19.06.2024

PROJEKT:

Program do analizy danych z gry

LABOLATORIUM JĘZYKI SKRYPTOWE {Python}

Artur Lisowski
Mateusz Drypa

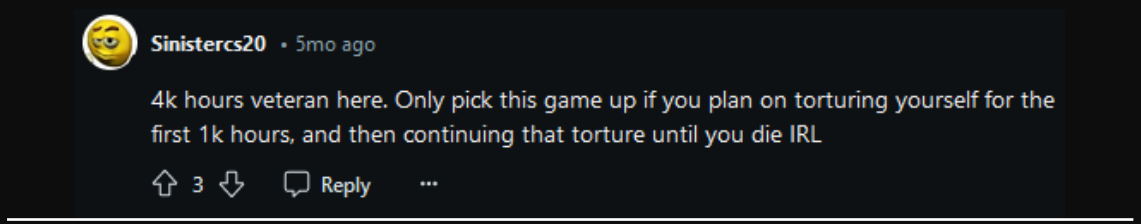
1. Wstep do projektowu

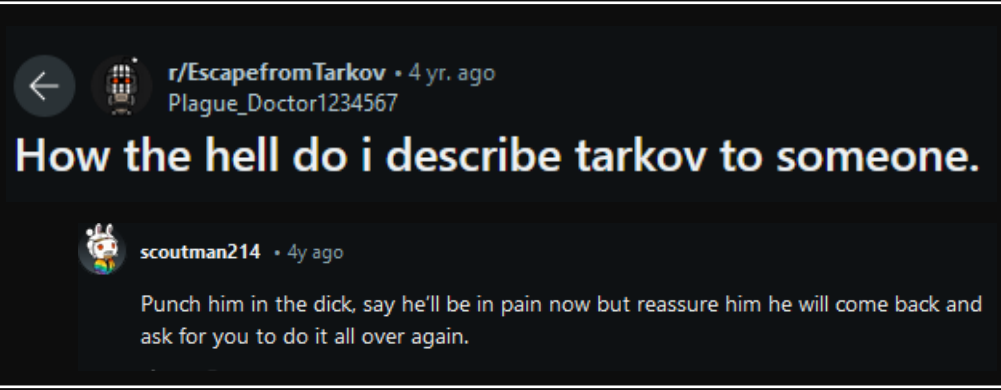
Projekt "GPSaviour" to autorski pomysł realizujący projekt z zajęć laboratoryjnych z przedmiotu Języki skryptowe. Celem programu jest wspomaganie graczy w grze "Escape from Tarkov" poprzez wyznaczanie współrzędnych na mapie na podstawie analizy zrzutów ekranu.

Opis gry

"Escape from Tarkov" to realistyczna, strzelanka FPS, w której określenie własnej pozycji na mapie jest kluczowe dla chociażby odrobiny poczucia ‘wygranej’. Gracz musi nawigować po rozległych, realistycznych oraz skomplikowanych terenach dążąc do punktu ucieczki. W międzyczasie wykonuje różne misje, konkurując z innymi o przeżycie i najlepsze zdobycze. Często próbując odnaleźć się w gęstym lesie czy zniszczonym mieście, jedyne co czeka na gracza to huk w tle oraz ekran informujący o śmierci postaci (i delikatny zawał).

Dobrze ilustrują to opinie graczy:





Nawigacja jest szczególnie trudna dla nowych graczy, sprawiając, że pierwsze kilkadziesiąt godzin gry mało ma wspólnego z przyjemnością, dla takich osób kierowany będzie ten program.

Opis projektu

Narzędzie " GPSaviour" analizuje zrzuty ekranu wykonane w grze, aby wyznaczyć współrzędne postaci gracza na mapie. Na podstawie nazwy pliku zrzutu ekranu, która zawiera zaszyfrowane współrzędne, narzędzie wyodrębnia te informacje i zapisuje do plików txt potrzebne informacje.

Projekt korzysta z pliku konfiguracyjnego map_config.json, który zawiera dane dotyczące różnych map używanych w grze. Plik ten definiuje parametry takie jak granice mapy w grze, granice mapy faktyczne, wynikające z plików, oraz wymiary mapy jako pixele.

Projekt używa map zapisanych w formacie SVG, które są łatwe do skalowania i manipulacji. Pliki SVG są konwertowane na format PNG za pomocą biblioteki cairosvg, co pozwala na wyświetlanie ich w aplikacji.

