自主學習成果分享

森森不息電腦程式遊戲創作

211胡馨予

動機

在高一的多元選修課中設計出一款桌遊，在學習Python程式語言後，便想嘗試用程式寫出來。

最終成果

計分卡、指示物

知識牌
(1 2 3)
三元素牌
(陽光 空氣 水)
功能牌
(菜蟲&辣椒水 土壤貧瘠&肥料 颱風&溫室)

題目卡(左上小樹苗)

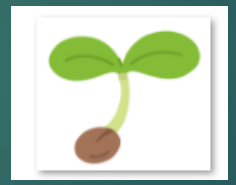
進階題目卡(左上大樹)

說明書

遊戲規則

高一所設計的桌遊因設計時沒有考慮到牌數的問題，導致結束一場遊戲需要花費許多時間。於是便參考一款知名的策略遊戲—**sleeping queen**，改變遊戲機制。

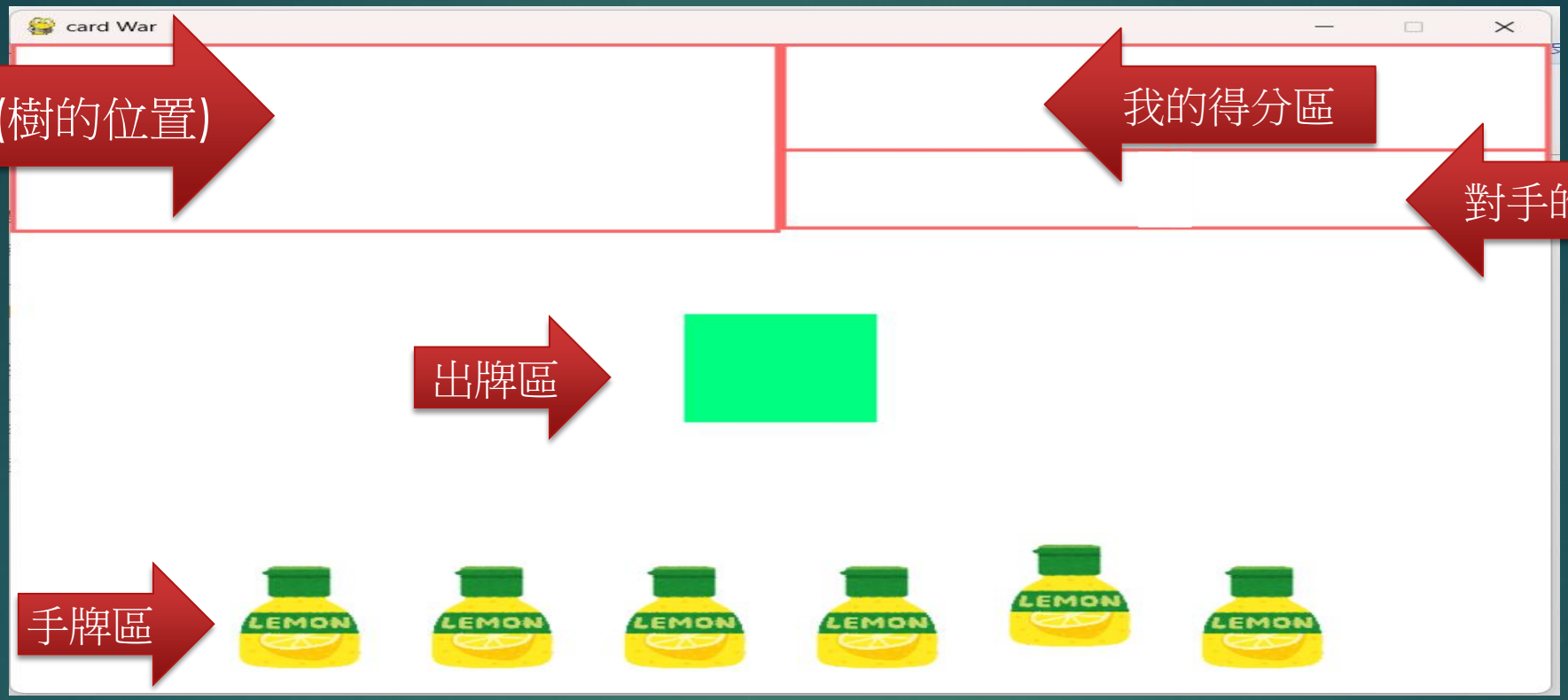
功能牌(進攻與防守)



