Odczyt i wizualizacja siatek trójkątnych

Dawid Kwapisz
dawid.kwapisz@student.pk.edu.pl
Nr indeksu: 152915

5 czerwca 2024

1 Wprowadzenie

Siatki trójkątne są powszechnie stosowane w grafice komputerowej, modelowaniu 3D, symulacjach fizycznych oraz wielu innych dziedzinach. Składają się one z wierzchołków (punktów) oraz trójkątów, które łączą te wierzchołki, tworząc powierzchnie. Efektywna wizualizacja siatek trójkątnych pozwala na analizę kształtów, struktur i właściwości modeli 3D.

Celem niniejszego projektu jest zbadanie i porównanie różnych narzędzi i bibliotek dostępnych w języku Python do odczytu oraz wizualizacji siatek trójkątnych. Podczas testowania skupiono się na pięciu popularnych bibliotekach:

- Matplotlib [1]
- Plotly [2]
- Open3D [3]
- VTK [4]
- Mayavi [5]

Projekt ten ma na celu dostarczenie przeglądu możliwości każdej z tych bibliotek, a także ich zastosowań w kontekście wizualizacji danych trójwymiarowych.

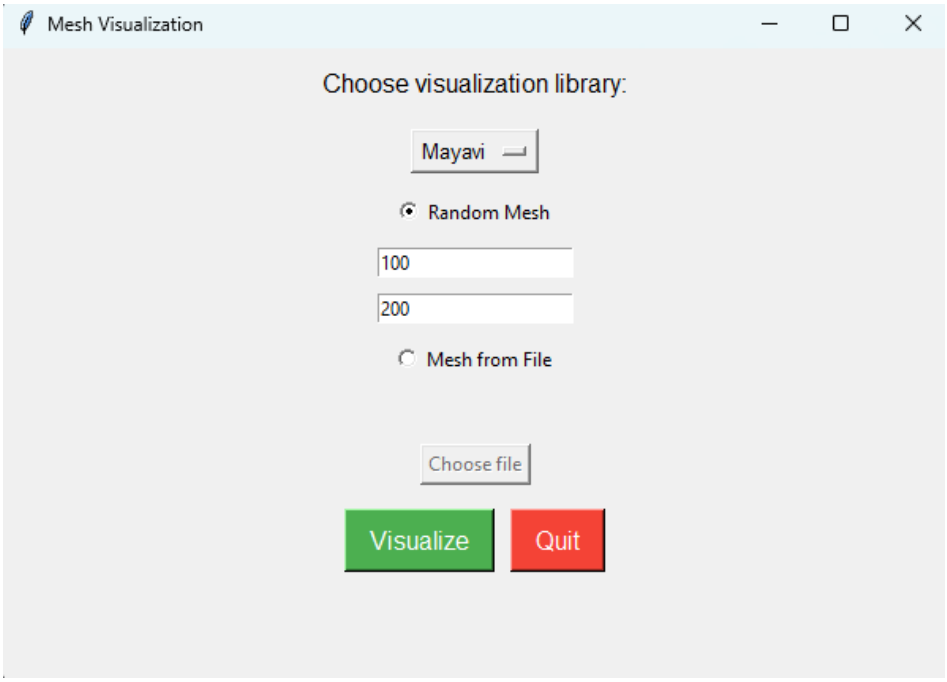
2 Aplikacja testowa

2.1 Specyfikacja aplikacji

Aplikacja została napisana w języku Python 3.8 oraz posiada prosty interfejs użytkownika zbudowany za pomocą biblioteki Tkinter. Interfejs umożliwia wybór biblioteki do wizualizacji, wybór trybu generowania siatek (losowy lub z pliku), wprowadzenie liczby punktów i trójkątów do wygenerowania (w trybie losowym), wybór pliku z siatką (w trybie z pliku) oraz uruchomienie wizualizacji. Jest ona dostępna do pobrania w serwisie Github [6].

