TIM OESTERREICH

BUILDING LARGE DYNAMIC TRUSS STRUCTURES

BUILDING LARGE DYNAMIC TRUSS STRUCTURES

TIM OESTERREICH

Using TrussFormer

April 2018 – version 4.5

Ohana means family.
Family means nobody gets left behind, or forgotten.

— Lilo & Stitch

Dedicated to the loving memory of Rudolf Miede.

1939 – 2005

ABSTRACT

Short summary of the contents in English... a great guide by Kent Beck how to write good abstracts can be found here:

https://plg.uwaterloo.ca/~migod/research/beckOOPSLA.html

ZUSAMMENFASSUNG

Kurze Zusammenfassung des Inhaltes in deutscher Sprache...

We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.

— Donald E. Knuth [1]

ACKNOWLEDGMENTS

Put your acknowledgments here.

Many thanks to everybody who already sent me a postcard!

Regarding the typography and other help, many thanks go to Marco Kuhlmann, Philipp Lehman, Lothar Schlesier, Jim Young, Lorenzo Pantieri and Enrico Gregorio[1], Jörg Sommer, Joachim Köstler, Daniel Gottschlag, Denis Aydin, Paride Legovini, Steffen Prochnow, Nicolas Repp, Hinrich Harms, Roland Winkler, Jörg Weber, Henri Menke, Claus Lahiri, Clemens Niederberger, Stefano Bragaglia, Jörn Hees, Scott Lowe, Dave Howcroft, José M. Alcaide, and the whole LaTeX-community for support, ideas and some great software.

Regarding LyX: The LyX port was intially done by *Nicholas Mariette* in March 2009 and continued by *Ivo Pletikosić* in 2011. Thank you very much for your work and for the contributions to the original style.

1 Members of GuIT (Gruppo Italiano Utilizzatori di TeX e LaTeX)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

1 INTRODUCTION

Personal fabrication devices, such as 3D printers, are already widely used for rapid prototyping and allow non-expert users to create interactive machines, tools and art. As consumer-grade 3D printers are usually desktop-sized, the size of these objects is, however, fairly limited. TrussFormer aims to enable users to create large-scale dynamic objects using desktop-sized 3D printers. Scale can be achieved by creating multiple small-sized objects and connecting them to each other. If all parts of a large object would be 3D printed, this process would take a long time and special large-size 3D printers would be needed. Our solution to this problem is to take ready-made objects, like empty plastic bottles, and only print the connectors that keep them together. To aid users in this process, we developed a software simulation that can create objects which are capable of handling the substantial forces large object intrinsically have. We achieve this by providing stable primitives which can be attached together. These primitives resemble truss structures - beam-based constructions creating closed triangle surfaces, which are intrinsically sturdy and material-efficient.
In order to build the simulated objects, we provide export-functionalities. Our software also provides tools to evaluate the magnitude of force acting on the links.
- TODO:
- node-link-structure
- export
- force

1.1 TRUSSFAB

- create big structures
- create them quickly and cheaply
- explain concept of nodes and edges

1.2 TRUSSFORMER

- make structures move
- observe forces during movement
- create animation
- define hinges

1.3 TRUSSCONTROL

- closed-loop movement control
- automatic conversion of simulation animation to arduino code

2 RELATED WORK

2.1 large-scale personal fabrication

2.2 construction kits

2.3 prototyping with ready-made objects

2.4 building with variable geometry trusses

- Steward Platform
- Walking Octa

2.5 software pipeline for animatronics

2.6 sketchup

3 WALKTHROUGH

3.1 DESIGNING STATIC STRUCTURES

- placement of structurally stable primitives
- importing previously built objects
- edit object (grow/shrink, move tool)

3.2 ADDING MOVEMENT TO THE STRUCTURES

- placing actuators
- placing primitives with variable geometry trusses
- demonstrate movement tool

3.2.1 *Force Analysis*

- check tension force on edges
- check acceleration and speed on nodes
- add loads to object
- check tension while moving
- automatically fix movement when object is exceeding force

3.3 CONTROLLING THE STRUCTURE

- closed-loop control -> more sophisticated and complex movements possible

3.3.1 *PID Control*

- short intro: how does PID work?
- how do we use it?
- i.e. position control of actuators
- forward reference to section 4 (setup of length measurement)

3.4 BUILDING THE FINAL OBJECT

After the object was sufficiently tested in the editor, it is time to print the connectors and assemble the final object.

3.4.1 *OpenSCAD*

At first, our abstract description of the object has to be converted into a physical representation. In order to achieve this, we used a modeling language called *OpenSCAD*. The *Export Hubs and Hinges* button will automatically morph the structure into a statically sound object, i.e. it will elongate and shorten edges so, that the ideal amount of movement is possible.

This needs to be more detailed for sure!!

The resulting arrangement of nodes and edges will be transferred to OpenSCAD. Using templates, we can create parameterized representations of hubs and hinges, which, when assembled, will exactly represent the object in the editor. This will be explained in more detail in 5.2.4.

3.4.2 *Printing the Parts*

Each OpenSCAD file represents a single part in the structure. These files can easily be converted to *.stl* files, which are typically used for 3D printing. These files have to be imported into any 3D printing software, arranged efficiently and send to a 3D printer.

put conversion script in here somehow

add some time reference here?