功能牌(得分)

數字牌

遊戲佈局

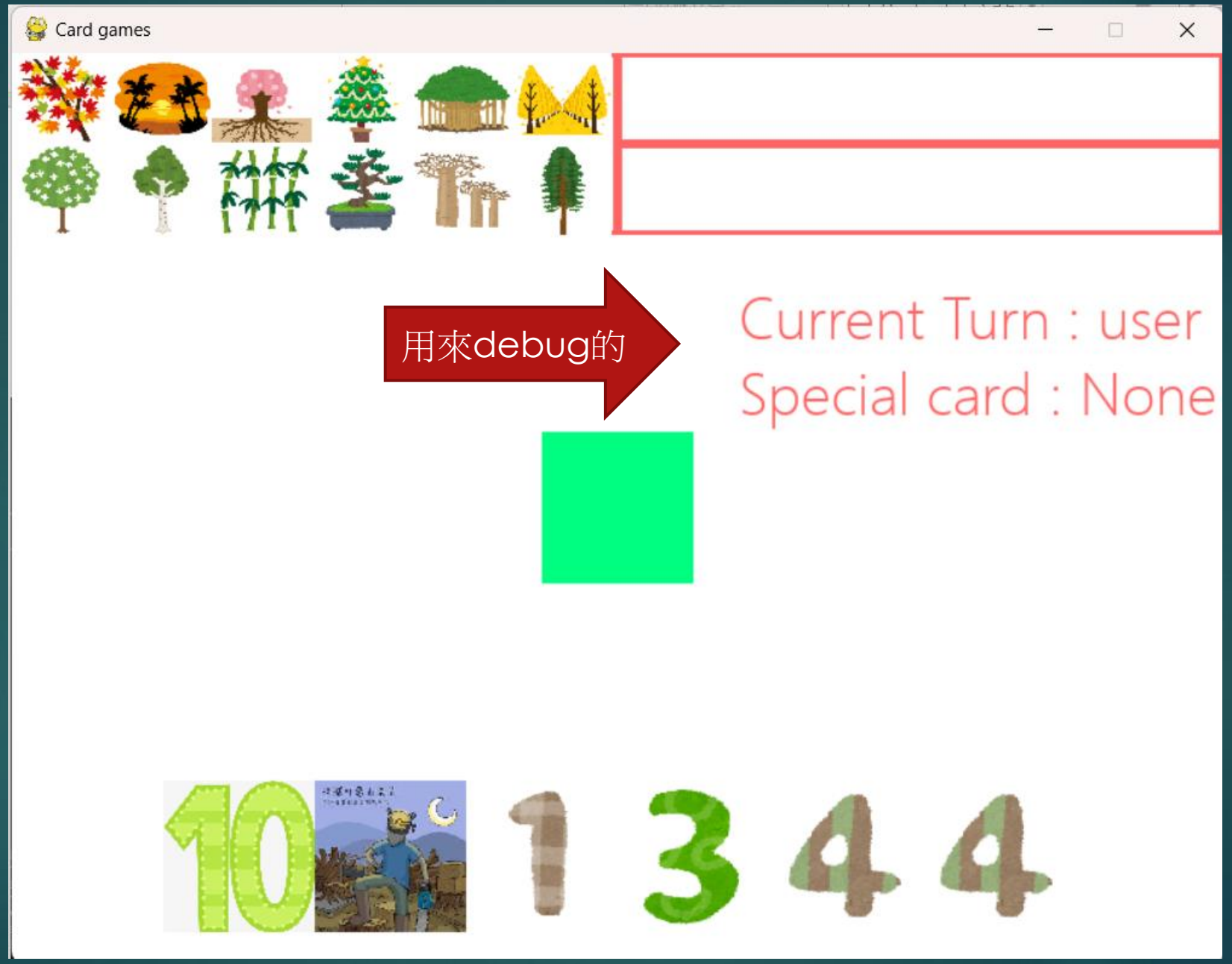



```
def draw_queen_area(window_surface):
    # draw queen area
    public_rect = pygame.Rect(PUBLIC_TREE_AREA_X, PUBLIC_TREE_AREA_Y, PUBLIC_TREE_AREA_WIDTH, PUBLIC_TREE_AREA_HEIGHT)
    pygame.draw.rect(window_surface, TREE_AREA_COLOR, public_rect, 3)

    user_rect = pygame.Rect(TREE_AREA_X, TREE_AREA_Y, TREE_AREA_WIDTH, TREE_AREA_HEIGHT)
    pygame.draw.rect(window_surface, TREE_AREA_COLOR, user_rect, 3)

    ai_rect = pygame.Rect(AI_TREE_AREA_X, AI_TREE_AREA_Y , AI_TREE_AREA_WIDTH, AI_TREE_AREA_HEIGHT)
    pygame.draw.rect(window_surface, TREE_AREA_COLOR, ai_rect, 3)
    #small_queen_rect2 = pygame.Rect(WINDOW_WIDTH - SMALL_TREE_AREA_WIDTH, TREE_AREA_HEIGHT - 3, SMALL_TREE_AREA_WIDTH, SMALL_TREE_AR
    #pygame.draw.rect(window_surface, TREE_AREA_COLOR, small_queen_rect2, 3)
```

