Politechnika Śląska w Gliwicach
Wydział Informatyki, Elektroniki i Informatyki

PROJEKT
Praktyka Programowania Python
Organizacja i rozwój projektów Open Source

System do planowania obserwacji satelitów orbity ziemskiej

Autor Maciej Gromek
Repozytorium:
https://github.com/mgromek99/System-do-planowania-obserwacji-satelit-w-orbity-ziemskiej
Rok akademicki 2024/2025
Kierunek Informatyka
Specjalizacja IGT
Data złożenia 11.02.2025

Spis treści

Rozdział 1

Wstęp

Na całym świecie istnieją setki obserwatorów, każde z nich prowadzące obserwację satelitów, zazwyczaj w celach badawczych. Jednak pojawia się wtedy problem, jak zorganizować tą sieć obserwatoriów tak, aby każdy obiekt był w danej chwili obserwowany przez jedno obserwatorium celem zwiększenia efektywności wykorzystania ich dostępnego czasu. Rozszerzeniem tego problemu jest upewnienie się, że to najważniejsze obiekty są obserwowane przez najdłuższy możliwy czas wykorzystując optymalne do tego obserwatoria. Mój projekt ma na celu usprawnić ich działanie w taki właśnie sposób.

Rozdziały tej pracy będą omawiały następujące tematy:

- Rozdział drugi - Analiza tematu, opisze cel projektu, przejdzie przez problemy związane z obserwacjami oraz wyodrębni te z nich, które wymagają brania pod uwagę w kontekście tej pracy.
- Rozdział trzeci - Wymagania i narzędzia, przejdzie przez funkcje, które będzie musiał posiadać efekt tej pracy, metodykę pracy wykorzystywaną podczas tworzenia go i używanych do tego narzędzi.
- Rozdział czwarty - Specyfikacja zewnętrzna, przejdzie przez kolejne elementy programu i jak one mogą działać.
- Rozdział piąty - Specyfikacja wewnętrzna, opisze najważniejsze elementy kodu programu.
- Rozdział szósty - Weryfikacja i walidacja, będzie opisywać sposób testowania efektu projektu, stanowisko na którym był on testowany oraz najważniejsze wykryte błędy,
- Rozdział siódmy - Podsumowanie i wnioski, przejdzie przez potencjalne możliwości rozbudowy rozwiązania.

Rozdział 2

Analiza tematu

2.1 Cel Projektu

Głównym celem projektu jest utworzenie programu komputerowego, który przetworzy dane otrzymane od użytkownika oraz na ich podstawie utworzy plan obserwacji, który pozwoli zwiększyć ilość dostępnego czasu obserwacji satelitów orbity ziemskiej dzięki zapewnieniu braku możliwości nakładania się czasów obserwacji różnych obserwatorów w przypadku jednoczesnego wystąpienia tego samego obiektu w ich polu widzenia co zagwarantuje zaimplementowanie funkcjonalności możliwości ustawiania priorytetów zarówno dla obiektów obserwowanych jak i również obserwatoriów. Zakładane jest także, że użytkownik będzie miał możliwość zapisania wygenerowanych danych w zależności od ich typu, które zostaną utworzone w plikach tekstowych lub graficznych. Ponadto program umożliwi dostęp do bardziej podstawowych danych, które są częściami składowymi planu takimi jak czasy umożliwiające obserwatoriom wykonywanie obserwacji, czy wszystkie przeloty obiektów w ich polu widzenia. Wszystkie te funkcjonalności dostępne będą dla użytkownika w przeznaczonym do tego interfejsie graficznym.

2.2 Problematyka wykonywania obserwacji z powierzchni Ziemi

Obserwacje obiektów znajdujących się poza atmosferą Ziemi wymagają brania pod uwagę wielu aspektów takich przedsięwzięć, niezależnie od tego, czy są one wykonywane hobbystycznie, czy w środowisku profesjonalnym, od tego z jakich punktów te są wykonywane i ich otoczenia, przez to jakim medium jest atmosfera Ziemska, czy w jakich momentach możliwe jest ich obserwowanie i kiedy te znajdują się w polu widzenia obserwatora. Ta sekcja przejdzie przez problemy, które mogą napotkać osoby zainteresowane nimi celem określenia, które z nich powinno się i można rozwiązać programem będącym celem tego projektu.

2.2.1 Refrakcja atmosferyczna

Jest to zjawisko, które odnosi się do zakrzywienia ścieżki po której porusza się światło, gdy przemieszcza się ono przez atmosferę. Pojawia się ono, gdyż atmosfera nie jest próżnią, a ośrodkiem niejednorodnym, mieszaniną gazów o różnych gęstościach i temperaturach, które wpływają na prędkość przepływających przez nie widzialnych promieni promieniowania elektromagnetycznego. Ostatecznym efektem zjawiska w tym kontekście jest wrażenie z perspektywy obserwatora pojawienia się obserwowanego obiektu delikatnie wyżej, niż gdzie ten rzeczywiście się znajduje. Efekt ten nasila się wraz ze zmniejszeniem kąta pomiędzy horyzontem, a obiektem, gdyż światło, które dociera do obserwatora musi przebyć dłuższą drogę przez zniekształcające otoczenie.

2.2.2 Zanieczyszczenie świetlne

Dotyczy ono rozjaśnienia nocnego nieba przez przykładowo oświetlenie uliczne, światło uciekające z mieszkań, czy inne źródła iluminacji stworzone przez człowieka, które zwiększają jasność ich okolicy. Osoby zainteresowane obserwacjami muszą więc wykorzystać miejsca, które znajdują się w wystarczającej odległości od takich zjawisk, wykorzystując dostępne źródła informacji jak mapy zanieczyszczeń świetlnych których przykładem mogą być te udostępniane przez na przykład serwisy internetowe "www.lightpollutionmap.info"[9], czy "www.darksitefinder.com"[3].

2.2.3 Warunki pogodowe i środowiskowe

Są one krytycznie ważnym elementem dla podejmowania obserwacji ze względu na to, że mogą one wpłynąć na zarówno jakość obserwacji jak i stan wykorzystywanych do nich narzędzi. Przede wszystkim problematyczne są chmury, czy turbulencje w atmosferze utworzone przez różnice w temperaturze i prędkości wiatru na różnych wysokościach co opisuje efekt nazwany w astronomii "seeing"[1].

2.2.4 Absorpcja atmosferyczna

Atmosfera Ziemi składa się z wielu gazów, które mogą tłumić niektóre długości fal elektromagnetycznych, tym samym utrudniając lub uniemożliwiając obserwację fal o specyficznych długościach. Zjawisko to tworzy tak zwane okna atmosferyczne [8], sprawiając, że duża część promieniowania ultrafioletowego (określane często jako UV, z języka angielskiego od słowa ultraviolet), część podczerwonego (określane często jako IR, z języka angielskiego od słowa infrared) i dalekiego podczerwonego (określane często jako FIR, z języka angielskiego od słowa far-infrared) nie jest z powierzchni Ziemi widzialna. Te z obserwatoriów, które osadzone są na powierzchni Ziemi położone są często w miejscach

o dużej wysokości nad poziomem morza, gdzie powietrze jest rzadsze i suchsze, aby zminimalizować te efekty, w szczególności dla IR. Najbardziej widocznymi długościami są te określone w oknie atmosferycznym dla światła widzialnego i części fal radiowych.

2.2.5 Prędkość względna

Obiekty, które są obserwowane w kontekście astronomii poruszają się z dużymi prędkościami względem siebie nawzajem i obserwatora na Ziemi. Jest to krytyczna dla wykonywania obserwacji koncepcja, ponieważ wprowadza wymóg brania poprawek o zjawiska takie jak obrót kuli ziemskiej, który sprawia, że obiekty widoczne na niebie oddają wrażenie poruszania się po nim, wpływając na to jak i kiedy są one obserwowane.

2.2.6 Światło dzienne

Najczęstszymi z pór dnia do przeprowadzania obserwacji są momenty, gdy słońce nie wpływa na nie w tak dużym stopniu jak za dnia, głównie odnosząc się do późnych pór dnia, gdy Słońce zachodzi poniżej horyzontu, pomiędzy zjawiskami określonymi jako zmierzchy. Można wyróżnić ich trzy rodzaje:

- Zmierzch cywilny, występujący, gdy słońce znajduje się względem horyzontu dla obserwatora poniżej 0 stopni, aż po 6 stopni.
- Zmierzch żeglarski, który ma miejsce, gdy słońce pojawia się pomiędzy 6 stopniami, a 12.
- Zmierzch astronomiczny, mający odpowiednio progi 12 oraz 18 stopni.

Dzieli się je również ze względu na to, czy rozpoczynają one się od zakończenia dnia, czy nocy:

- Te, które zaczynają się od końca dnia nazywane są zachodami.
- Kończące noc natomiast nazywa się wschodami.

Często wykorzystuje się do wykonywania obserwacji czasy, jak sama nazwa wskazuje, zmierzchu astronomicznego, ale dla najbardziej widocznych na niebie obiektów obserwacje można rozpocząć od zmierzchu żeglarskiego, którego nazwa wzięła się z wykorzystywania przez żeglarzy do nawigacji najbardziej znanych gwiazd, a horyzont mógł być jasno rozróżniony.

2.2.7 Satelity i ich orbity

Satelity są stworzonymi przez człowieka obiektami celem wykonywania szerokiej gamy zadań takich jak zastosowania telekomunikacyjne, obserwacja pogody, nawigacja, obserwacja Ziemi, czy pomiary w celach badawczych. Te, które używane są na orbitach osadzonych dookoła Ziemi (geocentrycznych) często charakteryzują swoje zastosowania w zależności od typu swojej orbity. Spośród głównych ich typów można wymienić [4]:

- Niska orbita okołoziemska (ang. Low Earth Orbit, LEO) to orbita typowo orbita na wysokości od 160 do 2000 kilometrów nad powierzchnią Ziemi. Satelity na niej osadzone mają krótkie okresy orbitalne, często od 90 do 120 minut, innymi słowy szybko okrążające Ziemię. Tego typu orbity są optymalne dla satelitów obserwacyjnych jak i również tych, które posiadają aparaturę badawczą, ponieważ umożliwiają zbieranie dużej ilości detali obserwacji. Niemniej jednak mogą mieć one ograniczone pole widzenia i wymagać zbudowania ich sieci celem zapewnienia ciągłości w pomiarach, a co więcej głównym wyzwaniem dla satelitów LEO jest opór atmosferyczny, który może skrócić ich żywotność orbitalną. Wymagają również częstych regulacji, aby utrzymać orbitę.

- Średnia orbita okołoziemska (ang. Medium Earth Orbit, MEO) natomiast to rodzaj orbity znajdujący się pomiędzy wysokościami 2000 i 35786 kilometrów nad powierzchnią. Ich okresy orbitalne wynoszą typowo kilka godzin i są wykorzystywane do zastosowań takich jak nawigacja, której przykładem może być Global Positioning System (GPS). Jest to orbita umożliwiająca osiągnięcie balansu pomiędzy pokryciem i opóźnieniami sygnałów.

