Todo list

Wilhelm-Ostwald-Schule, Gymnasium der Stadt Leipzig

Dokumentation zur Besonderen Lernleistung

Im Fachbereich

Informatik

Thema

Erzeugung eines 3D-Modells eines Gebäudes

anhand des Grundrisses

Vorgelegt von

Johann Bartel und Peter Oehme

Schuljahr

2016/2017

Externe Betreuer

Herr Prof. Dr. Gerik Scheuermann, Herr Tom Liebmann

Universität Leipzig Fakultät für Mathematik und Informatik

Interner Betreuer

Herr Rai-Ming Knospe

Leipzig, den 6.06.2017

Bibliographische Beschreibung

Bartel, Johann und Oehme, Peter

„Erzeugung eines 3D-Modells eines Gebäudes anhand des Grundrisses“

21 Seiten, Y Anlagen, 2 Abbildungen

Erzeugung eines druckbaren 3D-Modells eines Gebäudes anhand des Grundrisses

Die Zielstellung dieser BeLL ist es, den Grundriss eines Hauses, der aus einem Konstruktionsprogramm entnommen wurde, in eine druckbare 3D-Datei zu konvertieren. Diese Umwandlung wird mithilfe eines Programmes mit eingebetteten selbst entworfenen mathematischen Operationen realisiert.

Aus dem Grundriss, welcher eine 2D-Datenmenge darstellt, werden die digitalen Anweisungen für die 3D-Strukturen Wände, Grundflächen und Eckpfeiler berechnet. Die Berechnungen laufen so ab, dass an allen Elementen komplementäre Stecker angebracht werden, die zusammen als ein Stecksystem fungieren. Eckpfeiler dienen hierbei als Verbindungsstücke zwischen den Wänden und Bodenplatten die somit für die Stabilität des Objektes sorgen. Mit dem Stecksystem kann der gesamte 3D-Druck aus seinen einzelnen Bestandteilen aufgebaut werden. Dadurch entsteht ein Modell, welches aufgrund der genannten Modifikationen transportabel und geeignet für Präsentationen ist.

Immobilienkäufer und -verkäufer können dadurch die 3D-Darstellung der Immobilie nutzen, um mehr Eindruck über das Objekt erlangen und um eine mögliche Inneneinrichtung zu planen.

Johann Bartel und Peter Oehme

Inhaltsverzeichnis

1. Einleitung

In den letzten Jahren gewannen 3D-Drucker immer mehr Bedeutung, sowohl für wissenschaftliche als auch für wirtschaftliche Zwecke. Sie werden genutzt, um verschiedene Gegenstände oder Bauteile des Eigenbedarfs selbst herzustellen oder nach Belieben anzupassen. Entsprechend naheliegend war es, dass schnell die ersten Modelle nachgebildet wurden, oder man sich an beliebten Steckbausteinsystemen wie LEGO orientierte, um sich eigene Sets zu drucken.

Diese Eignung für den Modellentwurf und Modellbau erweckte auch die Idee, ein Modell eines Hauses zu drucken, welches in sich aus strukturierten Bauteilen zusammengesetzt ist und somit auch das Entfernen einzelner dieser Bauteile erlaubt, um einen einfacheren Einblick in das Modell zu erhalten. Kombiniert mit dem Interesse an der Architektur entstand die Überlegung, ob es möglich wäre, anhand eines Grundrisses, welchen man aus einem Konstruktionsprogramm wie beispielsweise AutoCAD in Form einer .dxf-Datei erhalten kann, ein 3D-Modell des Hauses zu erzeugen, welches mithilfe eines Programmes automatisch in die vorgesehenen Bauteile zerlegt wurde, das im Anschluss von einem 3D-Drucker gedruckt werden kann. Dem Nutzer wird demnach nur zuteil, den Grundriss einzuspeisen und die ausgegebenen Bauteile korrekt auszudrucken, was ihm einen aufwendigen Modellierungs- und Zerlegungsprozess erspart.

Ein solches Modell soll dann Architekten als Möglichkeit vorliegen, um ihren Kunden vor dem Kauf eines Hauses näheren Einblick in die Immobilie zu gewähren und mit ebenfalls 3D-gedruckten Möbeln bereits im Voraus erste Einrichtungsideen zu überprüfen. Diese Methode würde auf ein ausgeprägtes dreidimensionales Vorstellungsvermögen des Kunden verzichten und als Ergänzung zum vorgelegten Grundriss funktionieren.

