МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу

ЛАБОРАТОРНА РОБОТА №1
з дисципліни «Просторове моделювання та візуалізація»
на тему: «Моделювання криволінійних обводів»
Варіант №5

Виконала:
Аспірантка гр. КН-в31ф
Старовойт Тетяна Василівна
Перевірила:
д.т.н., доц. Аушева Н.М.

Київ – 2025

Моделювання криволінійних обводів кривими Безьє 5-го порядку

Мета: *оволодіти навичками побудови та керування криволінійними контурами*

Завдання: створити систему моделювання криволінійного контура з гладкістю першого порядку кривими Безьє 5-го порядку

Крива Безьє є параметричною кривою, яка базується на базисних поліномах Бернштейна (Bernstein basis polynomials). Вона широко використовується в комп'ютерній графіці, CAD/CAM системах, моделюванні контурів, а також у геометричному моделюванні.

$$P(t) = \sum_{i=0}^{n} J_{n,i}(t) b_i$$

деb_i є точками Безьє або контрольними точками та$J_{n,i}(t)$єith $n$thбазисна функція Бернштейна порядку, щоn– степінь базисної функції Бернштейна [27] .$J_{n,i}(t)$в інтервалі$[t_j, t_{j+1}]$представлений

$$J_{n,i}(t) = \binom{n}{i} (t_{j+1} - t_j)^{-n} (t_{j+1} - t)^{n-i} (t - t_j)^i, \quad t \in [t_j, t_{j+1}]$$

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}, \quad 0! \equiv 1, \quad (0)^0 \equiv 1, \quad i = 0, \ldots, n$$

Набір базисних функцій 5-го степеня, а також їх похідні першого та другого порядку, що використовуються для інтерполяції, показано на рис 1, 2, 3.

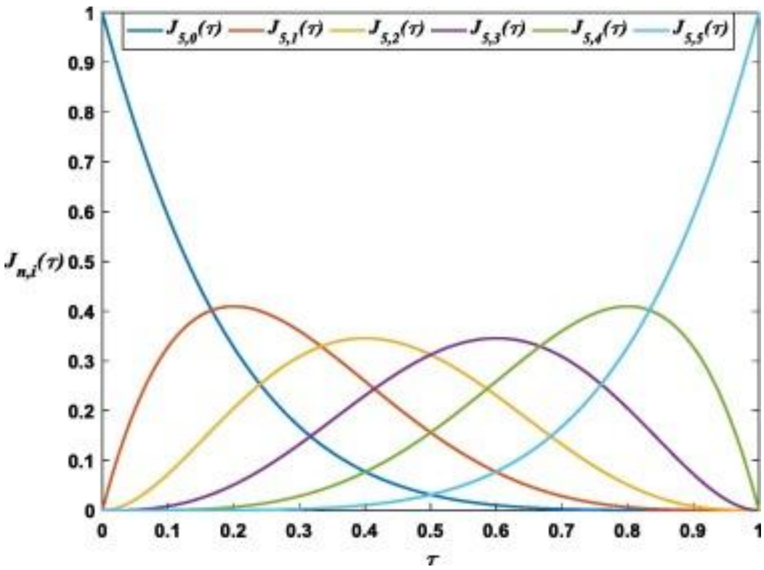

Рис. 1. Підготовлені базисні функції Бернштейна.

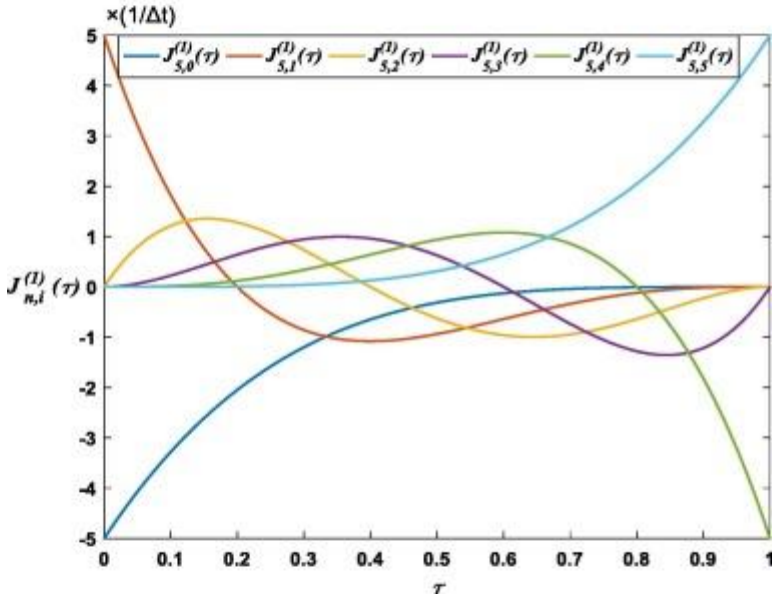

Рис. 2. Підготовлені базисні функції Бернштейна з похідними першого порядку.

