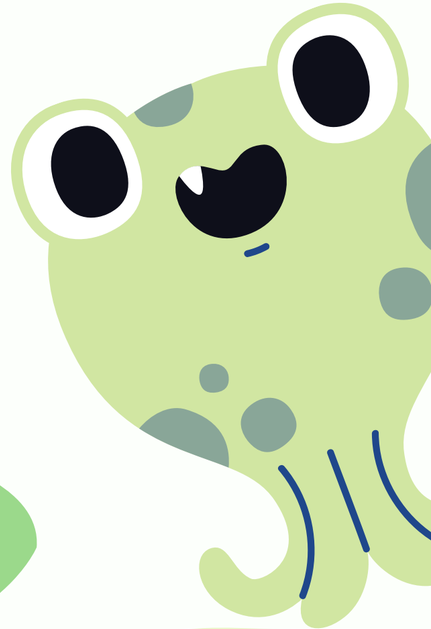
尋找機頭
(隨機變體版)

組員:412410093 侯善文
412416561 吳家盈
412411117 吳欣語



目錄

遊戲規則

- 遊戲目標：在限定步數內找到所有飛機的機頭
- 遊戲格子：10x10，每個格子可點擊翻開，步數+1
- 點擊格子後：空格 → 顯示白色　機身 → 顯示藍色　機頭 → 顯示紅色
- 遊戲結束條件：找到所有機頭 → 獲勝　超過最大步數 → 遊戲失敗
- 步數受難度限制:簡單→40 步　一般→30 步　困難→20 步

特殊玩法

隨機飛機形狀：

- 每架飛機的翅膀、尾翼、機身長度隨機生成
- 每局遊戲形狀可能不同

旋轉飛機：

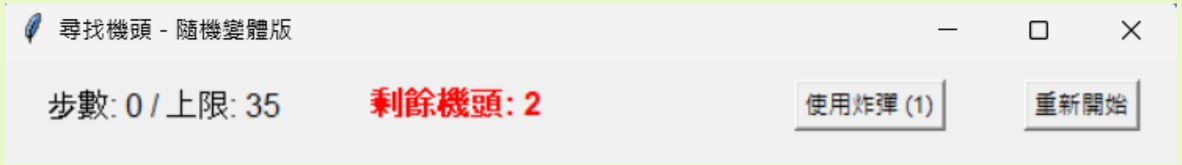
- 飛機可隨機旋轉 0°/90°/180°/270°

炸彈功能：

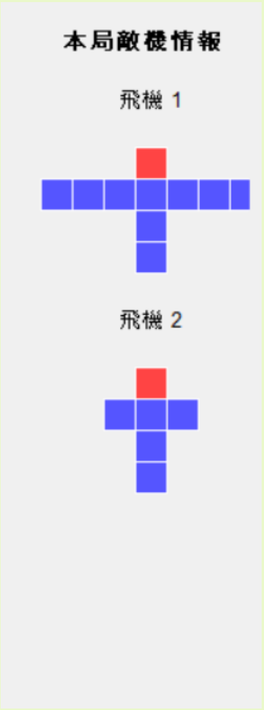
- 一次性使用
- 翻開 2x2 區域的格子
- 可炸到機頭或機身，加快搜尋速度

遊戲畫面概覽

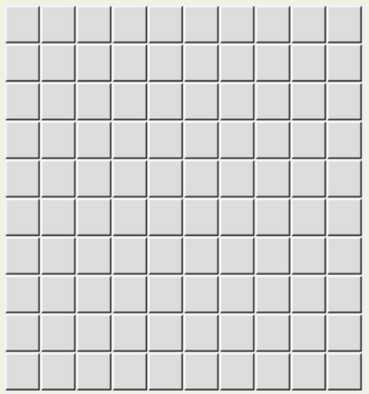
上方:

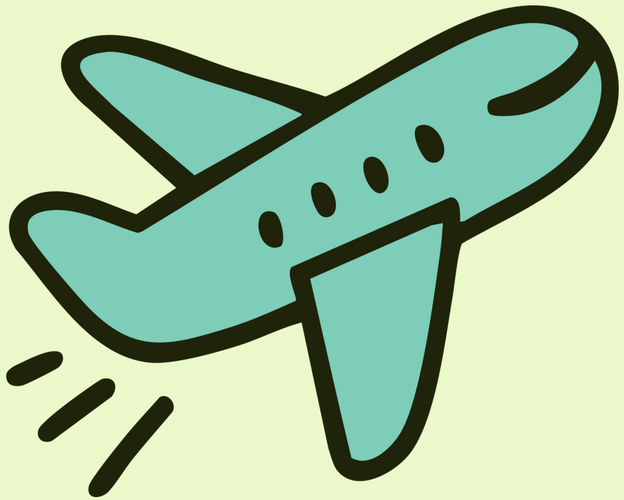


右方:



