**Python实现推箱子游戏**



#encoding:utf-8

import tkMessageBox

from Tkinter import Tk, Canvas, Frame

from PIL import ImageTk, Image

img = []

def load\_picture():

 global img

 for i in range(0, 10):

 file = "F:/source/" + str(i) + ".bmp"

 im = Image.open(file)

 im = ImageTk.PhotoImage(im)

 img.append(im)

class Grid:

 def \_\_init\_\_(self, master = None, window\_width = 450, window\_height = 500, width = 30):

 self.master = master

 self.width = width

 self.map = []

 self.load\_map(1)

 self.canvas = Canvas(master, width = window\_width, height = window\_height, bg = 'cyan')

 self.canvas.pack()

 self.display\_map()

 def load\_map(self, i):

 file = "F:/source/" + str(i) + ".txt"

 f = open(file, "r")

 lines = f.readlines()

 for k in range(0, len(self.map)):

 self.map.pop()

 for k in range(0, len(lines)):

 mp = []

 line = lines[k]

 for j in range(0, len(lines)):

 p = int(line[j])

 mp.append(p)

 if p == 6:

 self.x = k

 self.y = j

 self.map.append(mp)

 def display\_map(self):

 self.canvas.delete('all') #清空画布canvas

 m = len(self.map)

 n = len(self.map[0])

 for i in range(0, m):

 for j in range(0, n):

 self.create\_image(i, j)

 self.display\_text()

 def display\_text(self):

 self.canvas.create\_text(210, 460, text = "pageUp,pageDown切换关卡", font = 'BOLD', fill = 'black')

 self.canvas.create\_text(210, 480, text = "BackSpace后退", font = 'BOLD', fill = 'black')

 def create\_image(self, i, j):

 global img

 x = (2 \* i + 1) \* self.width / 2

 y = (2 \* j + 1) \* self.width / 2

 pic = self.map[i][j]

 self.canvas.create\_image(y, x, image = img[pic])

class Game(Frame):

 def \_\_init\_\_(self, master = None):

 Frame.\_\_init\_\_(self, master)

 self.grid = Grid(self.master)

 self.direction = 'Down'

 self.map = []

 self.get\_map() # 获取地图，用于标志地图中最初箱子放在目标位置

 self.stack = [1]

 self.level = 1

 self.dir\_x = [-1, 1, 0, 0]

 self.dir\_y = [0, 0, -1, 1]

 self.dir = {'Up':0, 'Down':1, 'Left':2, 'Right':3}

 self.bind\_all("<KeyPress>", self.KeyPress\_event)

 def get\_map(self):

 for k in range(0, len(self.map)):

 self.map.pop()

 for i in self.grid.map:

 mp = []

 for j in i:

 mp.append(j)

 self.map.append(mp)

 def KeyPress\_event(self, event):

 key = event.keysym

 if self.dir.has\_key(key):

 self.direction = key

 self.move()

 elif key == 'Prior':

 self.prior\_level()

 elif key == 'Next':

 self.next\_level()

 elif key == 'BackSpace':

 if len(self.stack) == 1:

 return

 cur\_dir = self.stack.pop() # 当前坐标的方向

 # 当前坐标

 mx = self.stack.pop()

 my = self.stack.pop()

 cur\_picture = self.stack.pop() # 当前坐标cur\_dir方向的图片

 next\_picture = self.stack.pop() # 前一个坐标以前图片

 pre\_dir = self.stack[len(self.stack) - 1]

 dx = self.dir\_x[cur\_dir]

 dy = self.dir\_y[cur\_dir]

 self.grid.map[mx][my] = cur\_picture

 self.grid.map[mx + dx][my + dy] = next\_picture

 self.grid.map[mx - dx][my - dy] = pre\_dir + 5

 self.grid.x = mx - dx

 self.grid.y = my - dy

 self.grid.display\_map()

 def move(self):

 index = self.dir[self.direction]

 x = self.grid.x # 原来的位置

 y = self.grid.y

 dx = self.dir\_x[index] # 位移

 dy = self.dir\_y[index]

 mp = self.grid.map

 self.grid.x += dx

 self.grid.y += dy

 manX = self.grid.x

 manY = self.grid.y

 origin\_picture = mp[manX][manY]

 next\_picture = mp[manX + dx][manY + dy]

 # dir方向是箱子，箱子dir方向是墙或箱子

 if (mp[manX][manY] == 4 or mp[manX][manY] == 3) and (mp[manX + dx][manY + dy] == 2 or mp[manX + dx][manY + dy] == 3 or mp[manX + dx][manY + dy] == 4):

 self.grid.x -= dx

 self.grid.y -= dy

 return

 # dir方向是墙

 if mp[manX][manY] == 2:

 self.grid.x -= dx

 self.grid.y -= dy

 return

 # dir方向是草地或者目的地

 if mp[manX][manY] == 1 or mp[manX][manY] == 9:

 mp[manX][manY] = index + 5

 # self.grid.create\_image(manX, manY)

 if self.map[x][y] == 9: # 原来位置是目的地

 mp[x][y] = 9

 else:

 mp[x][y] = 1 # 原来位置是草地

 # self.grid.create\_image(x, y)

 # dir方向是箱子，箱子dir方向是草地

 if (mp[manX][manY] == 4 or mp[manX][manY] == 3) and mp[manX + dx][manY + dy] == 1:

 mp[manX][manY] = index + 5

 # self.grid.create\_image(manX, manY)

 if self.map[x][y] == 9: # 原来位置是目的地

 mp[x][y] = 9

 else: # 原来位置是草地

 mp[x][y] = 1

 # self.grid.create\_image(x, y)

 mp[manX + dx][manY + dy] = 4 # 草地位置变箱子

 # self.grid.create\_image(manX + dx, manY + dy)

 # dir方向是箱子，箱子dir方向是目的地

 if (mp[manX][manY] == 4 or mp[manX][manY] == 3) and mp[manX + dx][manY + dy] == 9:

 mp[manX][manY] = index + 5

 # self.grid.create\_image(manX, manY)

 if self.map[x][y] == 9: # 原来位置是目的地

 mp[x][y] = 9

 else: # 原来位置是草地

 mp[x][y] = 1

 # self.grid.create\_image(x, y)

 mp[manX + dx][manY + dy] = 3 # 目的地位置变箱子

 # self.grid.create\_image(manX + dx, manY + dy)

 self.grid.display\_map() # 重绘画布

 if self.is\_pass():

 message = tkMessageBox.showinfo("提示", "恭喜过关！")

 if message == 'ok':

 self.next\_level()

 # 每走一步保存当前位置的坐标、以前的图片

 self.stack.append(next\_picture)

 self.stack.append(origin\_picture)

 self.stack.append(manY)

 self.stack.append(manX)

 self.stack.append(index)

 def destroy\_stack(self):

 for i in range(0, len(self.stack) - 1):

 self.stack.pop()

 def prior\_level(self):

 if self.level == 1:

 return

 self.level -= 1

 self.grid.load\_map(self.level)

 self.grid.display\_map()

 self.destroy()

 self.get\_map()

 def next\_level(self):

 if self.level == 13:

 return

 self.level += 1

 self.grid.load\_map(self.level)

 self.grid.display\_map()

 self.destroy()

 self.get\_map()

 def is\_pass(self):

 p = 0

 for i in range(0, 15):

 for j in range(0, 15):

 if self.grid.map[i][j] == 4:

 p += 1

 if p == 0:

 return True

 return False

if \_\_name\_\_ == "\_\_main\_\_":

 root = Tk("sokoban")

 load\_picture()

 game = Game(root)

 root.mainloop()