Programm 1: Drohnengetragenes Laserscanning

Zur Erstellung eines digitalen Oberflächenmodells des Gebäudes Steyrergasse 30 wurde ein nach unten gerichteter Laserscanner an eine Drohne montiert. Die Drohne flog eine vorgegebene Trajektorie über dem Projektgebiet ab und nahm dabei Punkte unter sich auf. Um die genaue Trajektorie der Drohne zu bestimmen, wurde diese von einem Tachymeter vollautomatisch getrackt.

Implementierung

Die Tachymeterbeobachtungen sind in der Datei `obsTachy.txt` enthalten. Das Dateiformat enthält folgende Spalten: *Zeit in Sekunden, Distanz in Meter, Zenitwinkel in Radiant, Azimutwinkel in Radiant*. Die Zeit bezieht sich dabei auf den Referenzzeitpunkt 2018-03-13 15:10:00. Die Beobachtungen sind zeilenweise gespeichert.

Die Beobachtungen des Laserscanners befinden sich in der Datei `obsDrone.txt`. Das Dateiformat ist gleich wie bei den Tachymeterbeobachtungen.

Daten einlesen

Implementiere zur Speicherung/Verarbeitung der Beobachtungsdaten eine Klasse `Epoch` mit einem Attribut `time`. Das Attribut soll als *Property* implementiert werden, so dass der Zugriff von außen kontrolliert über *private Get- und Set-Funktionen* erfolgt. In der Set-Funktion soll überprüft werden, ob der übergebene Wert dem Datentyp `datetime` des Python-Moduls `datetime` entspricht. Ist dies nicht der Fall, soll eine *Exception* mit einer verständlichen Fehlermeldung geworfen werden. Beim Anlegen eines Objekts dieser Klasse *muss* ein Zeitpunkt übergeben werden.

Implementiere zwei Unterklassen `PositionEpoch` und `PolarEpoch`, die von der Klasse `Epoch` erben und somit die Zeitfunktionalität übernehmen. Die Klasse `PositionEpoch` enthält zusätzlich drei Attribute für die 3D-Koordinaten `x, y, z`. Für die Klasse `PolarEpoch` kommen die Attribute `distance, zenith, azimuth` hinzu. Überlege, ob und nach welchen Kriterien die übergebenen Werte in beiden Klassen überprüft werden können. Etwaige Überprüfungen sollen ebenfalls in privaten Get- und Set-Funktionen implementiert werden, so dass der Zugriff in Form von Properties erfolgt. Beim Anlegen von Objekten der jeweiligen Klassen sollen neben dem (notwendigen) Zeitpunkt optional Werte für die Attribute übergeben werden können.

Die Beobachtungen des Tachymeters und des Laserscanners sollen jeweils als Liste von Objekten der Klasse `PolarEpoch` eingelesen werden.

Daten ordnen

Aufgrund eines Softwarefehlers im Tachymeter sind die Daten nicht zeitlich sortiert, sondern in zufälliger Reihenfolge ausgegeben worden. Implementiere eine Funktion `bubbleSort`, welche eine Liste von beliebigen Objekten "in place" (direkt die übergebene Liste, keine Kopie als Rückgabe) mittels Bubble-Sort-Algorithmus (siehe VO) sortiert. Erweitere die Klasse `Epoch` so, dass eine Liste von Epoch-Objekten zeitlich aufsteigend sortiert werden kann.

Implementiere eine einfache Klasse `Timer`, die zusammen mit dem `with`-Keyword zur Zeitmessung verwendet werden kann. Der aktuelle Zeitpunkt kann z.B. über die Funktion `time.process_time()` aus dem Modul `timeit` ermittelt werden.

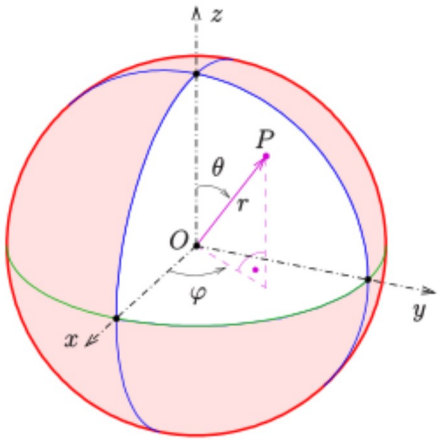
In [1]:
```
with Timer():
    pass # do something
# when finished, required time is automatically printed
```

Sortiere die Beobachtungen des Tachymeters mit der implementierten BubbleSort-Funktion und messe die dafür benötigte Zeit mit der selbst implementierten Timer-Klasse. Zur Veranschaulichung soll eine *Kopie* der Liste mit unsortierten Tachymeterbeobachtungen mit der Python-internen Sortierfunktion (`someList.sort()`) sortiert werden. Die dafür benötigte Zeit soll ebenfalls via Timer gemessen und mit der von BubbleSort verglichen werden.

