1 2 3 # 5 #
4  # 2 3 4 - 6 #
5  # 3 + * * 7 #
6  # 4 5 6 7 8 #
7  # # # # # # #

```

經過上面的分析，我們可以將佈局是否安全定義為蛇是否可以跟著蛇尾運動，也就是蛇吃完食物後，蛇頭和蛇尾間是否存在路徑，如果存在，我就認為是安全的。

OK，繼續。真蛇派出虛擬蛇去探路後，發現吃完食物後的佈局是安全的。那麼，真蛇就直奔食物了。等等，這樣的策略好嗎？未必。因為蛇每運動一步，佈局就變化一次。佈局一變就意味著可能存在更優解。比如因為蛇尾的消耗，原本需要繞路才能吃到的食物，突然就出現在蛇眼前了。所以，真蛇走一步後，更好的做法是，重新做BFS。然後和上面一樣進行安全判斷，然後再走。

接下來我們來考慮一下，如果蛇和食物之間不存在路徑怎麼辦？上文其實已經提到了做法了，跟著蛇尾走。只要蛇和食物間不存在路徑，蛇就一直跟著蛇尾走。同樣的，由於每走一步佈局就會改變，所以每走一步就重新做BFS得到最新佈局。

好了，問題又來了。如果蛇和食物間不存在路徑且蛇和蛇尾間也不存在路徑，怎麼辦？這個我是沒辦法了，選一步可行的路徑來走就是了。還是一個道理，每次只走一步，更新佈局，然後再判斷蛇和食物間是否有安全路徑；沒有的話，蛇頭和蛇尾間是否存在路徑；還沒有，再挑一步可行的來走。

上面列的好幾個問題裡都涉及到蛇的行走策略，一般而言，我們會讓蛇每次都走最短路徑。這是針對蛇去吃食物的時候，可是蛇在追自己的尾巴的時候就不能這麼考慮了。我們希望的是蛇頭在追蛇尾的過程中，盡可能地慢。這樣蛇頭和蛇尾間才能騰出更多的空間，空間多才有得發展。所以蛇的行走策略主要分為兩種：

- 1 1. 目標是食物時·走最短路徑
- 2 2. 目標是蛇尾時·走最長路徑

那第三種情況呢？與食物和蛇尾都沒路徑存在的情況下，這個時候本來就只是挑一步可行的步子來走，最短最長關係都不大了。至於人為地讓蛇走S形，我覺得這不是什麼好策略，最初版本中已經分析過它的問題了。（當然，除非你想使用最最無懈可擊的那個版本，就是完全不管食物，讓蛇一直走S，然後在牆邊留下一條過道即可。這樣一來，蛇總是可以完美地把所有食物吃完，然後佔滿整個空間，可是就很boring了。沒有任何的意思）

上面還提到一個問題：因為食物是隨機出現的，有沒可能出現無解的局面？答案是：有。我運行了程序，然後把每一次佈局都輸出到log，發現會有這樣的情況：

```

1  # # # # # # #
2  # * * * * * #
3  # * * - 0 * #
4  # * * # + * #
5  # * * * * * #
6  # * * * * * #
7  # # # # # # #

```

其中，+號是蛇頭，-號是蛇尾，*號是蛇身，0是食物，#號代表空格，外面一圈# 號代表牆。這個佈局上，食物已經在蛇頭面前了，可是它能吃嗎？不能！因為它吃完食物後，長度加1，蛇頭就會把0的位置填上，佈局就變成：

```

1  # # # # # # #
2  # * * * * * #
3  # * * - + * #
4  # * * # * * #
5  # * * * * * #
6  # * * * * * #
7  # # # # # # #