左方:





程式碼簡介

初始化格子 ⇨ 遊戲設定 ⇨ 開始遊戲 & 重置資料

⇩

生成飛機形狀

⇩

旋轉與放置飛機 ⇦ 點擊格子翻開 ⇦ 翻開格子邏輯

⇧

使用炸彈 ⇨ 勝利or失敗判斷

初始化格子

python 複製程式碼

```python
# 建立 10x10 遊戲格子
for r in range(GRID_SIZE):
    for c in range(GRID_SIZE):
        btn = tk.Button(self.game_frame, width=4, height=2, bg=COLOR_DEFAULT,
                        command=lambda row=r, col=c: self.on_click(row, col))
        btn.grid(row=r, column=c, padx=1, pady=1)
        self.buttons[r][c] = btn  # 儲存按鈕對應位置
```

註解：

- GRID_SIZE=10 → 建立 10x10 按鈕格子
- bg=COLOR_DEFAULT → 初始灰色

執行結果：

- 所有格子灰色，尚未翻開

遊戲設定

```python
# 玩家輸入飛機數量
num = simpledialog.askinteger("飛機數量", "輸入 2 或 3")
# 玩家選擇難度
difficulty = simpledialog.askstring("難度", "簡單/一般/困難")
```

註解：

- 玩家輸入飛機數量
- 選擇難度 → 決定最大步數

執行結果：

- 例如選 3 架飛機，難度「一般」 → 最大步數 30

開始遊戲 & 重置資料

```python
self.steps = 0              # 步數歸零
self.found_heads = 0        # 已找到機頭數歸零
self.bomb_available = 1     # 炸彈可用次數
self.lbl_steps.config(text=f"步數: {self.steps} / 上限: {self.max_steps}")
self.update_head_label()  # 更新剩餘機頭標籤
```

