FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

3D Mesh Unfolding

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Thorsten Korpitsch
Matrikelnummer 01529243

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ph.D. Hsiang-Yun Wu

Wien, 1. März 2019

Thorsten Korpitsch — Hsiang-Yun Wu

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

3D Mesh Unfolding

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Thorsten Korpitsch
Registration Number 01529243

to the Faculty of Informatics

at the TU Wien

Advisor: Ph.D. Hsiang-Yun Wu

Vienna, 1st March, 2019

Thorsten Korpitsch   Hsiang-Yun Wu

Erklärung zur Verfassung der Arbeit

Thorsten Korpitsch

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. März 2019

Thorsten Korpitsch

Danksagung

Ihr Text hier.

Acknowledgements

Enter your text here.

Kurzfassung

Ihr Text hier.

Abstract

Enter your text here.

Contents

CHAPTER 1

Introduction

- Describe what 3D Meshunfolding is
- What can it be used for? - Papercraft mostly, self folding materials
- What Problems arise? - Performance, Distortion, Overlaps

1.1 Motivation

- Can be used for Papercraft and Papercraft is fun
- Previous work mostly focused on unfolding
- Gluetags can improve reconstruction experience
- Can simple algorithms yield good results?

1.2 Goal

- Implement algorith to transform 3D models into planar patch
- add minimum amount of gluetags, to have stable structure
- detect overlaps and resolve them
- visualize all steps of the algorithm with opengl
- evaluate the findings

CHAPTER 2

Definitions

2.1 Dualgraph

- what is a dual graph
- describe what it is used for in this problem

2.2 Minimum Spanning Tree

- what is a minimum spanning tree
- how is it related to this problem

2.3 Simulated Annealing

- what is simulated annealing
- how can it be used for this problem

CHAPTER 3

Implementation/Methodology

3.1 Overview

- general approach - what steps do i have to do:
- read the data from files
- display and interact with 3d model
- calculation of gluetags
- unfolding of model

3.2 Datastructure

- cgal datastructure as a base - short explaination why polyhedron mesh is usefull -> halfedges linking faces together
- own datastructures description planarTriangles vs. 3D Triangles to "link" 2d triangle with 3d triangle
- gluetags are already calculated and kept in the datastructure for later unfolding

3.3 Unfolding

- explaining dual graph calculation
- explaining algorithm, how simulated annealing is adapted for this problem

- explaining calculation of spanning tree
- explaining unfolding of 3d triangles
- explaining overlap detection how it is done
- explaining decision if triangle unfolding was found -> gluetgs are calculated
- explaining decision if a gluetag is used or not
- gluetag overlap detection similar to triangle overlap -> gluetag consist of two triangles
- new energy < old energy we take this step

3.4 Conflict Detection

- conflict detection most important part
- description and example of problematic areas that often arise
- triangles checked against other triangles - write how i did it, calculating
- gluetags checked against triangles and other gluetags

CHAPTER 4

Evaluation

4.1 Performance

- table evaluating the performance against some example meshes
- maybe some pictures of unfoldings and original mesh??
- maybe graph for performance per polygon increase

4.2 Limitations

- random walk in annealing can be problematic, especially for big and complex meshes - sometimes finds fast solution, sometimes doesn't
- bigger meshes need far more iterations -> more sophisticated algorithm needed to scale better
- amount of gluetags cannot be calculated in advance - unfolding decides how many gluetags are needed

CHAPTER 5

Discussion

- comparison to bruteforcing the unfolding
- show some formulas that there are really really many possible unfoldings and trillions of possible gluetag positions and actually that my algorithm works rather well
- even sophisticated algorithms for unfolding without gluetags have performance problems if the mesh gets too big

CHAPTER 6

Conclusion

6.1 Summary

- ???? overall it works fast for smaller meshes, but has its limitations as discussed before

6.2 Future Work

- post process gluetags to resolve very small overlap areas to cut down computation time
- change minimum spanning tree to more sophisticated calculation to get an unfolding faster than with a random walk
- find a way to make search space smaller for gluetags to improve performance

List of Figures

List of Tables

List of Algorithms