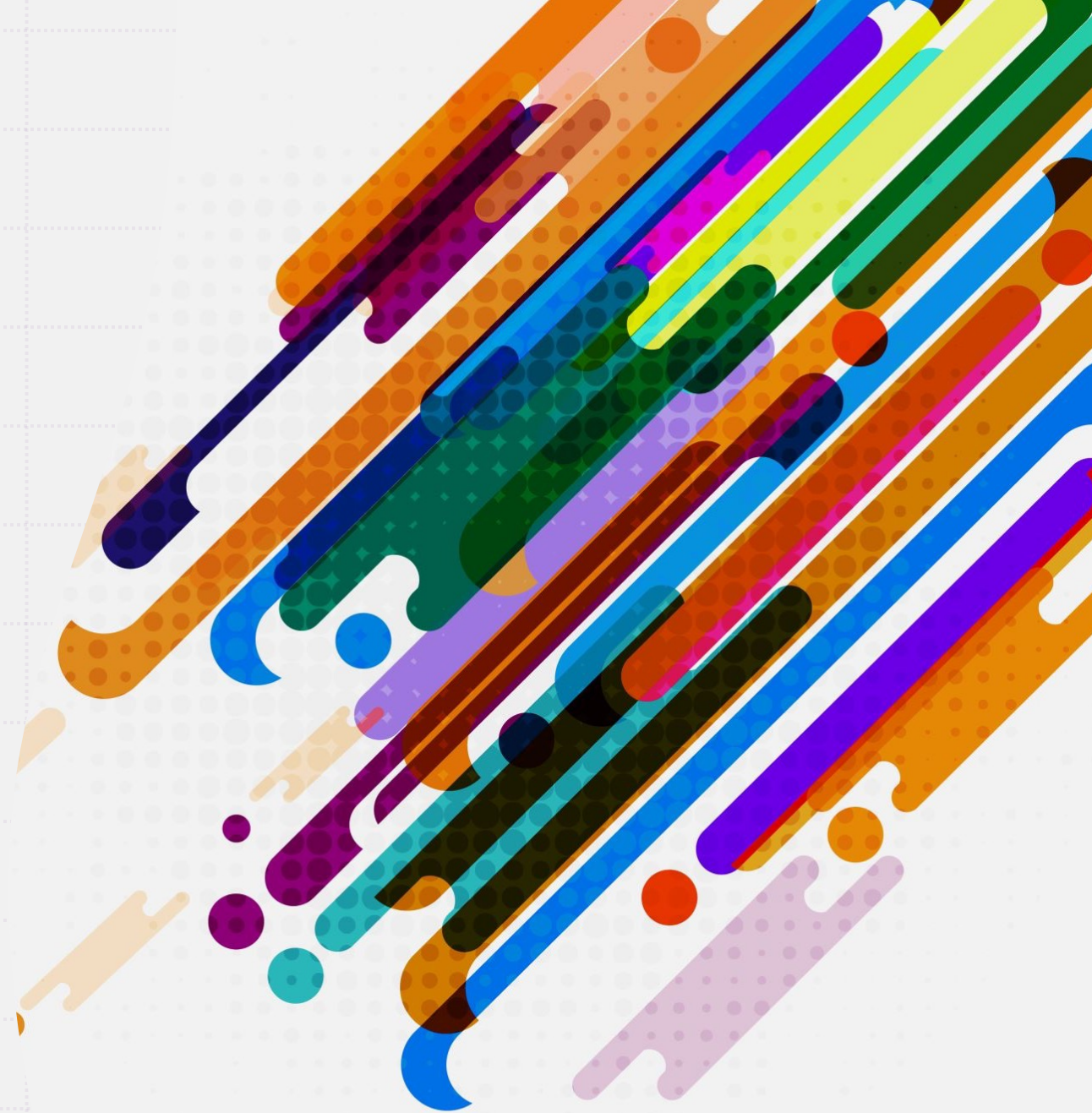


Comment coder AgarIO sur Python ?