2.2 Funkcje aplikacji

- **Generowanie siatek trójkątnych:** Aplikacja umożliwia generowanie losowych siatek trójkątnych. Użytkownik może określić liczbę punktów i trójkątów do wygenerowania.
- **Wczytywanie siatek z pliku:** Aplikacja umożliwia wczytywanie siatek trójkątnych z pliku. Format pliku musi być zgodny z wybraną biblioteką do wizualizacji.
- **Wizualizacja siatek:** Aplikacja umożliwia wizualizację siatek trójkątnych za pomocą różnych bibliotek do wizualizacji 3D: Matplotlib, Plotly, Open3D, VTK, Mayavi. Użytkownik może wybrać, którą bibliotekę chce użyć do wizualizacji.



Rysunek 1: GUI aplikacji

2.3 Struktura projektu

Projekt składa się z kilku modułów:

- **Main.py:** Główny moduł aplikacji, zawiera definicję interfejsu użytkownika i logikę aplikacji.
- **Utils/MeshGenerator.py:** Moduł odpowiedzialny za generowanie siatek trójkątnych - zarówno losowych, jak i z pliku.
- **Utils/Validation.py:** Moduł odpowiedzialny za walidację formatu pliku z siatką.
- **Visualizers/MatplotlibVisualizer.py, Visualizers/MayaviVisualizer.py, Visualizers/Open3DVisualizer.py, Visualizers/PlotlyVisualizer.py, Visualizers/VTKVisualizer.py:** Moduły odpowiedzialne za wizualizację siatek za pomocą różnych bibliotek. Każdy z tych modułów definiuje klasę `Visualizer`, która ma metodę `visualize()`.

3 Przetestowane biblioteki

W ramach projektu przetestowano kilka bibliotek do wizualizacji danych 3D w Pythonie. Każda z nich oferuje unikalne funkcje i możliwości, które mogą być przydatne w różnych scenariuszach. Poniżej przedstawiono krótki opis każdej z przetestowanych bibliotek.

3.1 Matplotlib

3.1.1 Opis

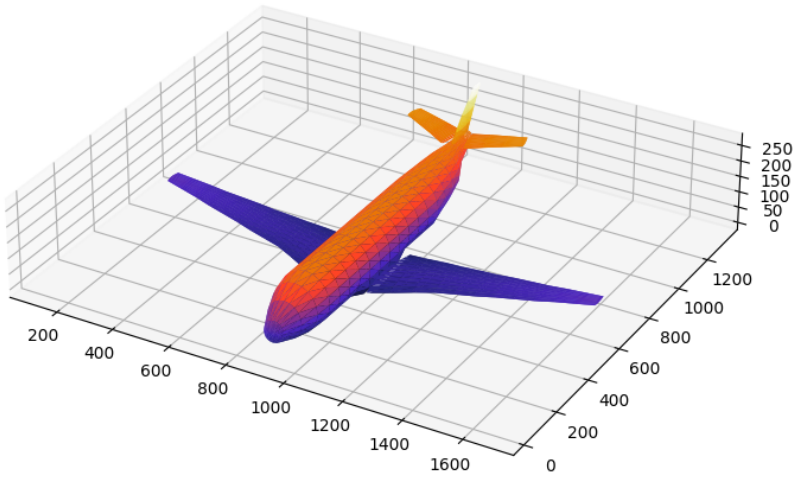
Matplotlib jest jedną z najprostszych bibliotek, jakie można wykorzystać w Pythonie do odczytu i wizualizacji siatek trójkątnych. Implementacja funkcji do wczytywania i wizualizacje jest bardzo prosta - wykorzystuje ona standardowy moduł **pyplot** z biblioteki Matplotlib oraz moduł **tri** do przeprowadzenia Triangulacji. Pozwala ona na wczytywanie plików z siatkami trójkątnymi o rozszerzeniach: *.vtk*, *.vtp*, *.ply*, *.stl* oraz *.obj*. Duża zaletą, a jednocześnie wadą tej biblioteki jest jej prostota - implementacja jest bardzo lekka, jednak nie oferuje zbyt wielu zaawansowanych opcji. Dodatkowo, dużym ograniczeniem jest brak interaktywności w wygenerowanej wizualizacji - wynikiem działania biblioteki jest obraz, który pozbawiony jest interaktywności (np. za pomocą myszki). Parametry takie jak obrót, przybliżenie można konfigurować jedynie z poziomu kodu.