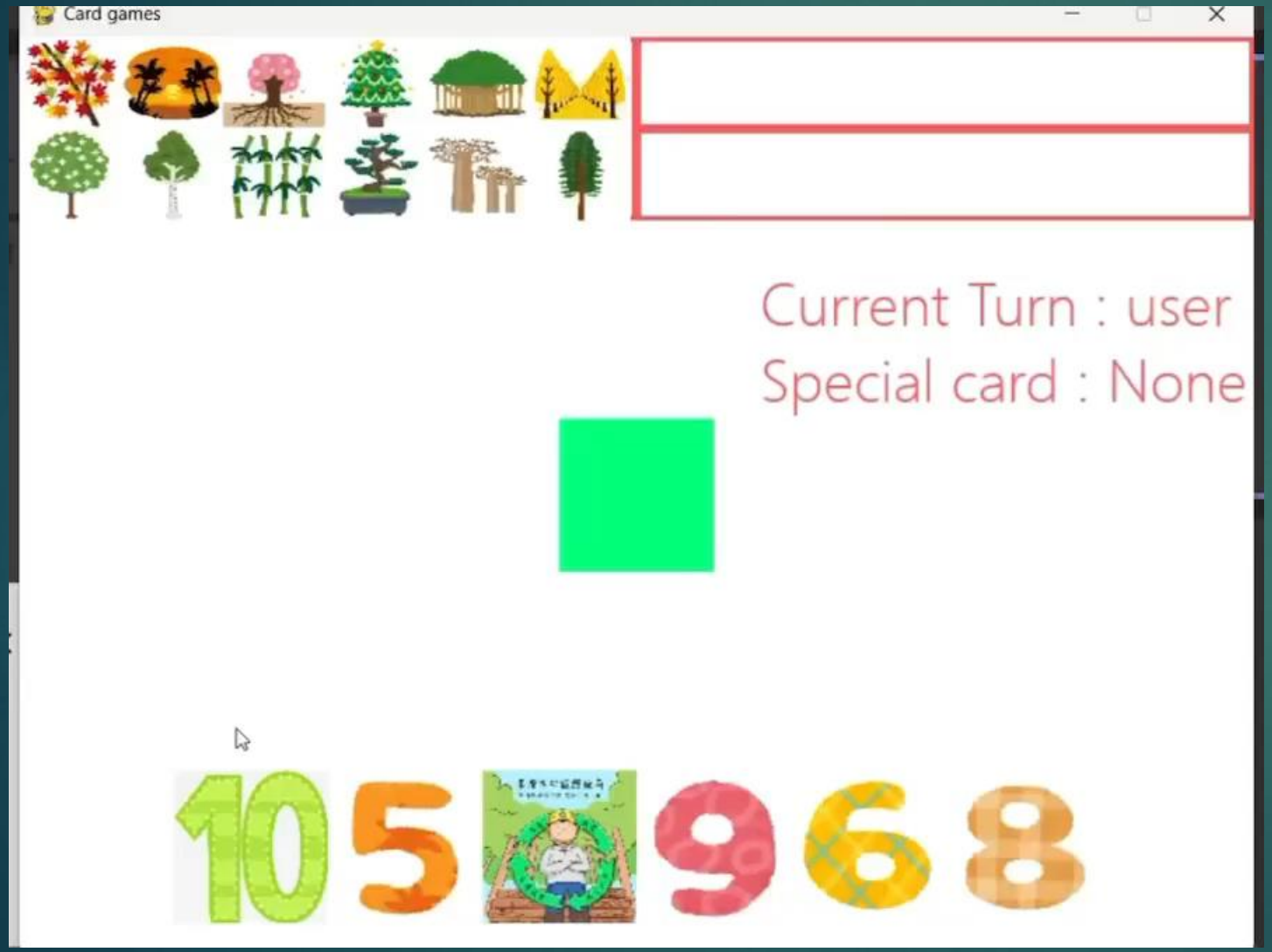
用xy座標來劃分區域

畫面

放上精美的圖片!


Card games
用來debug的
Current Turn : user
Special card : None
10
1 3 4 4

當滑鼠碰到手牌->手牌向上移動