Die Umsetzung des Programms erfolgt in der Programmiersprache Java. Um die Problemstellung zu bewältigen, musste zunächst eine systematisch

einzuhaltende Zerteilung des Modells festgelegt werden.

2. Grundlagen

2.1 Planare Graphen

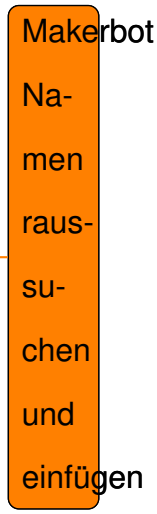
Ein planarer, auch plättbarer Graph ist ein Graph der in einer Ebene mithilfe von Punkten bzw. Knoten (1) und Kanten (2) dargestellt werden kann, ohne dass sich zwei oder mehr Kanten schneiden. Jede Fläche (3) des Graphen wird dabei durch mindestens drei verschiedene Kanten beschrieben, die den Rand dieser Fläche darstellen. Die Fläche um den Graphen herum, welche scheinbar unbegrenzt groß ist, wird äußerstes Gebiet genannt.

2.2 Doubly connected edge list

Um planare Graphen ohne Informationsverlust zu speichern werden in der Informatik Referenzen zwischen den einzelnen Bestandteilen des Graphen eingesetzt.
In der sogenannten „Doubly connected edge list“ (DCEL) erhält eine Kante, die aus einem Anfangsknoten und Endknoten besteht jeweils eine Vorgänger-, eine Nachfolger- und eine Zwillingskante. Jedem Knoten wird außerdem eine ausgehende Kante und allen Flächen eine anliegende Kante zugewiesen.
Diese zwischenobjektlichen Referenzierungen ermöglichen es, ausgehend von einem Element ohne umfangreiche Berechnungen auf alle anderen Objekte zu schließen, indem bei Knoten und Flächen die zugehörigen Kanten, beziehungsweise bei Kanten deren Vorgägner und Nachfolger betrachtet werden.

2.3 OpenSCAD

Um ein Modell mit dem vorliegenden 3D-Drucker „MakerBot X5098“ auszudrucken, muss den Windows eigenen Treibern eine .stl-Datei zur Verfügung stellen. Für dieses Format existieren jedoch keine grafikfähigen Bearbeitungsprogramme vor, die ein Modellieren erleichtern.
Deshalb muss das Modell zunächst im Programm „OpenSCAD“ modelliert und in einer .scad-Datei gespeichert werden. OpenSCAD generiert die 3D-Modelle auf der Basis einer simplen Sprache, die dem Nutzer leicht verständlich geometrische Elemente zur Verfügung stellt. OpenSCAD bietet zusätzlich zu zahlreichen Modellierungshilfen, wie dem Highlighten bestimmter Objekte oder dem Modularisieren bereits erstellter Modelle, im Anschluss an die Modellierung auch die Möglichkeit, Dateien im .scad-Format in .stl-Dateien umzuwandeln.
Von weiterem Vorteil ist außerdem, dass .scad-Dateien ähnlich wie Dateien des .txt-Formates sehr simpel aufgebaut sind und somit sehr leicht von separaten Programmen erstellt und bearbeitet werden können.



3. Vorgehen zur Problemlösung

3.1 Einlesen des Grundrisses

3.1.1 Funktionsweise der Bibliothek *kabeja*

Den Beginn der Verarbeitung markiert hierbei die Grundrissdatei, in welcher sämtliche Werte, welche im weiteren Verlauf des Programmes relevant werden, enthalten sind. Das Einlesen der Daten eines Grundrisses, wie in Abb. 5, erfolgt mit der Java-Bibliothek „kabeja“. Diese ermöglicht es, aus .dxf-Dateien alle DXF-Objekte eines bestimmten Typs zu erhalten und deren Werte in einer Liste zu speichern und später zu verarbeiten [6].

```
1  public static ArrayList<Line> getAutocadFile(String filePath)
       throws ParseException {
2        ArrayList<Line> vcs = new ArrayList<>();
3        Parser parser = ParserBuilder.createDefaultParser();
4        parser.parse(filePath, DXFParser.DEFAULT_ENCODING);
5        DXFDocument doc = parser.getDocument();
6        List lst = doc.getDXFLayer("0").getDXFEntities(
             DXFConstants.ENTITY_TYPE_LINE);
7        for (int index = 0; index < lst.size(); index++) {
8              DXFLine dxfline = (DXFLine) lst.get(index);
9              Line v = new Line(
10             new Vector(round2(dxfline.getStartPoint().getX()),
                   round2(dxfline.getStartPoint().getY())),
11             new Vector(round2(dxfline.getEndPoint().getX()),
                   round2(dxfline.getEndPoint().getY())));
12             vcs.add(v);
13       }
14
15       return vcs;
16 }
```