3.1.2 Implementacja

```python
class MatplotlibVisualizer:
    def __init__(self, mesh):
        self.mesh = mesh

    def visualize(self):
        fig = plt.figure(figsize=(12, 8))
        ax = fig.add_subplot(111, projection='3d')

        x = self.mesh.points[:, 0]
        y = self.mesh.points[:, 1]
        z = self.mesh.points[:, 2]

        triangles = tri.Triangulation(x, y, self.mesh.triangles)

        ax.plot_trisurf(triangles, z, cmap=plt.cm.CMRmap)

        ax.set_box_aspect([np.ptp(x), np.ptp(y), np.ptp(z)])

        plt.show()
```

Rysunek 2: Implementacja dla biblioteki Matplotlib

3.1.3 Wizualizacja



Rysunek 3: Wizualizacja przy użyciu biblioteki Matplotlib

3.2 Plotly

3.2.1 Opis

Biblioteka Plotly jest już nieco bardziej zaawansowana w stosunku do biblioteki Matplotlib - przede wszystkim oferuje interaktywność. Biblioteka dostarcza dość wysoki poziom abstrakcji, dzięki czemu tworzenie skomplikowanych wykresów jest znacznie ułatwione. Biblioteka ta dobrze współpracuje z narzędziami takimi jak Jupyter Notebook czy Dash oraz z różnymi językami programowania (Python, R, Julia), dzięki czemu jej znajomość może ułatwić pracę, gdy jest potrzeba wykorzystania różnych narzędzi czy języków programowania. Plotly wspiera takie rozszerzenia jak: *.vtk*, *.vtp*, *.ply*, *.stl* oraz *.obj*. Implementacja przedstawiona na rysunku 4 tworzy prostą wizualizację, która jest dostępna

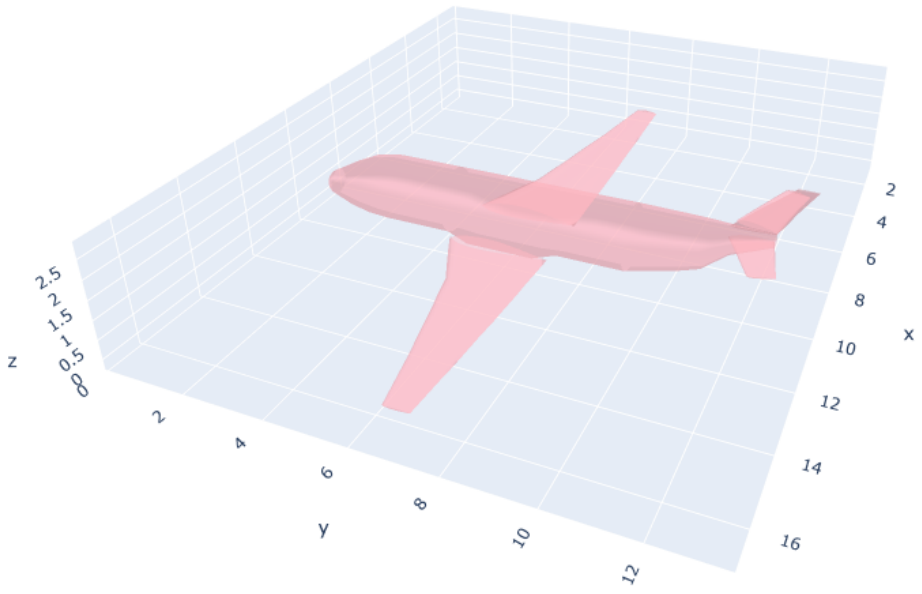
w przeglądarce pod adresem localhosta z odpowiednim portem. Wizualizacja jest interaktywna, jednak nie jest pozbawiona wad. Podczas implementacji wczytywania siatki przedstawionej na rysunku 5 konieczne było specjalne przygotowanie kodu, aby biblioteka wyświetliła wizualizacje w odpowiednich proporcjach, w innym przypadku obraz był dość mocno zniekształcony.