- Orbita geostacjonarna (ang. Geostationary Orbit, GEO) znajduje się dokładnie na wysokości 35786 kilometrów nad równikiem, ponieważ jest to wysokość na której jej okres orbitalny jest równy dobie sprawiając, że pozostają w jednej pozycji względem powierzchni Ziemi. Jej głównym problemem jest to, że jest tylko w tym konkretnym miejscu, więc ilość satelitów, które tam można osadzić jest ograniczona, a ponadto nie nadaje się w porównaniu do poprzednich do wykonywania obserwacji powierzchni Ziemi i wymagają osprzętu komunikacyjnego o większej mocy.

- Orbita heliosynchroniczna (ang. Sun-synchronous orbit, SSO) jest to orbita, która sprawia, że satelity będące na niej każdego dnia przelatują nad tą samą częścią Ziemi o tej samej lokalnej godzinie słonecznej lub innymi słowy jej płaszczyzna nie będzie daleka od płaszczyzny biegunowej Ziemi na danej szerokości geograficznej. Są one szczególnie przydatne dla satelitów wykonujących obrazowanie, rekonesans, czy badających pogodę ze względu na ten sam kąt oświetlenia przy każdym przelocie nad badanym punktem. Orbitę tego rodzaju osiąga się poprzez cofanie płaszczyzny orbity w kierunku wschodnim każdego dnia o około jeden stopień.

- Orbita silnie eliptyczna (ang. Highly Elliptical Orbits, HEO) to orbita o dużej ekscentryczności, czyli mierze określającej jej kształt opisanej przez wzór (2.1), co sprawia, że osadzone na takich orbitach satelity mogą pozostawać nad punktami zainteresowania przez długi okres czasu. Ze względu na to zjawisko mogą tego rodzaju orbity być wykorzystywane w celach komunikacyjnych, czy do długich obserwacji w pozycjach o dużej szerokości geograficznej, gdzie te satelity, które osadzone są na orbitach geostacjonarnych nie mają pokrycia.

$$e = \sqrt{1 + \frac{2EL^2}{m_{\text{red}}\alpha^2}}$$

Gdzie:

e	oznacza ekscentryczność,
E	oznacza energię całkowitą,
L	oznacza całkowity moment pędu,
m_{red}	oznacza masę zredukowaną układu dwóch ciał,
α	jest współczynnikiem siły centralnej prawa odwrotności kwadratu.

(2.1)

Warto zaznaczyć, że wynik wzoru (2.1) określa rodzaj orbity:

- e = 0, dla orbit okrągłych,
- 0 < e < 1, dla orbit eliptycznych,
- e = 1, dla orbit parabolicznych (paraboliczna orbita ucieczki),
- 1 < e, dla orbit hiperbolicznych.

Obserwator, który zamierza obserwować satelity orbity ziemskiej musi brać poprawkę na to jakiego kształtu oraz na jakiej wysokości są orbity satelitów i w jakiej pozycji w danym czasie na nich się one znajdują celem ich obserwowania.

2.3 Pozyskiwanie danych o pozycjach

Jakiekolwiek działania dążące do obserwacji muszą zostać rozpoczęte od zebrania informacji i ich przetworzenia. Ta sekcja przejdzie przez wykorzystane rodzaje danych służących do określenia informacji na temat satelitów, czy obserwatorów.

2.3.1 Two Line Element

Informacje na temat satelitów orbity ziemskiej można pozyskać wykorzystując dane TLE (ang. Two-Line Element, element dwuliniowy). Jest to format zapisu listy elementów

orbitalnych (parametrów) orbity keplerowskiej sztucznych satelitów orbity ziemskiej dla danego momentu w czasie. Jak sama nazwa wskazuje składają się one z dwóch linii danych numerycznych, które umożliwiają przewidywanie pozycji satelity w czasie. Linia pierwsza składa się z następujących komponentów:

- Numer seryjny satelity, numer katalogowy NORAD (ang. North American Aerospace Defense Command), czyli Dowództwa Obrony Północnoamerykańskiej Przestrzeni Powietrznej i Kosmicznej, który jest unikatowym identyfikatorem przydzielonym do wszystkich orbitujących wokół Ziemi obiektów,
- Międzynarodowy identyfikator satelity zawierający rok oraz numer startu,
- Data i czas epoki, które określają moment dla którego dane są aktualne,
- Pierwsza pochodna średniego ruchu, czyli zmiana prędkości orbitalnej,
- Druga pochodna średniego ruchu, czyli szybkość zmiany prędkości orbitalnej,
- Współczynnik oporu atmosferycznego BSTAR,
- Typ efemeryd, czyli określenie którego modelu należy użyć do generacji TLE,
- Numer zestawu elementów, czyli inkrementowana wartość oznaczająca łączną ilość aktualizacji TLE,
- Suma kontrolna modulo 10 weryfikująca poprawność linii.

Linia druga natomiast składa się z poniższych elementów:

- Ponowne podanie numeru katalogowego NORAD,
- Inklinacja, czyli nachylenie orbity względem równika,
- Rektascensja węzła wstępującego lub inaczej miejsce, gdzie orbita przecina równik, idąc na północ,
- Ekscentryczność orbity pokazująca, jak bardzo orbita odbiega od kształtu koła (wzór (2.1)),
- Argument perygeum, czyli punkt orbity, który znajduje się najbliżej Ziemi,
- Anomalia średnia, czyli pozycja satelity na orbicie w momencie aktualnej epoki,
- Średni ruch, czyli liczba obiegów satelity wokół Ziemi na dobę,
- Numer obiegu na epokę lub innymi słowy łączna liczba obiegów od momentu wystrzelenia do momentu epoki,

- Suma kontrolna modulo 10 dla sprawdzenia poprawności linii drugiej.

Dane te są krytyczne do śledzenia satelitów i powszechnie wykorzystywane zarówno przez obserwatorów w środowiskach hobbystycznych jak i profesjonalnych, którzy zajmują się operacjami kosmicznymi. Trzeba jednak zaznaczyć, że aby utrzymać ich wystarczającą dokładność w obliczeniach, wymagają one aktualizacji, gdy się zmienią, szczególnie kiedy zmieniają się często. [6]

2.3.2 Efemerydy

Efemerydami nazywamy dane, najczęściej w formie tabelarycznej, opisujące zestawienia współrzędnych obiektów astronomicznych w czasie, efektywnie ostatecznie dając ich trajektorię, zazwyczaj podawane jako współrzędne na niebie. Dzięki mocy obliczeniowej dzisiejszych komputerów te mogą być i są liczone na dziesiątki, czy nawet setki lat oraz dla czasu zarówno przyszłego jak i przeszłego tym samym sprawiając, że są niezastąpionym narzędziem dla obserwacji astronomicznych jak i innych dziedzin związanych z badaniem kosmosu. Dane te tworzone są na podstawie skomplikowanych modeli matematycznych przez organizacje takie jak Laboratorium Napędu Odrzutowego (ang. Jet Propultion Laboratory, JPL), które jest jednym z centrów badawczych NASA (Narodowa Agencja Aeronautyki i Przestrzeni Kosmicznej, ang. National Aeronautics and Space Administration). Przykładem może być właśnie tej agencji wypuszczony w 2008 roku model DE421, który pokrywał oryginalnie lata od 1900 do 2050 roku, a w 2013 roku został poszerzony o dane do 2200 roku. Poniższa tabela pokazuje rekomendacje dotyczące wybierania efemeryd:

Wypuszczone w:	**Krótkie**	**Średnie**	**Długie**
1997		de405 1600 do 2200 63 MB	de406 -3000 do 3000 287 MB
2008	de421 1900 do 2050 17 MB		de422 -3000 do 3000 623 MB
2013	de430_1850-2150 1850 do 2150 31 MB	de430t 1550 do 2650 128 MB	de431t -13200 do 17191 3.5 GB
2020	de440s 1849 do 2150 32 MB	de440 1550 do 2650 114 MB	de441 -13200 do 17191 3.1 GB

Tabela 2.1: Rekomendowane[5] wybieranie efemeryd spośród serii DE centrum badawczego JPL

2.3.3 Przewidywanie pozycji satelitów

Planowanie obserwacji nie może obejść się bez przewidywania tego, gdzie pojawią się satelity. Na szczęście istnieją narzędzia, które to umożliwiają - modele matematyczne takie jak SGP4 (uproszczony model orbity satelity dotyczący ogólnych zaburzeń, ang. Simplified General Perturbations Satellite Orbit Model 4). Jest to jeden z zestawu pięciu modeli matematycznych, które często określane są wspólnie jako SGP4 ze względu na to jak często ten z nich jest wykorzystywany z TLE. Algorytym ten wykorzystywany jest do obliczania prędkości i pozycji satelitów, uwzględniając najważniejsze perturbacje jak na przykład opór atmosferyczny, oddziaływania Słońca i Księżyca, czy nieidealny kształt Ziemi względem sfery. Posiada on błąd na poziomie około jednego kilometra w momencie epoki i rośnie od około jednego do trzech kilometrów dziennie, gdzie inne modele tego zestawu przeznaczone do większych orbit (powyżej 225 minut okresu orbitalnego) jak SDP4 (ang. Simplified Deep Space Perturbations model 4, uproszczony model zaburzeń głębokiej przestrzeni kosmicznej 4) mają błędy większe, w tym przypadku 10 kilometrów w momencie epoki.

2.3.4 Określenie pozycji obserwatora

Układy współrzędnych to systemy matematyczne określające pozycję obiektów w przestrzeni. Obserwator, który chce dokonywać obserwacji obiektów takich jak satelity musi wiedzieć jak określić swoją pozycję względem nich. Dla pozycji na powierzchni Ziemi można do tego wykorzystać system odniesienia WGS84 (ang. World Geodetic System 1984, Światowy System Geodyzyjny 1984), który definiuje następujące parametry:

- Model elipsoidy Ziemi, czyli matematyczne przybliżenie jej kształtu o określonej długości półosi małej (ekwatorialnej) oraz małej (polarnej),
- Geoidę, która jest hipotetycznym poziomem morza i jest wykorzystywana do określenia wysokości punktów na powierchni Ziemi,
- Datę geodezyjną, która sprawia, że można jednoznacznie określić pozycję na świecie,
- System trzech współrzędnych: szerokości (ang. latitude) i wysokości (ang. longitude) określanych w stopniach kątowych oraz wysokość (ang. altitude) w metrach, które umożlwiają określenie pozycji każdego punktu na Ziemi.

Posiada on wiele zastosowań w różnych dziedzinach kartografii, nawigacji satelitarnej, systemach informacji geograficznej, czy geodezji dzięki jednolitemu i precyzyjnemu określaniu pozycji, a ponadto jego globalne zastosowanie sprawia, że dane najczęściej są na całym świecie kompatybilne.