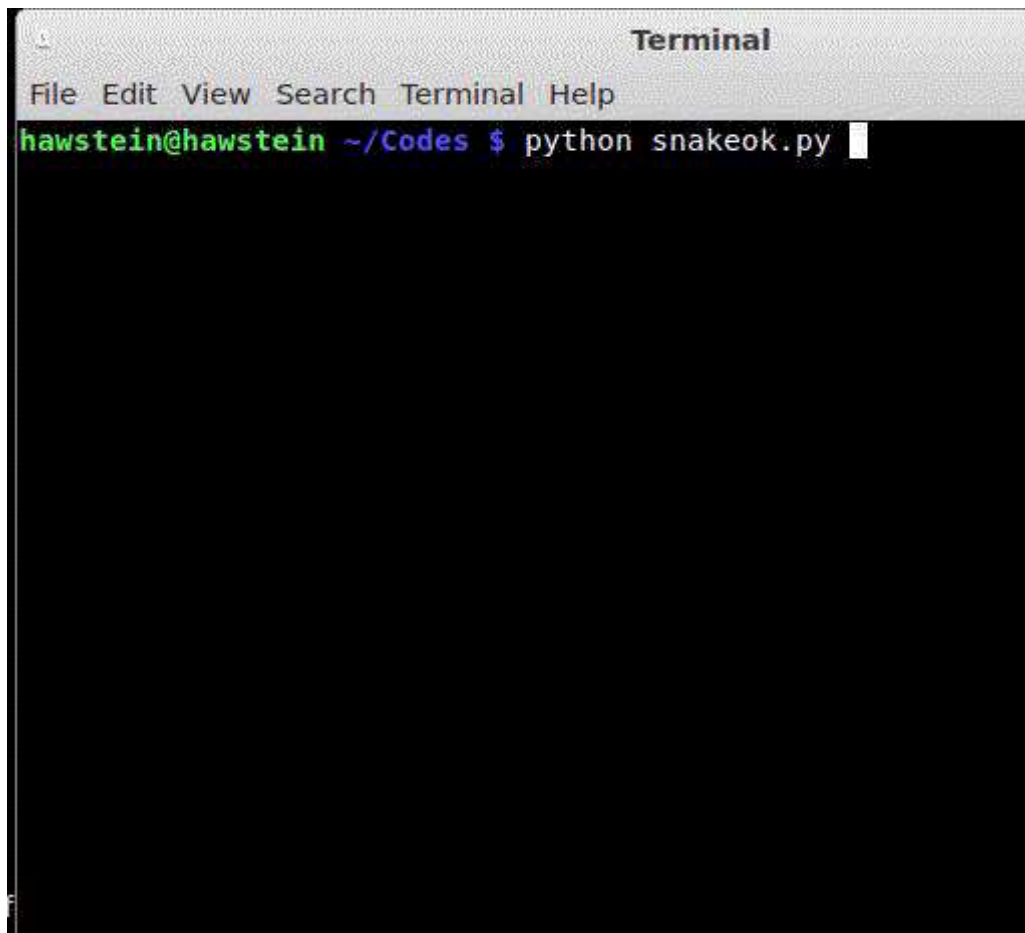
```

此時，由於蛇的長度加1，蛇尾沒有動，而蛇頭被自己圍著，掛掉了。可是，我們卻還有一個空白的格子#沒有填充。按照我們之前教給蛇的策略，面對這種情況，蛇頭就只會一直追著蛇尾跑，每當它和食物有路徑時，它讓虛擬的蛇跑一遍發現，得到的新佈局是不安全的，所以不會去吃食物，而是選擇繼續追著蛇尾跑。然後它就這樣一直跑，一直跑。死循環，直到你按ESC鍵為止。

由於食物是隨機出現的，所以有可能出現上面這種無解的佈局。當然了，你也可以得到完滿的結局，貪吃蛇把整個矩形都填充滿。

上面的最後一個問題，暴力法是否能得到最優序列。從上面的分析看來，可以得到，但不能保證一定得到。

最後，看看高瞻遠矚的蛇是怎麼跑的吧：



矩形大小10*20，除去外面的邊框，也就是8*18。Linux下錄完屏再轉成GIF格式的圖片，優化前40多M，真心是沒法和Windows的比。用下面的命令優化時，有一種系統在用生命做優化的感覺：