Codeauschnitt 1: DXF File Parser

Abbildung 1: Eins Grafik

In dieser Anwendung wird eine Funktion der Klasse „DXFReader“ verwendet, welche den Pfad zur .dxf-Datei als Parameter übergeben bekommt. Aus dieser Datei werden dann alle DXF-Objekte, die mit dem Typen `DXFLine` übereinstimmen, in einer Liste zurückgegeben. Die Koordinaten der Start- und Endpunkte der DXFLines in dieser Liste werden anschließend in eine Liste von Linien übertragen, welche im weiteren Programmablauf unter anderem für die Umwandlung des Graphen in die DCEL verwendet werden.

3.1.2 Funktionsweise der GUI

asdf

3.2 Erstellen der DCEL

3.2.1 Graph Klasse

Die wichtigste Klasse der Anwendung ist `Graph`. Sie verbindet alle grafischen Bestandteile und kann mit einer übergebenen Liste aus Linien mittels mehrerer Zwischenoperationen eine vollständige DCEL, also einen planaren Graph erstellen. Dafür sind drei Listen vom Typ `Edge`(Kante), `Node`(Knoten) und `Face`(Fläche) gespeichert, die eine DCEL repräsentieren.

3.2.2 Line-to-Edge Konvertierung

Initialisierend werden die Start- und Endpunkte der eingelesenen Linien in Knoten ohne eine Referenz auf eine anliegende Kante umgewandelt und diese als Start- und Endknoten der entsprechenden Kante gesetzt. Um für die Eindeutigkeit der Knoten zu sorgen, gibt die Funktion `createNode()` ein äquivalenten Knoten zu einem Punkt zurück und fügt ihn zu der Liste an Knoten hinzu, wenn er noch nicht in ihr existiert.

```
1 private void processData(ArrayList<Line> ls) {
2       for (Line l : ls) {
3             edges.add(new Edge(createNode(l.getP1()),
                  createNode(l.getP2())));
4       }
5 }
```

Codeauschnitt 2: Line-to-Edge Konvertierung

```
1 private Node createNode(Vector p) {
2       for (Node n : nodes) {
3             if (n.getOrigin().equals(p)) {
4                   return n;
5             }
6       }
7       nodes.add(new Node(p));
```

```
8         return (nodes.get(nodes.size() - 1));
9   }
```

Codeauschnitt 3: `createNode()` Funktion

Hier könnte man noch was zu sagen

3.2.3 Zwillingskantengenerierung

Um aus den entstandenen Kanten die invertierten Gegenstücke, auch als „Zwillingskanten“ bezeichnet, zu erhalten, werden alle Kanten, die in der Liste bereits vorhanden sind, betrachtet und neue Kanten hinzugefügt, die im Vergleich zu den ursprünglichen Kanten vertauschte Start- und Endknoten besitzen. Direkt nach dem Hinzufügen der neuen Kante wird jeweils eine Referenz erstellt, die in beiden Kanten auf den jeweils zugehörigen Zwilling verweist. In der Kantenliste existiert nun für jede Linie die der DXF-Reader eingelesen hat, zwei zueinander komplementäre Kanten.

```
1   private void computeTwins() {
2         int amount = edges.size();
3         for (int i = 0; i < amount; i++) {
4               edges.add(edges.get(i).generateTwin());
5               edges.get(i).setTwin(edges.get(edges.size() - 1));
6               edges.get(edges.size() - 1).setTwin(edges.get(i));
7         }
8   }
```