2.4 Osadzenie tematu w kontekście tematyki astronomii obserwacyjnej

Biorąc pod uwagę ogół tematyki wykonywania tego rodzaju obserwacji i cel projektu należy wybrać wspólne elementy. Program ten będzie przede wszystkim narzędziem planistycznym, więc będzie zajmował się określeniem czasów, gdy obserwacje są możliwe oraz zależnie od preferencji użytkownika tworzył z nich harmonogram dla określonych przedziałów czasowych. Tym samym najważniejszymi elementami, które będzie musiał przetworzyć będą te określone w:

- Części sekcji 2.2:
  - Podsekcji 2.2.5 Prędkość względna, która wpływa na to jak może poruszać się satelita względem obserwatora,
  - Podsekcji 2.2.6 Światło dzienne, które sprawia, że obserwacje nie będą utrudniane przez jego wpływ,
  - Podsekcji 2.2.7 Satelity i ich orbity, które określają jak długo satelity będą widoczne zależnie od typu,
- Całości sekcji 2.3:
  - 2.3.1 Two Line Element, gdyż to on określa elementy orbitalne satelitów,
  - 2.3.2 Efemerydy, które mogą zostać wykorzystane do stwierdzenia, kiedy w danym miejscu rozpocznie i zakończy się zmierzch,
  - 2.3.3 Przewidywanie pozycji satelitów, ponieważ bez przewidywania ich pozycji w czasie nie będzie możliwe stwierdzenie w jakich przedziałach czasowych te będą widoczne,
  - 2.3.4 Określenie pozycji obserwatora, ze względu na to, że dla każdej pozycji na Ziemi czasy widoczności danego satelity będą się różnić.

Rozdział 3

Wymagania i narzędzia

3.1 Wymagania funkcjonalne i niefunkcjonalne

Rozpoczęcie projektu programu komputerowego wymaga uprzedniego określenia tego co ten ma robić oraz jak będzie wchodził w interakcję z użytkownikiem. Problem ten omawiają wymagania funkcjonalne i niefunkcjonalne,

3.1.1 Wymagania funkcjonalne

- Określenie położenia użytkownika na kuli ziemskiej,
- Określenie przez użytkownika priorytetów danych punktów obserwacyjnych oraz satelitów,
- Obliczenie czasów zmierzchów, od zachodu do wschodu żeglarskiego, dla określonych punktów,
- Obliczenie czasów, gdy dany satelita znajduje się w polu widzenia obserwatora,
- Utworzenie z obliczonych czasów planu obserwacji według podanych priorytetów,
- Eksport danych do plików w określonym przez użytkownika miejscu.

3.1.2 Wymagania niefunkcjonalne

- Interfejs graficzny,
- Odporność na błędy użytkownika,
- Struktura umożliwiająca rozbudowę o nowe funkcje.

3.2 Metodyka pracy

Projekt rozpoczął się od określenia wymaganych do jego realizacji bibliotek, nauczenia się zawartych w nich narzędzi i wykorzystania ich do stworzenia wymaganych elementów programu. Wykorzystany został system kontroli wersji Git, który umożliwił tworzenie osobnych gałęzi dla tworzenia osobnych podprogramów, a następnie połączenia ich do wersji finalnej.

3.3 Opis wybranych narzędzi

3.3.1 Język programowania, moduły i biblioteki

Aplikacja została napisana w języku Python w wersji 3.11. Python został utworzony w roku 1991 przez Guido van Rossum w Centrum Wiskunde & Informatica w Holandii. Został on utworzony jako następca języka ABC. Charakterystyczną cechą języka Python jest to, że wszystkie jego wydania były typu Open Source, czyli o otwartym źródle. Python językiem interpretowanym lub innymi słowy różni się tym od języków takich jak C++ lub Java gdzie kod jest kompilowany do kodu maszynowego przed uruchomieniem, a zamiast tego jest wykonywany linia po linii przez interpreter. Konsekwencją tego jest to że zmienne w języku Python są dynamicznie typowane i są określane podczas działania programu, a nie podczas kompilacji. [2]

Moduł itertools

Jest to moduł dla języka Python, który zawiera narzędzia do obsługiwania iteratorów, czyli typów danych, które mogą być wykorzystane w pętli for do używania jego elementów jeden po drugim. Umożliwia on na generowanie kombinacji, czy permutacji danych i obliczeń opartych o iteratory z naciskiem na krótki czas wykonywania kodu, aby mogły zarządzać dużymi zestawami danych.

Moduł re

Wyrażenia regularne to inaczej wzorce opisujące łańcuchy symboli. Moduł re, którego nazwa pochodzi od angielskiej wersji tego wyrażenia, regular expressions, posiada narzędzia, które umożliwiają wyszukiwanie w liniach tekstu opisanych nimi słów i manipulację wartości z nimi związanych.

Moduł os

Moduł os (od ang. operating system) z kolei obsługuje funkcjonalności zależne od systemu operacyjnego. Korzystanie z niego umożliwia interakcje z nim takie jak obsługa plików, katalogów, a nawet wykonywania dostępnych dla niego komend.

Moduł datetime

Umożliwia on na zarządzanie datami i czasami na różne sposoby korzystając ze swoich klas, takie jak prosta arytmetyka, wydobywanie danych z linii tekstu, czy konwersja pomiędzy strefami czasowymi.

Moduł json

Daje on możliwość przekształcenia danych pomiędzy formatem tekstowym JSON (notacja obiektu JavaScript, od ang. JavaScript Object Notation), a strukturami danych języka Python. JSON często jest wykorzystywany do przesyłania danych aplikacji internetowych pomiędzy klientami, a serwerami.

Moduł copy

Wprowadza on funkcje, które kopiują obiekty zmienne takie jak listy, słowniki, czy własne struktury, aby wprowadzać zmiany bez wpływania na oryginał. Wspiera on kopie płytkie i głębokie, gdzie te pierwsze tworzą nowy obiekt, ale też referencje do tych, które znajdują się w oryginale, natomiast te drugie wykonują rekursywnie kompletną kopię oryginału, bez referencji do niego lub jego składowych elementów.

Moduł random

Tworzy on pseudo-losowe liczby różnych rodzajów takich jak liczby całkowite, zmiennoprzecinkowe lub sekwencje. Nadaje się on do zastosowań, które wymagają wprowadzenia zmiennych wyników przy różnych wywołaniach.

Biblioteka pytz

Biblioteka ta umożliwia precyzyjne operacje dla różnych stref czasowych. Jest ona szczególnie przydatna ze względu na to, że oferuje możliwość konwertowania obiektów datetime nieuwzględniających strefy czasowej, na takie, które je uwzględniają.

Biblioteka tkinter

Jest popularną biblioteką wykorzystywaną do tworzenia GUI (interfejs graficzny użytkownika, ang. Graphical User Interface) tym samym umożliwiając tworzenie komputerowych aplikacji okienkowych.

Biblioteka matplotlib

Matplotlib jest biblioteką w języku Python używaną w celach tworzenia wykresów i wizualizacji danych. Wspiera szeroką gamę różnych rodzajów wykresów takie jak liniowe,

słupkowe, punktowe, histogramy, kołowe i wiele innych jak i również posiada dużo opcji ich dostosowywania. Jej pliki wynikowe mogą być wyeksportowane w wielu formatach takich jak rastrowy PNG (ang. Portable Network Graphics, przenośna grafika sieciowa) o określonej rozdzielczości, czy wektorowym SVG (ang. Scalable Vector Graphics, skalowalna grafika wektorowa) i innych. Ze względu na jej popularność posiada też szereg zewnętrznych rozszerzeń

Biblioteka skyfield

Będąca efektywnie sercem tego projektu, biblioteka skyfield umożliwia wykonywanie wielu obliczeń krytycznych w zagadnieniach astronomii obserwacyjnej. Pomimo prostoty korzystania z niej, posiada ona narzędzia umożliwiające przewidywanie pozycji obiektów astronomicznych w czasie, takich jak gwiazdy, planety i co najważniejsze dla tego projektu - satelitów, czy też czasów, gdy te będą w spełniających wymagania użytkownika momentach. Jest zbudowana z myślą o wymaganej do tych zadań precyzją, korzystając z efemeryd udostępnionych przez JPL i implementacji algorytmu SGP4. Posiada też dużą bazę przykładów wykorzystania dostępnych w niej narzędzi w praktyce.

3.3.2 Oprogramowanie i usługi

Github

Github jest platformą internetową, która umożliwia hostowanie projektów, w tym za darmo dla prywatnych, wykorzystujących rozproszony system kontroli wersji Git. Dzięki temu, że istnieją kopie projektu na zewnętrznych serwerach w razie awarii prace można kontynuować pracę po zmianie maszyny roboczej. Kontrola wersji pozwala na zawężenie problemów pomiędzy nimi, a narzędzia umożliwiające zapisywanie problemów w projekcie ułatwiają pamiętanie o nich, aby nie pojawiły się w ostatecznych wersjach oprogramowania.

Github Desktop

Jest to program komputerowy od twórców platformy Github umożliwiający na proste tworzenie repozytoriów, zarządzanie gałęziami projektu, podgląd historii wpisów i tego co one dotyczyły, dzięki czemu umożliwia on przyspieszenie procesu zarządzania projektem.

Pycharm

PyCharm jest oprogramowaniem typu IDE (Integrated Development Environment, zintegrowane środowisko programistyczne) tworzone przez firmę JetBrains. Jego głównym celem jest zwiększenie produktywności poprzez oferowanie wielu przydatnych w czasie tworzenia projektów narzędzi. Spośród nich można wymienić inteligentny edytor kodu

uzupełniający w locie nazwy funkcji i zaznacza popełnione błędy w kodzie, czy graficzne narzędzia pokazujące wartości zmiennych w trybie debugowania i wiele innych.

space-track.org

Space-Track.org jest stroną i usługą umożliwiającą dostęp do danych na temat sytuacji w przestrzeni kosmicznej takich jak informacje o szczątkach, satelitach, czy innych obiektach orbitujących wokół Ziemi. Jedną z głównych usług dostarczanych przez tą stronę jest dostęp do informacji na temat aktualnych TLE satelitów dla użytkowników.

heavens-above

Są to aplikacja mobilna oraz serwis internetowy (heavens-above.com), które umożliwiają podgląd danych satelitów i umożliwiają sprawdzenie najbliższych przelotów nad pozycją podaną przez użytkownika. Serwis ten ułatwia stwierdzenie, z jakich miejsc można obserwować satelity, dzięki wizualizacji orbit.

kalendarz.livecity.pl

Serwis ten posiada przydatną dla weryfikacji funkcjonalności usługę, czyli opis zmierzchu zarówno dla wschodu i zachodu. Dane te podane są niemniej jednak dla konkretnych lokalizacji.

3.4 Środowisko testowe

Wymagania uruchomienia programu są stosunkowo w tym przypadku proste. O ile dane TLE mają ograniczoną datę ważności, zanim staną się przedawnione, wczytanie ich do programu i przetworzenie nim danych dla przykładowo najbliższych kilku dni jest, biorąc pod uwagę moc obliczeniową dzisiejszych komputerów, dosyć szybkie, nawet jeśli wymagałoby kilku minut obliczeń dla bardzo dużych zestawów danych. Jedyna maszyna na której program byłem w stanie uruchomić w czasie tworzenia projektu posiadała podzespoły średniej klasy cenowej - CPU (ang. Central Processing Unit, procesor) serii i5 z 2012 roku, który można uznać na dzisiejsze standardy pod względem mocy obliczeniowej za przestarzały. Maszyna ta jest w stanie przetworzyć kod tworzący plan na poziomie kilku do kilkunastu sekund na skali kilku do kilkunastu dni obliczeń.

Rozdział 4

Specyfikacja zewnętrzna

4.1 Wymagania sprzętowe

Wymogami uruchomienia programu są:

- Komputer będący w stanie kompilować kod użytych bibliotek i mający możliwość wyświetlania interfejsu graficznego,
- Zainstalowane wykorzystywane przez projekt biblioteki i ich zależności:
  - tkinter,
  - skyfield,
  - pytz,
  - matplotlib.