註解：

- 每局開始前清空資料
- UI 標籤更新步數和剩餘機頭

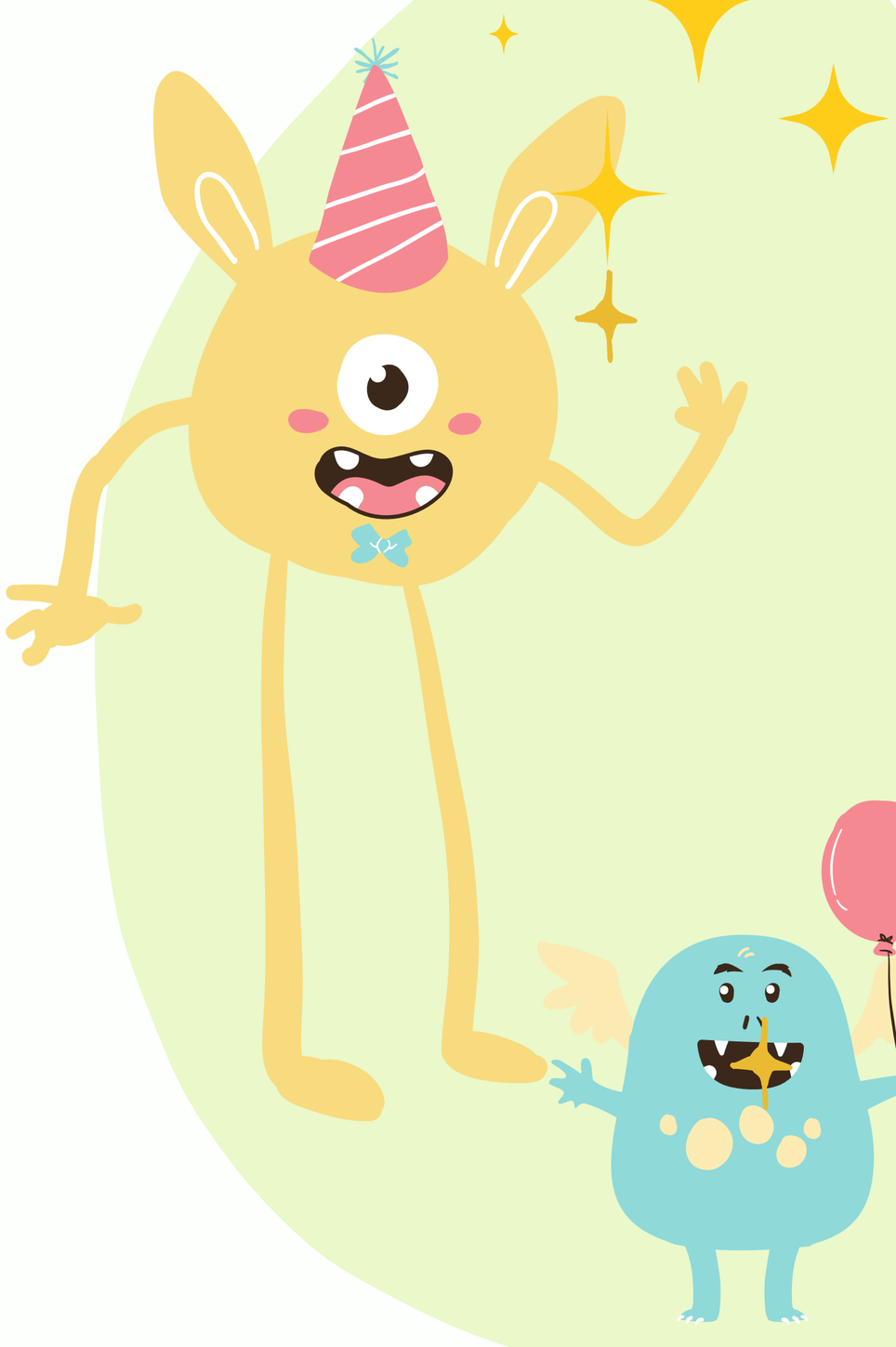
執行結果：

- 左上角顯示「步數: 0 / 上限: 30」
- 剩餘機頭顯示「剩餘機頭: 3」

生成飛機形狀

python 複製程式碼

```python
shape = [(0,0),(0,1)]
wing_len = random.randint(1,3)
body_len = random.randint(2,4)
for i in range(2, body_len+1): shape.append((0,i))
for i in range(1, wing_len+1):
    shape.append((-i,1))   # 左翅
    shape.append((i,1))    # 右翅
if random.choice([True,False]):
    tail_y = body_len
    shape.append((-1, tail_y))
    shape.append((1, tail_y))
```