3.2.2 Implementacja

```python
class PlotlyVisualizer:
    def __init__(self, mesh):
        self.mesh = mesh

    def visualize(self):
        scale_factor = 0.01

        x = self.mesh.points[:, 0] * scale_factor
        y = self.mesh.points[:, 1] * scale_factor
        z = self.mesh.points[:, 2] * scale_factor

        fig = go.Figure(data=[go.Mesh3d(x=x, y=y, z=z, i=self.mesh.triangles[:, 0], j=self.mesh.triangles[:, 1],
                                        k=self.mesh.triangles[:, 2], color='lightpink', opacity=0.50)])

        max_range = np.array([x.ptp(), y.ptp(), z.ptp()]).max() * 0.5

        mid_x = (x.max() + x.min()) * 0.5
        mid_y = (y.max() + y.min()) * 0.5
        mid_z = (z.max() + z.min()) * 0.5
        x_eye = mid_x + max_range
        y_eye = mid_y + max_range
        z_eye = mid_z + max_range

        fig.update_layout(width=1200, height=800, autosize=False,
                          scene=dict(aspectratio=dict(x=np.ptp(x), y=np.ptp(y), z=np.ptp(z)), aspectmode='manual',
                                     camera=dict(eye=dict(x=x_eye, y=y_eye, z=z_eye))))

        fig.show()
```

Rysunek 4: Implementacja dla biblioteki Plotly

3.2.3 Wizualizacja



Rysunek 5: Wizualizacja przy użyciu biblioteki Plotly

3.3 Open3D

3.3.1 Opis

Biblioteka Open3D jest już bardziej zaawansowana niż Plotly i Matplotlib. Oferuje ona pełną interaktywność wizualizacji, a przy tym umożliwia prostą implementacje odczytu danych - potrzebne jest dosłownie kilka linii kodu, jak to przedstawiono na rysunku 6. Jest to bazowa implementacja, aczkolwiek biblioteka oferuje znacznie więcej - od zaawansowanego przetwarzania i analizy siatek trójkątnych, przez obsługę chmur punktów, aż po złożone algorytmy rekonstrukcji scen 3D. Dostarcza ona narzędzia np. do filtracji, wygładzania, segmentacji czy klasteryzacji siatek trójkątnych. Dodatkowo, biblioteka pozwala na integrację z takimi narzędziami jak Tensorflow czy OpenCV. Przykładowe formaty danych wspierane przez Open3D to: *.ply*, *.stl*, *.obj*, *.off*, *.gltf* oraz *.glb*.

3.3.2 Implementacja

```python
class Open3DVisualizer:
    def __init__(self, mesh):
        self.mesh = mesh

    def visualize(self):
        mesh_o3d = o3d.geometry.TriangleMesh()
        mesh_o3d.vertices = o3d.utility.Vector3dVector(self.mesh.points)
        mesh_o3d.triangles = o3d.utility.Vector3iVector(self.mesh.triangles)

        vis = o3d.visualization.Visualizer()
        vis.create_window(width=1200, height=800)
        vis.add_geometry(mesh_o3d)
        vis.run()
        vis.destroy_window()
```

Rysunek 6: Implementacja dla biblioteki Open3D

3.3.3 Wizualizacja

Rysunek 7: Wizualizacja przy użyciu biblioteki Open3D

3.4 VTK

3.4.1 Opis

Biblioteka VTK (ang. *Visualization Toolkit* jest jedną z najbardziej zawansowanych bibliotek do wizualizacji 3D w Pythonie. Oferuje ona bogaty zestaw narzędzi do przetwarzania, analizy oraz wizualizacji danych przestrzennych. Podstawowa implementacja funkcji do wczytywania i wizualizacji siatek trójkątnych w VTK - przedstawiona na rysunku 8 - jest najbardziej skomplikowaną implementacją spośród wszystkich 5 przetestowanych bibliotek, co może być wadą przy potrzebie szybkiego zwizualizowania danych. VTK obsługuje takie formaty jak: *.vtk*, *.vtp*, *.ply*, *.stl* oraz *.obj*.