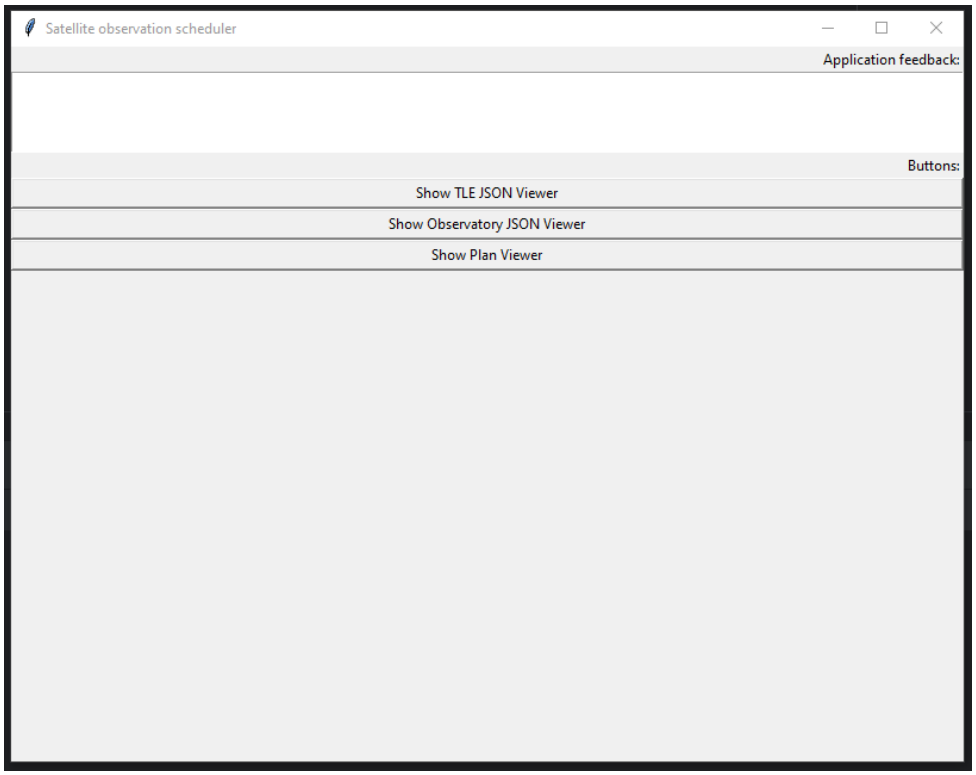
4.2 Uruchomienie programu

Użytkownik po zainstalowaniu wszystkich wymaganych zależności rozpoczyna pracę z programem poprzez uruchomienie kodu pliku main.py co pokazane jest na rysunku 4.1.

4.3 Opis interfejsu

Po uruchomieniu programu powinien pojawić się interfejs programu ilustrowany przez rysunek 4.2. Zaczynając od góry wyświetlonego okna posiada on następujące elementy:

Rysunek 4.1: Wywołanie kodu programu z konsoli



Rysunek 4.2: Interfejs programu po uruchomieniu

- Na górze ekranu można zauważyć okno "Application feedback:", pojawiają się w nim komunikaty na temat błędów działania aplikacji,

- Kolejnym elementem pod podpisem "Buttons:"znajdują się trzy przyciski otwierające następujące części programu:

  - "Show TLE JSON Viewer"Służący do otwarcia widoku otwierania i podglądu plików JSON zawierających dane TLE satelitów, które użytkownik chce obserwować oraz po załadowaniu dający możliwość ustawienia priorytetów,
  - "Show Observatory JSON Viewer"Służący do otwarcia widoku otwierania i podglądu plików JSON zawierających dane punktów obserwacyjnych, z których użytkownik chce dokonywać obserwacji oraz po załadowaniu dający możliwość ustawienia priorytetów,
  - "Show Plan Viewer"Służący do otwarcia widoku umożliwiającego tworzenie planu, listy przelotów satelitów, przedziały czasowe od zachodu do wschodu żeglarskiego, oraz eksport tych danych jako plik tekstowy lub w przypadku planu również graficzny.

- Wstępnie puste pole pod przyciskami jest miejscem, gdzie pojawiać będą się widoki z nimi powiązane.

4.3.1 Okno Application feedback

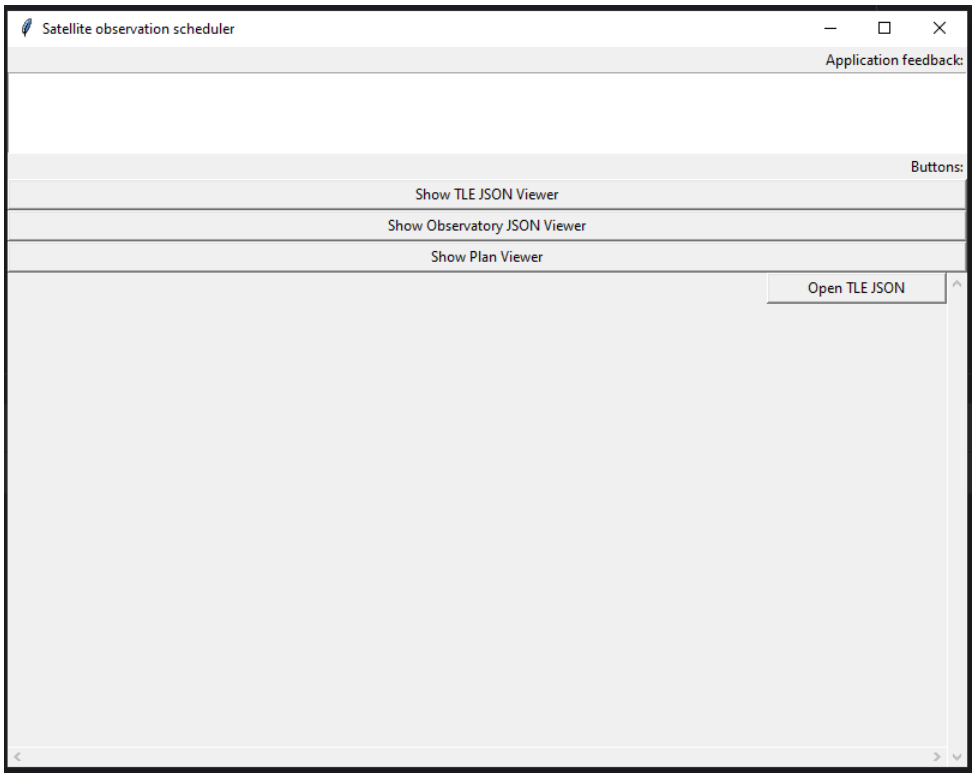
Okno to służy do wyświetlania napotkanych niespodziewanych błędów i ważnych wydarzeń, informując użytkownika, gdzie w programie wystąpił ostatni napotkany błąd lub jakie ważne wydarzenie nastąpiło. Nie wyświetla ono wszystkich błędów, gdyż te, które powiązane są z generowaniem pliku tekstowego znajdują się w jego podglądzie. Okno to może wyświetlać następujące komunikaty:

- "Start time must be earlier than end time.", gdy użytkownik próbuje wygenerować plan w widoku Plan Viewer wykorzystując datę zakończenia obserwacji mniejszą, niż jej początek,
- "Invalid date format: E", gdzie E oznacza napotkany błąd, gdy użytkownik wprowadzi datę w niepoprawnym formacie,
- "Selected folder: S", gdzie S oznacza ścieżkę, gdy użytkownik wybierze docelowy folder używany do eksportowania danych,
- "Please choose a location first.", gdy użytkownik będzie próbował dokonać eksportu danych bez uprzedniego wybrania ścieżki folderu,
- "Content exported successfully.", gdy program poprawnie dokona eksportu danych,
- "Failed to export content. Error: E", gdzie E oznacza napotkany błąd, gdy eksport danych nie powiedzie się,
- "Invalid datetime input.", gdy data początku lub zakończenia obserwacji nie istnieje,
- "First select export path", gdy użytkownik będzie próbował eksportować plan w postaci grafu Gantta bez uprzedniego ustawienia ścieżki docelowej,
- "Failed to use sorted plan.", gdy z niespodziewanych powodów nie uda się utworzyć danych wymaganych do wpisania w graf,
- "Error parsing date-time: V", gdy nie uda się załadować czasu podanego w pliku JSON.

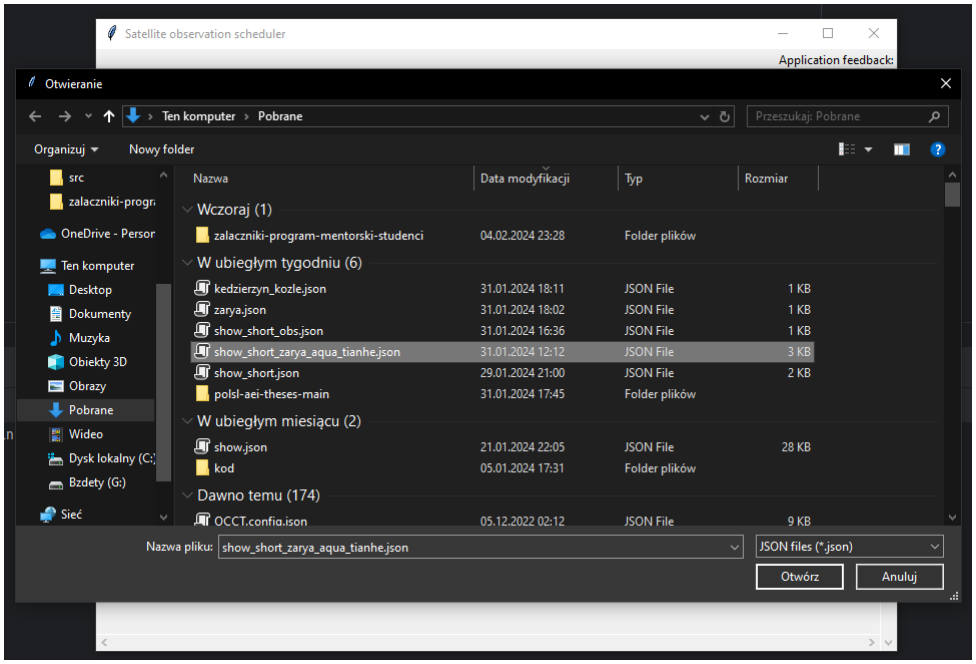
4.3.2 Widok TLE/Observatory JSON Viewer

Widoki te służą do wczytywania danych z plików JSON wybranych przez użytkownika. Jak widać na rysunku 4.3 pierw pokazuje on tylko przycisk "Open TLE JSON", który otwiera okno dialogowe do wybrania pliku do załadowania co przedstawia rysunek 4.4.