```
1  convert output.gif -fuzz 10% -layers Optimize optimised.gif
```

最後還是拿到Windows下用AE，三下五除二用圖片序列合成的動態圖片 (記得要在format options裡選looping，不然圖片是不會循環播放的)

Last but not least

如果對源代碼感興趣，請戳以下的鏈接： Code goes here (<https://github.com/Hawstein/snake-ai>)

另外，本文的貪吃蛇程序使用了curses模塊，類Unix系統都默認安裝的，使用Windows的童鞋需要安裝一下這個模塊，送上地址：需要curses請戳我 (<http://www.lfd.uci.edu/~gohlke/pythonlibs/#curses>)

以上的代碼仍然可以繼續改進(現在加注釋不到300行, 優化一下可以更少), 也可用pygame或是pyglet庫把界面做得更加漂亮, Enjoy!

聲明: 自由轉載-非商用-非衍生-保持署名 | 創意共享3.0許可證

(<http://creativecommons.org/licenses/by-nc-nd/3.0/deed.zh>), 轉載請註明作者及出處

出處: <http://hawstein.com/2013/04/15/snake-ai/> (/2013/04/15/snake-ai/)

PREVIOUS

PYGLET 教程 (/2013/03/31/PYGLET-TUTORIAL/)

NEXT

求兩個單鏈表的和 (/2013/06/30/ADD-SINGLY-LINKED-LIST/)

41 Comments

Hawstein's Blog

 Disqus' Privacy Policy

 Login ▾

 Favorite 10

 Tweet

 Share

Sort by Best ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name

Ron Prog • 7 years ago

我只学过一些java。我是为了能run您程序，在eclipse上面安装的pydev。helloworld啥的都能run了。我把你github上snake.py全都复制下来run的。run以后出了这些error message：

Traceback (most recent call last):

File "/Users/yufang/Documents/PyDev/Computer Programming Assignments/secondhello.py", line 336. in <module>

FEATURED TAGS (/archive/)

Linux (/archive/?tag=Linux)

Algorithm (/archive/?tag=Algorithm)

CC150 (/archive/?tag=CC150)

AlgoCasts (/archive/?tag=AlgoCasts)

Scala (/archive/?tag=Scala)

小結 (/archive/?tag=%E5%B0%8F%E7%BB%93)

Data Structure (/archive/?tag=Data+Structure)

Reading (/archive/?tag=Reading)

C++ (/archive/?tag=C%2B%2B)

Indie Hacker (/archive/?tag=Indie+Hacker)

Python (/archive/?tag=Python)

隨筆 (/archive/?tag=%E9%9A%8F%E7%AC%94)

(/archive/?tag=%E9%9A%8F%E7%AC%94)

(/archive/?tag=%E9%9A%8F%E7%AC%94)

(/archive/?tag=%E9%9A%8F%E7%AC%94)

(/archive/?tag=%E9%9A%8F%E7%AC%94)

(/archive/?tag=%E9%9A%8F%E7%AC%94)

(/archive/?tag=%E9%9A%8F%E7%AC%94)


(/archive/?tag=%E9%9A%8F%E7%AC%94)


(/archive/?tag=%E9%9A%8F%E7%AC%94)

(/archive/?tag=%E9%9A%8F%E7%AC%94)

(/archive/?tag=%E9%9A%8F%E7%AC%94)

(/archive/?tag=%E9%9A%8F%E7%AC%94)

(/archive/?tag=%E9%9A%8F%E7%AC%94)  (/feed.xml)

 (https://twitter.com/hawstein)  (http://weibo.com/hawstein)

 (https://github.com/hawstein)

Copyright © Hawstein's Blog 2021

AlgoCasts (<https://algo casts.io/>) | Debob (<https://debob.co/>) | Depop Bot

(<https://chrome.google.com/webstore/detail/depop-bot-debob/gnmpfkiolmopmndaljjgippfebbhfjam>)