3.4.2 Implementacja

```python
class VTKVisualizer:
    def __init__(self, mesh):
        self.mesh = mesh

    def visualize(self):
        points = vtk.vtkPoints()
        for point in self.mesh.points:
            points.InsertNextPoint(point)

        triangles = vtk.vtkCellArray()
        for triangle in self.mesh.triangles:
            vtk_triangle = vtk.vtkTriangle()
            vtk_triangle.GetPointIds().SetId(0, triangle[0])
            vtk_triangle.GetPointIds().SetId(1, triangle[1])
            vtk_triangle.GetPointIds().SetId(2, triangle[2])
            triangles.InsertNextCell(vtk_triangle)

        polydata = vtk.vtkPolyData()
        polydata.SetPoints(points)
        polydata.SetPolys(triangles)

        mapper = vtk.vtkPolyDataMapper()
        mapper.SetInputData(polydata)

        actor = vtk.vtkActor()
        actor.SetMapper(mapper)

        renderer = vtk.vtkRenderer()
        renderer.AddActor(actor)

        render_window = vtk.vtkRenderWindow()
        render_window.SetSize(1200, 800)
        render_window.AddRenderer(renderer)

        interactor = vtk.vtkRenderWindowInteractor()
        interactor.SetRenderWindow(render_window)

        render_window.Render()
        interactor.Start()
```

Rysunek 8: Implementacja dla biblioteki VTK

3.4.3 Wizualizacja

Rysunek 9: Wizualizacja przy użyciu biblioteki VTK

3.5 Mayavi

3.5.1 Opis

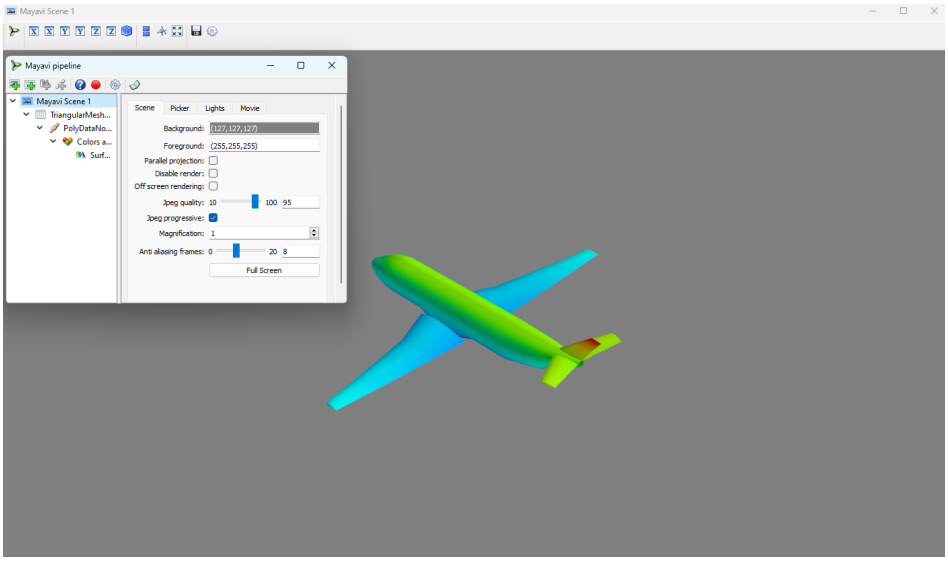
Biblioteka Mayavi to zaawansowane narzędzie do tworzenia interaktywnych wizualizacji 3D, które oferuje bardzo krótki i prosty kod do podstawowej implementacji wizualizacji siatek trójkątnych. Jest to jedna z jej głównych zalet, gdyż pozwala na szybkie uzyskanie wyników przy minimalnym wysiłku kodowania. Mayavi obsługuje formaty plików takie jak: *.vtk*, *.vtp*, *.ply*, *.stl* oraz *.obj*. Biblioteka ta wyróżnia się tym, że domyślnie udostępnia najwięcej opcji w okienku wizualizacji, co pozwala na intuicyjne dostosowanie widoku bez konieczności dodatkowego kodowania. Mayavi dobrze współpracuje z narzędziami takimi jak Jupyter Notebook, co czyni ją świetną opcją do szybkich analiz i prezentacji danych. Pomimo prostoty implementacji, Mayavi oferuje szeroki wachlarz zaawansowanych opcji, umożliwiając tworzenie skomplikowanych i estetycznych wizualizacji.

