FME HighEFF

Centre for an Energy Efficient and Competitive Industry for the Future

Deliverable D2.1_2017.04

Potential tube configurations design for 3D printing – status from 2017 FDM 3D printing activities

Delivery date: 2017-12-19

Organisation name of lead beneficiary for this deliverable:

SINTEF Energy Research

HighEFF- Centre for an Energy Efficient and Competitive Industry for the Future is one of Norway's Centre for Environment-friendly Energy Research (FME). Project co-funded by the Research Council of Norway and Industry partners. Host institution is SINTEF Energi AS.		
Dissemination Level		
PU	Public	X
RE	Restricted to a group specified by the consortium	

Deliverable number:	D2.1_2017.04
ISBN number:	
Deliverable title:	Potential tube configurations design for 3D printing – status from 2017 FDM 3D printing activities
Work package:	WP2.1 heat exchangers
Deliverable type:	Memo
Lead participant:	SINTEF Energy Research

Quality Assurance, status of deliverable		
Action	Performed by	Date
Verified (WP leader)	Geir Skaugen	
Reviewed (RA leader)	Armin Hafner	
Approved (dependent on nature of deliverable)*)	Armin Hafner	

*) *The quality assurance and approval of HighEFF deliverables and publications have to follow the established procedure. The procedure can be found in the HighEFF eRoom in the folder "Administrative > Procedures".*

Authors		
Author(s) Name	Organisation	E-mail address
Trond Andresen	SINTEF Energy Research	trond.andresen@sintef.no

Abstract
This deliverable is shared between HighEFF and KPN COPRO. The described activities are fairly fundamental in nature and a prerequisite for successful future activities in both projects.

Table of Contents

1 Introduction

Heat exchangers are components of crucial importance, and deserves focus on continued improvement on performance, cost, and energy- and material efficiency. A lot of development is undertaken by the manufacturers, but the majority is focussed on improving or optimizing specific heat exchanger technologies or "type". The research question asked in the HighEFF and COPRO projects is as follows: "What new potential could be realised if heat exchangers design and manufacture could be decoupled from current production methods".

There exist many types and technologies for heat exchangers, but all are based on a specific procedure for mass production. For example; plate heat exchangers consist of stacks of pair-wise identical stamped steel plates, soldered or clamped together to a unit. Material cost is typically the dominant cost factor. In a *shell-and-tube*, identical tubes are assembled in bundles, and either mechanically expanded or welded into end plates, supports, and baffle plates. The *Novel Heat Exchanger Activities* in HighEFF and COPRO aim to explore the possibilities and potential of "unrestricted" heat exchanger design for a few different applications, using both numerical modelling and experimental verification of custom manufactured heat exchanger samples in laboratory scale. The technology for additive manufacturing of designed concepts in metal alloys exist ("3D printing in metal").

There are however steps to take before experimental verification using metal samples;

1. Constructing the numerical pipeline - producing numerical 3D models directly or based on output from inhouse Heat Exchanger design model *FlexHX*.
2. Design and produce samples using conventional polymer 3D printing in order to accumulate experience on using this pipeline to produce actual samples – and adjust and upgrade both the heat exchanger design requirements and constraints, as well as the pipeline itself in order to achieve the best possible results.

In addition, 3D printing has been used to for physical visualization of results. Physical models seems to communicate well, beyond pictures and numbers alone. The picture below illustrates main results from the project KPN COMPACTS. A project goal was to minimize the weight and size of a waste heat recovery heat exchanger for offshore gas turbine exhaust. The red shape shows the base case, the current technology concept. The blue shape has the same performance, but considerably reduced size and weight.

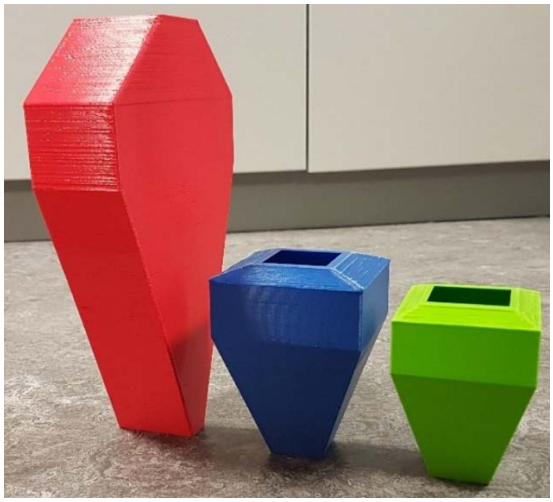
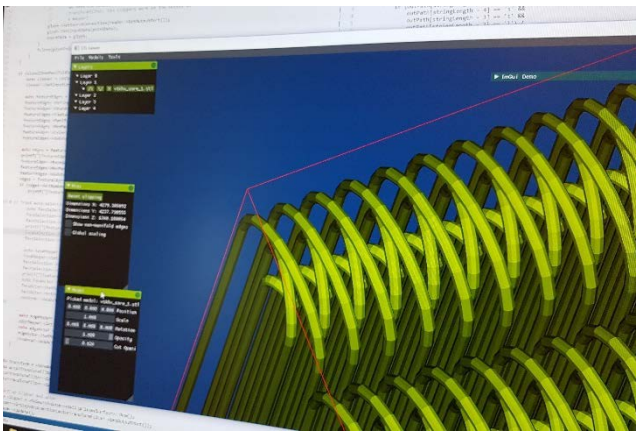