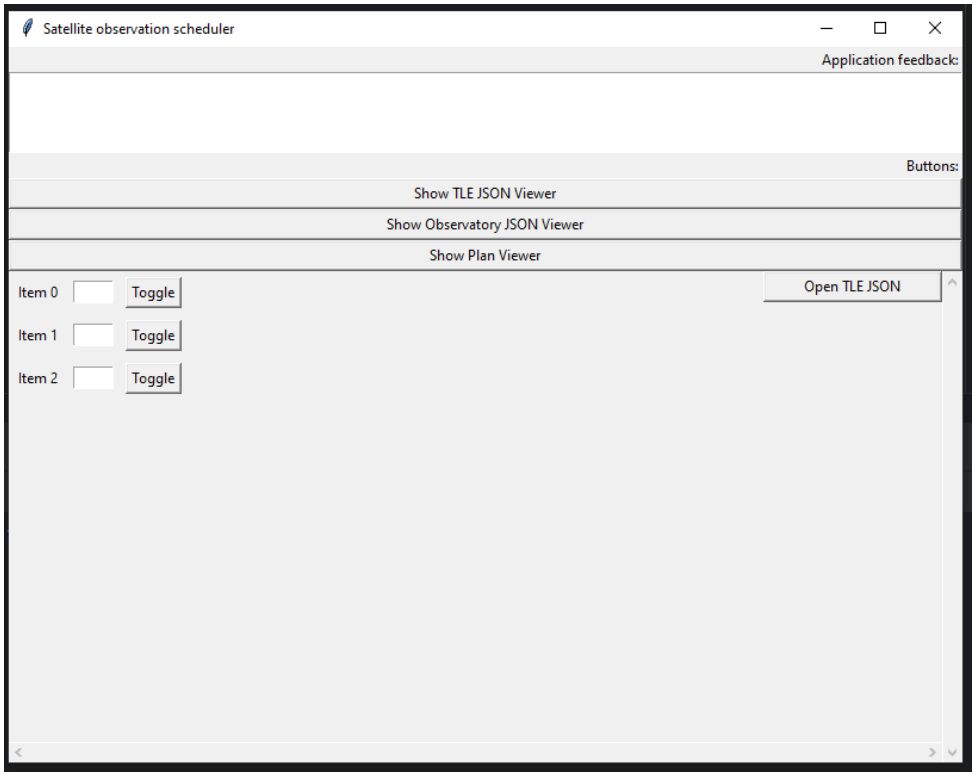
Po poprawnym otwarciu pliku użytkownik może zobaczyć listę otwartych obiektów jak na rysunku 4.5. Posiadają one liczbę porządkową, pole priorytetu oraz przycisk "Toggle", który po naciśnięciu pokazywać będzie dane danego obiektu tak jak na rysunku 4.6.



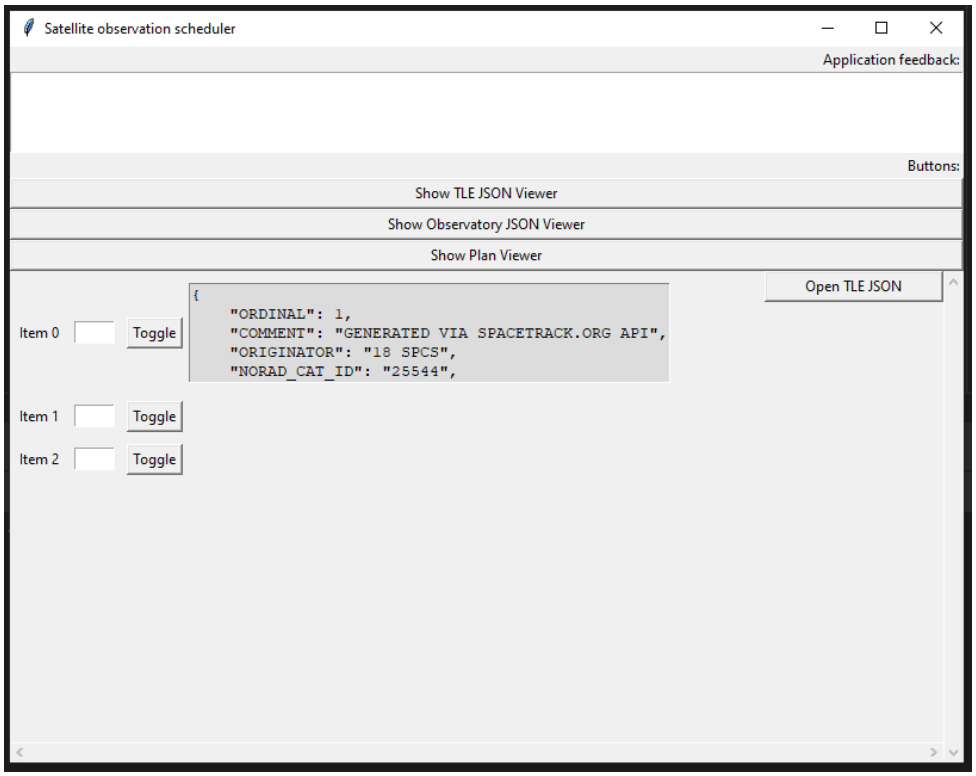
Rysunek 4.3: Interfejs programu po otwarciu jednego z widoków JSON Viewer



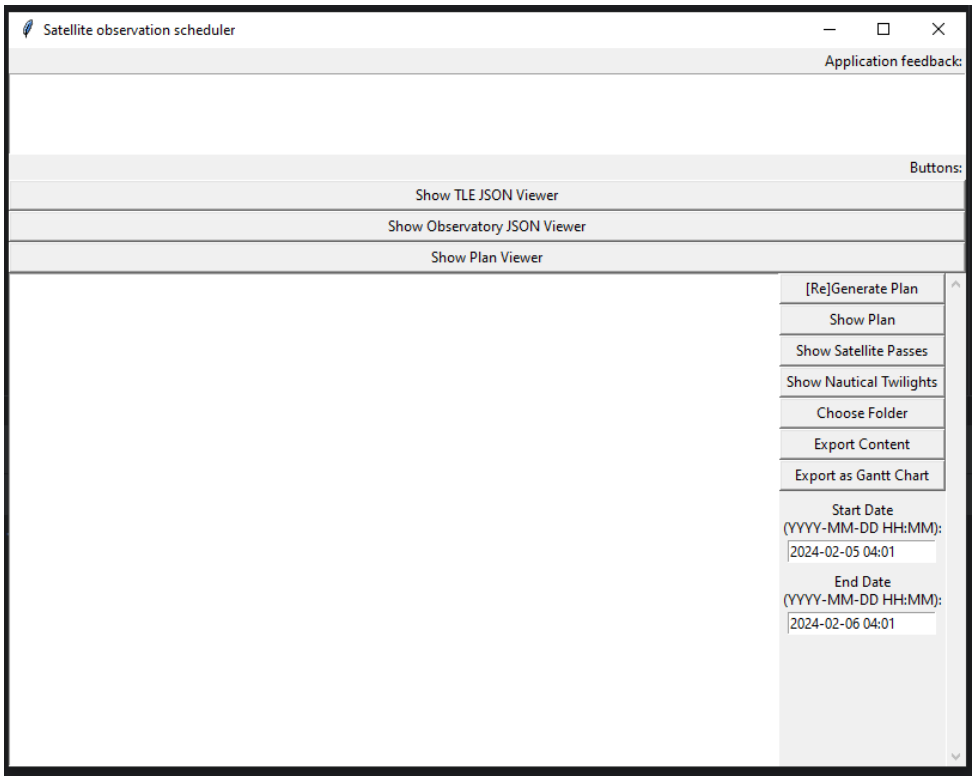
Rysunek 4.4: Interfejs programu po otwarciu okna dialogowego do wyboru pliku JSON



Rysunek 4.5: Interfejs programu po otwarciu pliku JSON



Rysunek 4.6: Interfejs widoku JSON Viewer po rozwinięciu detali przyciskiem Toggle



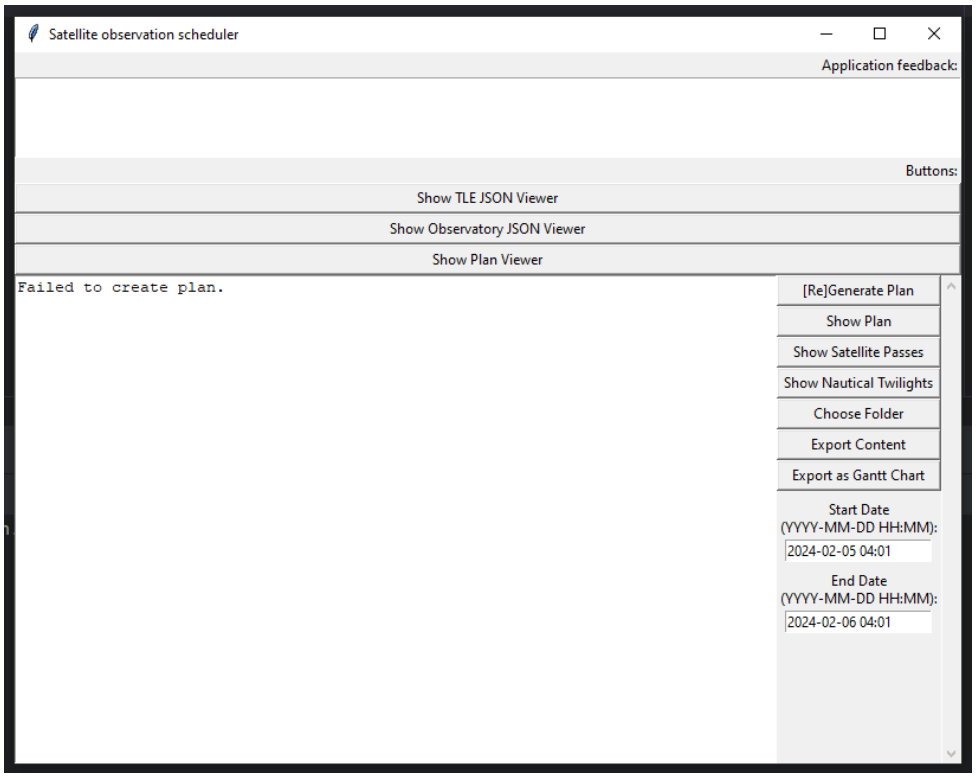
Rysunek 4.7: Interfejs programu po otwarciu widoku Plan Viewer

Priorytety mogą być jedynie liczbami całkowitymi. Trzeba zaznaczyć, że pozostawienie pola priorytetu pustym sprawi, że automatycznie zostanie przydzielony mu przy generacji priorytet najniższy.