Przeprowadzenie odpowiednich obliczeń pozwala precyzyjnie przeliczyć współrzędne z gry na współrzędne w pliku SVG mapy, program następnie dodaje punkt gdzie znajduje się gracz i wyświetla go w interfejsie użytkownika, umożliwiając graczom łatwe śledzenie swojej lokalizacji.

Obecny stan projektu

Projekt znajduje się we wczesnej fazie rozwoju. Aktualnie narzędzie w pełni obsługuje tylko jedną mapę, ale prace nad dodaniem obsługi kolejnych map są w toku. Dzięki użyciu bibliotek takich jak tkinter do tworzenia interfejsu graficznego oraz PIL do manipulacji obrazami, narzędzie jest intuicyjne i łatwe w obsłudze. W przyszłości planowane jest rozszerzenie funkcjonalności oraz dodanie obsługi wszystkich map (każda wymaga innego podejścia).

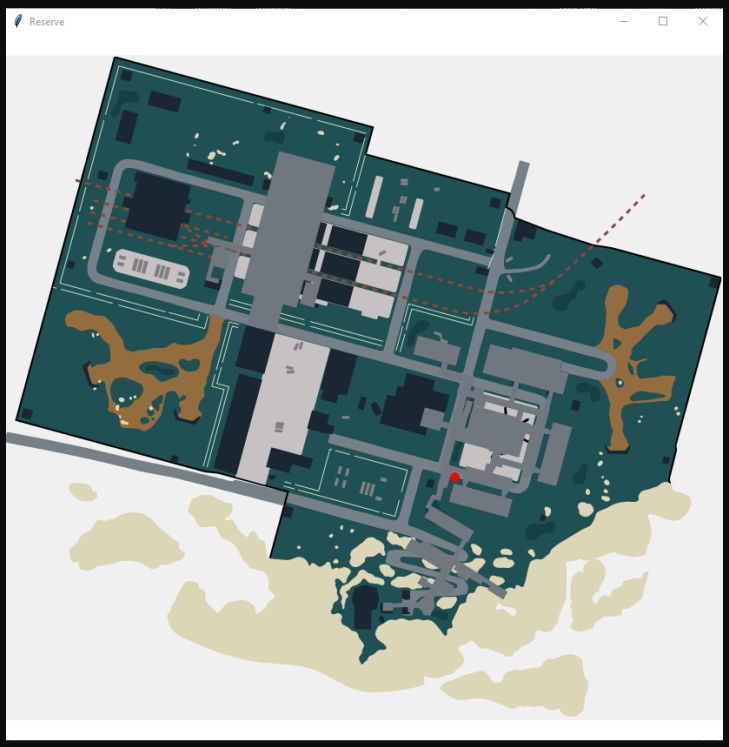
Wideo Prezentacja

Instrukcja użytkowania

1. Uruchomić program.
2. Wczytać zrzut ekranu (przykładowe dołączone do projektu).
3. Wybrać mapę na której jesteśmy (testowe dla mapy Reserve).
4. Pojawi się okienko z mapą i wyznaczonym czerwonym punktem.

Okno menu:

Okno mapy:



Opis poszczególnych funkcji

1. Funkcja load_map_config:

```
def load_map_config():
    with open('map_config.json', 'r') as file:
        return json.load(file)

MAP_CONFIG = load_map_config()
```

- Ładuje konfigurację map z pliku map_config.json i zwraca ją jako słownik.
- Przechowuje ją w zmiennej globalnej MAP_CONFIG

2. Funkcja upload_screenshot oraz save_screenshot_name:

```
def upload_screenshot():
    file_path = filedialog.askopenfilename(filetypes=[("Obraz PNG", "*.png")])
    if file_path:
        file_name = os.path.basename(file_path)
        save_screenshot_name(file_name)
        messagebox.showinfo("Sukces", f"Zaladowano zrzut ekranu: \n{file_name}")


def save_screenshot_name(screenshot_name):

    with open('nazwaZrzutu.txt', 'a') as file:
        file.write(f"{screenshot_name}\n")

    parts = screenshot_name.split('_')
    if len(parts) >= 3:
        values = parts[1].split(', ')
        if len(values) >= 3:
            value1 = values[0]
            value2 = values[2]
            with open('wspolrzedne.txt', 'a') as file:
                file.write(f"{value1}, {value2}\n")
```

