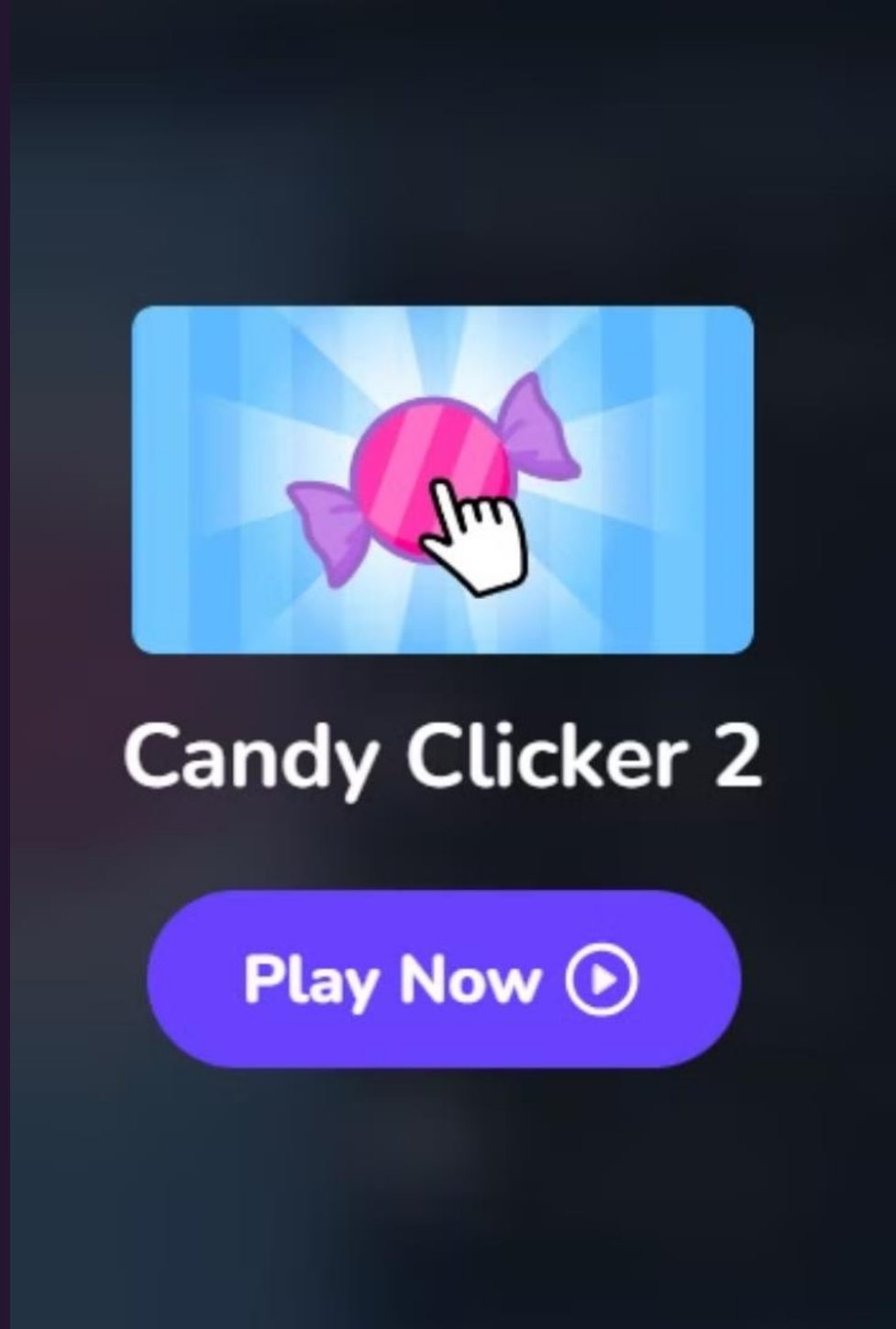
PyAutoGUI & Selenium: Clicker遊戲自動化

本 Candy Clicker 自動化專案的目的是利用 Python 程式語言中的 PyAutoGUI 函式庫，自動執行遊戲中的點擊動作。透過此自動化技術，我們可以提升遊戲的效率，完成重複性的任務，並獲得更有趣的遊戲體驗。