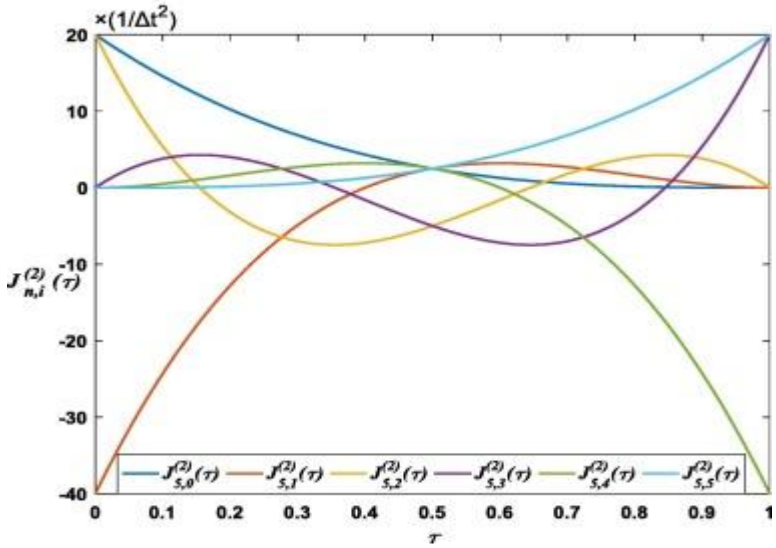

Рис. 3. Підготовлені базисні функції Бернштейна другого порядку за похідними.

Крива Без'є розглядається як інтерполяційна функція

$$P(t) = \sum_{i=0}^{5} J_{5,i}(t) b_i$$

Це можна розвинути до матричної форми як

$$P^{(d)}(t) = \mathbf{J}^{(d)}(\boldsymbol{\tau})\mathbf{b}, \quad t \in [t_j, t_{j+1}], \quad \boldsymbol{\tau} \in [0,1]$$

$$\mathbf{J}^{(d)}(\boldsymbol{\tau}) = \begin{bmatrix} J^{(d)}_{5,0}(\boldsymbol{\tau}) & J^{(d)}_{5,1}(\boldsymbol{\tau}) & J^{(d)}_{5,2}(\boldsymbol{\tau}) & J^{(d)}_{5,3}(\boldsymbol{\tau}) & J^{(d)}_{5,4}(\boldsymbol{\tau}) & J^{(d)}_{5,5}(\boldsymbol{\tau}) \end{bmatrix}$$

$$\mathbf{b} = [\, b_0 \quad b_1 \quad b_2 \quad b_3 \quad b_4 \quad b_5 \,]^T$$

Підготовка даних

Крок 1. Для побудови точного контуру акули була використана схема автоматичного розпізнавання та векторизації з растрового зображення:

Інструмент в ArcGIS Pro: Raster to Polygon

Вхід: растрове зображення акули (у форматі PNG або TIFF) зі збереженим контуром на прозорому або однотонному фоні.