註解：

- shape[0] 為機頭
- 隨機翅膀長度及尾翼

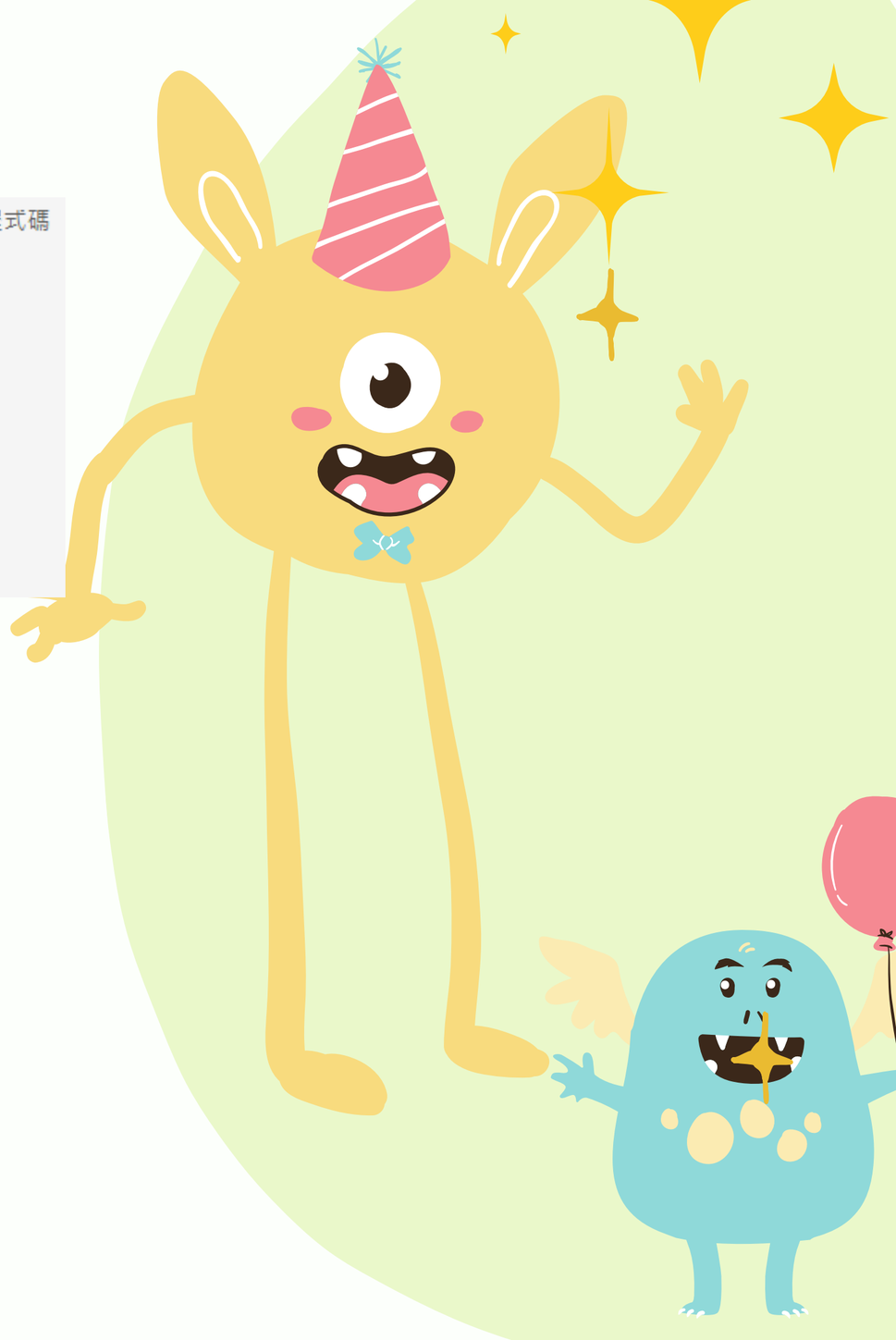
執行結果：

- 每局飛機形狀不同
- 預覽 Canvas 顯示形狀

旋轉與放置飛機

python 複製程式碼

```
rotation = random.choice([0,90,180,270])
rotated_shape = self.rotate_shape(raw_shape, rotation)
start_r = random.randint(0,9)
start_c = random.randint(0,9)
if self.is_valid_position(start_r, start_c, rotated_shape):
    self.add_plane_to_grid(start_r, start_c, rotated_shape)
```



註解：

- 隨機旋轉角度
- 隨機格子放置
- 檢查是否超出邊界或重疊

執行結果：

- 遊戲格子隱藏飛機
- UI 顯示灰色格子

點擊格子翻開

python 複製程式碼

```
self.steps += 1
if self.steps > self.max_steps:
    messagebox.showinfo("遊戲失敗")
self.reveal_cell(r,c)
```