E **by Eve Lee**



報告大綱

- 目標 & 專案動機
- 使用模組：PyAutoGUI 及 Selenium
- 專案流程
- 程式碼說明：ActionChains vs. PyAutoGUI
- 挑戰與解決方案 - 1
- 挑戰與解決方案 - 2
- 成果
- 結論：回顧與未來發展

目標

本 Candy Clicker 自動化專案的目的是利用 Python 程式語言中的 PyAutoGUI 函式庫，自動執行遊戲中的點擊動作。透過此自動化技術，我們可以提升遊戲的效率，完成重複性的任務，並獲得更有趣的遊戲體驗。

遊戲網址: https://www.crazygames.com/game/candy-clicker-2

專案動機

專案的主要動機在於探索如何利用程式語言輔助遊戲操作。藉由模擬滑鼠點擊動作，我們可以幫助玩家省下寶貴的時間與精力，並專注於遊戲的策略思考與樂趣體驗。這個自動化專案將展示如何運用 PyAutoGUI 技術，達成高效而有趣的遊戲體驗。

使用模組：PyAutoGUI及Selenium

PyAutoGUI

PyAutoGUI 是一個強大的 Python 函式庫，允許程式設計師輕鬆模擬滑鼠和鍵盤事件。它能夠精確地點擊、移動滑鼠，以及輸入文字，使其成為遊戲自動化的理想工具。

Selenium - WebDriver & By

Selenium 是一種 **網頁自動化測試** 工具（套件），開發者可以藉由操作 WebDriver 來自動化瀏覽器行為，如填寫表單、點擊按鈕或連結、取得和驗證網頁內容，而透過 By 模組可以定位與選取網頁中的元素。

專案流程

1

2

3

步驟 1：初始化 WebDriver 和瀏覽器設定

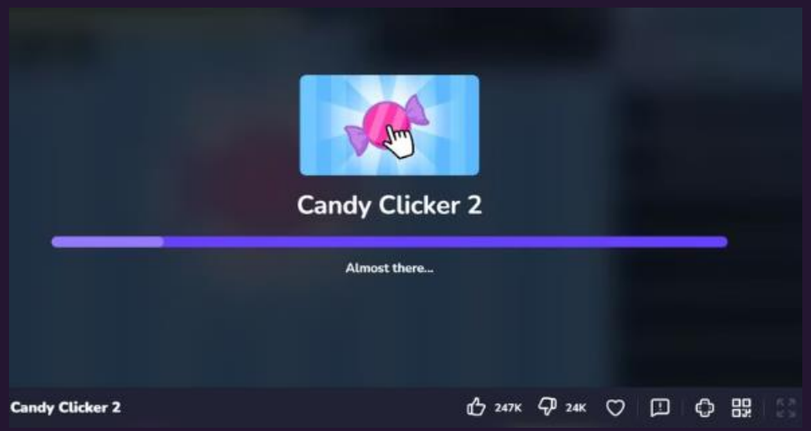
首先，使用 WebDriver_manager 初始化 Chrome 瀏覽器驅動程式。然後設定瀏覽器選項，例如最大化視窗並啟用隱私模式，以確保自動化過程的順暢進行。

