

# Algorytmy geometryczne

Przetwarzanie i przechowywanie opisu siatki  
trójkątnej na płaszczyźnie

Dominik Jastrząb

# Przechowywanie i przetwarzanie opisu siatki

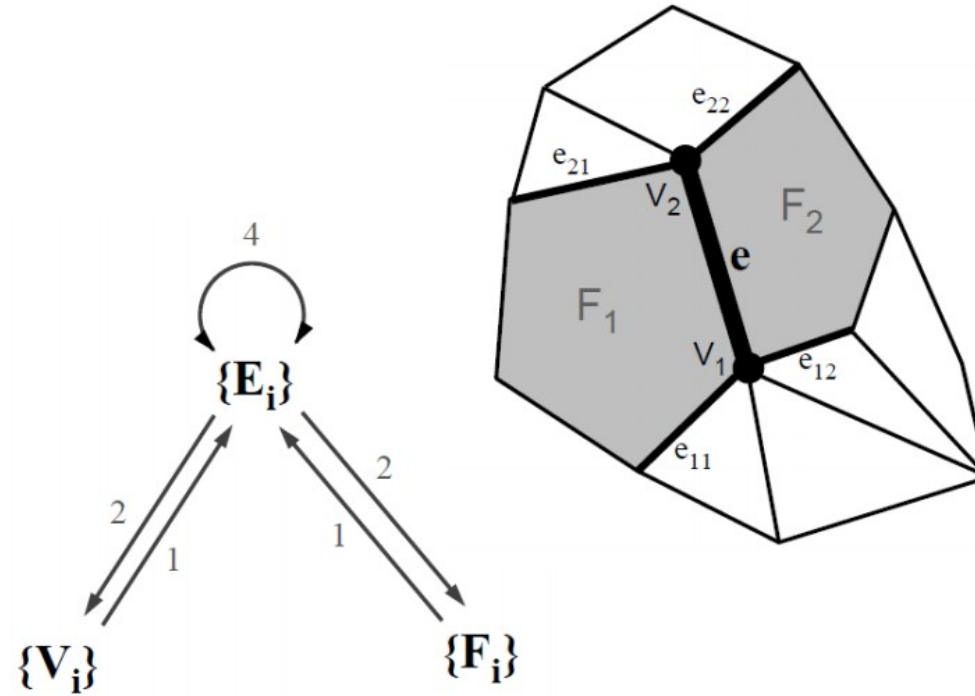
- Główną trudnością w przechowywaniu opisu siatki jest problematyczne, ze względu na konieczność przechowywania danych topologicznych i geometrycznych.
- Rozdzielenie danych geometrycznych od topologicznych pozwala osiągnąć mniejszą złożoność pamięciową
- Przechowywanie wszystkich powiązań topologicznych pozwala na wykonanie większości operacji na siatce w czasie liniowym. Jest niestety znacznie obciążające pamięciowo

# Przechowywanie i przetwarzanie opisu siatki

- Aby doprowadzić do kompromisu między złożonością pamięciową, a obliczeniową stosuje się określone struktury danych. Pozwalają one zoptymalizować oba aspekty, unikając jednocześnie redundancyjnych informacji

# Winged Edge Data Structure

## Winged Edge Data Structure



# Winged Edge Data Structure

- Struktura krawędzi jest głównym nośnikiem danych
- Tylko wierzchołki zawierają dane geometryczne
- Wierzchołki i ściany zawierają tylko należność do danej krawędzi (z danych topologicznych)
- Pozwala na realizację zapytań o sąsiedztwo (krawędzi, wierzchołków i ścian) w czasie liniowym