註解：

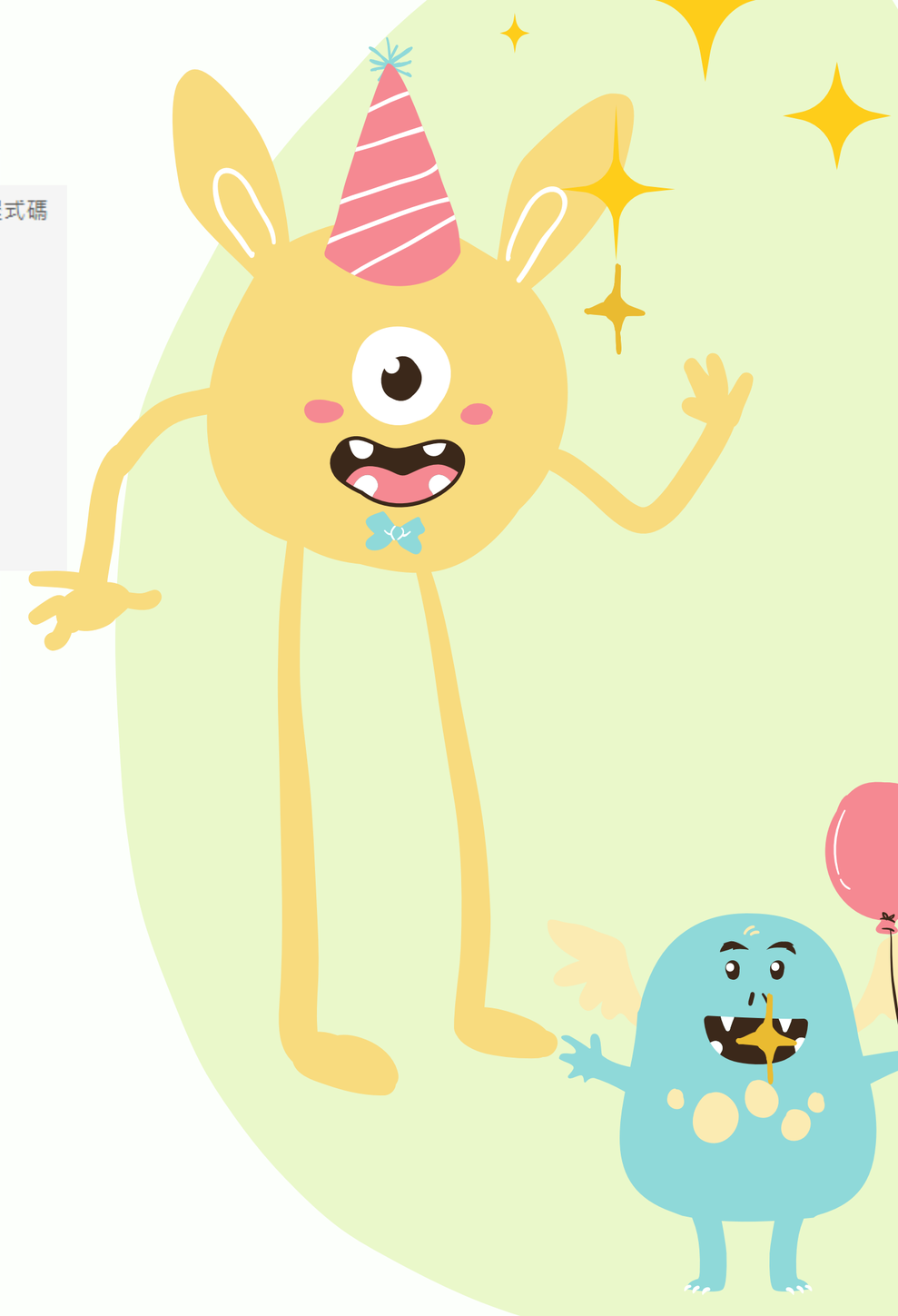
- 點擊格子 → 步數 +1
- 超過步數 → 遊戲失敗
- 否則翻開格子

執行結果：

- 格子顏色變化
- 步數更新

翻開格子邏輯

```python
if content is None: btn.config(bg=COLOR_MISS)    # 空格 → 白色
elif content=='B': btn.config(bg=COLOR_BODY)     # 機身 → 藍色
elif content=='H':
    btn.config(bg=COLOR_HEAD, text="X")          # 機頭 → 紅色 X
    self.found_heads += 1
    self.update_head_label()
```



執行結果：

- 白色：空格
- 藍色：機身
- 紅色 X：機頭，剩餘機頭標籤減一

使用炸彈

```python
row = simpledialog.askinteger(...)
col = simpledialog.askinteger(...)
for dr in range(2):
    for dc in range(2):
        self.reveal_cell(row+dr, col+dc)
self.bomb_available -= 1
self.btn_bomb.config(state=tk.DISABLED)
```

註解：

- 玩家點選要放置炸彈的位置
- 翻開 2x2 區域
- 更新炸彈狀態 → 只能用一次

執行結果：

- 2x2 格子同時顯示顏色
- 若炸到機頭 → 更新剩餘機頭

勝利判斷or失敗

成功:

```python
if self.found_heads == self.total_heads:
    messagebox.showinfo("獲勝！")
```

