EEE 202: Computer Programming II Project
Mogoa Duncan Begi: J174/1057/2022
Mwenda Victor Murithi: J174/1037/2022

# Product Documentation

Grid Duel (ft. Python's Playground) . Simple Computer Game with GUI.

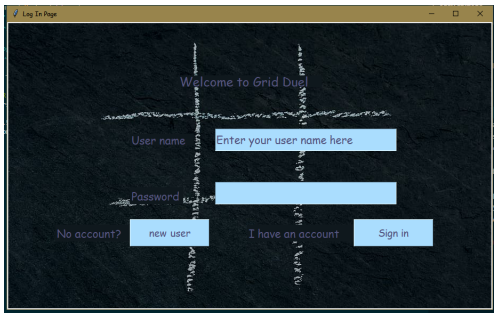
## Introduction

The following is our description of the project, having reached completion.

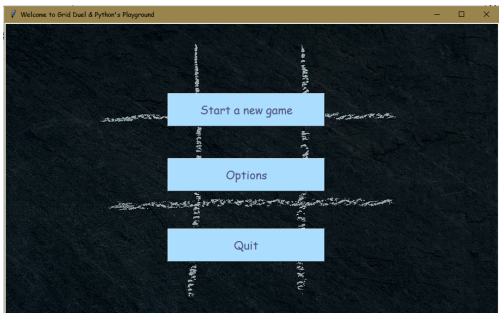
## Usage.

The Project's UI is shown below:

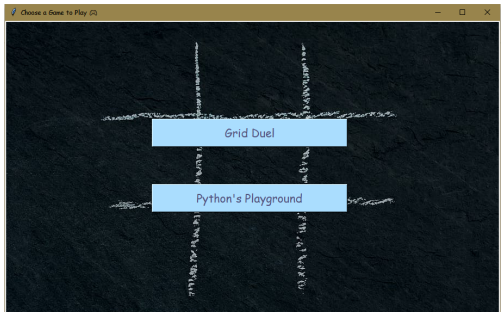
Sign In page



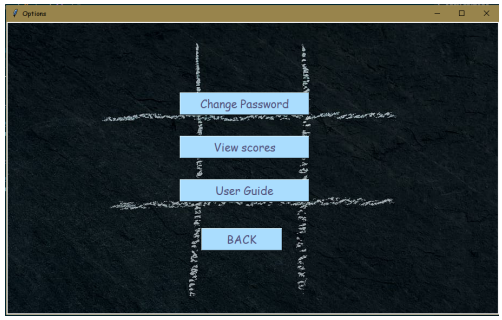

Home Page



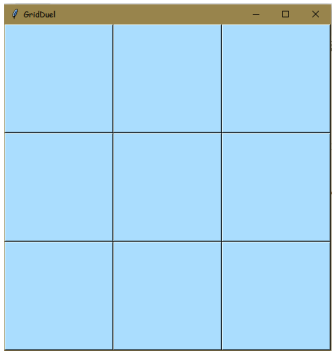

Choose a Game page.



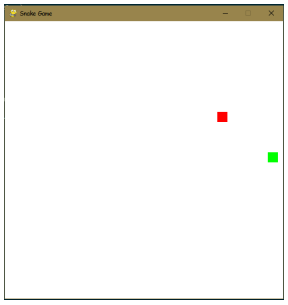

Options Page.




Grid Duel.




Python's Playground.




## Code Base.

Find the All the Source Code for our Project at the repository [here](): 

# Maingui.py

```
"""
Grid Duel (ft. Python's Playground) . Simple Computer Game with GUI.

Mogoa Duncan Begi: J174/1057/2022
Mwenda Victor Murithi: J174/1037/2022


Our project consists of the following modules:
    maingui.py          --> This script. initializes the first window instance
    v1log_in.py         --> Contains the Class 'LogIn', that defines the Login page
    v2game_home.py      --> Contains the Class 'GameHome' that defines the Home page
    v3start_game.py     --> Contains the Class 'StartGame' that defines the Choose-a-game
page
    v4game_grid.py      --> Contains the Class 'GridDuel' that defines game 1 : Grid Duel
    snake2.py           --> Contains functions that define game 2 : Python's Playground
    v5options.py        --> Contains the Class 'Options' that defines the Options Page

Our Project contains the following assets:
    wallpaper1.png      --> The image file used as the window background in Log in, Home,
Choose-a-game and Options pages
    UserDatabase.db     --> The database containing our user credentials, i.e the names and
passwords
    scoreDatabase.db    --> The database that stores the wins and draws everytime game 1
(Grid duel) is ran

"""
from v1log_in import LogIn
class App:
    def __init__(self):
        # Initialize the main window
          self.login_window = LogIn()
if __name__ == '__main__':
     app = App()
```