Codeauschnitt 4: Hinzufügen der Zwillingskanten

3.2.4 Nachfolger- und Vorgängerermittlung

Das Erstellen der Referenzen werden zuerst alle ausgehenden Edges der Nodes, das heißt alle Edges, die den jeweiligen Node als ihren Startnode besitzen, in einer zweidimensionalen ArrayList gespeichert. Die erste Dimension steht für den Index des Nodes in der erstellten Nodeliste für den

in der zweiten Dimension die jeweiligen ausgehenden Edges vorliegen. Da diese durch eine `for()` Schleife mit der oben stehenden Bedingung herausgesucht werden, sind die Edges im Array in zufälliger Reihenfolge, also nicht nach der Anordnung am Node gegenwärtig. Jetzt werden die Edges anhand des `atan2()` Winkel am vorliegenden Node im mathematisch positiven Drehsinn sortiert. Daraus ergibt sich, dass das vorherige bzw. nachfolgende Element einer Edge die Edge, die „links“ bzw. „rechts“ der Betrachteten liegt darstellt.
Für jede ausgehende Edge `e` können nun folgende Referenzen gesetzt werden:

- Das vorherige bzw. letzte Element der ArrayList, wenn die betrachtete Edge den Index 0 hat, stellt den Nachfolger der Twinedge von `e` dar.
- Der Twin der in der ArrayList nachfolgenden Edge bzw. ersten Edge, wenn die betrachtete Edge das letzte Element ist, ist der Vorgänger von `e`

Genannte Referenzen werden nun gesetzt, sodass die Verknüpfungen zwischen den Edges fertiggestellt sind.
code eventuell bilder definitiv

3.2.5 Flächenerstellung

Durch eine Schleife können die einzelnen Flächen herausgefiltert werden. Zuerst wird eine Boolean-ArrayList mit der selben Länge der Edgeliste erstellt, welche die Indices der schon behandelten Kanten repräsentiert. Folglich besteht die Liste anfangs nur aus `false` Werten. Jetzt wird fortlaufend eine nicht behandelte Kante herausgesucht und von dieser
keene Ahnung wie ich das mit der repräsentativen Liste formulieren soll

Ab diesem Schritt existieren alle Flächen vollständig in der Flächenliste. Das unendliche bzw. äußere Gebiet lässt sich

3.2.6 Vervollständigung der Knoten

Die letzte nötige Operation ist die Speicherung einer anliegenden Edge in den Nodes.

man könnte das im code auch vorher tun denke ich

3.3 Aufbau der Einzelbauteile

3.3.1 OpenSCAD Java Interface

Für die erleichterte Erstellung von OpenScad Objekten wurde ein Java Interface `ScadObject` erstellt, welches alle für das Projekt wichtigen Befehle enthält. Die Methode `toString()` stellt in den Klassen des Interfaces die Übergabe des OpenSCAD Befehlsstrings da. So kann man z.B. mit der Klasse `Cube` einen Quader mit Länge, Höhe und Breite erstellen der dann wie folgt mit `Cube.toString()` in einen String konvertiert wird:

```
cube([Länge, Breite, Höhe]);
```