Tien-thinh Tran-Truong et Jacques
Zhang



Courte introduction à GitHub et Vscode

- **Définition** : GitHub est une plateforme de développement collaboratif. Elle permet aux développeurs de travailler ensemble sur des projets tout en gérant et en suivant les modifications du code.
- *Caractéristiques principales :*
- **Gestion de dépôt (repository)** : Chaque projet est stocké dans un dépôt, qui peut être public ou privé.
- **Collaboration** : Les développeurs peuvent "forker" un dépôt (créer une copie) pour proposer leurs propres modifications.

Courte introduction à GitHub et Vscode

- **Définition** : Visual Studio Code (VSCode) est un éditeur de code développé par Microsoft.
- *Caractéristiques principales :*
- **Support multilingue** : Prise en charge de nombreux langages de programmation comme JavaScript, Python, C++, etc.
- **Extensions** : Possibilité d'ajouter des fonctionnalités via des extensions, permettant d'adapter l'éditeur à n'importe quel langage.
- **Intégration Git** : Capacité native à gérer les dépôts Git, avec la possibilité de cloner, pusher, puller et voir les modifications directement depuis l'interface.

Courte introduction à GitHub et Vscod

Principales commandes pour interagir avec GitHub depuis un terminal Vscod

- **Clonage d'un dépôt GitHub** : Pour obtenir une copie locale d'un dépôt GitHub sur votre machine :

`git clone [URL_du_dépôt]`

eg : `git clone https://github.com/mon-compte/mon-projet.git`

- **Commiter vos modifications** :

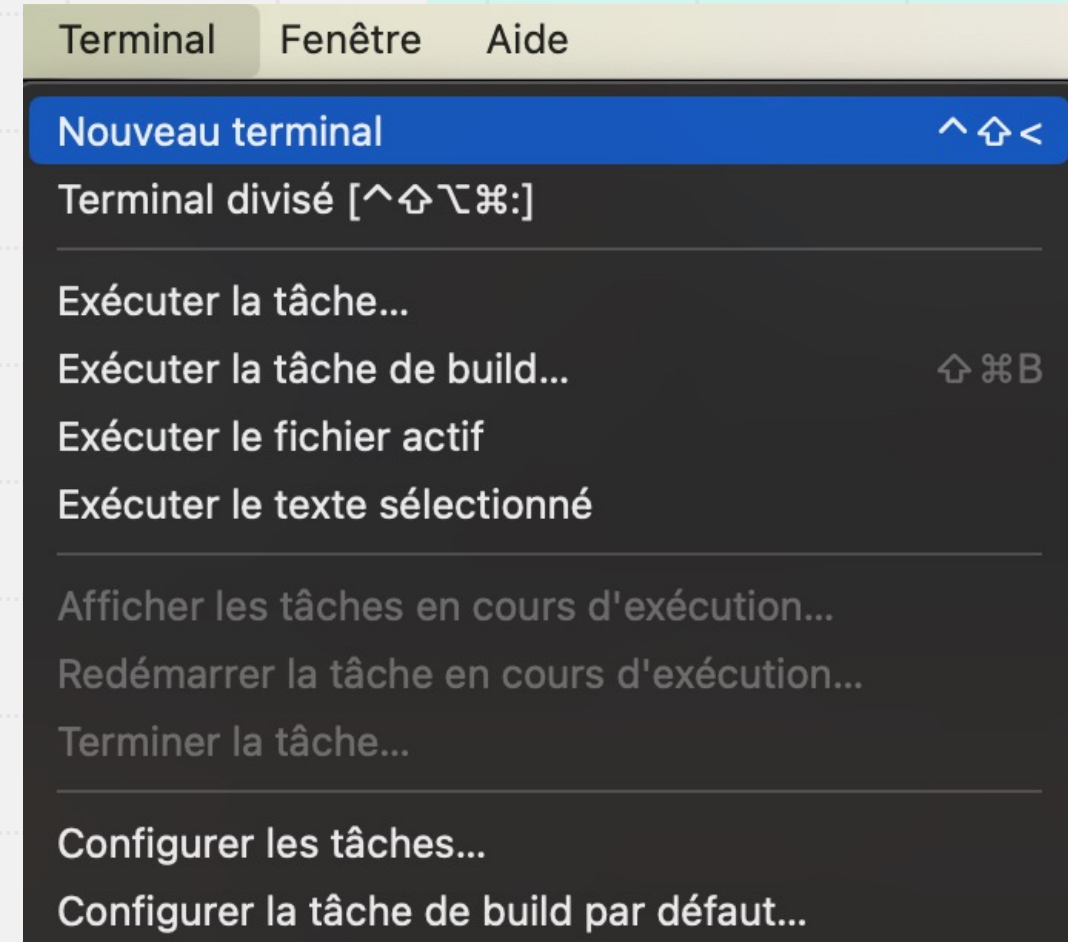
`git commit -m "Message de commit explicatif"`