## v1Log_in.py

```
from tkinter import *
from tkinter import messagebox  #import message box library
from v2game_home import GameHome
from ourUserDatabase import *
class LogIn():
    def __init__(self):
        self.gui()
        self.run()

    def gui(self ):
        #This function defines our window, and all the widgets it'll contain
        #Window Widget
        self.w1 = Tk()  #the tkinter window widget, w1, will be an instance attribute, hence the "self."
        self.w1.geometry('910x539')
        self.w1.title('Login Page')
        self.w1.maxsize(910,539)    #We set a  max size to prevent scaling of the window above our background image size.
        #A Canvas widget, for our window background
        self.background = PhotoImage(file='wallpaper1.png')
        self.bcanvas = Canvas(self.w1, width=910, height=539)
        self.bcanvas.pack( expand=True, fill='both', anchor='center')
        self.bcanvas.create_image(0,0, image=self.background,anchor='nw')   #set image in canvas
        #Entry boxes
        self.username_entry = Entry(self.w1, fg = "#55557f", bg ='#aaddfe', font = ( "Comic Sans MS", 16))
        self.username_entry.place(x = 390, y = 200, width = 340, height = 42)
        self.username_entry.insert(0, "Enter your user name here")  #Place holder text
        self.password_entry = Entry(self.w1, fg = "#55557f", bg ='#aaddfe', font = ( "Comic Sans MS", 16), show='*')
        self.password_entry.place(x = 390, y = 300, width = 340, height = 42)
        ## self.password_entry.insert(0, "Enter your Password here")    #Placeholder text
        #Buttons
        self.newUser_button = Button(self.w1, text = "new user", fg = "#55557f", font = ( "Comic Sans MS", 14), activebackground='#aaddfe',
bg='#aaddfe' )
        self.newUser_button.place(x = 230, y = 370, width = 150, height = 52)
        self.newUser_button['command'] = self.create_newuser
        self.signIn_button = Button(self.w1, text = "Sign in", fg = "#55557f", font = ( "Comic Sans MS", 14),activebackground='#aaddfe',
bg='#aaddfe')
        self.signIn_button.place(x = 650, y = 370, width = 150, height = 52)
        self.signIn_button['command'] = self.signIn
        #labels
        self.label_username = Label(self.w1, text = "User name", anchor='w', fg = "#55557f",bg='#0f181d' ,font = ( "Comic Sans MS", 16))
        self.label_username.place(x = 230, y = 210, width = 120, height = 22)
        self.label_password = Label(self.w1, text = "Password", anchor='w', fg = "#55557f",bg='#0f181d', font = ( "Comic Sans MS", 16))
        self.label_password.place(x = 230, y = 310, width = 100, height = 32)
```

```
        self.label_noAccount = Label(self.w1, text = "No account?", anchor='w', fg = "#55557f",bg='#0f181d', font = ( "Comic Sans MS", 16))
        self.label_noAccount.place(x = 90, y = 380, width = 140, height = 32)
        self.label_ihaveAccount = Label(self.w1, text = "I have an account", anchor='w', fg = "#55557f",bg='#0f181d', font = ( "Comic Sans MS",
16))
        self.label_ihaveAccount.place(x = 450, y = 380, width = 200, height = 32)
        self.label_welcome = Label(self.w1, text = "Welcome to Grid Duel", anchor='w', fg = "#55557f",bg='#0f181d', font = ( "Comic Sans MS",
18))
        self.label_welcome.place(x = 320, y = 100, width = 270, height = 22)
    #run the window
    def run (self):
        self.w1.mainloop()
    #destroy the window
    def remove (self):
        self.w1.destroy()

    #create a new user by adding the name and password to the db
    def create_newuser(self):

        newUserName = self.username_entry.get()
        newUserPassword = self.password_entry.get()
        #check if the credentials already exist, and proceed normally if they do
        if self.check(newUserName, newUserPassword):
            self.remove()
            #open the next window (game_home)
            self.signin_window = GameHome()
        else:
            add_data(newUserName,newUserPassword)
            self.remove()
            self.signin_window = GameHome()

    def signIn(self):
        #verify that the details entered are in the db
        userName = self.username_entry.get()
        userPassword = self.password_entry.get()
        if self.check(userName, userPassword):
            self.remove()
            #opens the next window (game_home)
            self.signin_window = GameHome()
        else:
            messagebox.showerror(title='Access Denied', message='Credentials not found!')
   def check(self, a,b):
      return query_database(a,b)
```