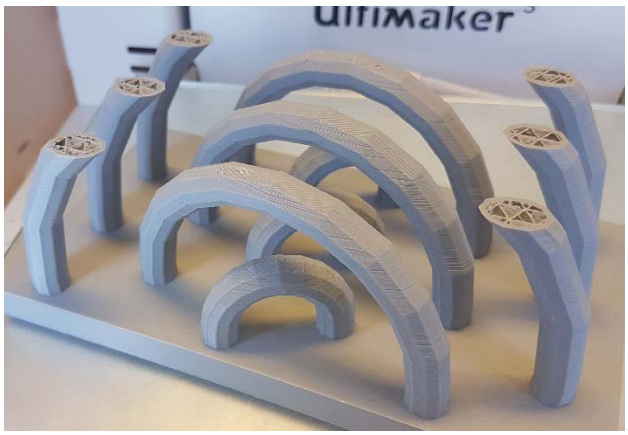
Figure 1: Visualization of heat exchanger external size comparisons

2 From numerical to physical model

SINTEF Energy Research's inhouse design and optimization model for heat exchangers is based on actual geometry parameters, and have the capability for on-screen visualization of the heat exchanger designs. Therefore, a lot of the required information needed for physical model construction was already in place. A joint summer internship between HighEFF and COPRO began constructing a work flow from code and model output, to physical model using a 3D printer. The pictures below show the on-screen visualization of the tube layout in a finned tube heat exchanger (a), and a partial 3D print of a detail from the same model (b).



a) On-screen visualization and editing



b) Partially completed print of detail from same model

Figure 2: Detail from finned-tube-heat exchanger

Many industrial scale heat exchangers contain a lot of geometry details, way beyond what is practical or useful to physically visualize completely. Printing specific details is often more purposeful. The internship report can be found in Appendix A (in Norwegian). The earlier mentioned pipeline for producing physical models from a heat exchanger design code may be different for polymer 3D printing versus additive manufacturing in metal, but we expect there also to be significant similarities.

This following sections briefly summarizes some initial conclusions on model requirements as well as technical experience with the 3D printing itself.

2.1 Model requirements for successful 3D printing

A model to be 3D printed using the common FDM (fused deposition modeling) technology is pre-processed by a software that "slices" it in horizontal sections, depending on the selected layer height in the printer settings. The actual printing process lays down one slice (layer of model) at the time, and thus building the model from the base and up.

Overhangs and bridges in the model are more challenging to print well, and care should be taken both in model design, and in placement and orientation of the model in the slicing software. Too fine details (in relation to the printer settings and capabilities) can also increase the risk of print failure.

The current 3D printer acquired in a cooperation between HighEFF and the KPN COPRO project, use 0.4mm diameter extrusion nozzles as standard. For details much smaller than the nozzle diameter, there is

increased risk for the slicing software to omit that detail. As an example; the below profile is called a multiport extruded (MPE) tube that is used in some types of heat exchanger. The inner diameter of the channels in this profile can be well below 1 mm, and the wall between two channels (called the *web*) is typically a fraction of the diameter. A mpe can thus have a web thickness of 0.1-0.2 mm. In the below example, the slicing software happily processed the 3D model and started the print, but on closer inspection, the actual web was omitted at its narrowest point, leaving a large section of the model without material for one or several layers. The print subsequently failed, as the rest of the model was no longer in physical contact with the lower layers.