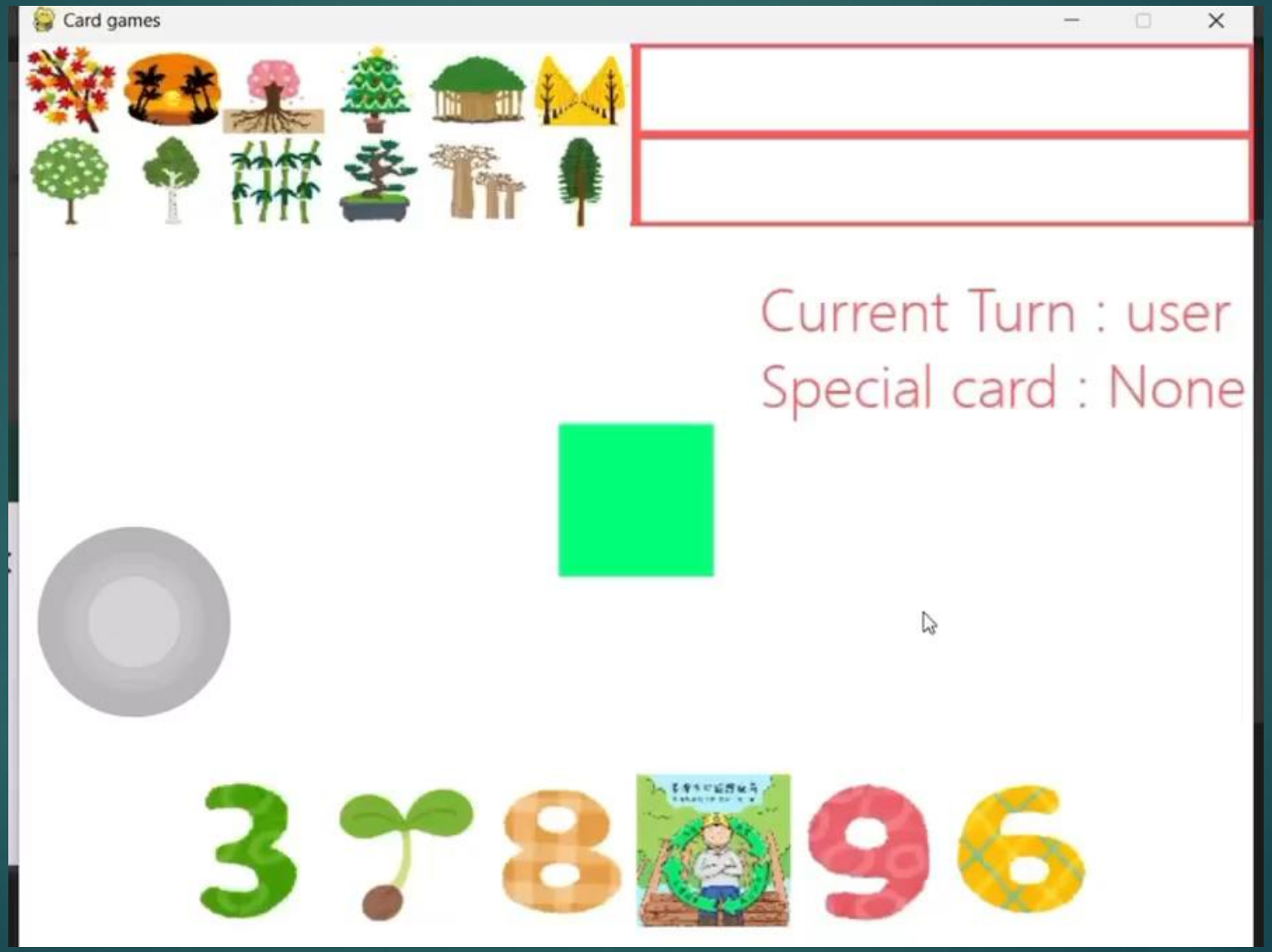

```
def hover(self):
    self.rect.y = self.position_y - ONMOUSE_OFFSET
    self.hover_status  = True
```