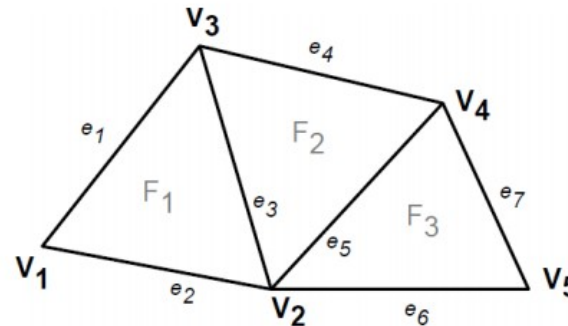
# Winged Edge Data Structure

## Winged Edge Data Structure

| EDGE TABLE     |                               |                               |                |                |                |                |                |
|----------------|-------------------------------|-------------------------------|----------------|----------------|----------------|----------------|----------------|
|                |                               |                               |                | 11             | 12             | 21             | 22             |
| e <sub>1</sub> | V <sub>1</sub> V <sub>3</sub> |                               | F <sub>1</sub> | e <sub>2</sub> | e <sub>2</sub> | e <sub>4</sub> | e <sub>3</sub> |
| e <sub>2</sub> | V <sub>1</sub> V <sub>2</sub> | F <sub>1</sub>                |                | e <sub>1</sub> | e <sub>1</sub> | e <sub>3</sub> | e <sub>6</sub> |
| e <sub>3</sub> | V <sub>2</sub> V <sub>3</sub> | F <sub>1</sub> F <sub>2</sub> |                | e <sub>2</sub> | e <sub>5</sub> | e <sub>1</sub> | e <sub>4</sub> |
| e <sub>4</sub> | V <sub>3</sub> V <sub>4</sub> |                               | F <sub>2</sub> | e <sub>1</sub> | e <sub>3</sub> | e <sub>7</sub> | e <sub>5</sub> |
| e <sub>5</sub> | V <sub>2</sub> V <sub>4</sub> | F <sub>2</sub> F <sub>3</sub> |                | e <sub>3</sub> | e <sub>6</sub> | e <sub>4</sub> | e <sub>7</sub> |
| e <sub>6</sub> | V <sub>2</sub> V <sub>5</sub> | F <sub>3</sub>                |                | e <sub>5</sub> | e <sub>2</sub> | e <sub>7</sub> | e <sub>7</sub> |
| e <sub>7</sub> | V <sub>4</sub> V <sub>5</sub> |                               | F <sub>3</sub> | e <sub>4</sub> | e <sub>5</sub> | e <sub>6</sub> | e <sub>6</sub> |

| FACE TABLE     |                |
|----------------|----------------|
| F <sub>1</sub> | e <sub>1</sub> |
| F <sub>2</sub> | e <sub>3</sub> |
| F <sub>3</sub> | e <sub>5</sub> |

| VERTEX TABLE   |                |                |                |                |
|----------------|----------------|----------------|----------------|----------------|
| V <sub>1</sub> | X <sub>1</sub> | Y <sub>1</sub> | Z <sub>1</sub> | e <sub>1</sub> |
| V <sub>2</sub> | X <sub>2</sub> | Y <sub>2</sub> | Z <sub>2</sub> | e <sub>6</sub> |
| V <sub>3</sub> | X <sub>3</sub> | Y <sub>3</sub> | Z <sub>3</sub> | e <sub>3</sub> |
| V <sub>4</sub> | X <sub>4</sub> | Y <sub>4</sub> | Z <sub>4</sub> | e <sub>5</sub> |
| V <sub>5</sub> | X <sub>5</sub> | Y <sub>5</sub> | Z <sub>5</sub> | e <sub>6</sub> |