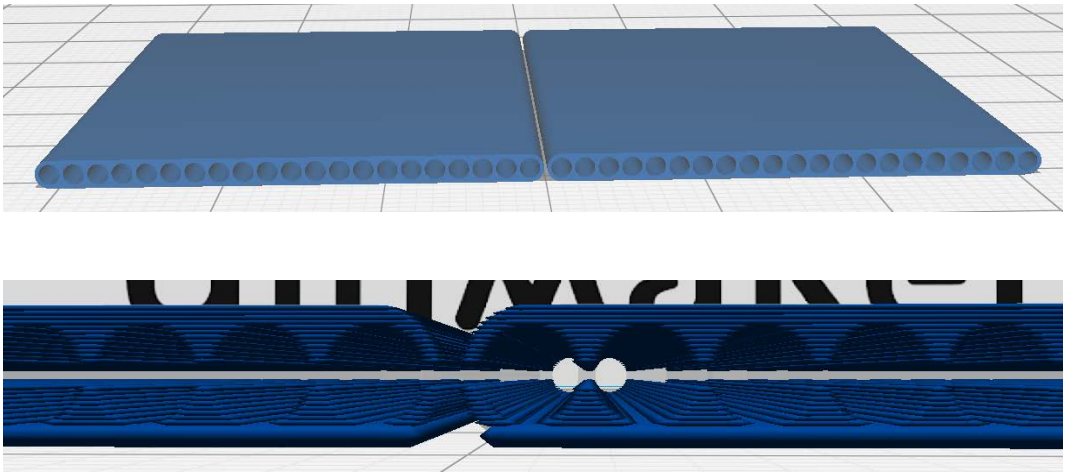
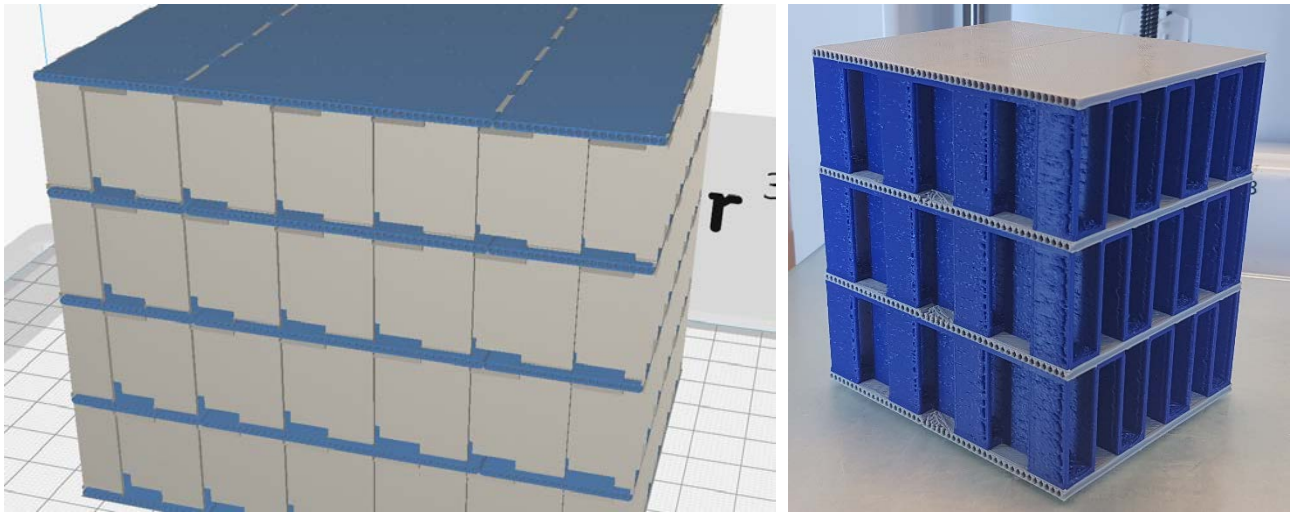


Figure 3: MPE used as "plate" in plate-and-fin heat exchanger. Too small wall thickness between channels is omitted from printing which leads to complete failure during print.

In this case, both print-settings adjustments and slight up-scaling of the model was required in order to successfully produce the sample below.



a) Model in slicing software b) Completed print

Figure 4: Plate-and-fin heat exchanger. Dual extruders allow printing two materials/colors simultaneously

The 3D printer cannot detect if it extrudes material without making contact with either the base plate or material below, and will happily continue spewing polymer threads into the air until it has completed all layers. An example of such an unhappy result can be seen below.