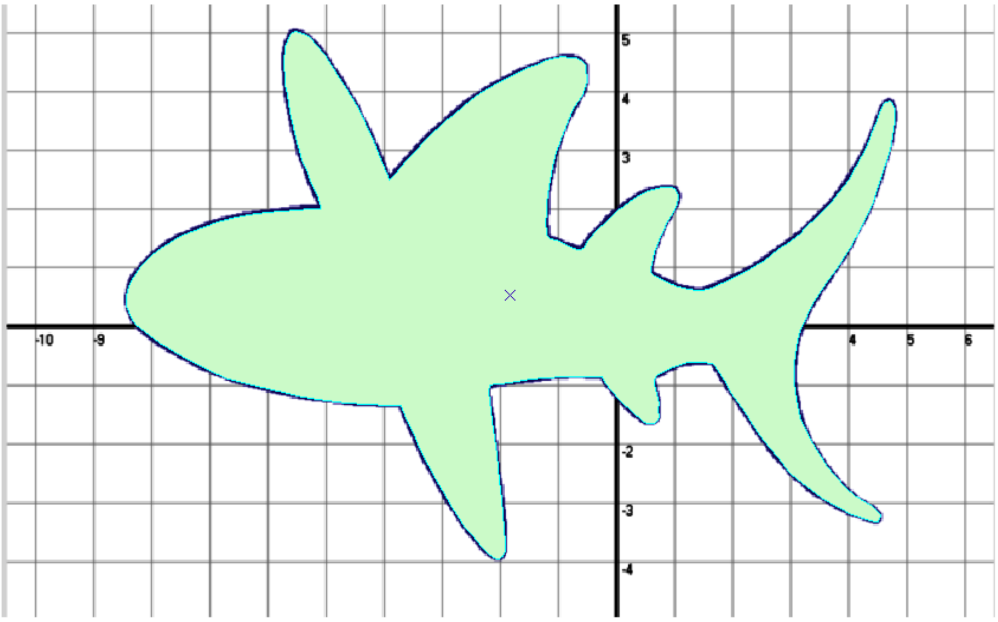

Рисунок 1. – Векторизовані полігональні контури силуету акули в ArcGIS Pro

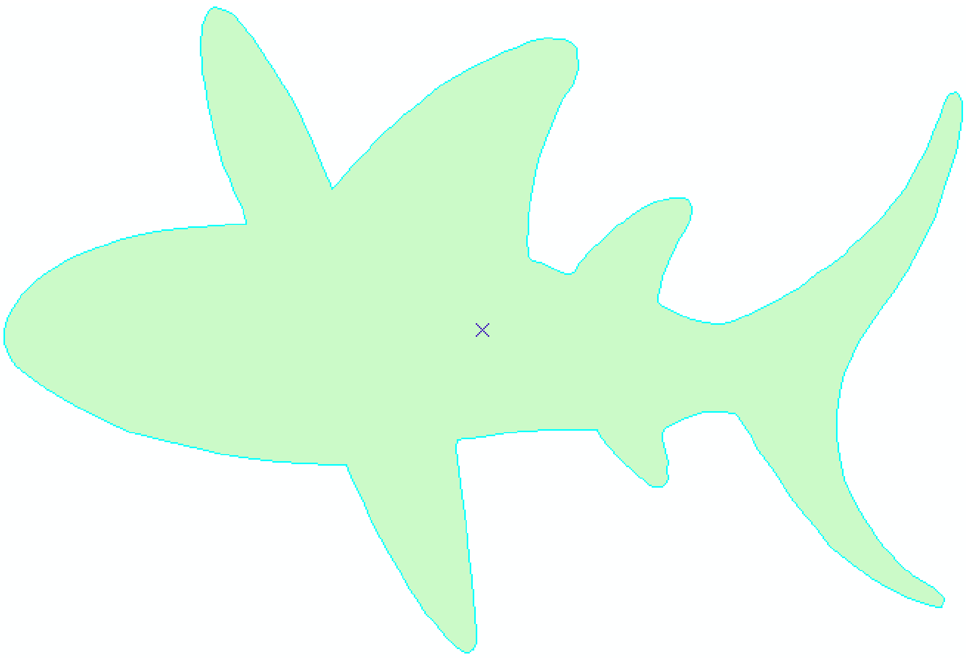

Рисунок 2. – Векторизовані полігональні контури силуету акули в ArcGIS Pro — результат векторизації, що включає точні контури об’єкта