# V2game_home.py

```
from tkinter import *
from v3start_game import StartGame
from v5options import Options
class GameHome():
    def __init__(self):
        self.gui()
        self.run()
    def run (self):
        self.w1.mainloop()
    def remove (self):
        self.w1.destroy()
    def gui(self):
        #This function defines our window, and all the widgets it'll contain
        #Window Widget
        self.w1 = Tk()
        self.w1.geometry('910x539')
        self.w1.title("Welcome to Grid Duel & Python's Playground")
        self.w1.maxsize(910,539)
        #A Canvas widget, for our window background
        self.background = PhotoImage(file='wallpaper1.png')
        self.bcanvas = Canvas(self.w1, width=910, height=539)
        self.bcanvas.pack( expand=True, fill='both', anchor='center')
        self.bcanvas.create_image(0,0, image=self.background,anchor='nw')
        #Our 3 buttons for start game, options, and quit
        self.startGame_button = Button(self.w1, text = "Start a new game", fg = "#55557f", font = ("Comic Sans MS", 16), activebackground='#aaddfe', bg='#aaddfe')
        self.startGame_button.place(x = 300, y = 130, width = 290, height = 62)
        self.startGame_button['command'] = self.start_game
        self.options_button = Button(self.w1, text = "Options", fg = "#55557f", font = ("Comic Sans MS", 16), activebackground='#aaddfe', bg='#aaddfe')
        self.options_button.place(x = 300, y = 250, width = 290, height = 62)
        self.options_button['command'] = self.open_options
        self.quit_button = Button(self.w1, text = "Quit", fg = "#55557f", font = ("Comic Sans MS", 16), activebackground='#aaddfe', bg='#aaddfe')
        self.quit_button.place(x = 300, y = 380, width = 290, height = 62)
        self.quit_button['command'] = self.quit_game

    def start_game(self):
        #opens the next window (start_game)
        self.remove()
        self.gamehome_window = StartGame()

    def open_options(self):
        #opens the options window
        self.remove()
        self.options_window = Options()

    def quit_game(self):
        self.remove()
```

## V3start_game.py

```
from tkinter import *
from v4game_grid import GridDuel
from snake2 import start_game

class StartGame():
    def __init__(self):
        self.gui()
        self.run()
    def run (self):
        self.w1.mainloop()
    def remove (self):
        self.w1.destroy()
    def gui(self):
        #This function defines our window, and all the widgets it'll contain
        #Window Widget
        self.w1 = Tk()
        self.w1.geometry('910x539')
        self.w1.title('Choose a Game to Play 🎮')
        self.w1.maxsize(910,539)
        #A Canvas widget, for our window background
        self.background = PhotoImage(file='wallpaper1.png')
        self.bcanvas = Canvas(self.w1, width=910, height=539)
        self.bcanvas.pack( expand=True, fill='both', anchor='center')
        self.bcanvas.create_image(0,0, image=self.background,anchor='nw')
        #Two buttons for single player and multiplayer.
        self.singleplay_button = Button(self.w1, text = "Grid Duel", fg = "#55557f", font =( "Comic Sans MS", 16), activebackground='#aaddfe',
bg='#aaddfe')
        self.singleplay_button.place(x = 270, y = 180, width = 360, height = 52)
        self.singleplay_button['command'] = self.game_gridDuel
        self.Twoplay_button = Button(self.w1, text = "Python's Playground", fg = "#55557f", font =( "Comic Sans MS", 16),
activebackground='#aaddfe', bg='#aaddfe')
        self.Twoplay_button.place(x = 270, y = 300, width = 360, height = 52)
        self.Twoplay_button['command'] = self.game_pythonsPlayground

    def game_gridDuel(self):
        self.remove()
        self.game = GridDuel()

    def game_pythonsPlayground(self):
        self.remove()
        start_game(1)
```