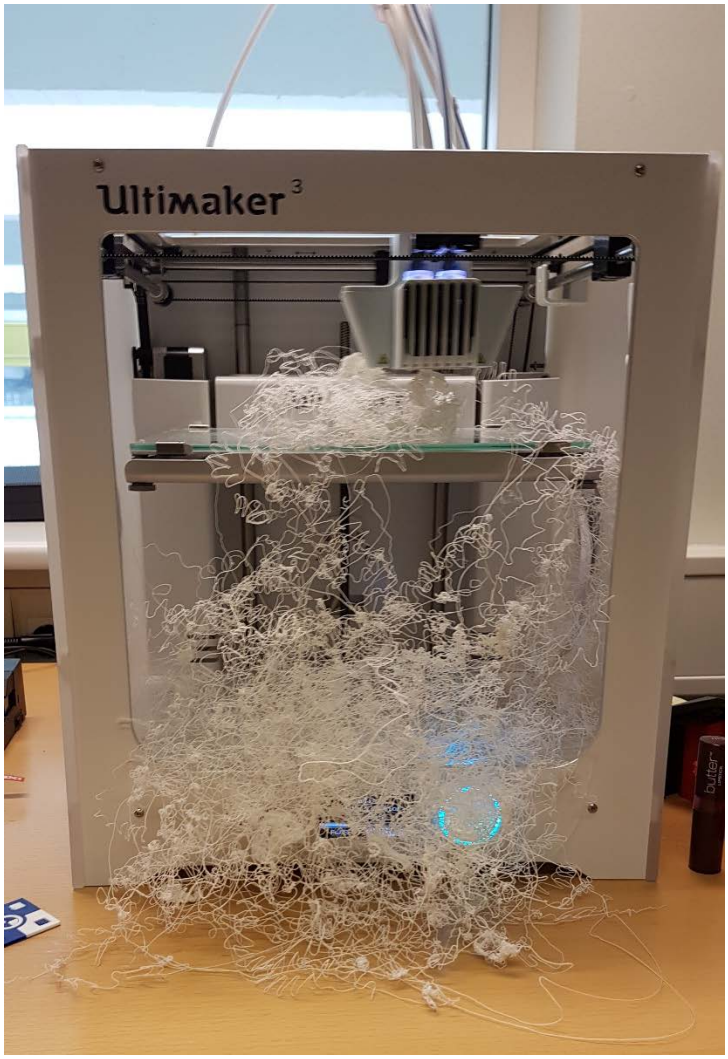


Figure 5: Print failure due to bad slicing

2.2 Random pieces of technical experiences

2.2.1 General

It is highly recommended to visually investigate the sliced model layers. This is the best opportunity to detect errors that will cause the print to fail. A typical example is thin walls or other small features. A minimum thickness of the model feature is required in order for it to be printed, as seen in the MPE-figure above.

The model should be analysed in the slicing software with focus on potential issues. Basic aspects to look for is the so-called 3 "S"es;

- Surface area – sufficient contact area promotes good adhesion to the build plate
- Stability – beware overhangs that shift the centre of mass beyond the base area during printing
- Supports – parts of the model that need structural supports in order to print correctly (e.g. overhangs)

2.2.2 Monitoring

Monitoring the printing process is highly recommended, because accidents and errors do occur. Print failures waste time and resources, and can also endanger the equipment

3D prints take a long time – the model in figure 4b took approximately 50 hours. Thus, in-situ monitoring is also impractical. Our current setup has the printer connected to a computer that can be remotely controlled, both within and outside office hours. In addition to a built in camera, an additional webcam is used for video monitoring. Two camera angles has shown to be very useful to monitor printing progress and early detection of errors.