3.5.2 Implementacja

```python
class MayaviVisualizer:
    def __init__(self, mesh):
        self.mesh = mesh

    def visualize(self):
        mlab.figure(size=(1200, 800))
        mlab.triangular_mesh(self.mesh.points[:, 0], self.mesh.points[:, 1], self.mesh.points[:, 2],
                             self.mesh.triangles)
        mlab.show()
```

Rysunek 10: Implementacja dla biblioteki Mayavi

3.5.3 Wizualizacja



Rysunek 11: Wizualizacja przy użyciu biblioteki Mayavi

4 Podsumowanie

Każda z przetestowanych bibliotek posiada zalety i wady. Matplotlib jest zdecydowanie najprostszą biblioteką, jest pozbawiona interaktywności, jednak świetnie sprawdzi się przy generowaniu wizualizacji, które są potrzebne do umieszczenia w pracach naukowych czy raportach. Plotly jest nieco bardziej zaawansowaną biblioteką, zapewnia interaktywność, oraz umożliwia na przeprowadzenie wizualizacji w przeglądarce, co może być przydatne w momencie, gdy jest potrzeba umieszczenia wizualizacji np. na stronie internetowej. Open3D wyróżnia się pełną interaktywnością i zapewnia zaawansowane narzędzia, które mogą być pomocne podczas analizy danych z siatek trójkątnych. Dodatkowo, Open3D wspiera dodatkowe rozszerzenia plików, co jest jego zaletą w porównaniu do pozostałych 4 bibliotek. VTK oferuje wiele narzędzi do pracy z danymi przestrzennymi, jednak wymaga największego nakładu pracy do wykonania prostej wizualizacji. Ostatnia z przetestowanych bibliotek - Mayavi - zapewnia zdecydowanie największą interaktywność w stosunku do wymaganego kodu, potrzebnego do uruchomienia wizualizacji. Biblioteka ta, udostępnia najbardziej zaawansowane menu wizualizacji w porównaniu do pozostałych bibliotek, dzięki czemu domyślnie są dostępne takie narzędzia jak np. *Mayavi Pipeline*, które umożliwia tworzenie i manipulacje danych 3D z poziomu menu.

Bibliografia

[1] *Dokumentacja biblioteki Matplotlib*. URL: `https://matplotlib.org/stable/index.html` (visited on 06/03/2024).

[2] *Dokumentacja biblioteki Plotly*. URL: `https://plotly.com/python/` (visited on 06/03/2024).

[3] *Dokumentacja biblioteki Open3D*. URL: `https://www.open3d.org/docs/release/` (visited on 06/03/2024).

[4] *Dokumentacja biblioteki VTK*. URL: `https://vtk.org/documentation/` (visited on 06/03/2024).

[5] *Dokumentacja biblioteki Mayavi*. URL: `https://docs.enthought.com/mayavi/mayavi/` (visited on 06/03/2024).

[6] *Mesh Visualizer - implementacja projektu*. URL: `https://github.com/dkwapisz/PiADM-Projekt` (visited on 06/03/2024).