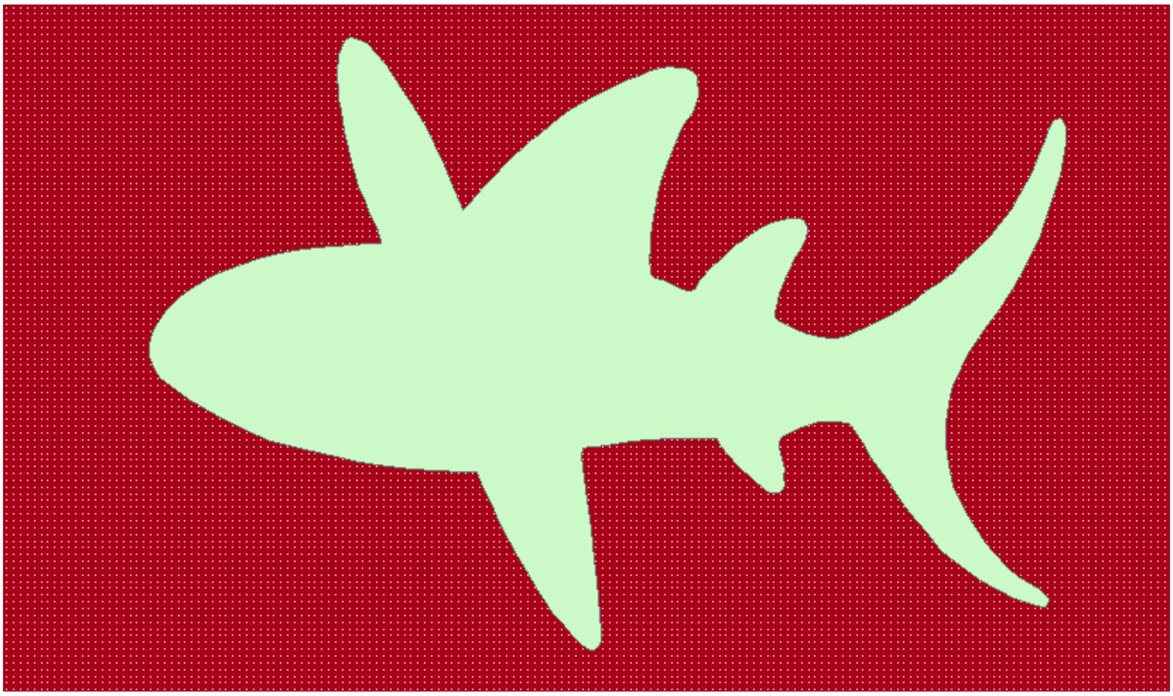

Рисунок 3. – Растрова класифікація фону та фігури для подальшої векторизації

Отриманий полігон експортується як Shapefile або GeoPackage для подальшої обробки в Python. До цього зображення застосовується інструмент Raster to Polygon з такими параметрами: Simplify Polygons: True; Field: Value (1 — фігура, 0 — фон).

Покроковий процес реалізації:

1) Зчитуємо контур фігури, як масив координат.

```
[4]  shp_path = "shark_contour.shp"
     with fiona.open(shp_path, 'r') as shapefile:
         shape = shapely.geometry.shape(next(iter(shapefile))["geometry"])
         coords = np.array(shape.exterior.coords)
```

2) Контур переміщуємо в центр координат та масштабуємо для зручності подальшої обробки.

```
0 сек.  [5]  centered_coords = coords - coords.mean(axis=0)
             scale = max(np.ptp(centered_coords[:, 0]), np.ptp(centered_coords[:, 1]))
             normalized_coords = centered_coords / scale * 20
```

3) Відображаємо початковий контур фігури для візуальної перевірки точності векторизації.

```
[7] plt.figure(figsize=(7, 7))
    plt.plot(normalized_coords[:, 0], -normalized_coords[:, 1], color='blue')
    plt.title("Shark outline from shapefile")
    plt.axis('equal')
    plt.grid(True)
    plt.show()
```

Отриманий результат вказаний на рисунку 4.