## V4options.py

```
from tkinter import *
from tkinter import messagebox  #import message box libray
from ourScoreDatabase import *
#initialize the database connection once
scoreDB = sqlite3.connect('ScoreDatabase.db')
score_cursor = scoreDB.cursor()
class Options():
    def __init__(self):
        self.gui()
        self.run()
    def run (self):
        self.w1.mainloop()
    def remove (self):
        self.w1.destroy()
    def gui(self):
        #This function defines our window, and all the widgets it'll contain
        #Window Widget
        self.w1 = Tk()
        self.w1.geometry('910x539')
        self.w1.title('Options')
        self.w1.maxsize(910,539)
        #A Canvas widget, for our window background
        self.background = PhotoImage(file='wallpaper1.png')
        self.bcanvas = Canvas(self.w1, width=910, height=539)
        self.bcanvas.pack( expand=True, fill='both', anchor='center')
        self.bcanvas.create_image(0,0, image=self.background,anchor='nw')
        #Buttons
        self.changepass_button = Button(self.w1, text = "Change Password", fg = "#55557f", font = ("Comic Sans MS",
16), activebackground='#aaddfe', bg='#aaddfe')
        self.changepass_button.place(x = 320, y = 130, width = 240, height = 42)
        self.changepass_button['command'] = self.change_password
        self.help_button = Button(self.w1, text = "User Guide", fg = "#55557f", font = ("Comic Sans MS", 16),
activebackground='#aaddfe', bg='#aaddfe')
        self.help_button.place(x = 320, y = 290, width = 240, height = 42)
        self.help_button['command'] = self.open_userguide
```

```
        self.back_button = Button(self.w1, text = "BACK", fg = "#55557f", font = ("Comic Sans MS", 16),
activebackground='#aaddfe', bg='#aaddfe')
        self.back_button.place(x = 360, y = 380, width = 150, height = 42)
        self.back_button['command'] = self.back_Options
        self.scores_button = Button(self.w1, text = "View scores", fg = "#55557f", font = ("Comic Sans MS", 16),
activebackground='#aaddfe', bg='#aaddfe')
        self.scores_button.place(x = 320, y = 210, width = 240, height = 42)
        self.scores_button['command'] = self.open_scores

    def change_password(self):
        #opens login window
        self.remove()
        from v1log_in import LogIn
        self.login_window = LogIn()

    def open_scores(self):
        #shows a message box for wins, loses, draws
        messagebox.showinfo(title="Scores ", message=get_scores()) #see the ourscoresdatabse.py file

    def open_userguide(self):
        #opens a message box for Options help
        messagebox.showinfo(title="User Guide", message="""✨Welcome to Grid duel |now with Snake!|  ✨\nIts super
easy, just align Xs or Os in a straight line 😊!\nAre you ready to become the ultimate champion? 😎 """)

    def back_Options(self):
        #opens the game_home window and closes Options
        self.remove()
        from v2game_home import GameHome    #prevent circular import traceback
        self.signin_window = GameHome()
```