Drohnentrajektorie berechnen

Berechne die Trajektorie der Drohne basierend auf den Tachymeterbeobachtungen. Die berechneten Koordinaten sollen zeitlich sortiert als Liste von Objekten der Klasse `PositionEpoch` gespeichert werden. Die Formeln zur Berechnung von 3D-Koordinaten aus Tachymeterbeobachtungen (Distanz r, Zenitwinkel θ, Azimutwinkel φ) lauten:

$$x = r \cdot \sin\theta \cdot \cos\varphi$$
$$y = r \cdot \sin\theta \cdot \sin\varphi$$
$$z = r \cdot \cos\theta$$



Weitere Informationen unter: https://de.wikipedia.org/wiki/Kugelkoordinaten (https://de.wikipedia.org/wiki/Kugelkoordinaten)

Die Koordinaten des Tachymeters sind zeitlich konstant und bekannt: `x = -51.280 m, y = -4.373 m, z = 1.340 m`
Etwaige Exzentrizitäten sind bereits an die Beobachtungen angebracht und müssen nicht berücksichtigt werden.

Bodenpunkte berechnen und gittern

Berechne die mittels Laserscanner gemessenen Bodenpunkte ausgehend von den Positionen der Drohne zur jeweiligen Epoche (Messzeitpunkt). Da es sich um die gleiche Art von Beobachtungen wie beim Tachymeter handelt, können erneut die oben angeführten Formeln verwendet werden. Die Anzahl der gemessenen Bodenpunkte unterscheidet sich je nach Epoche. Es können auch Epochen ohne Laserscannerbeobachtungen vorkommen. Daher ist es notwendig, jeder Epoche (Position der Drohne zu einem bestimmten Zeitpunkt) die entsprechenden Laserscannerbeobachtungen zuzuordnen.

Die berechneten Bodenpunkte sollen in ein regelmäßiges Raster mit einer Auflösung von 50 cm in X- und Y-Richtung gegittert werden. Kommen in einer Gitterzelle mehrere Bodenpunkte vor, so ist der Mittelwert der Z-Werte aller Punkte zu berechnen und der Gitterzelle zuzuordnen.

Visualisierung

Die Ergebnisse sollen auf ansprechende Weise in verschiedenen Varianten mittels `matplotlib` grafisch dargestellt werden. Interessant sind unter anderem die Trajektorie der Drohne (z.B. Horizontal- und Vertikalkomponenten), das Oberflächenmodell (z.B. 2D mit farbkodierter Höheninformation, 3D, original vs. gegittert, Aufnahmezeitpunkt farbcodiert, etc.) oder auch die Beobachtungsgrößen. Kombinierte Darstellungen (Oberfläche, Trajektorie, ...) sind ebenfalls wünschenswert.

Abgabe

Die Abgabe besteht aus zwei Teilen, dem **Quellcode** und dem **technischen Bericht**. Der Quellcode (aber nicht die gegebenen Beobachtungsdateien) und der technische Bericht (als `pro01DroneLaserScanning.pdf`) müssen in einem Archiv `pro01DroneLaserScanning.zip` bis Donnerstag, 2. Mai 2019, 13:00 Uhr im TeachCenter (https://tc.tugraz.at/main/course/view.php?id=938) abgegeben werden.

Quellcode

Die Formatierung und Präsentation des Quellcodes soll dem Verständnis förderlich sein, achte deshalb auf folgende Gesichtspunkte:

- Dateikopf mit Namen des Erstellers und Projekt in jeder Datei.
- Verwendung deskriptiver Variablennamen, z. B. `tachymeterEpoch` anstatt `te`.
- Erklärende Kommentare. Wieso ist dieser Quellcode so aufgebaut wie er ist? Welche Formel wird hier implementiert? Welches Ergebnis steht am Ende des Quellcodeabschnittes?

Weiters soll der Quellcode auch inhaltlich einer erkennbaren Struktur folgen:

- Sinnvoller Einsatz von Funktionen z. B. zur Vermeidung von Wiederholungen. Klar definierte Schnittstellen auch und speziell für Klassen (Mit Beschreibung von Parametern und zu erwartenden Rückgabewerten).
- Strukturierte Abfolge der Anweisungen, z. B. zuerst Abarbeitung eines Aufgabenteils gefolgt von Anweisungen zum nächsten Aufgabenteil.
- Variablen so lokal wie möglich deklarieren, im Idealfall erst bei der ersten Verwendung.

Technischer Bericht - Form

Einige formelle Kriterien sollen eingehalten werden:

1. Die Länge des technischen Berichts sollte 8 Seiten Inhalt (ohne Titelblatt, Inhaltsverzeichnis, Anhänge) nicht überschreiten.
2. Der technische Bericht besitzt ein Titelblatt. Dieses enthält:
    - Lehrveranstaltungsname, -nummer und -jahr