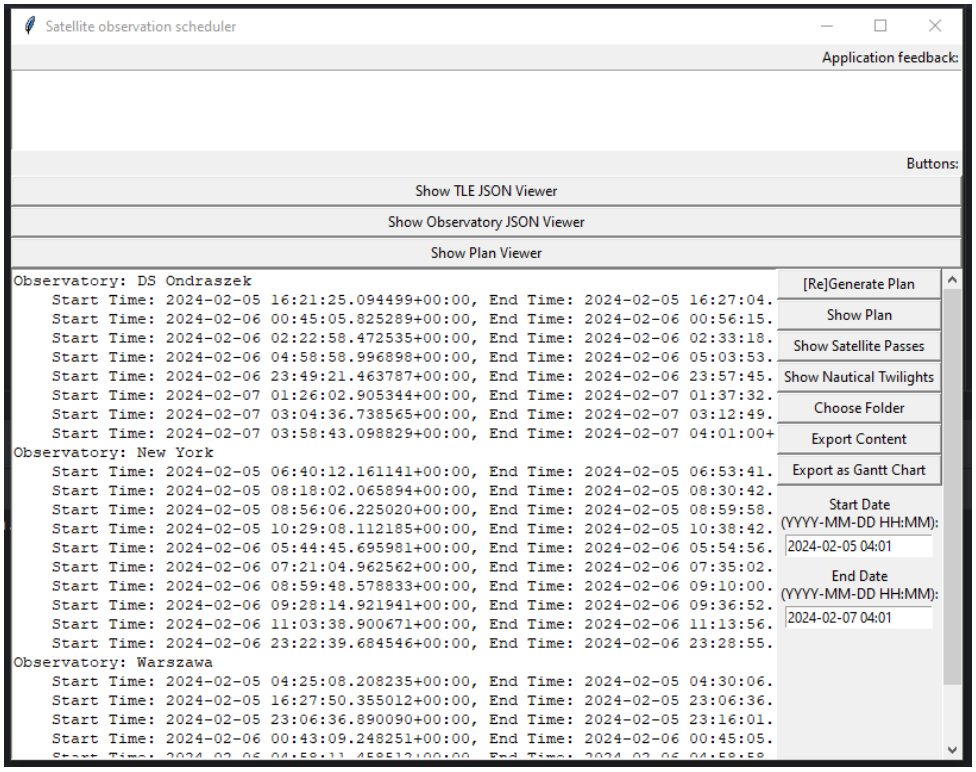
4.3.3 Widok Plan Viewer

Pierwszym co zobaczy użytkownik po otwarciu tego widoku jest widoczne na rysunku 4.7, czyli listę funkcjonalności tego widoku. Te są następujące:

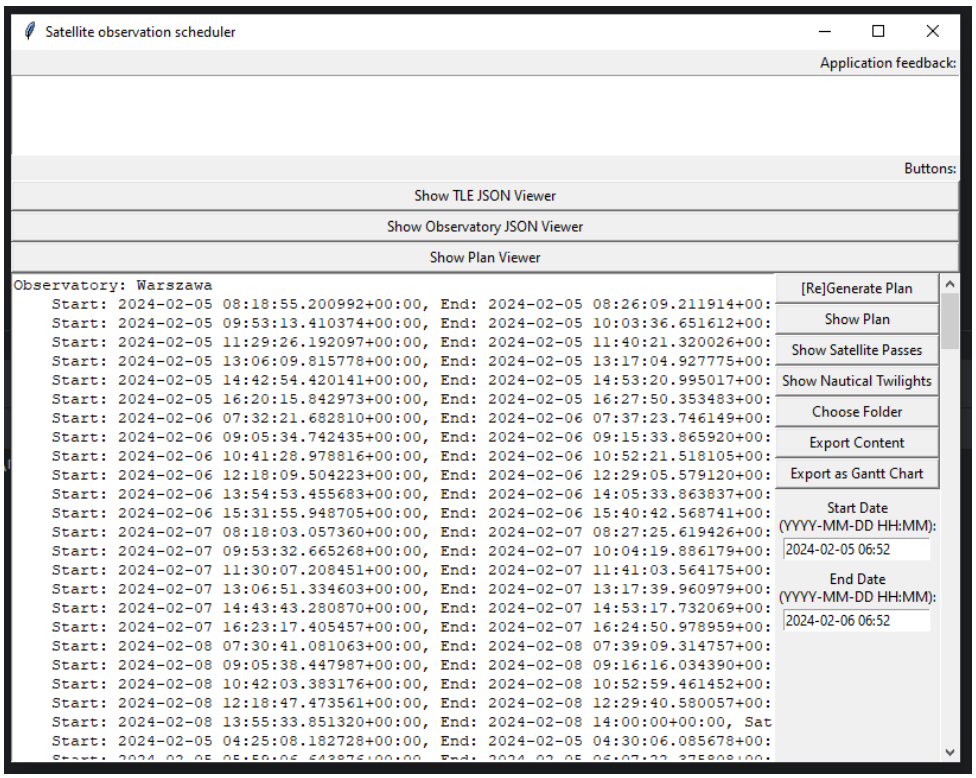
- Z lewej strony widoczny jest podgląd generowanego pliku tekstowego, którego zawartość będzie zależna od wywołanej funkcji,
- Z prawej strony natomiast są przyciski wywołujące odpowiednie funkcje oraz pola tekstowe służące do podania dat rozpoczęcia i zakończenia obserwacji. Kolejne, zaczynając od góry, elementy to:
  - Przycisk "[Re]Generate Plan", który generuje plan korzystając z danych poprzednich dwóch widoków. Użycie go może mieć dwa skutki:
    - Pierwszy to zwrócenie wartości "Failed to create plan.", która występuje, gdy ze względu na podane wartości wejściowe generowanie planu nie powiodło się. Sytuacja ta pokazana jest na rysunku 4.8,



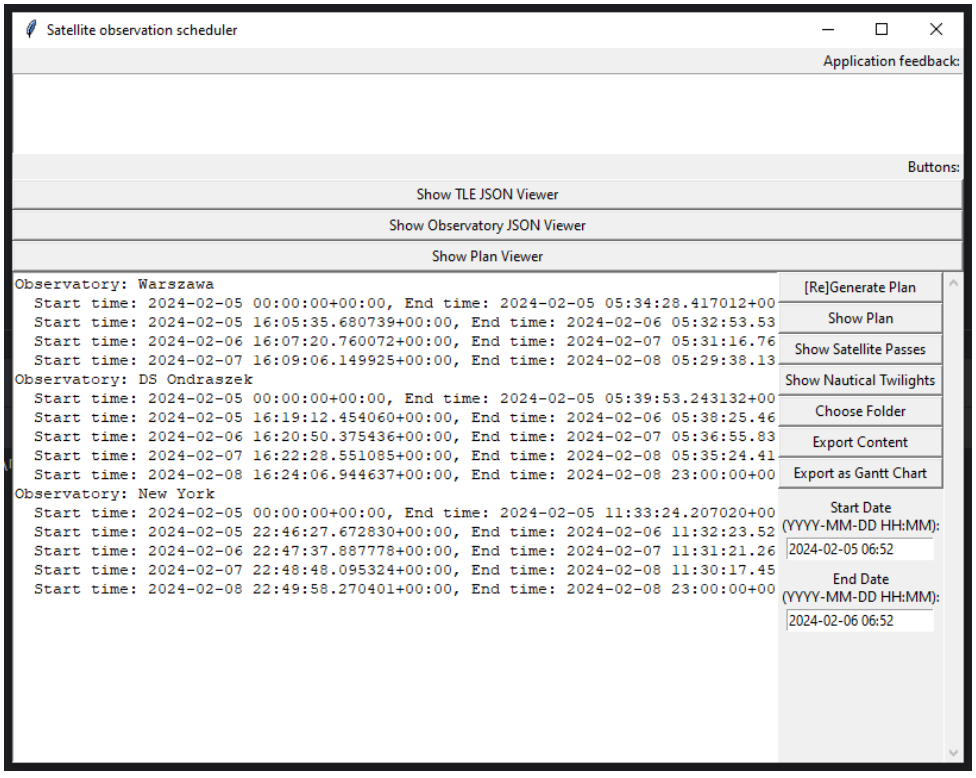
Rysunek 4.8: Interfejs programu po nieudanej próbie generacji planu



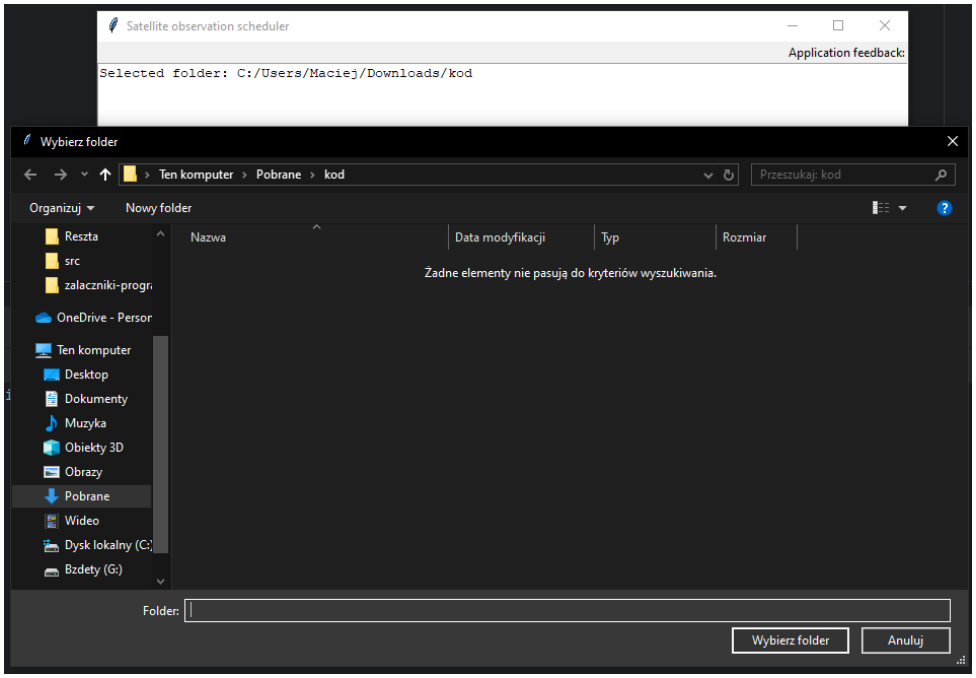
Rysunek 4.9: Interfejs programu po udanej próbie generacji planu



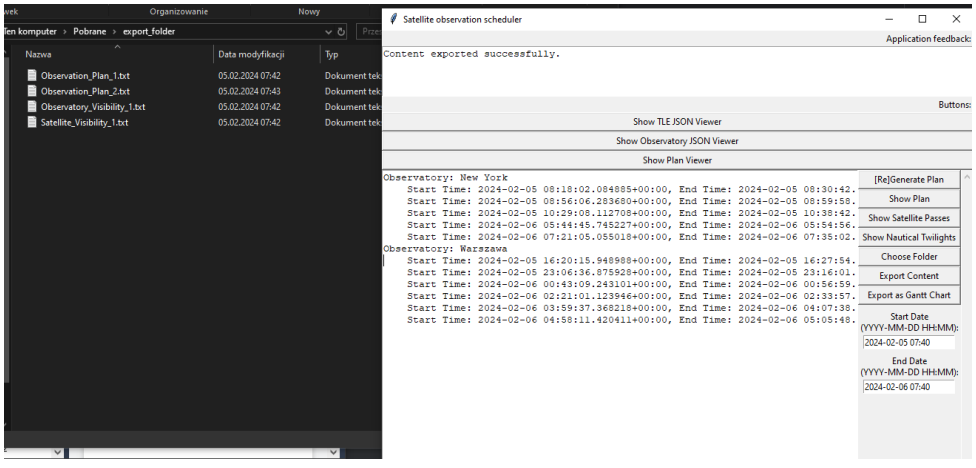
Rysunek 4.10: Interfejs programu po udanej próbie generacji wszystkich przelotów satelitów nad punktem