當滑鼠碰到手牌，手牌要上移

```
def cancel_hover(self):
    self.rect.topleft = (self.position_x, self.position_y)
    self.hover_status = False
```

上移後滑鼠碰到則不需要在上移;
若手牌在出牌區，滑鼠碰到也不須上移

滑鼠點擊->手牌移置出牌區



滑鼠點擊->手牌移置出牌區

```
# 當該卡牌被選中，根據他現在是第幾張被選中的牌來計算位置
def selected(self):
    global selected_num
    selected_num += 1
    self.select_no = selected_num
    self.rect.topleft = (SELECTED_X + (self.select_no - 1) * self.width, SELECTED_Y)
    self.hover_status = False
```

手牌被點擊後會移到出牌區。
卡牌位子根據出牌區現在有
幾張牌來計算x座標

```
def cancel_select(self):
    global selected_num
    self.rect.topleft = (self.position_x, self.position_y)
    self.select_no = 0
    selected_num -= 1
```

在出牌區點擊卡牌，卡牌會回
到手牌區，出牌區的牌數-1

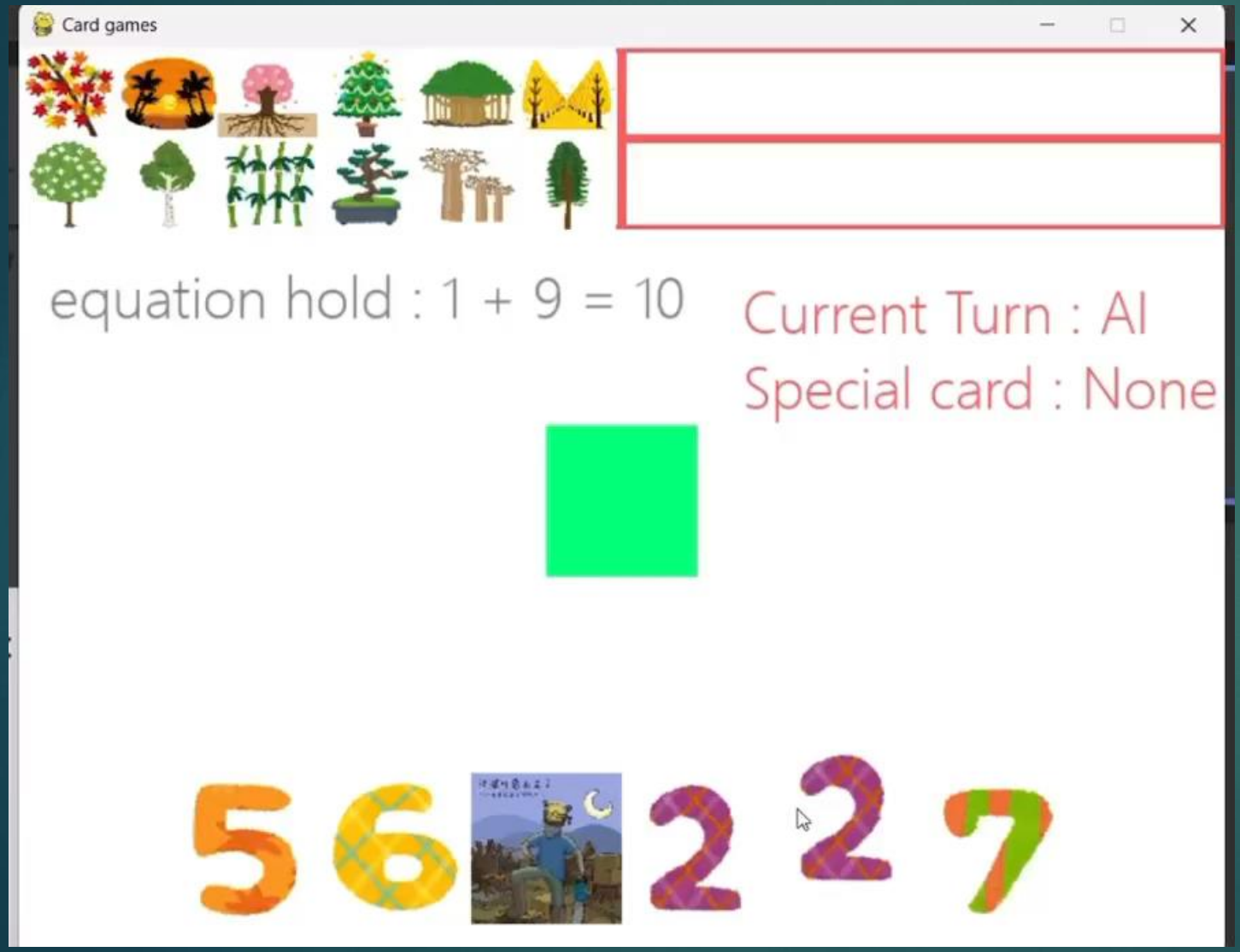
```
# 當從待出牌區移除卡牌時，可能會需要移動其他卡的位置
def cancel_select_helper(self):
    #if self.select_no:
    #self.rect.topleft = (SELECTED_X + (self.select_no - 1) * self.width, SELECTED_Y)
    if self.rect.x >= SELECTED_X + (self.select_no - 1) * self.width:
        self.rect.x -= 10
```

卡牌回到手牌區後，出牌區的
牌需重新計算位子

```
def show_play_button(cards, window_surface):
    # show button
    if any([card.select_no != 0 for card in cards]):
        play_rect = pygame.Rect(PLAY_BUTTON_X, PLAY_BUTTON_Y, PLAY_BUTTON_WIDTH, PLAY_BUTTON_HEIGHT)
        pygame.draw.rect(window_surface, TREE_AREA_COLOR, play_rect, 3)
        window_surface.blit(FONT.render("出牌", True, (128,128,128)), play_rect.topleft)
        return play_rect #有牌在出牌區=>把出牌的按鈕秀出來
```

若有牌在出牌區則要出示"出牌"按鈕

確認可否出牌



第一次:
出2、5、7
2+5=7 是正確數學式子，判斷可以出牌
出牌後顯示 equation hold:2+5=7

第二次:
出2、2、6
2+2!=6 是不正確數學式子，判斷不可以出牌
出牌後顯示 equation does not hold:2+5=7

確認可否出牌

```python
