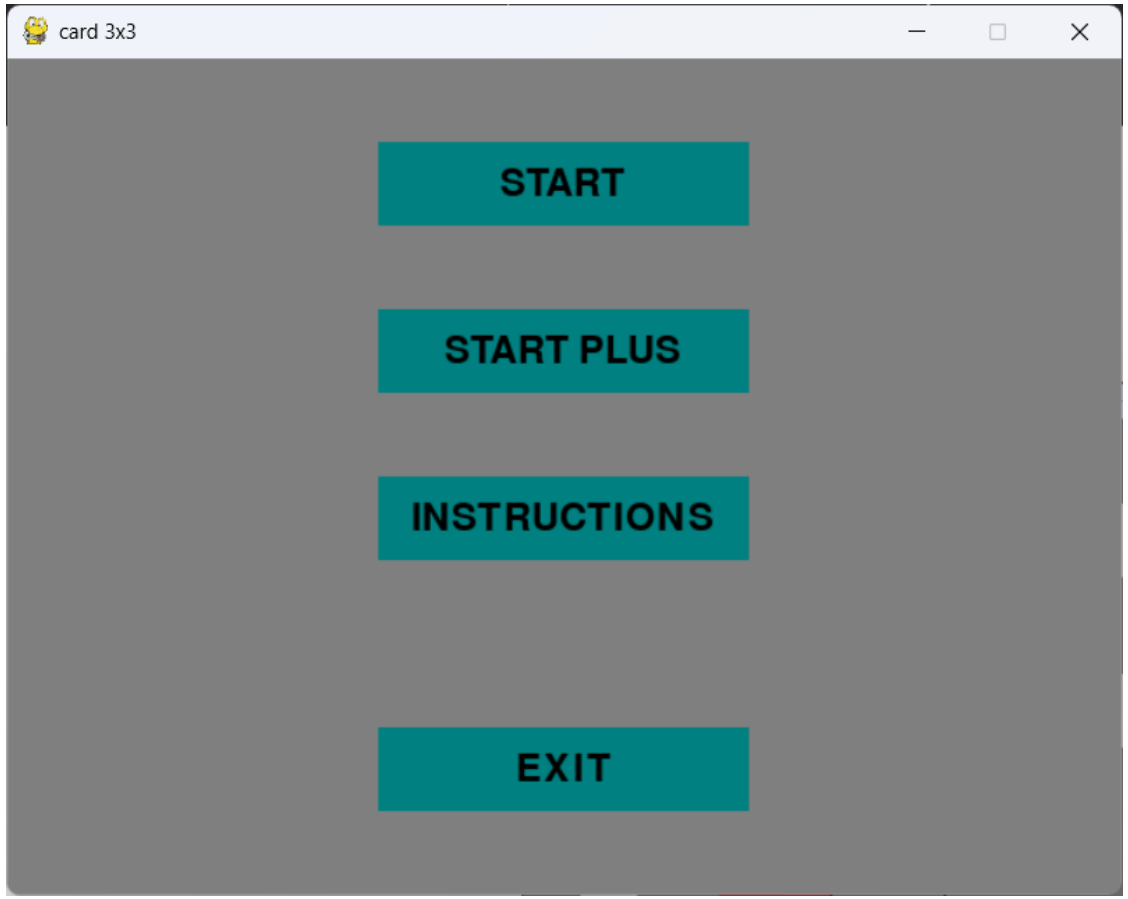
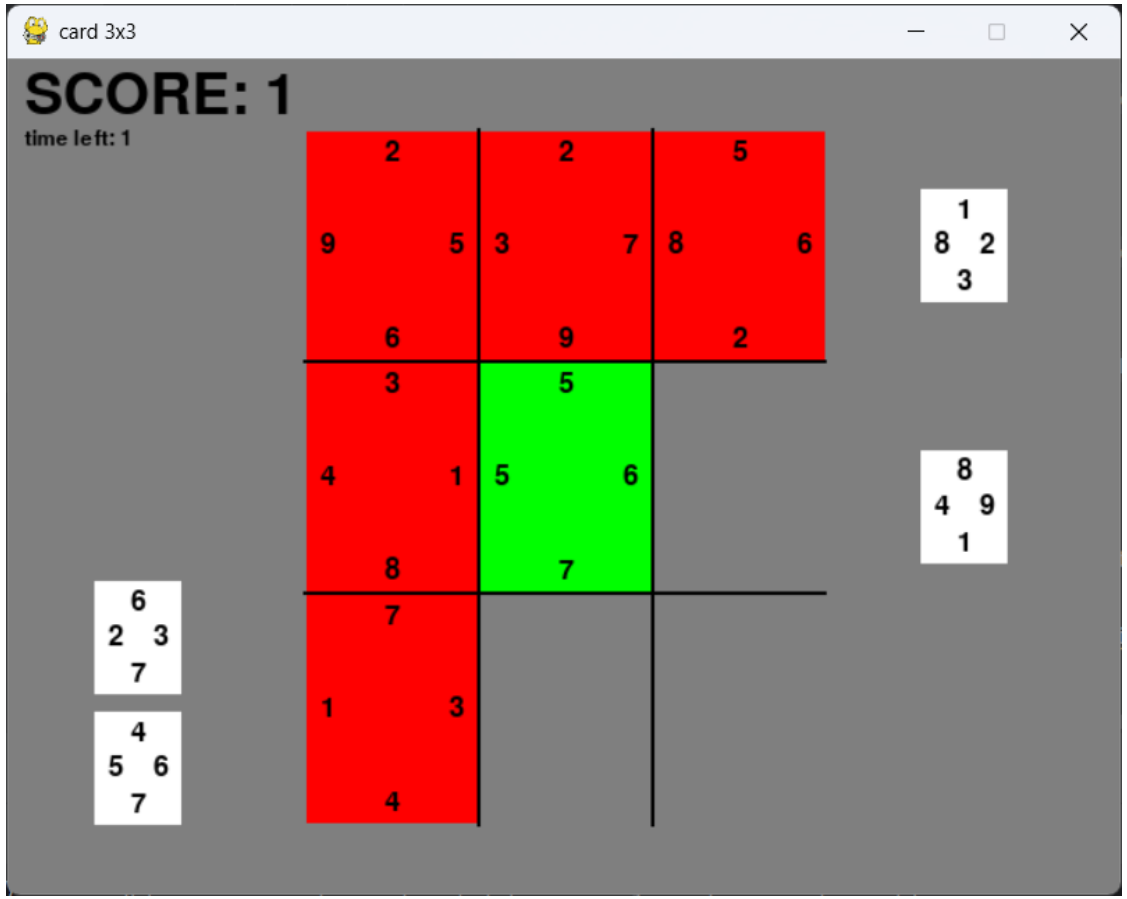