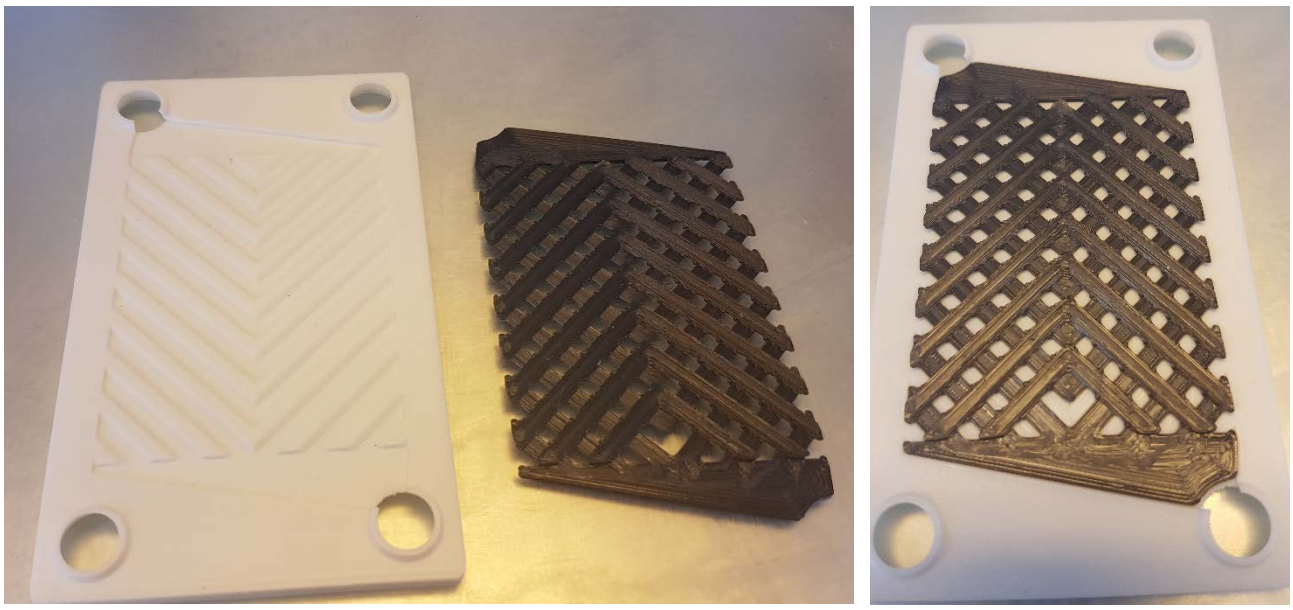
2.2.3 Filaments

Even filaments of the same material (e.g. PLA) behave differently in printing. There is not yet enough basis to indicate that some brand names are better than other, but it is clear that at the very least, different printer settings may be required to achieve similar and acceptable:

- Adhesion to base plate
- Consistent and correct flow rate at the nozzle (to avoid over/under-extrusion, blobs or holes in model)
- Surface finish

3 Additional examples

3.1 Plate heat exchanger



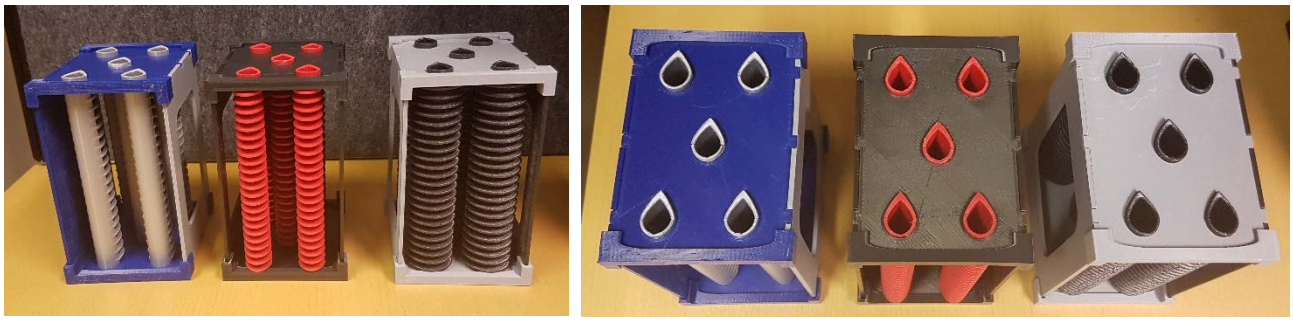
a) A chevron plate, with the fluid cross section area formed between two plates shown in black

b) Chevron plate with embedded flow pattern

Figure 6: Chevron plate and typical flow pattern/wetted perimeter

3.2 Novel finned tube heat exchanger for waste heat recovery

Some novel ideas for concepts suitable for dirty gas heat sources is under exploration. One idea is to combine drop-shaped tubes with partially or completely surrounding fins depending on operating conditions and gas composition, in order to avoid fouling or scaling.