def play_card_check(all_cards):
    played_card = [card for card in all_cards if card.select_no != 0]

    if len(played_card) == 1:

        # draw a new tree
        if played_card[0].cate == 'get_tree':
            public_tree = [ tree for tree in all_trees if tree.holder == 'public']
            tree = public_tree.pop(random.randrange(len(public_tree)))
            tree.set_state('user')

        elif played_card[0].cate in ['ant', 'mouse']:
            # if ai has at least one tree
            if any([tree.holder == 'ai' for tree in all_trees]):
                global special_card
                special_card = played_card[0].cate

        ''' throw card and draw a card'''
        out_cards.append(played_card[0])
        played_card[0].kill()
        #print([card.cate for card in out_cards])
        play_card(1, all_cards, decks)
        return f'play card : {played_card[0].cate}'

    elif len(played_card) == 3:
        # if all cards are number
        if all([type(card.cate) == int for card in played_card]):
            played_card.sort(key = lambda x : x.cate)
            # the equation hold
            if played_card[0].cate + played_card[1].cate == played_card[2].cate:
                for card in played_card:
                    out_cards.append(card)
                    card.kill()
                play_card(3, all_cards, decks)
                return f'equation hold : {played_card[0].cate} + {played_card[1].cate} = {played_card[2].cate}'
            else:
                return 'the equation does not hold'
        else:
            return 'the cards must be all number card'
    return 'Invalid card numbers'
```