執行結果：

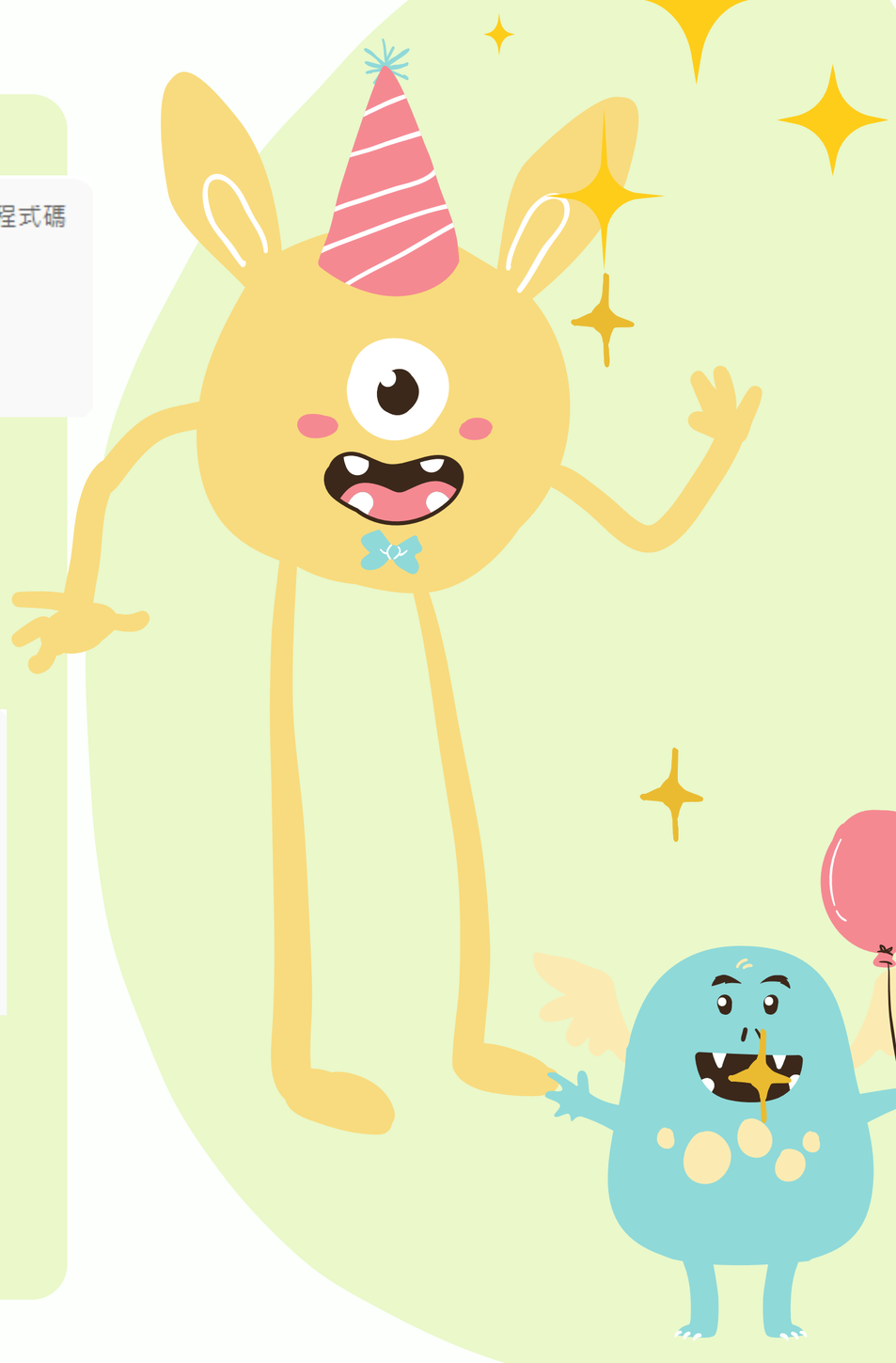
- 找到所有機頭 → 彈出「恭喜！你用了 XX 步」

失敗:

```python
if self.steps > self.max_steps:
    messagebox.showinfo("遊戲失敗")
    for row in self.buttons:
        for b in row: b.config(state=tk.DISABLED)
```