a) Side view

b) Top view

Figure 7: Three variations on a principle idea for finned tube heat exchanger

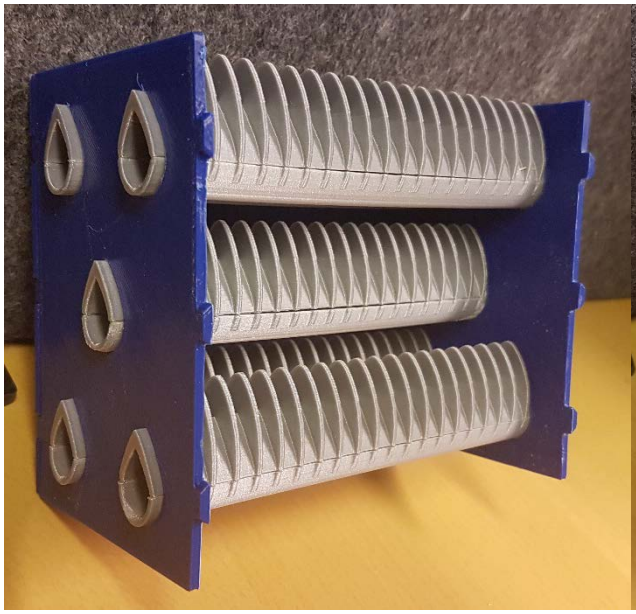


Figure 8: Trailing edge fins only

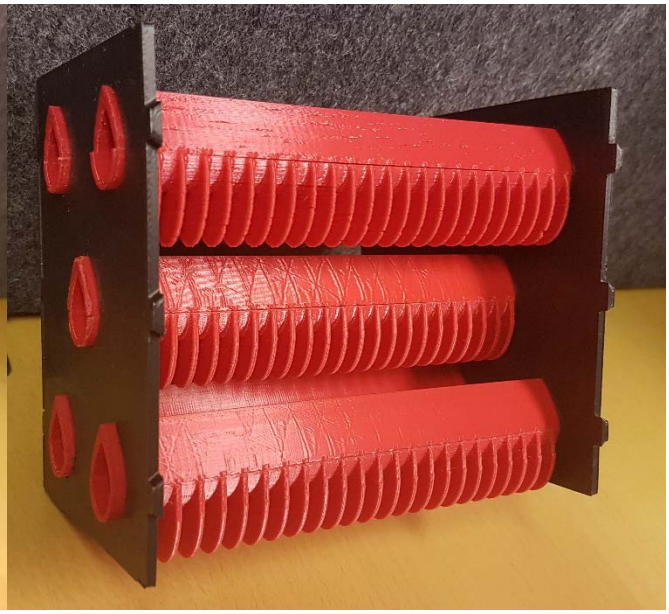


Figure 9: Leading edge fins only

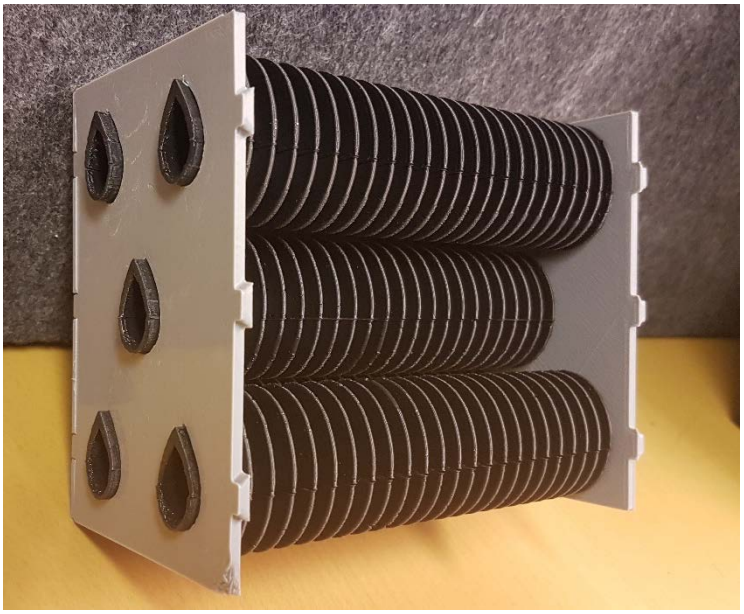


Figure 10: Complete, oval fins

3.3 Novel MPE based concept for gas-source waste heat recovery

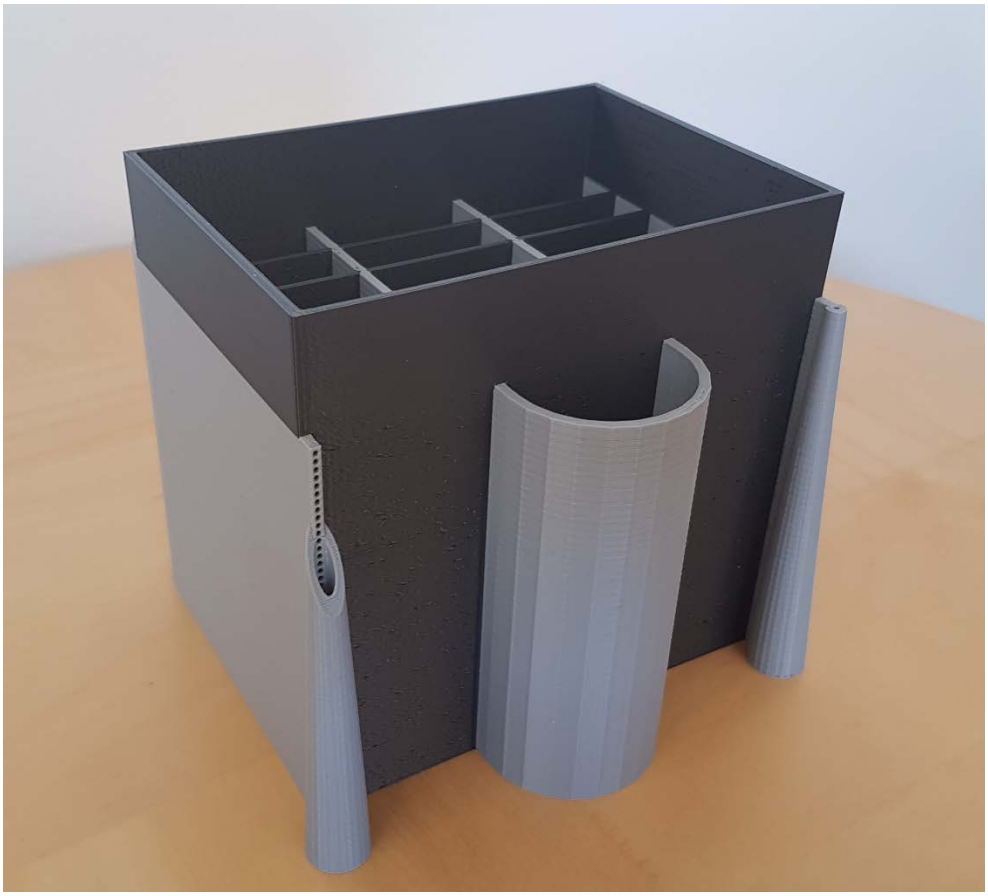


Figure 11: Novel MPE-based heat recovery concept

Appendix A. Summer intern Simon Høgås' report: "Fra data – til fysisk modell" (in Norwegian)

3D-printers: From data to physical model

Simon Høgås

3D Printing

The development of more efficient heat exchangers might give rise to more complex geometrical designs that are more difficult to visualize and are impossible to produce using conventional methods.

