Übungen zur Vorlesung

Simulationswerkzeuge

Dr. S. Lang, D. Popović

Besprechung am 17. Mai 2009 in der Übung

Übung 8 OpenCascade – Konstruktion von Solids

In der letzten Übung haben wir ein Programm auf Basis des CAD-Kernels von *OpenCASCADE* entwickelt, mit dem CAD-Geometrien eingelesen und die topologischen Basisstrukutren in eindeutigen Maps gespeichert werden können.

Die Testgeometrien `bearing.iges` und `hammer.iges` sind reine Randbeschreibungen, d.h. die CAD-Modelle enthalten keine `Solids`. Dies wollen wir nun ändern, indem wir mit *OpenCASCADE* ein `Solid` konstruieren. Wir erweitern daher die Klasse `OCC_GEOM` um eine Methode `int ConstructSolid()`, die aus den `Faces` oder `Shells` einer eingelesenen Shape ein `Solid` erstellt. Dazu verwenden wir die Klassen `BRepOffsetAPI_Sewing` und `BRepBuilderAPI_MakeSolid`. In der Datei `occ_geom.hh` müssen entsprechende Veränderungen vorgenommen werden:

- Definieren Sie in der Klasse `OCC_GEOM` eine Methode `int ConstructSolid()`.
- Inkludieren Sie die Header-Dateien `BRepOffsetAPI_Sewing.hxx` und `BRepBuilderAPI_MakeSolid.hxx`.

Nun können wir mit beiden Klassen arbeiten und die Methode `ConstructSolid` implementieren. Die Methode soll

- testen, ob Solids im CAD-Modell vorhanden sind,
- falls ja, die Anzahl der Solids zurückgeben,
- falls nein, das Modell also eine reine Randbeschreibung ist, ein `Solid` konstruieren.

Im letzten Fall erzeugen wir ein `Solid` in zwei Schritten:

1. Wir bedienen uns eines Objekts der Klasse `BRepOffsetAPI_Sewing` und fügen mittels der `Add`-Methode der Klasse alle `Faces` hinzu. Mit der Methode `Perform` werden diese dann topologisch verbunden.
2. Ein Objekt der Klasse `BRepBuilderAPI_MakeSolid` generiert uns dann ein `Solid`, indem wir diesem Objekt alle `Shells` mit der `Add`-Methode hinzufügen. Nun können wir dieses Objekt der Shape-Datenstruktur des `Occ_Geom`-Objekts (`this->ashape`) zuweisen, um eine `Shape` mit einem `Solid` zu erhalten.

Die Methode `ConstructSolid` implementieren wir in der Datei `occ_geom.cxx`. Wir gehen dabei nach den oben genannten Schritten vor:

- Definition eines Iterators vom Typ `TopExp_Explorer` für `Solids` und `Faces`.
- Testen, ob auch wirklich keine `Solids` in der Shape vorhanden sind.
- Ein Block erledigt das Verbinden der `Faces`:

```
BRepOffsetAPI_Sewing sewedShape;
for (face.Init(this->ashape,TopAbs_FACE); face.More(); \
    face.Next())
  sewedShape.Add (TopoDS::Face(face.Current()));
sewedShape.Perform();
```

- Ein weiterer Block fügt die `Shells` zu einem `Solid` zusammen:

```
BRepBuilderAPI_MakeSolid solidShape;
for (solid.Init(sewedShape.SewedShape(),TopAbs_SHELL); \
    solid.More(); solid.Next())
  solidShape.Add(TopoDS::Shell(solid.Current()));
```

- Falls eine Geometrie beim Import keine `Solids` enthält, soll die `Import_Geometrie`-Methode selbständig ein `Solid` erzeugen!

Nachdem Sie das Programm geschrieben haben, testen Sie es mit der *IGES*-Geometrien. Rufen Sie dazu in der `main`-Funktion die Methode `ConstructSolid` für das `OCC_GEOM`-Objekt auf.

Tipps:

- Laden Sie wie immer die Einstellungen mittels `source occ_debian.sh` und übersetzen Sie mit `make`.
- Dokumentation zu den verwendeten Klassen finden Sie im „Modeling Algorithm User's Guide".
- Orientieren Sie sich zur Definition der Iteratoren an den Iteratoren in der letzten Übung, wie sie z.B. in den Schleifen-Durchläufen zur Erzeugung der `IndexMapOfShapes` benutzt wurden.

Übung 9 OpenCascade – Geometrische Datenstrukturen

Um mit Vertices nicht im topologischen, sondern geometrischen Kontext zu arbeiten, benötigen wir deren Punktkoordinaten. Für einen Vertex können wir die Koordinaten wie folgt bestimmen:

```
gp_Pnt gp = BRep_Tool::Pnt(Vertex)
std::cout << "Point=␣" << gp.X() << "␣" << gp.Y() << "␣" << gp.Z() <<
    std::endl;
```

Hierbei ist `Vertex` ein topologischer Vertex, während die Klasse `gp_Pnt` die Datenstruktur für geometrische Punkte ist. Um die Klasse `BRep_Tool` zu verwenden, muss in `occ_geom.hh` die entsprechende Header-Datei inkludiert werden:

```
#include <BRep_Tools.hxx>
```

Erweitern Sie die Methode `OCC_GEOM::BuildIndexedMaps` zur Erstellung der Maps um die Ausgabe der Koordinaten von Vertices!