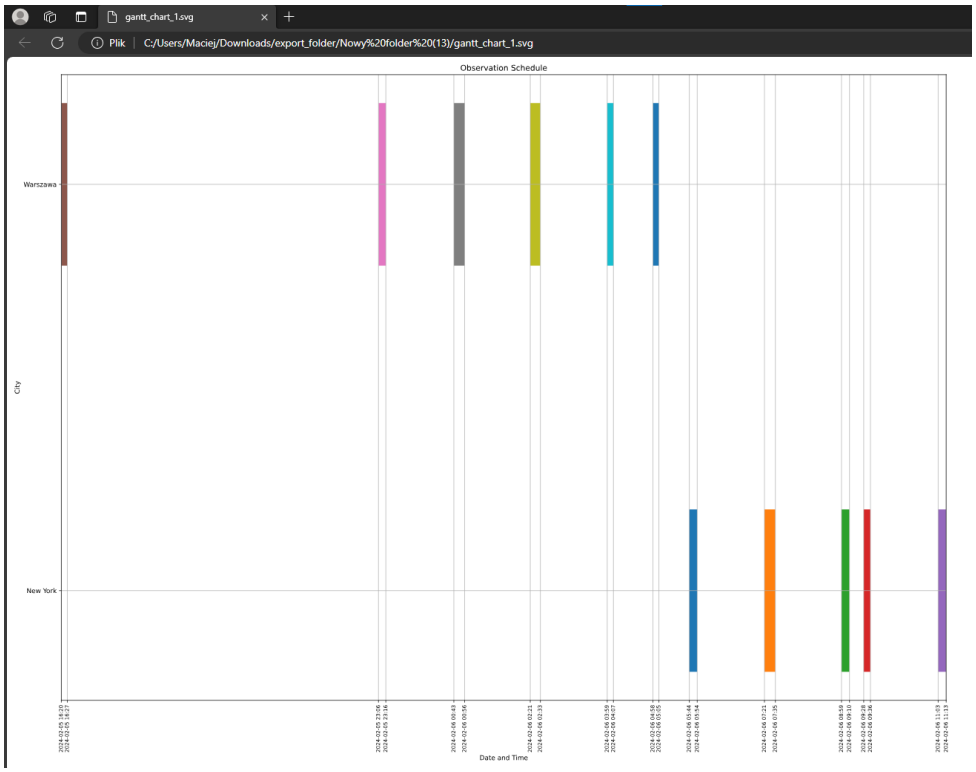
Rysunek 4.11: Interfejs programu po udanej próbie generacji wszystkich przedziałów czasowych od zachodu do wschodu żeglarskiego



Rysunek 4.12: Interfejs programu w czasie wybierania nowej ścieżki oraz z efektem jej wybrania w oknie Application Feedback



Rysunek 4.13: Interfejs programu po eksporcie do pliku tekstowego oraz efekt eksportu różnych danych



Rysunek 4.14: Eksportowany plik graficzny

  - Drugi to zwrócenie planu, gdy generacja skończy się powodzeniem, co pokazuje rysunek 4.9

- Przycisk "Show Plan", który zależnie od tego kiedy zostanie wciśnięty będzie zwracał jedną z dwóch wartości:
  - "The plan is not yet generated.", gdy plan nie został jeszcze wygenerowany,
  - Wygenerowany plan, podobnie jak w przypadku rysunku 4.9.
- Przyciski "Show Nautical Twilights"oraz "Show Satellite Passes"bez uprzedniego wygenerowania planu zwrócą wartość "Generate plan first.". W przypadku, gdy ten został już wygenerowany zwracają one wartość:
  - Wszystkich przelotów satelitów nad punktami obserwacyjnymi dla przycisku "Show Satellite Passes", tak jak przedstawia to rysunek 4.10,
  - Wszystkich przedziałów czasowych pomiędzy zachodem, a wschodem żeglarskim dla danych punktów obserwacyjnych w przypadku przycisku "Show Nautical Twilights", co pokazuje rysunek 4.11
- Przycisk "Choose Folder"otworzy okno do wybierania ścieżki, a po jej wybraniu wyświetli ją w oknie Application Feedback. Oba te przypadki można zaobserwować w rysunku 4.12.
- Przycisk "Export Content"eksportuje dane znajdujące się w oknie podglądu do pliku tekstowego o odpowiedniej nazwie, inkrementując co generację planu co

przedstawia rysunek 4.13.

– Przycisk "Export as Gantt Chart"eksportuje dane planu w formie wykresu Gantt'a w formie widocznej pod rysunkiem 4.14.

– Pola tekstowe "Start Date"oraz "End Date"opisują czas rozpoczęcia i zakończenia obserwacji według podanego wzorca. Ich domyślne wartości to odpowiednio aktualna data systemowa i jeden dzień do przodu.

4.4 Struktura wejściowych plików JSON

Program ten wykorzystuje dwa typy plików wejściowych JSON. Pierwszy z nich opisuje dane TLE satelitów, których obserwacją użytkownik jest zainteresowany, pozyskiwany ze strony dostarczającej je - space-track, a dokładniej plik wykorzystujący klasę tle_latest [7]. Drugi natomiast opisuje dane pozycji z których te będą obserwowane, więc zakładane jest, że użytkownik go zbuduje.

Name	**Description**
PRIORITY	Liczba całkowita większa lub równa 0 .
COMMENT	Komentarz dotyczący lokalizacji, zakłada się przykładowo miasto.
START_TIME	Pierwsza, wcześniejsza data określająca początek dostępności obserwacji w danym miejscu.
END_TIME	Druga, późniejsza data określająca koniec dostępności obserwacji w danym miejscu .
ANGLE	Kąt od którego można obserwować w danym punkcie obserwować satelity.
LATITUDE	Szerokość geograficzna punktu.
LONGITUDE	Wysokość geograficzna punktu.
TIMEZONE_OFFSET	Jest to pozostałość po funkcji uwzględniającej strefę czasową, zakłada się wartość 0.

Tabela 4.1: Struktura pliku JSON punktów obserwacyjnych

Rozdział 5

Specyfikacja wewnętrzna

5.1 Przegląd ważniejszych fragmentów kodu

5.1.1 generate_plan.py

Plik ten zawiera funkcję opisującą sposób tworzenia pliku wyjściowego programu poprzez wczytanie danych startowych i iteracyjne wycinanie z nich przedziałów, które interesują użytkownika.

```
1     obs_time = []
2         for obs in observatories:
3             start_time = obs['START_TIME']
4             end_time = obs['END_TIME']
5             tz = []
6             if obs['TIMEZONE_OFFSET']:
7                 tz = pytz.FixedOffset($60 * obs['TIMEZONE_OFFSET
                      '])
8             else:
9                 tz = pytz.FixedOffset(0)
10            start_time = start_time.replace(tzinfo=tz)
11            end_time = end_time.replace(tzinfo=tz)
12            obs_time.append([(start_time, end_time)])
```

Powyższy fragment kodu inicjuje listę list czasów punktów obserwacyjnych. Dla upewnienia się, że kod będzie kompatybilny z biblioteką skyfield przypisywana jest strefa czasowa (która, zakłada się przy aktualnym działaniu programu, wynosi 0).

```
1     start_time_sat = start_time_sat.replace(tzinfo=tzp)
2     end_time_sat = end_time_sat.replace(tzinfo=tzp)
3     for sat in satellites:
4         sat_time.append([(start_time_sat, end_time_sat)])
```

Kod ten podobnie do poprzedniego odpowiedzialny jest za inicjalizację listy list przedziałów czasu, z których będą wydzielane obliczone czasy widoczności.

```
1         for time_range_sat_orig, sat in zip(sat_time, satellites
              ):
2         plan_sat = []
3             for time_range_obs_orig, obs in zip(obs_time,
                  observatories):
4                 time_range_obs = time_range_obs_orig.copy()
5                 time_range_sat = time_range_sat_orig.copy()
```

Główne pętle po których program będzie przechodził. Jako, że pierwszą pętlą są satelity i ich listy przedziałów czasu, będą one wpisywać się iteracyjnie do kolejnych punktów obserwacyjnych zapełniając pierw swój czas zaczynając od najwyższych priorytetów, tym samym zapewniając, że nie będą pojawiać się w tym samym czasie dla różnych punktów. Operowanie na kopiach sprawia, że listy będą dopiero edytowane, gdy zostaną znalezione przedziały interesujące użytkownika.

```
1     ov = observatory_visibility(obs['START_TIME'], obs['END_TIME
          '], obs['LATITUDE'], obs['LONGITUDE'], obs['
          TIMEZONE_OFFSET'], event_type_plan)
2     ovv = []  # observatory_visibility_verified
3     for time_frame in ov:
4         if time_frame:
5             process_overlaps(time_frame, time_range_obs, ovv)
6
7     ovvv = []
8     for time_frame in ovv:  # look for overlaps of visibility
          and available satellite time (verified, twice)
9         if time_frame:
10            process_overlaps(time_frame, time_range_sat, ovvv)
```

Kod ten sprawdza, kiedy punkt obserwacyjny będzie miał możliwość wykonywania obserwacji ze względu na wystąpienie przedziałów pomiędzy zachodem, a wschodem żeglarskim. Następnie wycinane są te kawałki z czasu obserwacji satelitów i obserwatoriów (wyciągana część wspólna).

```
1     for tf in ovvv:
2         pl = satellite_visibility(sat['TLE_LINE1'], sat['
              TLE_LINE2'], tf[0], tf[1], obs['ANGLE'], obs['
              LATITUDE'], obs['LONGITUDE'])
3         if pl:
```

```
4             for pass_ in pl:  # for now, append start and end
                  times
5                 if len(pass_) in {5, 4}:
6                     ps.append((pass_[0], pass_[2]))
7                 elif len(pass_) == 3:
8                     ps.append((pass_[0], pass_[1]))
```

W czasie wspólnym poprzednich operacji wyszukiwane są momenty, gdy satelita będzie w polu widzenia obserwatora i dopisuje je do listy przedziałów obserwacji.

5.1.2 process_overlaps.py

```
1 def process_overlaps(input_tuple, tuple_list, overlap_list):
2     input_start, input_end = input_tuple
3     new_tuples = []
4     for idx, (start, end) in enumerate(tuple_list):
5         if start < input_end and end > input_start:
6             # Calculate the overlap
7             overlap_start = max(start, input_start)
8             overlap_end = min(end, input_end)
9             overlap = (overlap_start, overlap_end)
10
11            # Add overlap to overlap_list if not already present
12            if overlap not in overlap_list:
13                overlap_list.append(overlap)
14
15                # Modify the original tuple_list
16                start_check = start == overlap_start
17                end_check = end == overlap_end
18                if not (start_check and end_check):
19                    if start_check:  # overlap at start, append
                          2nd part as free
20                        new_tuples.append((overlap_end, end))
21                    elif end_check:  # overlap at end, append 1
                          st part as free
22                        new_tuples.append((start, overlap_start)
                              )
23                    else:  # overlap in middle, append 1st and 3
                          rd part as free
```

```
24                              new_tuples.append((start, overlap_start)
                                    )
25                              new_tuples.append((overlap_end, end))
26
27          else: # If no changes will be made, append old data to
                new list
28              new_tuples.append((start, end))
29
30      tuple_list.clear()
31      tuple_list.extend(new_tuples)
```

Funkcja ta szuka wejściowego przedziału czasu w liście przedziałów, a gdy je znajdzie dopisuje je do drugiej listy. Ostatecznie lista w której dokonywano wyszukiwania jest zastąpiona przez nowo stworzoną listę bez przedziałów wspólnych.