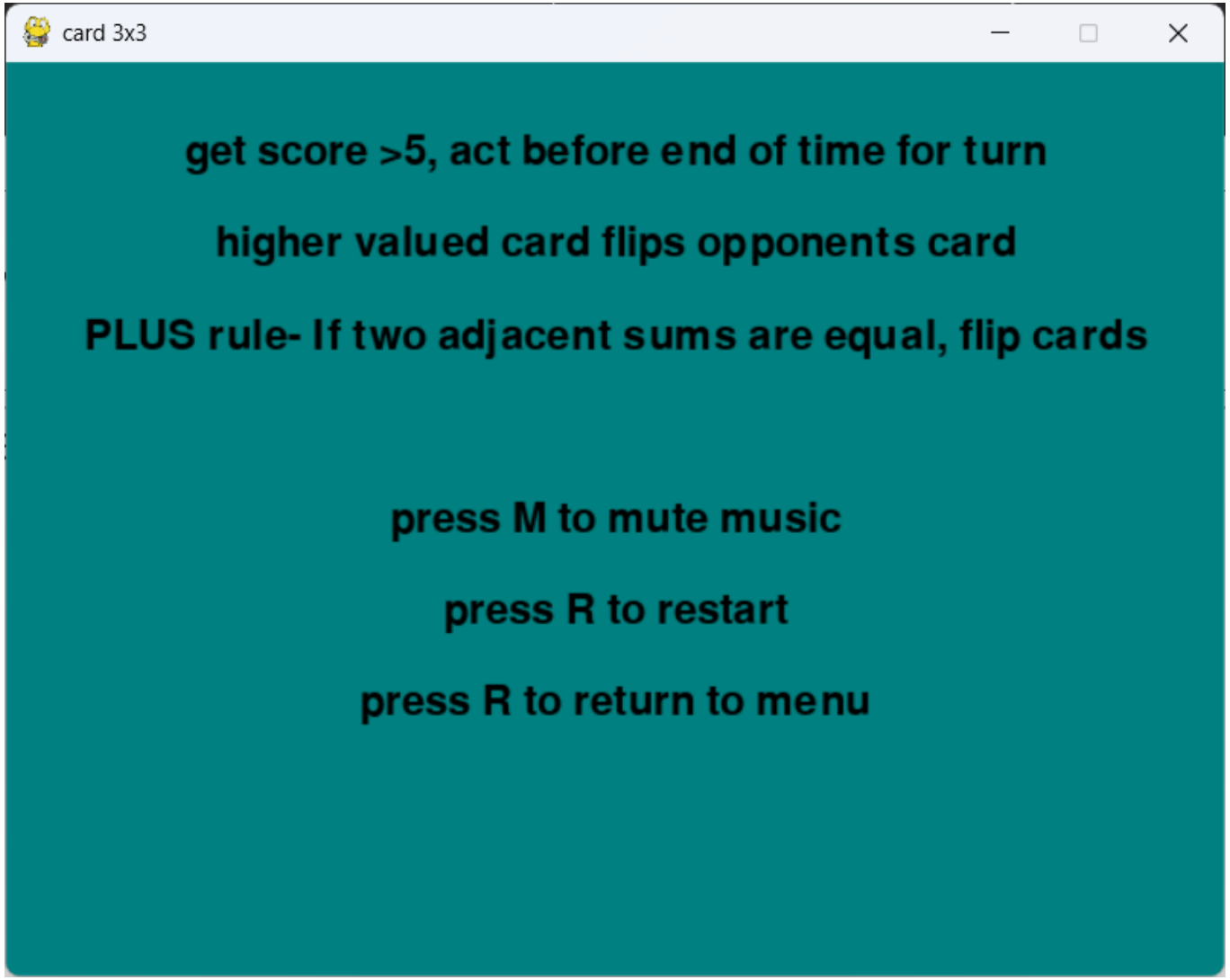
Card 3x3
Litwiński Dawid s27814