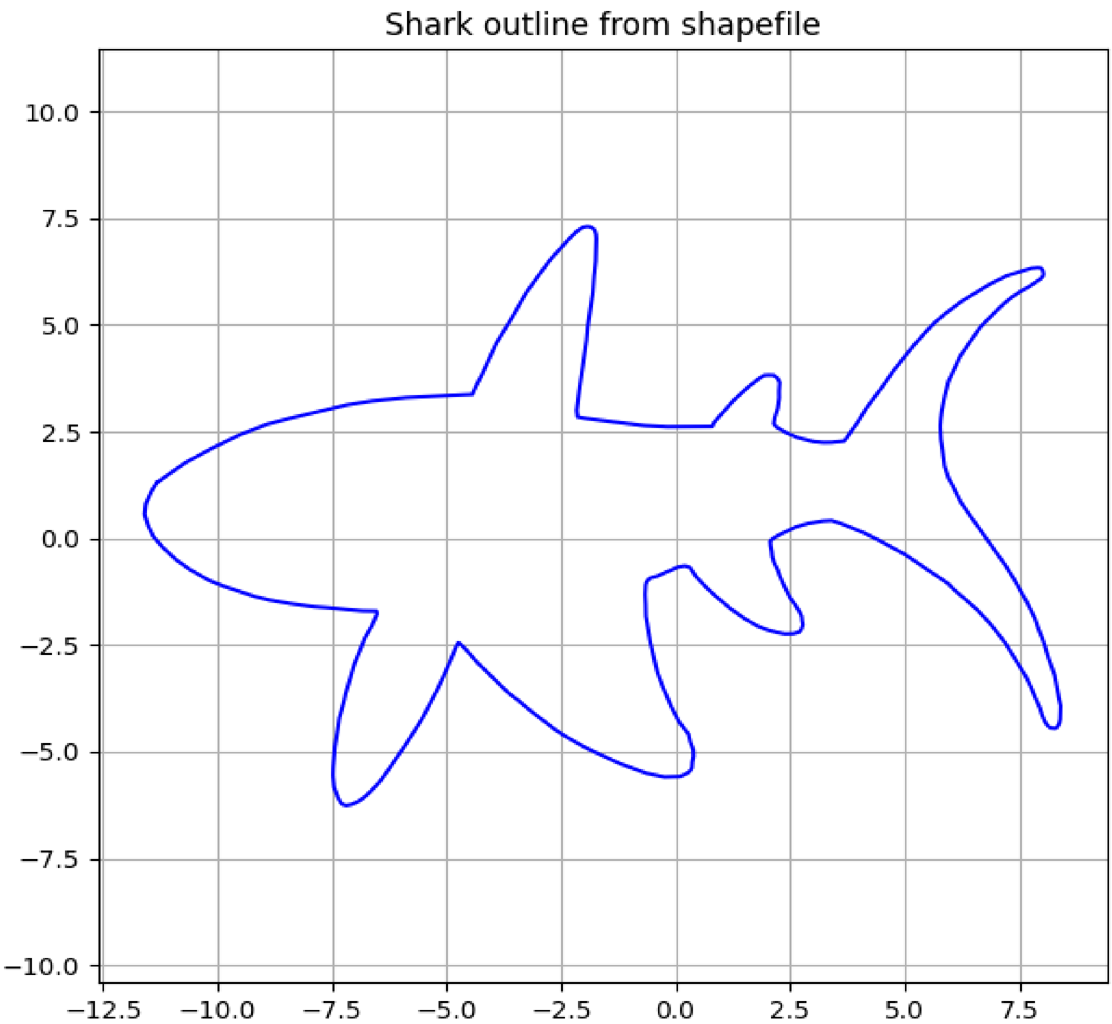

Рисунок 4. – Контур акули отриманий за допомогою просторової конвертації з растру у вектор

4) Побудова кривих Безьє 5-го порядку

Контур розділено на сегменти по 6 точок із замиканням у циклі:

```
coords = coords - coords.mean(axis=0)

def chunk_closed(points, chunk_size=6):
    total = len(points)
    segments = []
    for i in range(0, total, chunk_size):
        if i + chunk_size <= total:
            segments.append(points[i:i+chunk_size])
        else:
            remaining = points[i:]
            needed = chunk_size - len(remaining)
            segment = np.vstack([remaining, points[:needed]])
            segments.append(segment)
    return segments

segments = chunk_closed(coords, 6)
t_vals = np.linspace(0, 1, 300)
```

Побудова гладкого обводу

- Побудовано кожен сегмент окремо з нумерацією:

```
fig, ax = plt.subplots(figsize=(8, 8))
ax.set_aspect('equal')
ax.grid(True, linestyle='--', alpha=0.3)
ax.set_title("Silhouette of a shark, reproduced by coordinates from a shapefile")

for idx, seg in enumerate(segments):
    curve = Bezier.Curve(t_vals, seg)
    ax.plot(curve[:, 0], -curve[:, 1], color='blue')
    midpoint = Bezier.Point(0.5, seg)
    ax.text(midpoint[0], -midpoint[1], f"S{idx+1}", fontsize=8, color='darkred')

plt.show()
```

