Documentation de project jeu de vie

Jeu de vie documentation

Pour jouer ce jeu on fait la sélection de cellule, puis on presser le caractère ‘r ’ en clavier

Je pense à créer trois class pour fait ce jeu : Cell\_class1, window \_game, class et main

Cell\_class1 : classe qui détermine le caractère de Cell et contient les fonctions suivantes :

Import pygame : importer le packages pygame pour utiliser l’interface graphique

class Cell: Création de class Cell

\_\_init\_\_(self, surface, grid\_x, grid\_y): constructeur qui créer de cellule par default mort, et possède de longueur et largeur 20 et prend de coordonner sélectionner par l’utilisateur

self.alive = False : premièrement le cellule est mort de couleur blanche

self.surface = surface :cellule possède de surface déterminer par longueur et par largeur

self.grid\_x = grid\_x position de l’abscisses sur le fenêtre

self.grid\_y = grid\_y position de l’ordonner

self.image = pygame.Surface((20,20)) : créer de nouveau surface de longueur 20 et de largeur 20

self.rect = self.image.get\_rect(): retourner un nouveau rectangle couvre toute le surface

self.neighbours = []:sera contient toute le Neighbors(voisins) de cellule sélecter

self.alive\_neighbours = 0: compteur qui compte le nombre de voisine vivre par défaut égal zéro

update(self): fonction qui met à jour une zone de l'affichage à partir du

rectangle en haut à gauche)

draw(self): fonction pour dessiner sur une surface.et colorer le cellule vivant par de couleur noir et le cellule mort par le couleur blanche

blit() : fonction qui fait la changement simplement la couleur des pixels de l'écran. Les pixels ne sont pas ajoutés ou déplacés, c'est seulement la couleur de certains pixels qui est modifiée.

get\_neighbours(self, grid):fonction qui détermine le voisine de cellule et contrôler pour rester dans la cadre de fenêtre

neighbour\_list = [[1, 1], [-1, -1], [-1, 1], [1, -1], [0, -1], [0, 1], [1, 0], [-1, 0]]: list coordonner de 8 voisine

for neighbour in neighbour\_list:

neighbour[0] += self.grid\_x

neighbour[1] += self.grid\_y

for neighbour in neighbour\_list: boucle pour fait l’affectation de voisine si le voisine hors de fenêtre il fait l’affectation convenable pour retourne au cadre de fenêtre

if neighbour[0] < 0:

neighbour[0] += 30

if neighbour[1] < 0:

neighbour[1] += 30

if neighbour[1] > 29:

neighbour[1] -= 30

if neighbour[0] > 29:

neighbour[0] -= 30

live\_neighbours(self): fonction qui compte le nombre de voisine vivante de cellule sélecte

Documentation de la classe : Game\_Window qui contenu la fonction et le package suivante :

import pygame: pour utiliser l’interface graphique

import copy :pour fait copie de la fenêtre

from cell\_class1 import \* pour importer toute le fonction de classe Cell dans class Window\_game

\_\_init\_\_(self, screen, x, y):constructeur pour créer de l’interface graphique (Windows) qui possède de 30,ligne and colonne, ce Windows est diviser en cellule de longueur et largeur 20.

evaluate(self):fonction qui fait l'application directe de règle de la jeu de vie(CONWAY) est retourner un nouveau grid

documentation de la méthode dans la main file

WIDTH, HEIGHT = 800, 600 : longueur et largeur de la fenêtre

BACKGROUND = (199, 199, 199): couleur de la background

FPS = 60# speed de la transition

pygame.display.set\_caption ('Jeu de vie !') : titre de la window

# -------------------Setting mode----------------------------

get\_events():pour retourner l’évent appliquer sur le cellule par l’utilisateur

global running variable

pygame.event.get (): retourner list de eventment applique par l’utisateur

event. type == pygame.QUIT:# lorsque l'utilisateur appuie au bouton quitter en haut de la fenêtre à droite

mouse\_pos = pygame.mouse.get\_pos() :appel de fonction get\_pos() qui retourner la position cliquer par l’utilisateur

if mouse\_on\_grid(mouse\_pos): fonction qui test la position dans la matrice de la fenêtre ou o hors de la fenêtre

update (): fonction qui fait update de écran

draw ():fonction qui dessiner le fenêtre

# ------------------Running mode-----------------------

running\_get\_events(): fonction qui fait exécution, prend l’événement de utilisateur , quitter le window ou l’utilisateur select de cellule , ce fonction fait l’appelé de la fonction mouse\_on\_grid qui test si l’utilisateur fait la sélection dans le window ou hors de la fenêtre.

event.type == pygame.MOUSEBUTTONDOWN: choisir de cellule par l’utilisateur a partir de sélection par la souris

mouse\_on\_grid(mouse\_pos): tester si la position de cellule sélecter dans le cadre de la fenêtre ou non . Si oui fait l’appelé de la fonction click\_cell(pos)

click\_cell(pos):ce fonction calcule la position de cellule sélectionner en fonction de abscisse et ordonner ,puis quand on clique sur le cellule , si cellule est vivre devient morte et vice versa

grid\_pos = [pos[0] - 100, pos[1] - 0] :position de l’abscisse après diminution le marges de la fenêtre

grid\_pos[0] = grid\_pos[0] // 20 :diviser sur 20 pour obtenir un nombre entre 0 et 29 pour le abscisse

grid\_pos[1] = grid\_pos[1] // 20 : diviser sur 20 pour obtenir un nombre entre 0 et 29 pour l’ordonner

pygame.init():pour initialiser tous les modules pygame importés.

game\_window = Game\_window (window, 100, 0): determine le marge de la fenêtre

state = 'setting'

# -----------------Setting mode----------------------------

if state == 'setting': # this setting to set the game before running

# ------------------Running mode-----------------------

keyboard.is\_pressed('r') : dans le running mode si on sélectionner puis on taper r sur le clavier pour commencer le jeu de vie

clock.tick(FPS): frame per seconds ,performance de jeu

pygame.quit() :fonction pour quitter le jeu quand on cliquer sur le boutons X