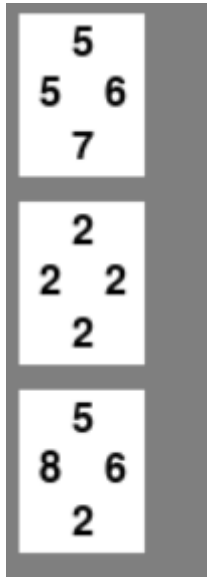


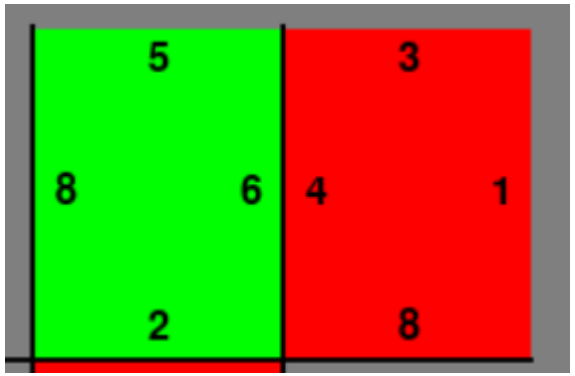
Rules


card 3x3
get score >5, act before end of time for turn
higher valued card flips opponents card
PLUS rule- If two adjacent sums are equal, flip cards
press M to mute music
press R to restart
press R to return to menu

Card

```
class Card:  3 usages  mirquLAP
    def __init__(self, ID, N, E, S, W, ir
        self.ID = ID
        self.N = N
        self.E = E
        self.S = S
        self.W = W
        self.img = img
        self.colour = colour
        self.is_hidden = is_hidden
```





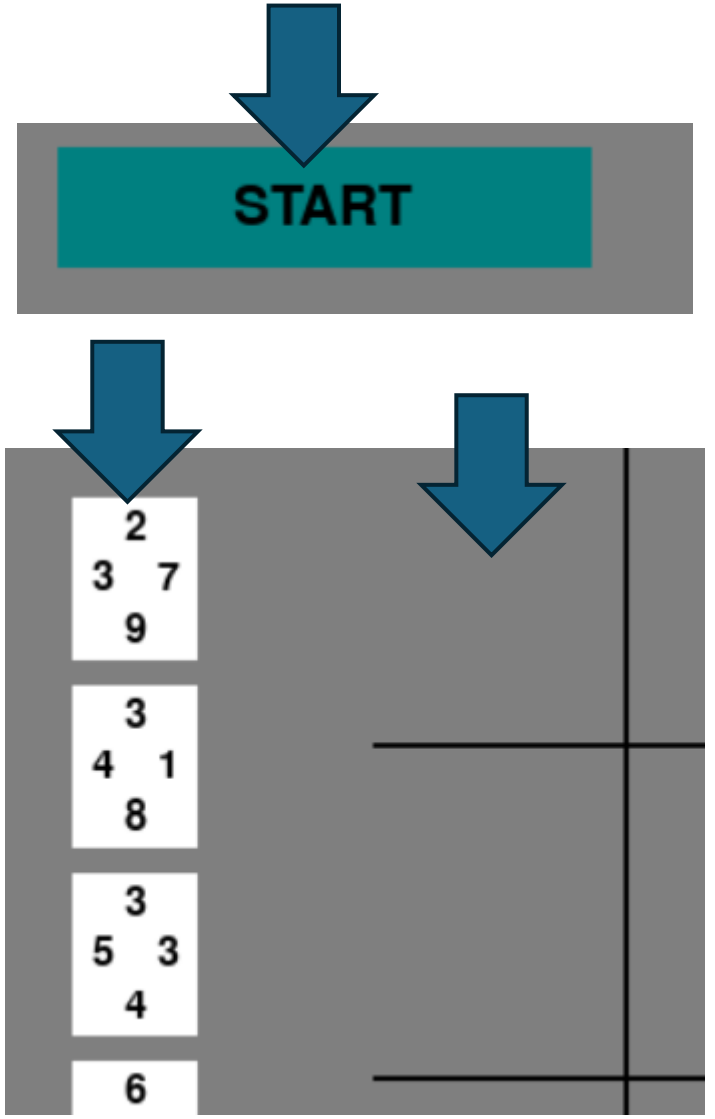
Button

```
class Button():  8 usages  mirquLAP
    def __init__(self, x, y, width, height):...

    def draw_card(self, surface, card):...

    def draw_empty(self, surface, colour):...

    def draw_text(self, surface, text_in, colour):...
```