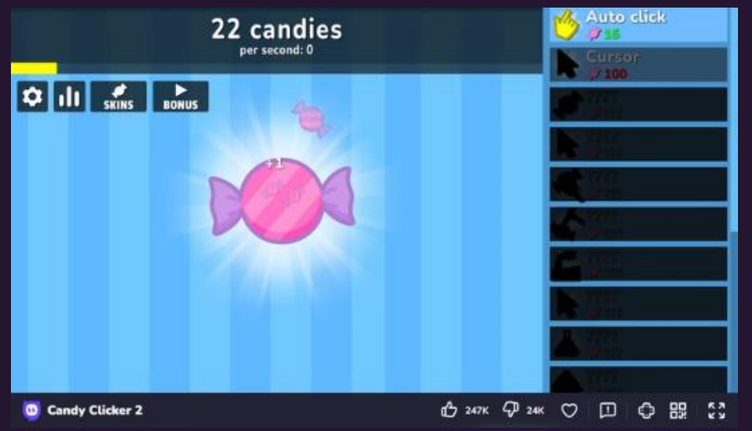
步驟 2：使用 PyAutoGUI 自動化

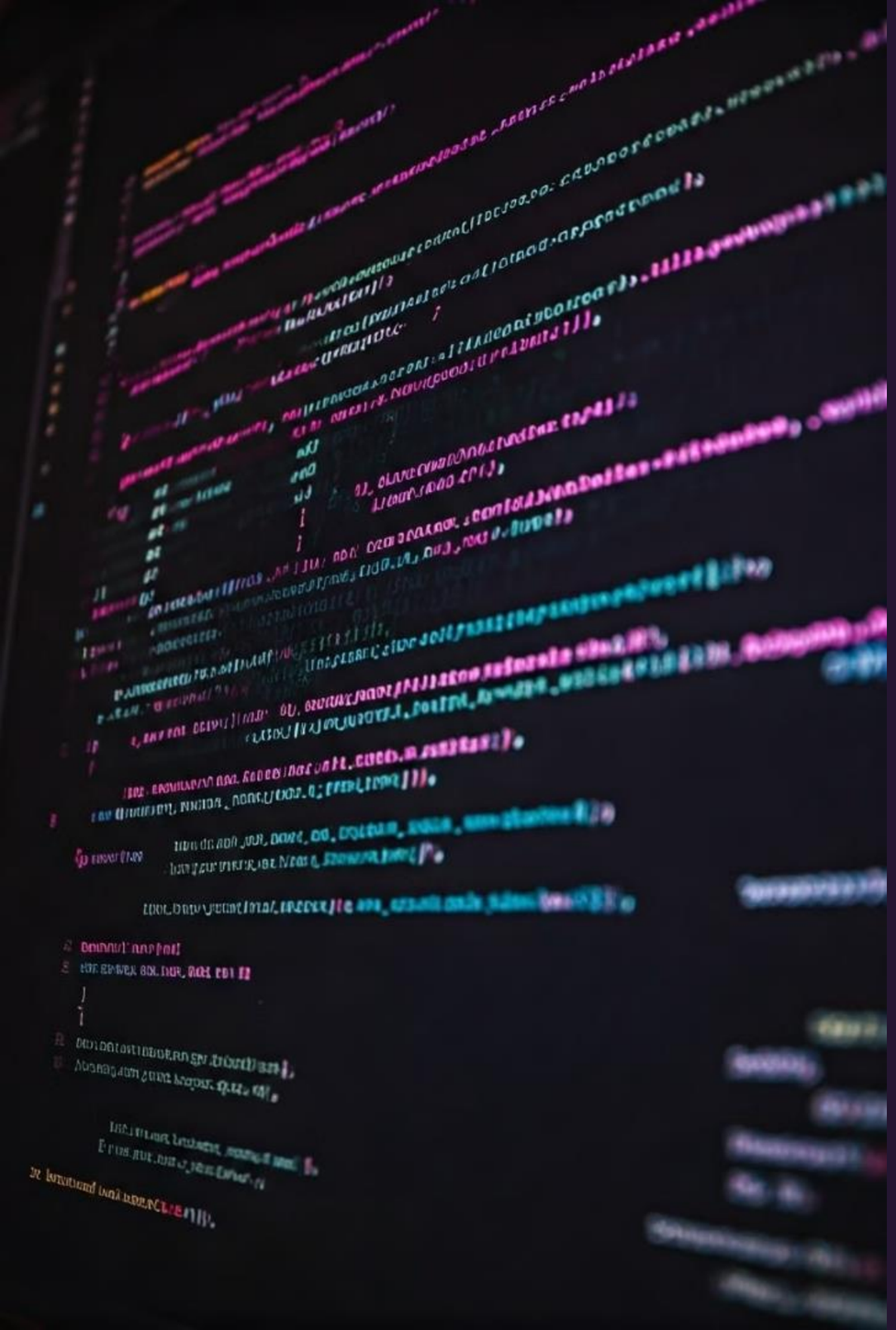
接下來，利用 PyAutoGUI 自動化滑鼠點擊。通過定位遊戲中的點擊目標，執行點擊動作，以達到自動化的目的。

