Descrizione delle funzioni

Questo codice MATLAB definisce una funzione chiamata `dentro\_list\_vect\_002` che determina quali punti di un insieme `P` si trovano all'interno di uno o più triangoli definiti da tre punti `A`, `B` e `C`. Ecco una spiegazione passo passo:

### Passo 1: Calcolo dei Vettori dei Triangoli

```matlab

v0 = C - A; % Vettore dal punto A al punto C

v1 = B - A; % Vettore dal punto A al punto B

```

Qui vengono calcolati due vettori che rappresentano i lati del triangolo formato dai punti `A`, `B` e `C`.

### Passo 2: Preparazione dei Punti P

```matlab

m = size(P, 1);

n = size(A, 1);

```

`m` è il numero di punti in `P` e `n` è il numero di triangoli (il numero di righe in `A`, `B`, e `C`).

### Passo 3: Espansione dei Punti

```matlab

A\_expanded = reshape(A, [n, 1, 3]);

P\_expanded = reshape(P, [1, m, 3]);

```

I punti `A` vengono espansi per consentire operazioni di broadcasting con i punti `P`. Così facendo, `A` diventa un array 3D che facilita le operazioni di sottrazione.

### Passo 4: Calcolo dei Prodotti Scalari

```matlab

dot00 = sum(v0\_expanded .\* v0\_expanded, 3);

dot01 = sum(v0\_expanded .\* v1\_expanded, 3);

dot11 = sum(v1\_expanded .\* v1\_expanded, 3);

dot02 = sum(v0\_expanded .\* v2, 3);

dot12 = sum(v1\_expanded .\* v2, 3);

```

Vengono calcolati i prodotti scalari necessari per determinare le coordinate baricentriche. `dot00`, `dot01`, e `dot11` sono prodotti scalari tra i vettori del triangolo, mentre `dot02` e `dot12` sono prodotti scalari tra i vettori del triangolo e i vettori che collegano i vertici del triangolo ai punti in `P`.

### Passo 5: Coordinate Baricentriche

```matlab

u = (dot11 .\* dot02 - dot01 .\* dot12) ./ denom;

v = (dot00 .\* dot12 - dot01 .\* dot02) ./ denom;

```

Qui vengono calcolate le coordinate baricentriche `u` e `v`. Queste coordinate indicano come un punto `P` si trova relativamente all'interno di un triangolo.

### Passo 6: Verifica della Posizione dei Punti

```matlab

inside = (u > 0) & (v > 0) & (u + v < 1);

```

Questa condizione verifica se i punti sono all'interno dei triangoli. `inside` sarà una matrice booleana che indica per ogni punto se è all'interno di un triangolo.

### Passo 7: Identificazione dei Punti all'Interno dei Triangoli

```matlab

points\_inside\_any\_triangle = any(inside, 1);

indices\_of\_points\_inside = find(points\_inside\_any\_triangle);

```

Qui si verifica quali punti sono dentro almeno un triangolo. `points\_inside\_any\_triangle` identifica i punti, mentre `indices\_of\_points\_inside` fornisce gli indici di questi punti.

### Associazione tra Triangoli e Punti

```matlab

[triangle\_indices, point\_indices] = find(inside);

```

Questa parte trova le associazioni tra triangoli e punti.

### Verifica della Visibilità

```matlab

idx\_pt = triangle\_indices < point\_indices;

```

Qui viene effettuata una verifica per assicurarsi che il triangolo esaminato sia davanti alla faccia oscurata.

### Output Finale

```matlab

IDX\_Facce\_ill\_vect = Tri\_ok;

```

La funzione restituisce un vettore che contiene gli indici dei triangoli in cui si trovano i punti.

In sintesi, questa funzione è utile per determinare rapidamente quali punti si trovano all'interno di triangoli definiti nello spazio tridimensionale, considerando anche la visibilità dei triangoli rispetto ai punti.

function IDX\_Facce\_ill\_vect = dentro\_list\_vect\_002(A, B, C, P,Sat)

%% Passo 1: Calcola i vettori dei triangoli

v0 = C - A; % n x 3

v1 = B - A; % n x 3

% Passo 2: Prepara i punti P

m = size(P, 1);

n = size(A, 1);

% Passo 3: Calcola v2 usando broadcasting esplicito

% A\_expanded avrà dimensione n x 1 x 3, mentre P avrà dimensione 1 x m x 3

A\_expanded = reshape(A, [n, 1, 3]);

P\_expanded = reshape(P, [1, m, 3]);

% Sottrazione vettoriale con broadcasting per ottenere v2 (n x m x 3)

v2 = P\_expanded - A\_expanded;

% Passo 4: Calcola i prodotti scalari

% Espandi v0 e v1 per farli combaciare con v2

v0\_expanded = reshape(v0, [n, 1, 3]); % v0 diventa n x 1 x 3

v1\_expanded = reshape(v1, [n, 1, 3]); % v1 diventa n x 1 x 3

% Prodotti scalari tra v0 e v2, v1 e v2 (dot02, dot12 avranno dimensione n x m)

dot00 = sum(v0\_expanded .\* v0\_expanded, 3); % n x 1

dot01 = sum(v0\_expanded .\* v1\_expanded, 3);

dot11 = sum(v1\_expanded .\* v1\_expanded, 3);

dot02 = sum(v0\_expanded .\* v2, 3); % n x m

dot12 = sum(v1\_expanded .\* v2, 3);

% Passo 5: Calcola le coordinate baricentriche

denom = dot00 .\* dot11 - dot01 .\* dot01;

u = (dot11 .\* dot02 - dot01 .\* dot12) ./ denom;

v = (dot00 .\* dot12 - dot01 .\* dot02) ./ denom;

% Leva le sovrapposizioni

u(1:n+1:end) = -1;

v(1:n+1:end) = -1;

% Passo 6: Determina se i punti sono all'interno dei triangoli

inside = (u > 0) & (v > 0) & (u + v < 1);

% Passo 7: Identifica i punti all'interno di almeno un triangolo

points\_inside\_any\_triangle = any(inside, 1);

indices\_of\_points\_inside = find(points\_inside\_any\_triangle);

% Associazioni tra triangoli e punti

[triangle\_indices, point\_indices] = find(inside);

% Verifica che il triangolo in esame sia davanti alla faccia oscurata

idx\_pt = triangle\_indices < point\_indices;

Tri\_ok = triangle\_indices;

Pt\_ok = point\_indices;

Tri\_ok(idx\_pt) = Pt\_ok(idx\_pt);

IDX\_Facce\_ill\_vect = Tri\_ok;

end