- **Pousser vos modifications sur GitHub** :

`git push`

- **Récupérer les dernières modifications depuis GitHub** :

`git pull`



Librairie python utilisée pour faire un jeu

Qu'est-ce que pygame ? pygame est une bibliothèque libre de Python dédiée à la création de jeux vidéo. Elle fournit des fonctionnalités qui facilitent le développement de jeux et de simulations graphiques. C'est un excellent choix pour les débutants qui souhaitent s'aventurer dans le développement de jeux grâce à sa simplicité et à sa documentation exhaustive.

Pour installer pygame si vous avez python, il suffit d'exécuter la commande : `pip install pygame`

Librairie python utilisée pour faire un jeu





Dans un projet typique de pygame, le code est souvent organisé sur plusieurs fichier.py. Nous pouvons avoir:

- **constants.py** contenant les valeurs constantes du jeu. (dimension de l'écran, les couleurs, etc)
- **functions.py** contenant les différentes définitions de fonctions utilisées dans les autres fichiers.py.
- **game.py** où l'ont défini la logique du jeu
- D'autres fichiers.py qui permettent de faire la liaison entre serveur et client, etc

Définition de la logique d'AgarIO

- **Logique de la caméra (centrée sur le joueur)**
- 🎯 **But** : - Garder le joueur centré à l'écran.
 - Afficher l'environnement du jeu qui bouge autour du joueur.
- 🔍 **Comment ça marche ?** - Utilisation de la position du joueur et de la taille de la fenêtre pour déterminer comment les autres éléments du jeu doivent être dessinés.
- 🎮 **Calculs clés** : - Position Relative en X : $xrel = xobjet - xjoueur + \frac{W}{2}$
 - Position Relative en Y : $yrel = yobjet - yjoueur + \frac{H}{2}$
- **Fenêtre de jeu** : - Taille : $W \times H$
 - c'est la portion de la carte du jeu visible à tout moment.
- **Carte complète du jeu** : - Taille : $map_width \times map_height$
 - C'est l'espace total où les éléments du jeu existent et se déplacent.

Définition de la logique d'AgarIO

- **Logique de Mouvement basée sur la Position de la Souris**
-  **But** : - Diriger le joueur en fonction de la position de la souris.
-  **Comment ça marche ?** - On détermine la direction du mouvement en calculant la différence entre la position de la souris et le centre de l'écran. - Cette direction indique où et comment le joueur doit se déplacer.
-  **Calculs clés** : - **Composante horizontale** : $dx = x_{mouse} - x_{center}$
- **Composante verticale** : $dy = y_{mouse} - y_{center}$
-  **Dimensions à considérer** : - **Fenêtre de jeu** : Taille : $W \times H$
- **Le point central où le joueur est centré** :
(x_{center}, y_{center}) avec $x_{center} = \frac{W}{2}$ et $y_{center} = \frac{H}{2}$.
- **Position de la souris** : C'est le point vers lequel le joueur se dirige. Représenté par (x_{mouse}, y_{mouse}).