Button use

```
if game_state == "left_place_card":
    for i, btn in enumerate(field_buttons):
        if cards_on_field[i] is None:
            if btn.draw_empty(DISPLAYSURF, BACKGROUND_COLOR):
                selected_spot = i
                game_state = "move_card"
        else:
            btn.draw_card(DISPLAYSURF, cards_on_field[i])
```

```
if game_state == "left_select_card":
    previous_human_move = True
    for i, btn in enumerate(left_buttons):
        if btn.draw_card(DISPLAYSURF, left_hand[i]):
            selected_card = i
            left_hand[i].colour = TEAL
            game_state = "left_place_card"
```

```
start_button = Button(WINDOW_WIDTH // 3, WINDOW_HEIGHT // 10, WINDOW_WIDTH // 3, WINDOW_HEIGHT // 10)
start_plus_button = Button(WINDOW_WIDTH // 3, (WINDOW_HEIGHT // 10) * 3, WINDOW_WIDTH // 3, WINDOW_HEIGHT // 10)
guide_button = Button(WINDOW_WIDTH // 3, (WINDOW_HEIGHT // 10) * 5, WINDOW_WIDTH // 3, WINDOW_HEIGHT // 10)
exit_button = Button(WINDOW_WIDTH // 3, (WINDOW_HEIGHT // 10) * 8, WINDOW_WIDTH // 3, WINDOW_HEIGHT // 10)
if start_button.draw_text(DISPLAYSURF, text_in: "START", TEAL):
    game_state = "left_select_card"
elif start_plus_button.draw_text(DISPLAYSURF, text_in: "START PLUS", TEAL):
    rule_plus = True
    game_state = "left_select_card"
```

Logic

- game_state controls game state
- Logic tied to FPS

```
if start_button.draw_text(DISPLAYSURF, text_in: "START", TEAL):
    game_state = "left_select_card"
elif start_plus_button.draw_text(DISPLAYSURF, text_in: "START PLUS", TEAL):
    rule_plus = True
    game_state = "left_select_card"
elif guide_button.draw_text(DISPLAYSURF, text_in: "INSTRUCTIONS", TEAL):
    game_state = "instructions"
elif exit_button.draw_text(DISPLAYSURF, text_in: "EXIT", TEAL):
    pygame.quit()
    sys.exit()
else:
```

Opponent’s decision making

- Attempts N random moves for current board, chooses best

```
if simulate_counter < DIFFICULTY:
    simulate=True
    if simulate_counter == 0:
        previous_cards_on_field = deepcopy(cards_on_field)
        previous_score=score
    simulate_counter += 1
    if previous_score-score > best_advantage:
        best_spot_to_move = selected_spot
        best_card_to_move = selected_card
    cards_on_field=deepcopy(previous_cards_on_field)
    selected_card = random.choice(available_cards)
    selected_spot = random.choice(available_spots)
    if simulate_counter == 1:
        best_card_to_move, best_spot_to_move = selected_card, selected_spot
    cards_on_field[selected_spot] = copy(right_hand[selected_card])
    cards_on_field[selected_spot].colour = RED
    game_state = "check_move_results"
else:
    simulate=False
    cards_on_field = deepcopy(previous_cards_on_field)
```

End