With the explosion of the hobby 3D-printer market, 3D-printers have become both cheaper and easier to use. This have made them more available to prototyping as well. There is also a growing interest in using 3D-printers for creating complex components, and the aerospace industry and automotive industry are already printing metal components for production use.

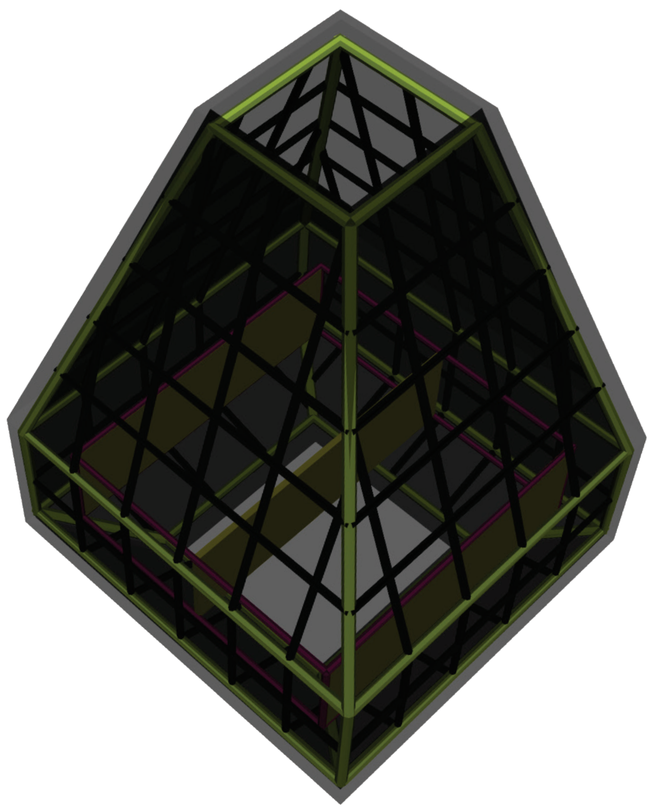
Project goal

- Understand an existing modelling program
- Explore possibilities and limits of current visualization techniques
- Create a workflow from visual representation to finished print

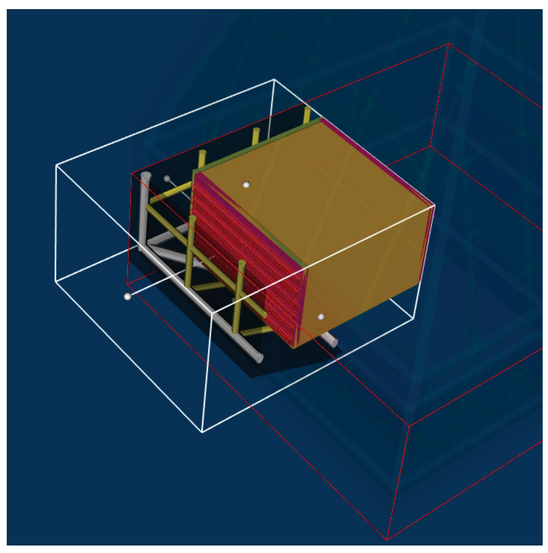
Workflow

- Start with geometry parameters from data model

- Model description uses geometry parameters to build polygon mesh of closed surfaces
  - Abstractions in data model could be used more directly for mesh generation
  - Minimize extra work from data abstraction to complete mesh