# V4game_grid.py

```
from tkinter import *
from tkinter import messagebox
from v2game_home import *
from ourScoreDatabase import *
# Import SQLite for database functionality
import sqlite3

# Initialize the database connection once
scoreDB = sqlite3.connect('ScoreDatabase.db')
score_cursor = scoreDB.cursor()

# Define our class
class GridDuel:
    def __init__(self):
        # Initialize the main window
        self.w1 = Tk()

self.w1.title("GridDuel")

        # Initialize the game variables
        self.current_player = 'X'
        self.board = [' ' for _ in range(9)]

        # Create buttons for the game grid
        self.buttons = [None] * 9
        for i in range(9):
            row, col = divmod(i, 3)
            self.buttons[i] = Button(self.w1, text='', width=12, height=5,

font=("Comic Sans MS", 16),

bg='#aaddfe', activebackground='#aaddfe',

command=lambda i=i: self.make_move(i))

self.buttons[i].grid(row=row, column=col)

        # Display user guide
        self.userguide()

        # Run the main loop
        self.run()

    def make_move(self, i):
        # Handle player moves
        if self.board[i] == ' ':
            self.board[i] = self.current_player

self.buttons[i].config(text=self.current_player)

            # Check for a win
            if self.check_win(self.current_player):

messagebox.showinfo("Grid Duel", f"Player {self.current_player} wins!")

                # Update the scores database

update_scores('wins')

                # Reset the game
                if messagebox.askyesno(title='Game Over', message='Play again?'):

self.reset_game()
                else:

self.remove()
                    self.back()

            # Check for a draw
            elif ' ' not in self.board:

messagebox.showinfo("Grid Duel", "It's a draw!")

update_scores('draws')

                # Reset the game
                if messagebox.askyesno(title='Game Over', message='Play again?'):

self.reset_game()
                else:

self.remove()
                    self.back()
            else:
                # Switch players

self.current_player = 'O' if self.current_player == 'X' else 'X'

    def userguide(self):
        # Display a user guide message box

messagebox.showinfo(title="User Guide", message="""✨just align Xs or Os in a straight line 😊!""")

    def check_win(self, player):
        # Check if a player has won
        winning_combinations = [(0, 1, 2), (3, 4, 5), (6, 7, 8),

(0, 3, 6), (1, 4, 7), (2, 5, 8),

(0, 4, 8), (2, 4, 6)]
        for combo in winning_combinations:
            if all(self.board[i] == player for i in combo):
                return True
        return False

    def reset_game(self):
        # Reset the game board and player
        for i in range(9):
            self.board[i] = ' '

self.buttons[i].config(text='')
        self.current_player = 'X'

    def remove(self):
        # Close the game window
        self.w1.destroy()

    def back(self):
        # Go back to the main menu
        self.back_to_main = GameHome()

    def run(self):
        # Run the main loop
        self.w1.mainloop()
```

## ourUserDatabase.py

```
# Import the SQLite module
import sqlite3

# Connect to our database
ourDB = sqlite3.connect('UserDatabase.db')

# Initialize a cursor for database operations
a_cursor = ourDB.cursor()

# Function to add records to our database
def add_data(name, password):
    # Execute an SQL command to insert data
into the 'users' table
    a_cursor.execute('''INSERT INTO users
(name, password) VALUES (?, ?)''', (name,
password))
    # Commit the changes to the database
    ourDB.commit()

# Function to query the database for the
presence of a user record
def query_database(name, password):
    # Execute an SQL command to select data
from the 'users' table based on name and
password
    a_cursor.execute('''SELECT * FROM users
WHERE name = ? AND password = ?''', (name,
password))
    # Fetch one result
    result = a_cursor.fetchone()
    # Check if a record was found
    if result is not None:
        return True
    else:
        return False

# Function to close the database connection
def close_database_connection():
    # Close the database connection
    ourDB.close()
    return

# Function to remove records from our database
def remove_data(name):
    # Execute an SQL command to delete data
from the 'users' table based on name
    a_cursor.execute('''DELETE FROM users
WHERE name = ?''', (name,))
    # Commit the changes to the database
    ourDB.commit()
```