PYGAME CHEAT SHEET!

Getting set up!

```
# Import the pygame module
import pygame

# Initialise pygame
pygame.init()
```

The Game window!

```
# Create the game window
size_x = 800
size_y = 600
screen = pygame.display.set_mode((size_x, size_y))

# Update the game window
pygame.display.update()
```

Writing to the screen!

```
# Write size 36 turquoise text to the screen
colour = (0, 255, 255)
font = pygame.font.Font(None, 36)
location = (300, 10)
screen.blit(font.render("Flippy Bird", True, colour), location)
```

Using Images

```
# Load an image and draw it to the game window
my_image = pygame.image.load("my_image.png")
my_image_x = 0
my_image_y = 0
screen.blit(my_image, (my_image_x, my_image_y))

# Get the height of an image
image_height = my_image.get_rect().size[1]

# Flip an image
my_image_flipped = pygame.transform.flip(my_image, False, True)

# Get the bounding rectangle of an image
pipe_rect = pipe_image.get_rect().move(pipe['x'], pipe['y'])
bird_rect = bird_image.get_rect().move(bird_x, bird_y)
```

Events!

```
# Get the list of events
events = pygame.event.get()

# Check to see if the event is a pressed or released key
if events[0].type == pygame.KEYDOWN:
    print("A key was pressed!")
elif events[0].type == pygame.KEYUP:
    print("A key was released!")

# Check to see which key was pressed
if events[0].key == pygame.K_UP:
    print("The up arrow key was pressed!")
elif events[0].key == pygame.K_DOWN:
    print("The down arrow key was pressed!")
elif events[0].key == pygame.K_q:
    print("The letter q was pressed!")
```

Close the window

```
# Import the system module
import sys

# Close the window and exit
pygame.display.quit()
sys.exit()
```

Pygame Events

```
pygame.QUIT
pygame.ACTIVEEVENT
pygame.KEYDOWN
pygame.KEYUP
pygame.MOUSEMOTION
pygame.MOUSEBUTTONDOWN
pygame.MOUSEBUTTONDOWN
```

Different Keys

Key

pygame.K_BACKSPACE	backspace
pygame.K_CLEAR	clear
pygame.K_RETURN	return
pygame.K_PAUSE	pause
pygame.K_ESCAPE	escape
pygame.K_SPACE	space
pygame.K_a	a
pygame.K_b	b
pygame.K_c	c
pygame.K_d	d
pygame.K_e	e
pygame.K_f	f
pygame.K_g	g
pygame.K_h	h
pygame.K_i	i
pygame.K_j	j
pygame.K_k	k
pygame.K_l	l
pygame.K_m	m
pygame.K_n	n
pygame.K_o	o
pygame.K_p	p
pygame.K_q	q
pygame.K_r	r
pygame.K_s	s
pygame.K_t	t
pygame.K_u	u
pygame.K_v	v
pygame.K_w	w
pygame.K_x	x
pygame.K_y	y
pygame.K_z	z
pygame.K_DELETE	delete
pygame.K_KP0	keypad 0
pygame.K_KP1	keypad 1
pygame.K_KP2	keypad 2
pygame.K_KP3	keypad 3
pygame.K_KP4	keypad 4
pygame.K_KP5	keypad 5
pygame.K_KP6	keypad 6
pygame.K_KP7	keypad 7
pygame.K_KP8	keypad 8
pygame.K_KP9	keypad 9
pygame.K_UP	up arrow
pygame.K_DOWN	down arrow
pygame.K_RIGHT	right arrow
pygame.K_LEFT	left arrow
pygame.K_INSERT	insert
pygame.K_HOME	home
pygame.K_END	end