5.1.3 satellite_visibility.py

```
1       from skyfield.api import Topos, load, EarthSatellite
2   import pytz
3
4   def satellite_visibility(line1, line2, start_time, end_time,
        angle, latitude, longitude):
5       # Observer's location
6       if end_time <= start_time or not start_time or not end_time:
7           return []
8       observer = Topos(latitude_degrees=latitude,
            longitude_degrees=longitude)
9
10      # Load timescale and satellite
11      ts = load.timescale()
12      satellite = EarthSatellite(line1, line2)
13
14      # Find events
15      t0 = ts.utc(start_time.year, start_time.month, start_time.
            day, start_time.hour, start_time.minute,
16                  start_time.second)
17      t1 = ts.utc(end_time.year, end_time.month, end_time.day,
            end_time.hour, end_time.minute, end_time.second)
18      times, events = satellite.find_events(observer, t0, t1,
            altitude_degrees=angle)
```

```
19
20      # Process events into tuples
21      passes = []
22      pass_events = []
23      for t, event in zip(times, events):
24          pass_events.append(t.utc_datetime())
25          if event == 2 and len(pass_events) == 3:  # set event
26              passes.append((pass_events[0], pass_events[1],
                    pass_events[2], satellite.model.satnum, True))
27              pass_events = []
28      tz = pytz.FixedOffset(0)
29      if len(pass_events) == 2:
30          passes.append((pass_events[0], pass_events[1], end_time.
                replace(tzinfo=tz), satellite.model.satnum))
31      elif len(pass_events) == 1:
32          passes.append((pass_events[0], end_time.replace(tzinfo=
                tz), satellite.model.satnum))
33      return passes
```

Funkcja ta wyszukuje w podanym przedziale określonego satelity, gdy dla danego punktu obserwacyjnego ten przekroczy daną ilość stopni kątowych względem horyzontu. Zwraca listę wydarzeń i ich czasów spełniających warunki, które korzystając z algorytmu SGP4 są przewidywane, ze specjalnymi warunkami, gdy te się dalej będą mieścić w wymaganiach bez znalezionych wydarzeń końcowych.

5.1.4 PlanViewer.py

Kod źródłowy PlanViewer.py dodatku **??**
Jest to klasa wywołująca pozostałe funkcje pozwalające na obliczanie, wyświetlanie i eksport danych tworzonych przez program.

5.1.5 UniquePriorityPicker.py

```
1   import random
2
3   class UniquePriorityPicker:
4       def __init__(self, init_integer):
5           self.init_integer = init_integer
6           self.multiplier = 1
7           self.calculate_multiplier()
```

```
8           self.integer_list = self.create_integer_list()
9       def create_integer_list(self):
10          # Creates a list from 0 to init_integer
11          return list(range(self.init_integer + 1))
12
13      def calculate_multiplier(self):
14          # Multiplies self.multiplier by 10 until it's larger
                than init_integer
15          while self.multiplier <= self.init_integer:
16              self.multiplier *= 10
17
18      def get_random_integer_and_multiplier(self):
19          # Returns a tuple of a random integer from the list (and
                removes it) and the multiplier
20          if not self.integer_list: # Checks if the list is empty
21              return None, self.multiplier # Returns None if the
                    list is empty
22
23          random_integer = random.choice(self.integer_list)
24          self.integer_list.remove(random_integer)
25          return random_integer, self.multiplier
```

UniquePriorityPicker to prosta klasa generująca unikatowe losowe liczby i mnożniki sprawiające, że dodawane do priorytetów liczby wyciągane z jej listy nie będą zmieniać priorytetów użytkowników.

5.1.6 App.py

Kod źródłowy klasy App **??**

Jest to klasa, która umożliwia zmianę widoków w aplikacji. Zawiera również kilka funkcji testowych ułatwiających szukanie błędów, które standardowo ukryte są poprzez przełącznik self.debug wymagający zmiany w kodzie.

5.1.7 JsonFileViewer.py

Kod źródłowy klasy JsonFileViewer **??** Klasa umożliwiająca wczytywanie danych z plików JSON, rzucanie na odpowiednie typy, tworząca GUI do obsługi priorytetów obiektów i zwracanie kopii danych z dodanym elementem losowości dla tych, które mają ten sam priorytet.

Rozdział 6

Weryfikacja i walidacja

6.0.1 Sposób testowania pracy

Praca przez okres jej tworzenia była aktywnie testowana przyrostowo po dodaniu nowych funkcjonalności. Eksperymenty, które były przeprowadzane sprawdzały zarówno typowe działanie programu jak i jego warunki brzegowe. Szczególny nacisk został nałożony na testowanie funkcji obliczających przedziały czasowe korzystając z SGP4 i efemeryd do określenia przedziałów czasowych w których pozycje satelitów względem obserwatora spełniały warunki teoretyczne do wykonywania obserwacji oraz funkcja generująca plan.

6.0.2 Stanowisko testowe

Testowanie funkcjonalności programu zostało przeprowadzone na stacji roboczej wyposażonej w następujące podzespoły i oprogramowanie:

- System operacyjny: Windows 10 Education, Wersja 22H2, 64-bit
- CPU: Intel Core i5-3570K
- GPU: NVidia Geforce GTX 1060 3gb
- Pamięć RAM: 16GB DDR3 1600Mhz
- Płyta główna: Asus Z77 Sabertooth

6.0.3 Najważniejsze wykryte błędy i ich rozwiązania

Środowiskiem testowym był program uruchomiony przez IDE Pycharm w trybie debugowania. Testy były wykonywane względem danych pozyskanych z serwisów internetowych heavens-above.com dla testowania momentów pojawienia się satelit nad lokalizacją oraz kalendarz.livecity.pl dla testowania pór zmierzchów.

- Opóźnienia na poziomie minut dla przedziałów czasowych przelatujących w polu widzenia obserwatora satelitów. Powodem okazały się przestarzałe dane TLE opisujące orbity. Błąd ustąpił po ich aktualizacji.
- Nieskończona pętla generowania planu. Powodem okazało się ciągłe dopisywanie do listy po której program iterował. Rozwiązaniem okazało się zmienienie sposobu działania funkcji, aby ta operowała pierw na kopii danych, a dopiero po znalezieniu nowych przedziałów dopisywała pozostałości zużytych przedziałów na koniec listy.

Ponadto w czasie tworzenia oprogramowania zostało zlikwidowanych wiele pomniejszych problemów takich jak niepoprawne sposoby wyświetlania danych poprzez ich przesortowanie, zmianę sposobu dzielenia czasów z powodu wykorzystania nieodpowiedniej funkcji logicznej oraz wiele innych jak widać na rysunku **??**.

Rozdział 7

Podsumowanie i wnioski

Cel projektu, czyli stworzenie projektu i implementacji systemu do planowania obserwacji satelitów orbity ziemskiej został osiągnięty. Określone cele funkcjonalne i niefunkcjonalne zostały zrealizowane.
O ile system wykonuje swoje zadania zauważyć można, że dodatkowe funkcje mogłyby być dalej implementowane takie jak wizualizacje podobne do tych jakie oferują serwisy jak heavens-above. Ponadto prócz aktualnego, opartego o określone przedziały czasowe sposobu wyszukiwania wydarzeń dotyczących satelitów możliwym byłoby wyszukiwanie określonej ilości najbliższych wydarzeń.
Jednym z aspektów, które mogłyby być także poprawione jest sam wygląd aplikacji. W aktualnym jego stanie spełnia swoje zadanie, ale jest bardzo prosty, bez charakteru. Dodanie chociażby animacji, które dawałyby znak użytkownikowi, że obliczenia są w trakcie wykonywania, a nie, że program się zaciął poprawiłoby jego czytelność.
Oczywiście ze względu na to, że jest to program zajmujący się obserwacją obiektów astronomicznych inną drogą rozbudowy mogłoby być przewidywanie innych typów obiektów takich jak planety, czy gwiazdy.
Najtrudniejszym w projekcie definitywnie było zrozumienie tematyki na poziomie umożliwiającym wykonanie projektu oraz nauczenie się nowego języka programowania.

Bibliografia

[1] The Editors of Encyclopaedia Britannica. *"seeing". Encyclopedia Britannica*. 2024. URL: https://www.britannica.com/science/seeing.

[2] Python Software Foundation. *History and License*. 2024. URL: https://docs.python.org/3/license.html (term. wiz. 05. 02. 2024).

[3] David Lorenz. *Dark site finder*. 2024. URL: https://darksitefinder.com/maps/world.html (term. wiz. 04. 02. 2024).

[4] NASA. *"Catalog of Earth Satellite Orbits"*. 2009. URL: https://earthobservatory.nasa.gov/features/OrbitsCatalog/page2.php (term. wiz. 04. 09. 2009).

[5] Brandon Rhodes. *Skyfield: Generate high precision research-grade positions for stars, planets, moons, and Earth satellites*. Wer. 1.17. Lut. 2020. URL: https://ui.adsabs.harvard.edu/abs/2019ascl.soft07024R.

[6] SAIC. *Basic Description of the Two Line Element (TLE) Format*. 2024. URL: https://www.space-track.org/documentation#/tle (term. wiz. 05. 02. 2024).

[7] SAIC. *Description of tle_latest class*. 2024. URL: https://www.space-track.org/basicspacedata/modeldef/class/tle_latest/format/html (term. wiz. 05. 02. 2024).

[8] NASA Science. *Introduction to the Electromagnetic Spectrum*. 2010. URL: http://science.nasa.gov/ems/01_intro (term. wiz. 04. 02. 2024).

[9] Jurij Stare. *Light Pollution Map*. 2024. URL: https://www.lightpollutionmap.info/ (term. wiz. 04. 02. 2024).

Dodatki

Spis skrótów i symboli

UV Ultrafiolet (ang. *Ultraviolet*)

IR Podczerwień (ang. *infrared*)

FIR Daleka Podczerwień (ang. *far-infrared*)

LEO Niska orbita okołoziemska (ang. *Low Earth Orbit*)

MEO Średnia orbita okołoziemska (ang. *Medium Earth Orbit*)

GEO Orbita geostacjonarna (ang. *Geostationary Orbit*)

SSO Orbita heliosynchroniczna (ang. *Sun-synchronous orbit*

TLE Element dwuliniowy (ang. *Two-Line Element*)

SGP4 Uproszczony model orbity satelity dotyczący ogólnych zaburzeń (ang. *Simplified General Perturbations Satellite Orbit Model 4*)

NORAD Dowództwo Obrony Północnoamerykańskiej Przestrzeni Powietrznej i Kosmicznej (ang. *North American Aerospace Defense Command*)

NASA Narodowa Agencja Aeronautyki i Przestrzeni Kosmicznej (ang. *National Aeronautics and Space Administration*)

JPL Laboratorium Napędu Odrzutowego (ang. *Jet Propulsion Laboratory*)

WGS84 Światowy System Geodyzyjny 1984 (ang. *World Geodetic System 1984*)

SDP4 Uproszczony model zaburzeń głębokiej przestrzeni kosmicznej 4 (ang. *Simplified Deep Space Perturbations model 4*),

JSON ang. JavaScript Object Notation,

GUI ang. Graphical User Interface,

PNG ang. Portable Network Graphics,

SVG ang. Scalable Vector Graphics,

IDE ang. Integrated Development Environment,

CPU ang. Central Processing Unit

Spis rysunków

Spis tabel