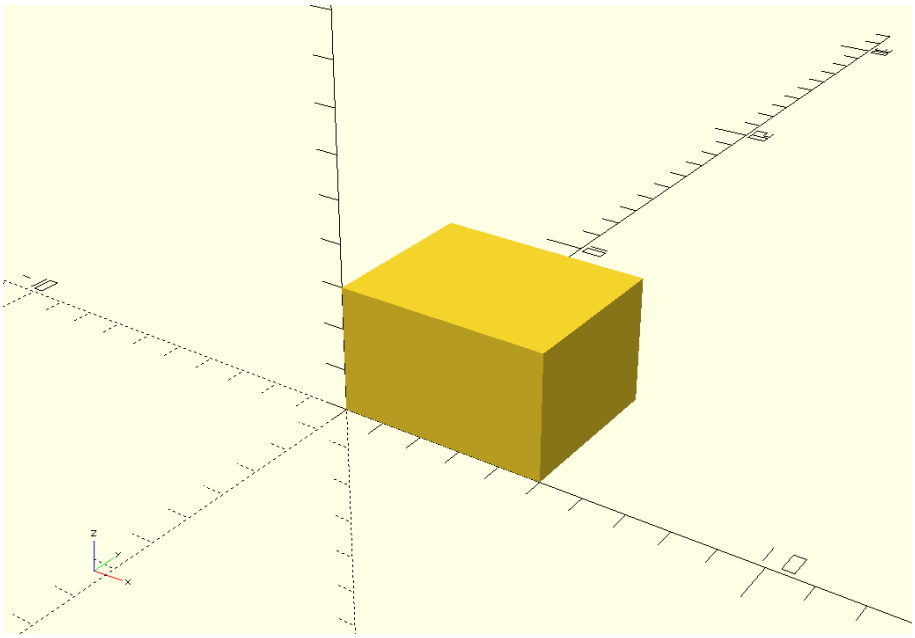


Abbildung 2: Ergebnis von `new Cube(3, 4, 5).toString()`

3.3.2 Corner

3.3.3 Wall

3.3.4 BasePlate

3.4 Druck

3.4.1

4. Ausblick

Im aktuellen Entwicklungsstand ist es nur möglich, alle Bauteile einzeln auszudrucken. Dies erhöht jedoch den Filamentverbrauch des 3D-Druckers um ein Vielfaches, weshalb eine Kombination mehrerer Bauteile für einen Druckvorgang zwecks der Reduktion des verwendeten Filaments für den Druck unterstützende Elemente als sinnvoll anzusehen ist. Dafür bietet sich beispielsweise ein gemeinsamer Druck von Wandteilen oder Eckpfeilern anbieten, da diese Objekte weitestgehend ähnliche Ausmaße besitzen und somit eine recht effektive Kombination möglich ist. Außerdem liegen momentan lediglich Bauteile vor, welche nur auf einer Druckplatte fester Größe gedruckt werden können. Sollte das zu druckende Objekt größer als die Druckplatte sein, muss es zum Drucken skaliert werden, was jedoch unbedingt vermieden werden soll, da dadurch die Verhältnisse der Stecker zueinander verändert werden und so ein sachgemäßer Aufbau verhindert wird. Um diesen Umstand zu verhindern, soll es in der weiteren Entwicklung möglich sein, überdimensionierte Bauteile weiter in kleinere Untereinheiten zu teilen und so eine Wahrung des Maßstabs zu garantieren. Hierfür muss jedoch ein weiteres Stecksystem, sowie weitere Logik zur Umsetzung und Umwandlung der alten Bauteile konzipiert und implementiert werden. Als ferne Zukunftskonzeption, die an den Rahmen der Besonderen Lernleistung anschließt, lässt sich die Umsetzung von 3D-Modellen festmachen. Hierzu zählen kompliziertere Wände mit Schrägen, Fenstern oder Verstrebungen und Dachgestelle, welche als Abschluss auf dem Modell angebracht werden können. Die Komplexität der Aufgabenstellung wird dadurch aber um ein Vielfaches gesteigert, weshalb diese Problematik kein Bestandteil der Besonderen Lernleistung sein wird.

Literaturverzeichnis

[1] de.wikipedia.org/wiki/Gau%C3%9Fsche_Trapezformel (Stand: 21.03.2017, 12:00 Uhr)

[2] cs.sfu.ca/~binay/813.2011/DCEL.pdf (Stand: 25.04.2017, 12:00 Uhr)

[3] en.wikipedia.org/wiki/Doubly_connected_edge_list (Stand: 25.04.2017, 12:00 Uhr)

[4] en.wikibooks.org/wiki/OpenSCAD_User_Manual (Stand: 25.04.2017, 12:00 Uhr)

[5] docs.oracle.com/javase/8/docs/api/index.html (Stand: 25.04.2017, 12:00 Uhr)

[6] kabeja.sourceforge.net/ (Stand: 12.10.2017, 10:00 Uhr)

[7] http://stackoverflow.com/questions/6740736/reading-a-dxf-file-wit (Stand: 09.05.2017, 14:00 Uhr)

[8] http://www.journaldev.com/864/java-open-file (Stand: 28.05.2017, 17:15)

Abbildungsverzeichnis

Selbstständigkeitserklärung

Johann Bartel:

Ich versichere, dass ich die Arbeit selbstständig angefertigt, nur die angegebenen Hilfsmittel benutzt und alle Stellen, die dem Wortlaut und dem Sinn nach anderen Werken entnommen sind, durch Angabe der Quellen als Entlehnung kenntlich gemacht habe.
Mit der schulinternen Verwendung der Arbeit bin ich einverstanden.

_____

Ort, Datum, Unterschrift

Peter Oehme:

Ich versichere, dass ich die Arbeit selbstständig angefertigt, nur die angegebenen Hilfsmittel benutzt und alle Stellen, die dem Wortlaut und dem Sinn nach anderen Werken entnommen sind, durch Angabe der Quellen als Entlehnung kenntlich gemacht habe.
Mit der schulinternen Verwendung der Arbeit bin ich einverstanden.

_____

Ort, Datum, Unterschrift