步驟 3：使用 Sleep 處理延遲

為了確保指令按照順序執行，並避免錯誤，在不同的動作之間使用 Sleep 函式引入適當的延遲，讓程式碼有足夠的時間完成每個指令。







程式碼說明ActionChains vs. PyAutoGUI

ActionChains

ActionChains 是 Selenium 提供的一種功能，用於在 WebDriver 中模擬複雜的使用者操作。它可以模擬滑鼠移動、點擊、拖曳、右鍵點擊選單，以及鍵盤輸入文字或按鍵等操作。不過，由於 Candy Clicker 2 是使用 Canvas 開發的，遊戲中的元素無法通過標準的 CSS 選取器 來定位，因此無法使用 Selenium 和 ActionChains 來進行自動化。

PyAutoGUI

而 PyAutoGUI 是一個能控制電腦滑鼠與鍵盤的工具，適用於圖形化介面的操作，並不依賴於瀏覽器或網頁中的元素。因此，在這個專案中，選擇使用 PyAutoGUI 來實現自動點擊，因為它能夠根據螢幕上的圖形元素進行精準操作，而不受限於瀏覽器的元素選取。

挑戰與解決方案-1

瀏覽器限制

某些瀏覽器可能會有彈出視窗或其他限制，影響自動化執行。

➠可以透過設定瀏覽器選項來禁用彈出視窗，並確保自動化程式碼能正常運作。

動態遊戲元素

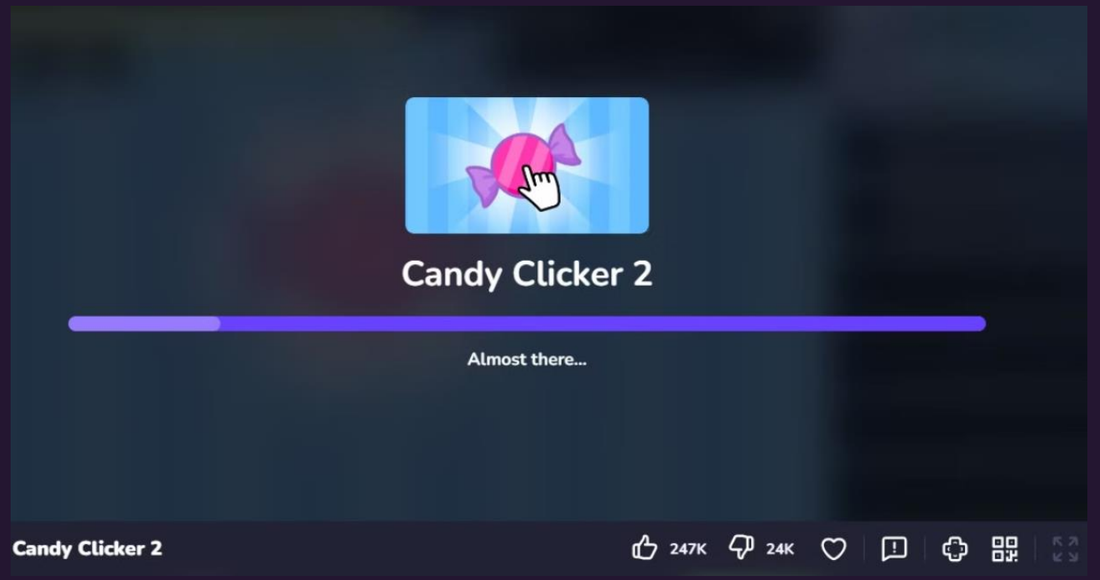
遊戲介面中的元素可能會動態改變，例如按鈕位置或大小。