- Otwiera okno dialogowe do wyboru pliku PNG
- Jeśli plik zostanie wybrany, zapisuje jego nazwę do pliku nazwaZrzutu.txt i wyświetla komunikat o sukcesie
- Ekstraktuje współrzędne z nazwy pliku i zapisuje je do pliku wspolrzedne.txt

3. Funkcja take_game_coordinates:

```
def take_game_coordinates():

    with open('wspolrzedne.txt', 'r') as file:
        last_line = file.readlines()[-1]
        x, y = last_line.strip().split(', ')
        return float(x), float(y)
```

- Odczytuje ostatnią linię z pliku wspolrzedne.txt i zwraca współrzędne jako krotkę liczb zmiennoprzecinkowych

4. Funkcja show_map:

```
def show_map(selected_map):
    svg_path = os.path.join("mapa", f"{selected_map}.svg")
    coordinates = take_game_coordinates()
    game_x, game_y = coordinates

    png_data = cairosvg.svg2png(url=svg_path)
    image_stream = io.BytesIO(png_data)
    map_image = Image.open(image_stream)
    map_photo = ImageTk.PhotoImage(map_image)
    img_width, img_height = map_image.size

    new_window = Toplevel(root)
    new_window.title(selected_map)
    new_window.geometry(f"{img_width}x{img_height + 50}")
    new_window.configure(bg='white')

    canvas = Canvas(new_window, width=img_width, height=img_height)
    canvas.pack(expand=True)
    canvas.create_image(0, 0, anchor=tk.NW, image=map_photo)
    canvas.image = map_photo

    map_config = MAP_CONFIG.get(selected_map, MAP_CONFIG["Default"])
    center_x = (map_config["centerMaxX"] + map_config["centerMinX"]) / 2
center_y = (map_config["centerMaxY"] + map_config["centerMinY"]) / 2

    real_x = img_width / 2 - ((game_x - center_x) * img_width /
(map_config["pointMaxX"] - map_config["pointMinX"]))
    real_y = img_height / 2 + ((game_y - center_y) * img_height /
(map_config["pointMaxY"] - map_config["pointMinY"]))

    point_estetic(canvas, int(real_x), int(real_y), "red")

# --- Dodaje punkt na mapie ---
def point_estetic(canvas, x, y, color="red"):
    canvas.create_oval(x - 5, y - 5, x + 5, y + 5, fill=color, outline=color)
```

- Wylicza pozycję punktu na mapie na podstawie współrzędnych gry i konfiguracji mapy
- Rysuje punkt na mapie
- Ładuje plik SVG mapy i konwertuje go na obraz PNG
- Tworzy nowe okno i wyświetla mapę

5. Konfiguracja GUI:

```
root = tk.Tk()
root.title("EFT-GPJesus")
root.geometry("200x480")
root.configure(bg='light gray')

button_style = {
    "width": 20,
    "height": 2,
    "bg": 'steel blue',
    "fg": 'white',
    "activebackground": 'royal blue',
    "activeforeground": 'white',
    "font": ('Helvetica', 10, 'bold')
}
button_style_maps = {
    "width": 10,
    "height": 2,
    "bg": 'steel blue',
    "fg": 'white',
    "activebackground": 'royal blue',
    "activeforeground": 'white',
    "relief": tk.FLAT,
    "font": ('Helvetica', 10, 'bold')
}

upload_button = tk.Button(root, text="Wczytaj zrzut ekranu",
command=upload_screenshot, **button_style)
upload_button.pack(pady=10, padx=20)
frame = tk.Frame(root, bg='light gray')
frame.pack(fill=tk.BOTH, expand=True, padx=20, pady=20)
maps = ["Customs", "GroundZero", "Interchange", "Lighthouse", "Reserve", "Shoreline",
"Woods"]

for map_name in maps:
    button = tk.Button(frame, text=map_name, command=lambda mn=map_name:
show_map(mn), **button_style_maps)
    button.pack(pady=5)
```

- Tworzy główne okno aplikacji
- Definiuje styl dla przycisków
- Tworzy przycisk do wczytywania zrzutu ekranu oraz każdej mapy

5. Główna pętla programu

```
root.mainloop()
```

- Uruchamia główną pętlę aplikacji, która czeka na interakcje użytkownika