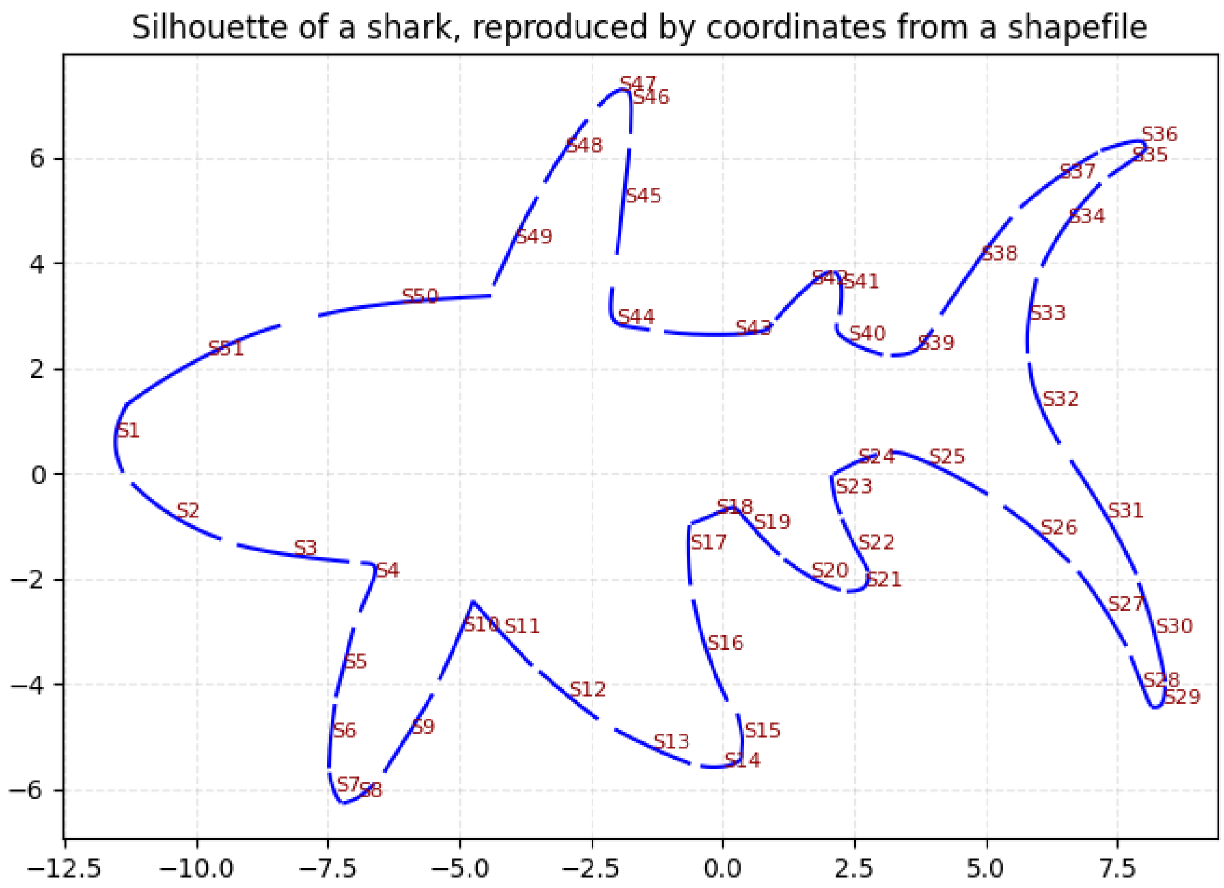

Рисунок 5. – Гладкий силует акули з кривих Безьє 5-го порядку з підписами сегментів (S1–Sn).

В результаті отримано повний гладкий контур силуету акули;

- Система моделювання дозволяє:
  - Автоматично зчитувати координати з shapefile;
  - Центрувати та масштабувати контур;
  - Формувати безперервну лінію з кривих Безьє 5-го порядку;
  - Виводити графічне представлення з підписаними сегментами.

Висновки:

1. **Метод кривих Безьє п'ятого порядку** виявився високоефективним для побудови складних замкнутих контурів. Він забезпечує достатній рівень гладкості та керованості при моделюванні.
2. **Попередня обробка в ArcGIS Pro** дозволила точно виділити силует фігури з растрового зображення, перевести його у векторний формат (shapefile) та забезпечити подальше точне використання координат у Python.

3. **Зчитування shapefile та нормалізація координат** забезпечили масштабованість і центрованість побудованої фігури, що дозволило зберегти правильні пропорції акули при виведенні.

4. **Функція розбиття на сегменти з 6 точок** дозволила автоматизувати формування контрольних точок для кривих Безьє. Завдяки цьому вдалося відтворити обвід без видимих розривів.

5. **Візуалізація кривих із підписами** допомогла проаналізувати відповідність окремих сегментів загальному контуру і спростила налагодження системи.

6. **Отримана система легко масштабована на інші контури та методи побудови** — її можна використовувати для моделювання будь-яких фігур з достатньою кількістю контрольних точок.