# Winged Edge Data Structure

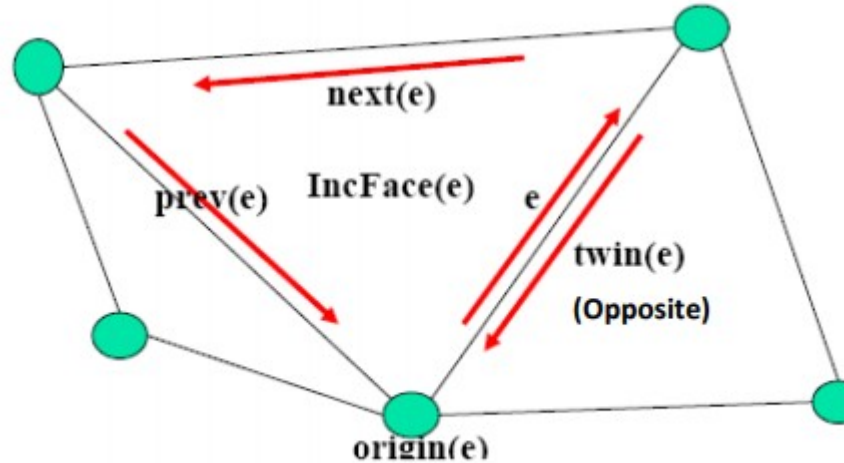
```
class Edgew:
    def __init__(self, vOrg, vDest, fl=None, fr=None, elcw=None, elcww=None, ercw=None, ercww=None):
        self.vertexOrigin=vOrg
        self.vertexDestination=vDest
        self.faceLeft=fl
        self.faceRight=fr
        self.edgeLeftCw=elcw
        self.edgeLeftCww=elcww
        self.edgeRightCw=ercw
        self.edgeRightCww=ercww

class VertexW:
    def __init__(self, x, y, edge):
        self.x=x
        self.y=y
        self.edge=edge

class FaceW:
    def __init__(self, edge):
        self.edge=edge
```

# Half Edge Data Structure

## Half Edge Data Structure



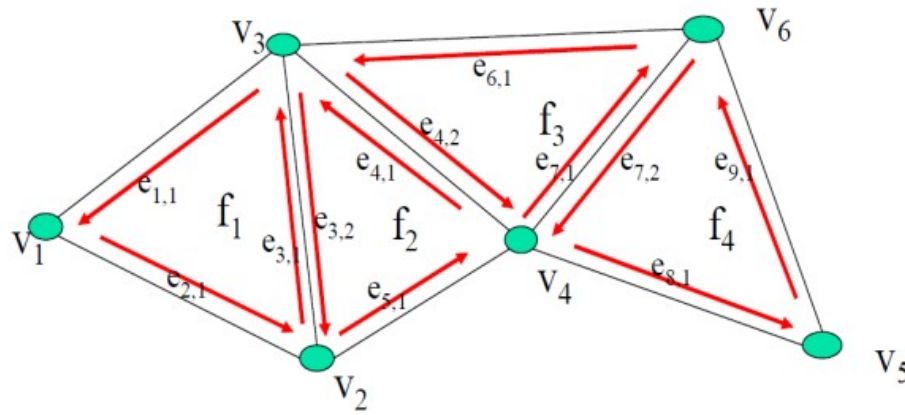


# Half Edge Data Structure

- Tylko wierzchołki zawierają dane geometryczne
- Wierzchołki zawierają również jedną, wychodzącą z nich krawędź
- Krawędzie (półkrawędzie) zawierają wierzchołek z którego wychodzą, ścianę która do nich przylega, oraz następną, poprzednią i bliźniaczą krawędź
- Optymalna dla algorytmów

# Half Edge Data Structure

## Half Edge Data Structure



| Half-edge | origin | Opposite  | IncidentFace | next      | prev      |
|-----------|--------|-----------|--------------|-----------|-----------|
| $e_{3,1}$ | $v_2$  | $e_{3,2}$ | $f_1$        | $e_{1,1}$ | $e_{2,1}$ |
| $e_{3,2}$ | $v_3$  | $e_{3,1}$ | $f_2$        | $e_{5,1}$ | $e_{4,1}$ |
| $e_{4,1}$ | $v_4$  | $e_{4,2}$ | $f_2$        | $e_{3,2}$ | $e_{5,1}$ |
| $e_{4,2}$ | $v_3$  | $e_{4,1}$ | $f_3$        | $e_{7,1}$ | $e_{6,1}$ |



# Half Edge Data Structure

# Half Edge Data Structure