3.4.3 *Assembling the Structure*

The resulting hubs and hinges contain an ID system for easy assembly. Each part of a node has the node ID printed on. That way it is easy to find out which hinge-parts belong together. Additionally, each "extended" edge-line (elongation) contains the id of the connected edge.

Verlängerung einer Edge, also quasi die Elongation. FIND A BETTER NAME!

A compound elongation, which is the usual case for a hinge, is therefore assembled by finding two parts with the same node and edge ID. For static hubs, this concept is similar, but of course these do not have to be assembled.
Two connectors with different node IDs but the same edge IDs will be connected by a link.

4 HARDWARE

- chapter will talk about challenges we faced in finding stable connectors
- material used: PLA (biodegradable, sturdy enough, ...)
- assembled based on ID system
- no special requirements to printer
- we used: UltiMaker3, UltiMaker2 and

remember name of other printer

4.1 BUILDING PARTS

We can differentiate between three essential building parts for our truss structures. *Links* are the connecting and shaping parts. We used PET bottles for these parts, because they are readily available, cheap and sturdy.
These links can be connected in two different ways. If the truss primitive is static, i.e. it does not allow deformation, we connect them by *hubs*. Hubs are single-part connectors for an arbitrary number of edges. They do not allow movement.

make sure people understand what that means

If the structure is intended to allow deformation, we can not use this single-part approach. In this case, movement is created having multiple parts that can hinge around each other. These *hinge chains* are generated according to the number of edges connected to the node and the angle of each edge relative to each other edge. In contrast to hubs, hinge chains have to be assembled manually using nuts and bolts.
Links and hubs or hinge chains, respectively, are connected by specially-printed connecting *cuffs*, which fit over the bottles thread and a fitting counter-part on the connecting end of the node.

4.1.1 *Links*

We opted to use 1l (big) and 0.5l (small) reusable PET bottles because of their intrinsic stability and abundant availability. Two bottles are connected on their bottom side by a wood screw, which is inserted using a special long-necked screwdriver. The resulting link-lengths are:

1. 60 cm - two big bottles
2. 53 cm - one big and one small bottle
3. 46 cm - two small bottles

4.1.2 *Hubs*

4.1.3 *Hinge Chains*

- beginning: open hinge chains
- later: closed hinge loops

4.1.4 *Cuffs*

In order to connect links to nodes, we developed a custom coupling system. These cuffs fit exactly over the neck of the bottle and special connecting parts on the hubs. - something about sizes of bottle neck
- size of connecting part
- little dimple for extra stability

4.2 CONTROLS

4.2.1 *Electric vs. Pneumatic Actuators*

4.2.2 *Open Loop vs. Closed Loop*

5 IMPLEMENTATION

We implemented TrussFormer as a plug-in for the 3D modeling software *SketchUp*. It is primarily written in Ruby and JavaScript.

Do we assume TrussFormer is a new product, which uses similar functionality as TrussFab, or do we say it is an improvement?

5.1 ARCHITECTURE

The software can be divided into four components. The most user-facing one is the designer. The other components handle the physics simulation of the created structures, minimization logic for the created hubs and hinges and the 3D print export.
Designer:
- designer keeps track of connections between components
- written in ruby
- components stored in graph structure

show diagram?

- tight coupling to SketchUp (uses SketchUp Elements, SketchUp rendering engine, ...)Physics Simulation:

Maybe the details should be subsections?

- Based on *MSPhysics* by Anton Synytsia - Ruby wrapper around C++ physics engine *Newton Dynamics* - Implemented as a *SketchUp Animation* - implements *nextFrame* method - this method is called every time SketchUp has finished rendering a frame - this method does:

1. tell Sketchup to render new frame (SketchUp will render the positions calculated in the previous world update: make use of calculate new update while sketchup already renders new positions)
2. call *update_world*, which does, world_iterations times:
   - update forces, i.e. call apply predetermined forces (e.g. weights on hubs, calculations of PID controller)
   - call *world.advance*: Tell MSPhysics, that a new world update is available and let it calculate new forces after positional updates
   - record tensions on links, for visualization later. This has to be recorded, because for each render step, a number of world updates are done. We don't want to miss crucial force updates
   - visualize forces: send color information to SketchUp, indicating the strength of the tension on links
3. update entity positions: tell SketchUp where components have to be rendered next time
4. send data to ui: send sensor data to ui charts, if needed

5.2 TRUSSFAB DESIGNER

The TrussFab Designer provides static sketching functionalities. It can create and display different predefined models, has knowledge about the connections of different components and can modify the resulting objects structure.

5.2.1 *User Interface*

5.2.2 *Structure Creation*

5.2.3 *Modifying the Structure*

5.2.4 *OpenSCAD Export*

5.3 TRUSSFORMER PHYSICS ENGINE

5.3.1 *Simulation*

5.3.2 *Automatic Actuator Placement (if it works soon-ish)*

5.4 FORCE CONTROL

5.4.1 *PID*

6 CONCLUSION

BIBLIOGRAPHY

[1] Donald E. Knuth. "Computer Programming as an Art." In: *Communications of the ACM* 17.12 (1974), pp. 667–673.

DECLARATION

I certify that the material contained in this thesis is my own work and does not contain unreferenced or unacknowledged material. I also warrant that the above statement applies to the implementation of the project.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Ich erkläre hiermit weiterhin die Gültigkeit dieser Aussage für die Implementierung des Projekts.

Potsdam, April 2018

Tim Oesterreich