

# 4D-GML

Marek Krótkiewicz, Marcin Jodłowiec

5 maja 2020

## 1 Cel

Opracowanie składni abstrakcyjnej i semantyki dla języka operowania w przestrzeni trójwymiarowej z uwzględnieniem osi czasu na obiektach reprezentowanych przez grafy.

## 2 Przestrzeń

Przestrzeń jest trójwymiarowa  $(X, Y, Z)$ , w której określono kartezjański układ współrzędnych. Dodatkowo zakłada się dyskretną oś czasu.

## 3 Obiekt

$$G = (N, E) \quad (1)$$

gdzie:

$N$  – zbiór wierzchołków,

$E$  – zbiór krawędzi.

Wierzchołek:

$$n \in N \quad (2)$$

$$n = (point, vpoint, apoint) \quad (3)$$

$$point = (x, y, z, \theta, \phi, \psi, t) \quad (4)$$

$$vpoint = (v_x, v_y, v_z, \omega_\theta, \omega_\phi, \omega_\psi, dt) \quad (5)$$

$$apoint = (a_x, a_y, a_z, \alpha_\theta, \alpha_\phi, \alpha_\psi, ddt) \quad (6)$$

$$v_x = \frac{dx}{dt}; v_y = \frac{dy}{dt}; v_z = \frac{dz}{dt} \quad (7)$$

$$\omega_\theta = \frac{d\theta}{dt}; \omega_\phi = \frac{d\phi}{dt}; \omega_\psi = \frac{d\psi}{dt} \quad (8)$$

$$a_x = \frac{dv_x}{dt} = \frac{d^2x}{dt^2}; a_y = \frac{dv_y}{dt} = \frac{d^2y}{dt^2}; a_z = \frac{dv_z}{dt} = \frac{d^2z}{dt^2} \quad (9)$$

$$\alpha_\theta = \frac{d\omega_\theta}{dt} = \frac{d^2\alpha_\theta}{dt^2}; \alpha_\phi = \frac{d\omega_\phi}{dt} = \frac{d^2\alpha_\phi}{dt^2}; \alpha_\psi = \frac{d\omega_\psi}{dt} = \frac{d^2\alpha_\psi}{dt^2} \quad (10)$$

Przykład: przyspieszenie kątowe w osi  $x$  węzła  $n_1$  określone dla punktu czasowego 2.56 sek. wynosi  $2.3 \text{ m/s}^2$ .

$$n_1.apoint.a_x(2.56) = 2.3 \text{ m/s}^2 \quad (11)$$

Krawędź:

$$e \in E \quad (12)$$

$$e = (startnode, endnode) \quad (13)$$

$$startnode, endnode \in N \quad (14)$$

## 4 Przykładowe wyrażenia

### 4.1 CREATE

```
graph = GRAPH()  
node = NODE(x, y, z, theta, phi, psi)  
edge = EDGE(node1, node2)
```

### 4.2 READ

```
valx = graph.nodes["n1"].point.x  
valtheta = graph.nodes["n1"].apoint.alpha_theta
```

### 4.3 UPDATE

```
graph.ADDNODE(node)  
graph.ADDEDGE(edge)  
graph.UPDATENODE(node1, node2)  
graph.UPDATEEDGE(edge1, edge2)  
graph.MERGEGRAPH(graph1)  
  
//-----//  
graph.nodes["n1"].point.x = valx  
graph.nodes["n1"].vpoint.vy = valy
```

### 4.4 DELETE

```
graph.DELETE()  
graph.DELETENODE(node)  
graph.DELETEEDGE(edge)
```

### 4.5 EXECUTEMOVEMENT

```
graph.EXECUTEMOVEMENT()
```

## 5 Struktury danych

(pseudo-python)

### 5.1 Graf

```
class GRAPH:

    def __init__(self):
        self.nodes = {}
        self.edges = {}
```

### 5.2 Węzeł

```
class NODE:
    class POINT:
        def __init__(self, x, y, z, alpha, beta, gamma):
            self.x = x
            self.y = y
            self.z = z
            self.alpha = alpha
            self.beta = beta
            self.gamma = gamma

    class VPOINT:
        def __init__(self, vx,vy,vz,om_theta,om_phi,om_psi):
            self.vx = vx
            self.vy = vy
            self.vz = vz
            self.om_theta = om_theta
            self.om_phi = om_phi
            self.om_psi = om_psi

    class APOINT:
        def __init__(self, ax,ay,az,alpha_theta,alpha_phi,alpha_psi):
            self.ax = ax
            self.ay = ay
            self.az = az
            self.alpha_theta = alpha_theta
            self.alpha_phi = alpha_phi
            self.alpha_psi = alpha_psi

    def __init__(self, x, y, z, alpha, beta, gamma):
        self.point = POINT(x, y, z, alpha, beta, gamma)
        self.vpoint = VPOINT(0,0,0,0,0,0)
        self.apoint = APOINT(0,0,0,0,0,0)
```

### 5.3 Krawędź

```
class EDGE:

    def __init__(self, node1, node2):
```

```
self.node1 = node1  
self.node2 = node2
```