Studio preliminare

L'obiettivo del codice contenuto nel file `mapper.jl` è quello di implementare diverse primitive parametriche, incluse curve, superfici e solidi incorporati in 2D o 3D.
Ciascun metodo contenuto in questa classe utilizza un approccio costruttivo, ossia genera una scomposizione semplice o cubica di un dominio geometrico semplice nello spazio parametrico *u,v* o *u,v,w*. Quindi un cambio di coordinate, ad es. da coordinate cartesiane a coordinate polari o cilindriche, si applica ai vertici del complesso cellulare decomponendo il suo dominio.
Il dominio mappato in questa maniera produce una collezione di curve nello spazio 2D o 3D. Per ottenere una superficie curva chiusa, cioè una collezione senza bordi, come nel caso di una `sfera` in 3D, o della superficie `toroidale` in 3D, viene eseguita un'opportuna identificazione di punti mappati coincidenti.

Concetti di base di LAR

L'esempio più semplice di una figura geometrica implementata da LAR è quella di un `quadrato` con un vertice nell'origine:

```
julia> square=([[0.; 0] [0; 1] [1; 0] [1; 1]], [[1,2,3,4]], [[1,2], [1,3], [2,4], [3,4]])
([0.0 0.0 1.0 1.0; 0.0 1.0 0.0 1.0], Array{Int64,1}[[1, 2, 3, 4]], Array{Int64,1}[[1, 2],
[1, 3], [2, 4], [3, 4]])
```

È anche possibile memorizzare gli array contenenti le coordinate dei vertici in delle variabili con nomi convenzionali (`V`, `FV`, `EV`):

```
julia> V,FV,EV = square
([0.0 0.0 1.0 1.0; 0.0 1.0 0.0 1.0], Array{Int64,1}[[1, 2, 3, 4]], Array{Int64,1}[[1, 2],
[1, 3], [2, 4], [3, 4]])
```

Illustrazione delle principali primitive

In questa sezione vengono descritte le principali primitive contenute nel package `mapper.jl` oggetto del lavoro di ottimizzazione previsto dal progetto.

Lar.approxVal

Lo scopo di questa funzione è molto semplice: fornendo in input un valore e un numero di cifre significative approssima il valore al numero di cifre richieste. Qualora il risultato fosse `-0.0` il segno viene cambiato.

Lar.simplifyCells

A causa di errori numerici sulle coordinate mappate potrebbero verificarsi duplicazioni di vertici. Questa funzione ausiliaria trova ed elimina vertici duplicati e celle errate. I paramentri di input sono CV (vettore di vettori) e V (matrice).
Viene creato un dizionario vertDict con chiave una colonne della matrice e valore

un contatore incrementale. Per ogni vettore incell di CV la funzione crea un vettore vuoto outcell e per ogni elemento di v di incell preleva la colonna corrispondente di V. Applica la funzione `approxVal` ad ogni elemento della colonna. Se la colonna non è presente nel dizionario, incrementa l'indice e la inserisce nel dizionario con chiave il nuovo valore dell'indice. Inoltre aggiunge l'indice ad outcell e la colonna alla matrice W di output.
Se invece la colonna appartiene già al dizionario, aggiunge ad outcell l'indice corrispondente presente nel dizionario. Infine l'eliminazione dei duplicati da outcell avviene mediante la creazione di un `Set` che per definizione non può contenere valori uguali.

Lar.circle

Calcola l'approssimazione di una circonferenza centrata nell'origine prendendo come input raggio, angolo e numero di segmenti utilizzati per l'approssimazione. Per la creazione di vertici e spigoli sfrutta la funzione `cuboidGrid` descritta nel modulo `largrid.jl`.
La funzione moltiplica ogni vertice per il rapporto fra l'angolo e i segmenti inseriti in input. Mappa ogni vertice come segue:

$$\begin{cases} x = a \cdot \cos(t); \\ y = a \cdot sen(t); \end{cases}$$

e inserisce le coppie così generate in un vettore. Dopodiché affianca ogni vettore ottenendo una matrice con un numero di elementi doppi rispetto alla precedente. Infine applica `simplifyCells` al risultato.

Lar.toroidal

Calcola una approssimazione del toro, ovvero una superficie di rotazione ottenuta dalla rivoluzione di una circonferenza in uno spazio tridimensionale intorno a un asse ad essa complanare. Gli input richiesti sono raggio e angolo della circonferenza e della superficie di rotazione, oltre al numero di segmenti richiesti per l'approssimazione.
La descrizione della funzione è simile alla precedente. Per la creazione di vertici e spigoli sfrutta la funzione `simplexGrid` descritta nel modulo `simplexn.jl`.
La matrice V ottenuta possiede due righe ciascuna delle quali rappresenta i vertici di ciascuna delle due figure. Viene effettuato un prodotto fra V e una matrice 2 x 2 con il rapporto fra l'angolo e il numero segmenti inseriti in input rispettivamente per la prima e la seconda figura sulla diagonale principale e degli 0 sulla diagonale secondaria.
Vengono estratte una per volta tutte le colonne di V mappandole nel seguente modo:

$$\begin{cases} x(u,v) = (R + r\cos u)\cos v \\ y(u,v) = (R + r\cos u)\sin v \\ z(u,v) = r\sin u, \end{cases}$$

Infine applica `simplifyCells` al risultato.

Lar.cuboid

Restituisce un cubo d-dimensionale dove d è la dimensione comune degli array di input `minpoint` e `maxpoint`.
Per la creazione di vertici e spigoli sfrutta la funzione `cuboidGrid` descritta nel modulo `largrid.jl`. Dopodiché tramite le funzioni esterne `s` e `t` crea delle matrici, le moltiplica fra loro e applica la funzione esterna `apply` al risultato.