➠PyAutoGUI 提供了精確點擊設定，可以根據元素的實際位置進行點擊，避免誤操作。

時間問題

程式碼執行速度和遊戲介面的顯示速度可能會不一致，導致點擊動作不準確。

➠使用 time 模組設定適當的延遲，確保點擊動作在遊戲畫面load完後進行。



挑戰與解決方案-2

遊戲限制

由於此遊戲必須在畫面完全載入後，滑鼠需要有移動才會觸發有效的點擊操作，因此在自動化過程中，我們使用了兩個 `for` 迴圈來適應這個設定。

- 第一個 `for` 迴圈負責確保滑鼠在每次點擊前都有輕微的移動，這可以模擬真實的用戶行為，避免無效點擊。
- 第二個 `for` 迴圈則負責在確認遊戲load完後執行點擊操作，確保能夠有效地與遊戲中的元素互動。

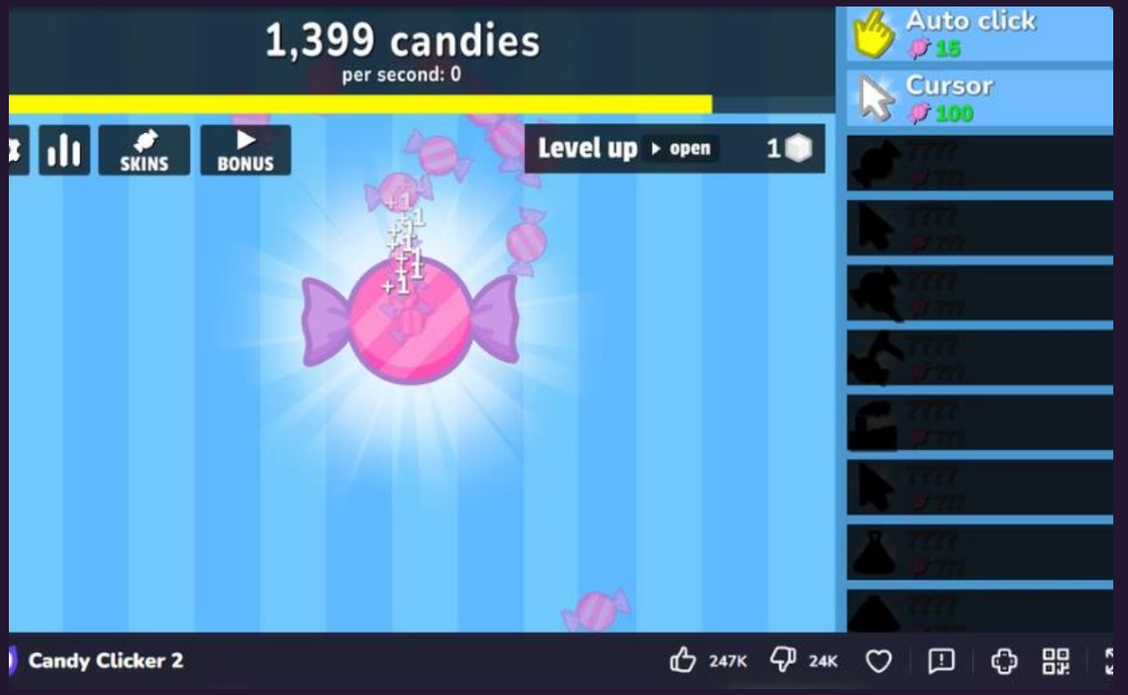
這樣的結構可以應對遊戲中需要滑鼠移動的情況，確保每次點擊都會被遊戲正常識別並生效。

```python
# 按"Start"座標附近，遊戲開始load
pyautogui.click(x=601, y=567)
sleep(15)

# 第一次連續點擊; 微調座標以利遊戲進行
for i in range(50):
    pyautogui.click(x=613, y=539)

sleep(10)

# 第二次連續點擊; 微調座標以利遊戲進行
for i in range(500):
    pyautogui.click(x=613, y=540)
```