出一張牌:

有三種情況，分別為功能牌(得分)、功能牌(進攻和防守)、數字牌。皆可出牌。只有功能牌(得分)和功能牌(進攻)需做出相對的動作。

得分: 得到一棵樹

進攻: 攻擊對方的樹

出三張牌:

只有數字牌在這種情況下是有效的。所以要先判斷是否全為數字牌。判斷全為數字牌後要判斷是否是正確的數學式。首先將三張牌按照小到大順序排放，再將第一張+第二張，若等於第三張則判斷可出牌。

出非一或三張牌則判斷不可出牌

反制牌

```
def show_confirm_button(window_surface): #show是否要出反制牌
    confirm_button_width = 100
    confirm_button_height = 30
    confirm_rect = pygame.Rect(PLAY_BUTTON_X, PLAY_BUTTON_Y, confirm_button_width , confirm_button_height)
    pygame.draw.rect(window_surface, TREE_AREA_COLOR, confirm_rect, 2)
    window_surface.blit(CONFIRM_FONT.render('打出反制牌', True, (100,200,0)), confirm_rect)
    cancel_rect = pygame.Rect(PLAY_BUTTON_X + 150, PLAY_BUTTON_Y, confirm_button_width, confirm_button_height)
    pygame.draw.rect(window_surface, TREE_AREA_COLOR, cancel_rect, 3)
    text_position = cancel_rect
    text_position[0] += 20
    window_surface.blit(CONFIRM_FONT.render('不出牌', True, (128,128,128)), text_position)
```

一方打出功能牌(進攻)；另一方若有功能牌(防守)則可決定是否出牌。此時會出現"打出反制牌"、"不出牌"按下任一個按鈕則會執行對應動作。

洗牌

```
''' throw card and draw a card'''
out_cards.append(played_card[0])
played_card[0].kill()
#print([card.cate for card in out_cards])
play_card(1, all_cards, decks)
return f'play card : {played_card[0].cate}'
```

出牌後則將出的牌從牌堆裡kill掉，並加到out_cards

```
def refresh_decks():
    global out_cards
    global decks
    decks += out_cards
    for card in decks:
        card.select_no = 0
    out_cards = []
```

當牌堆沒牌時再把out_cards加入牌堆，
並將out_cards清空

需改進的地方

- 攻擊與防守的牌的功能可以更多樣化。

  現階段遊戲的攻擊只能把對方的樹變不見，但其實可以試試看把對方的樹變成自己的樹、讓對方的樹回去公共區等等，讓遊戲更刺激。

- 得分的方式

  目前得分是用random的方式隨機抽一個樹，較沒有互動感。可以將遊戲改成一開始看不到樹的種類，出功能牌(得分)的時候，將樹移到畫面中間，讓玩家自己選擇。

心得反思與未來展望

- 在一年級「創意遊戲式學習的發想與設計」課程中自我創作的桌遊，作品從無到有的製作過程是非常充實且有趣的，最後成品完成也感到無比的成就感。之後自主學習Python 程式設計，一開始也是懵懵懂懂、跌跌撞撞，一直在trial and error的過程，後來逐漸對程式設計產生興趣後，就想把自己的桌遊創作實現在電腦上，設計的過程也是相當燒腦，但我是樂在其中，從完成初步架構再逐步完善，越做越有成就感。
- 在完成線上版遊戲後，我可以明顯感覺到自己對程式的熟悉度以及邏輯思考、分析能力的提升。有時在看到一道數學題，腦中出現的反而是如何用程式碼解出答案(直接中毒)。很幸運的是，我們身處在這資訊世代，遇到不懂、不熟悉的程式碼可以直接上網查詢，節省了許多時間。
- 希望未來再多學習網頁程式設計，將單機版遊戲，升級為web線上版，讓更多人可以玩。