## ourScoreDatabase.py

```
import sqlite3
scoreDB = sqlite3.connect('ScoreDatabase.db')
score_cursor = scoreDB.cursor()

def initialize_score_database():    #call this
only to ensure our database is properly set up
    score_cursor.execute("""
        CREATE TABLE IF NOT EXISTS scores (
            id INTEGER PRIMARY KEY,
            wins INTEGER DEFAULT 0,
            draws INTEGER DEFAULT 0
        )
        """)
    scoreDB.commit()

def get_scores():
    score_cursor.execute('SELECT wins, draws
FROM scores LIMIT 1')
    row = score_cursor.fetchone()
    if row is not None:
        return 'Wins : ' + str(row[0]) + ',
Draws : ' + str(row[1])
    else:
        return 'Wins : 0 , Draws : 0'

def update_scores(result):
    # Check if the row with id = 1 already
exists
    score_cursor.execute('SELECT 1 FROM scores
WHERE id = 1')
    existing_row = score_cursor.fetchone()

    if existing_row:
        # Row with id = 1 exists, update the
wins or draws count
        if result == 'wins':
            score_cursor.execute('UPDATE
scores SET wins = wins + 1 WHERE id = 1')
        elif result == 'draws':
            score_cursor.execute('UPDATE
scores SET draws = draws + 1 WHERE id = 1')
    else:
        # Row with id = 1 does not exist,
insert a new row
        if result == 'wins':
            score_cursor.execute('INSERT INTO
scores (id, wins) VALUES (1, 1)')
        elif result == 'draws':
            score_cursor.execute('INSERT INTO
scores (id, draws) VALUES (1, 1)')

    scoreDB.commit()
def reset_scores():
    score_cursor.execute('UPDATE scores SET
wins = 0, draws = 0 WHERE id = 1')
    scoreDB.commit()
def close_score_database_connection():
    scoreDB.close()
```

## Snake2.py

```
import pygame, sys, random
from tkinter import messagebox

def start_game(difficulty):
    # Initialize Pygame
    pygame.init()

    # Constants
    WIDTH, HEIGHT = 550, 550
    GRID_SIZE = 20
    GRID_WIDTH = WIDTH // GRID_SIZE
    GRID_HEIGHT = HEIGHT // GRID_SIZE

    # Colors
    WHITE = (255, 255, 255)
    GREEN = (0, 255, 0)
    RED = (255, 0, 0)

    # Initialize the screen
    screen = pygame.display.set_mode((WIDTH,
HEIGHT))
    pygame.display.set_caption("Snake Game")

    # Initialize the snake
    snake = [(GRID_WIDTH // 2, GRID_HEIGHT //
2)]
    snake_direction = (1, 0)

    # Initialize the food
    food = (random.randint(0, GRID_WIDTH - 1),
random.randint(0, GRID_HEIGHT - 1))

    # Game loop
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()

            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_UP
and snake_direction != (0, 1):
                    snake_direction = (0, -1)
                if event.key == pygame.K_DOWN
and snake_direction != (0, -1):
                    snake_direction = (0, 1)
                if event.key == pygame.K_LEFT
and snake_direction != (1, 0):
                    snake_direction = (-1, 0)
                if event.key == pygame.K_RIGHT
and snake_direction != (-1, 0):
                    snake_direction = (1, 0)

        # Move the snake
        new_head = (snake[0][0] +
snake_direction[0], snake[0][1] +
snake_direction[1])
        snake.insert(0, new_head)

        # Check for collisions
        if snake[0] == food:
            food = (random.randint(0,
GRID_WIDTH - 1), random.randint(0, GRID_HEIGHT
- 1))
        else:
            snake.pop()

        if (
            new_head[0] < 0 or new_head[0] >=
GRID_WIDTH or
            new_head[1] < 0 or new_head[1] >=
GRID_HEIGHT or
            new_head in snake[1:]
        ):
            show_game_over_popup(difficulty,
len(snake))

        # Draw the background
        screen.fill(WHITE)

        # Draw the food
        pygame.draw.rect(screen, RED, (food[0]
* GRID_SIZE, food[1] * GRID_SIZE, GRID_SIZE,
GRID_SIZE))

        # Draw the snake
        for segment in snake:
            pygame.draw.rect(screen, GREEN,
(segment[0] * GRID_SIZE, segment[1] *
GRID_SIZE, GRID_SIZE, GRID_SIZE))

        # Update the display
        pygame.display.update()

        # Delay to control the game speed
based on difficulty
        pygame.time.delay(100 // difficulty)

def show_game_over_popup(difficulty, score):
    result =
messagebox.askretrycancel(title="Game Over",
message=f"You hit the wall or
fouled!\nRetry?")
    if result:
        if
messagebox.askyesno(title='Difficulty',message
='Retry at a harder level?') :
            start_game(3)
        else:
            start_game(1)
    else:
        pygame.quit()
        sys.exit()
```