成果

自動化遊戲

自動化點擊大幅提升了遊戲效率，自動化的點擊速度遠超手動點擊，快速累積大量的分數。



快速達到遊戲目標

手動點擊效率較低，點擊速度受限於玩家的反應時間，需要花費更多時間才能累積相同數量的糖果。

結論：回顧與未來發展

分析遊戲

依遊戲設計，需要先分析遊戲設計並使用合適的模組以達到目的。

未來發展

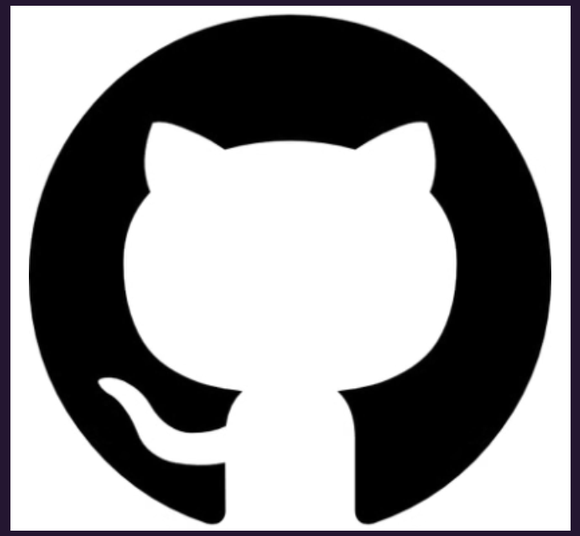
可探索其他遊戲的自動化或加入更複雜的動作。

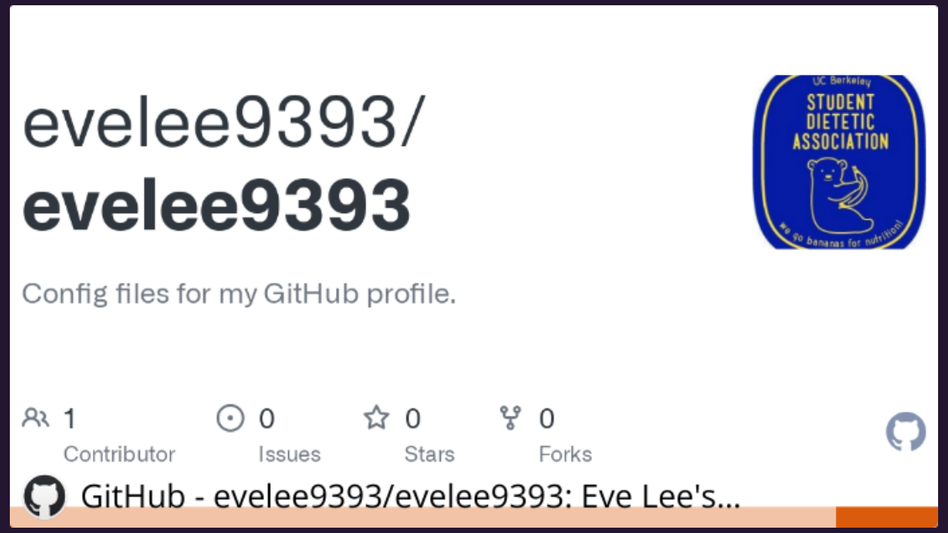
進階應用

加入更多複雜動作，可以點擊右側level up工具，更快速得到高分。

李敬怡的 GitHub 頁面

歡迎到我的 GitHub 頁面，查看這個專案的原始碼和更多有趣的作品。





https://github.com/evelee9393/evelee9393

Thank you!