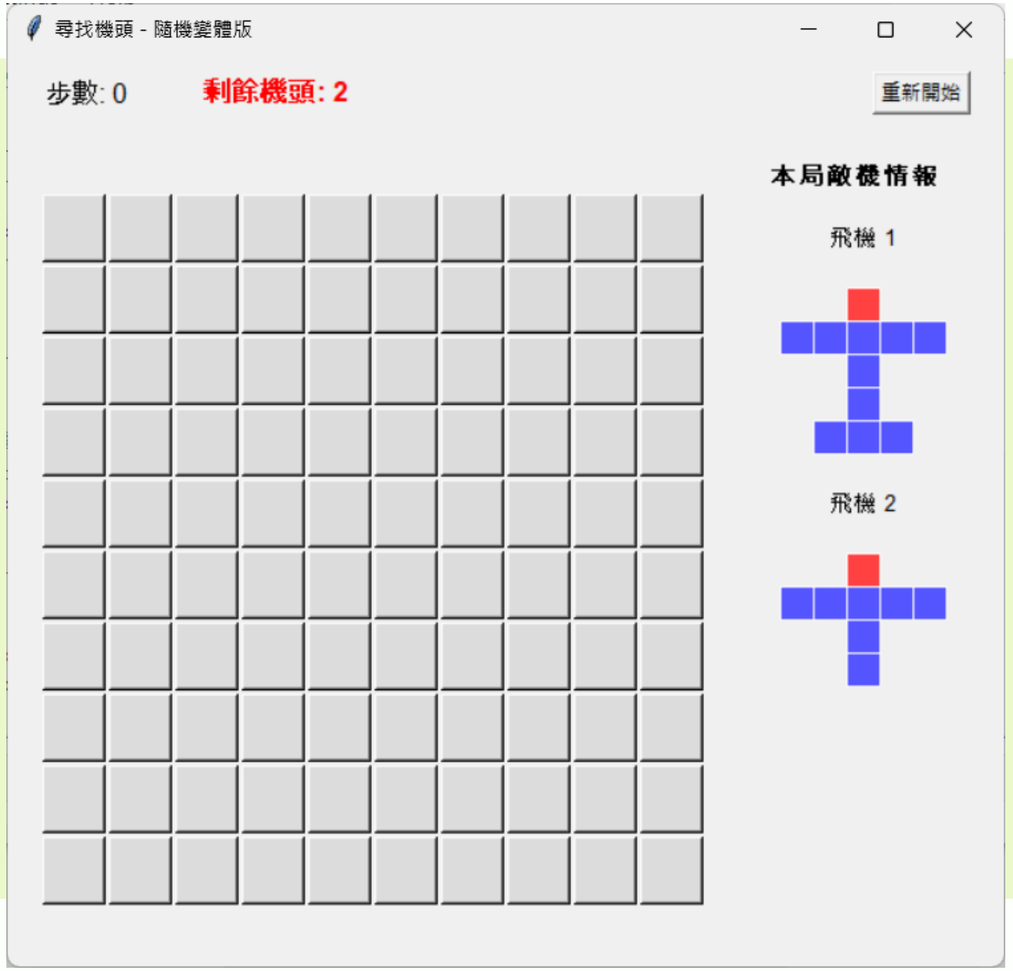
執行結果：

- 步數超過 → 遊戲結束
- 所有格子無法再點擊



COMMIT

第一版:基礎玩法



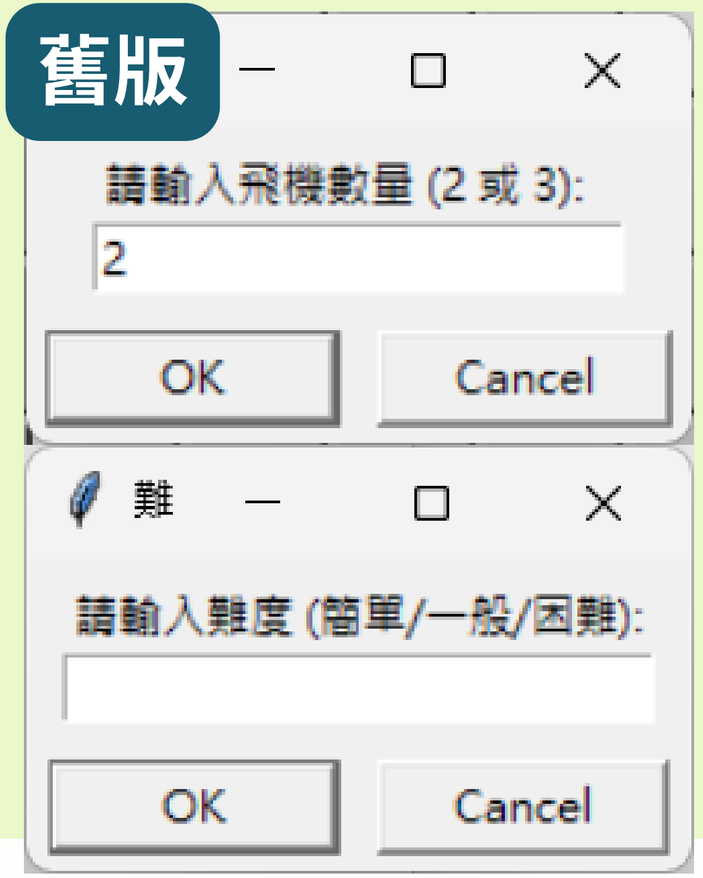
COMMIT

第二版:新增限制步數及炸彈玩法

COMMIT

第三版:更改選擇模式
視窗為點選模式

舊版



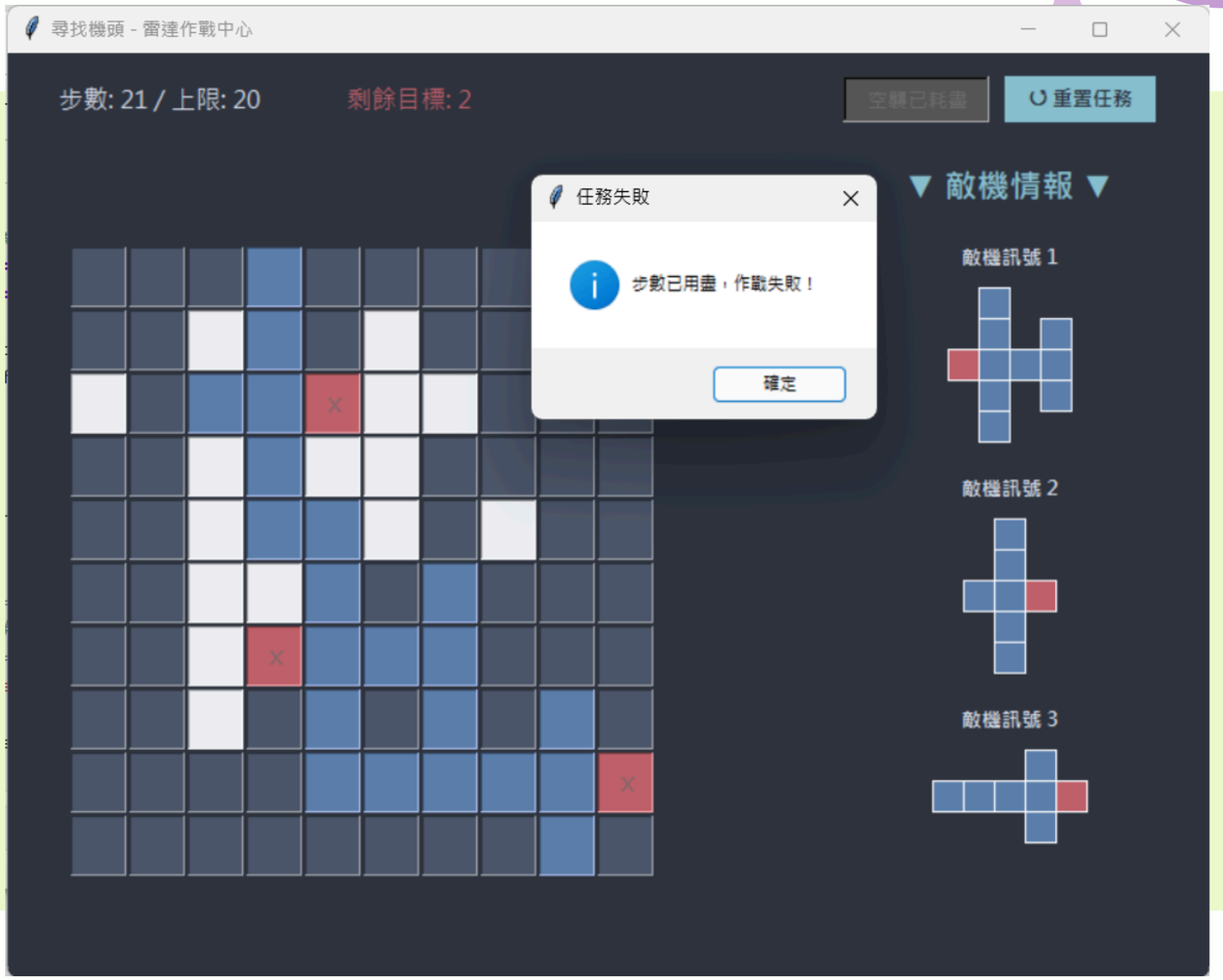
新版

COMMIT

第四版:美化版面

COMMIT

第四版:遊戲結束後
顯示所有的飛機



CODE REVIEW

原程式碼:

```
def place_planes(self, count):
    placed = 0
    while placed < count:
        shape = self.rotate_shape(self.generate_random_shape(), random.choice([0, 90, 180, 270]))
        r, c = random.randint(0, GRID_SIZE-1), random.randint(0, GRID_SIZE-1)
        if self.is_valid_position(r, c, shape):
            self.add_plane_to_grid(r, c, shape)
            self.planes.append(shape)
            placed += 1
```

修正後的程式碼:

```
def place_planes(self, count):
    placed = 0
    attempts = 0 # 【新增】安全計數器，用來計算嘗試了幾次

    # 【修改】while 迴圈加入 `and attempts < 1000`
    # 原因：防止因隨機位置一直重疊導致程式陷入死循環 (Infinite Loop) 卡住
    while placed < count and attempts < 1000:
        attempts += 1
        shape = self.rotate_shape(self.generate_random_shape(), random.choice([0, 90, 180, 270]))
        r, c = random.randint(0, GRID_SIZE-1), random.randint(0, GRID_SIZE-1)
        if self.is_valid_position(r, c, shape):
            self.add_plane_to_grid(r, c, shape)
            self.planes.append(shape)
            placed += 1

    # 【新增】如果嘗試了 1000 次還是放不完，印出警告 (方便除錯)
    if placed < count:
        print("警告：無法放置所有飛機 (嘗試次數過多)")
```

分工

組員	工作
侯善文	製作PPT、git commit、 增加創新玩法、code review、報告
吳家盈	製作PPT、建立GitHub Repository、git commit、增加創新玩法、code review
吳欣語	製作PPT、遊戲發想、git commit、修改遊戲細節、code review、報告


THANK
YOU FOR
LISTENING!
See You Next Time!