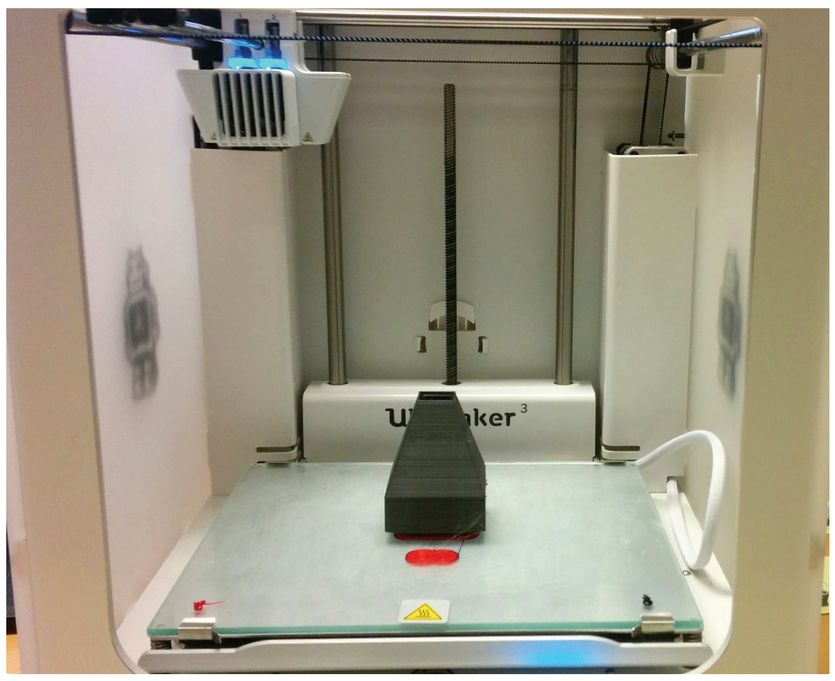


- Mesh can be viewed and modified on the computer
  - Testing of methods for model processing
  - Scale, crop and combine models, simplifications because of huge scale factors
  - Based on the Visualization Toolkit
  - Consider existing 3D modelling software



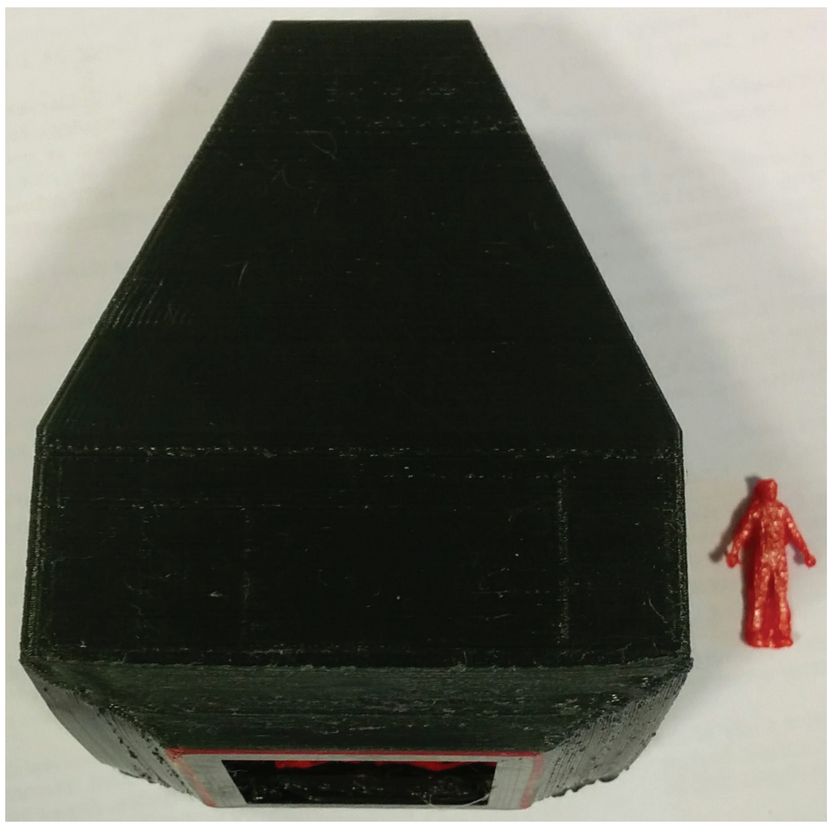
Cropping and scaling of model before print

- Modified or original mesh is converted to 3D-printer instructions
  - Mesh read from portable format



Finished 3D print in printer

- Print the 3D model
  - Show scale by comparing to known models
  - Small details, limited resolution



Casing of heat recovery steam generator with to-scale human model

Thanks to

Trond Andresen and Brede Hagen as sypervisors during toe project

Fra data – til fysisk modell

1 Introduksjon

Sommerprosjektet «Fra data – til fysisk modell» hadde som mål å se på muligheter rundt visualisering av 3D-geometri med 3D-skrivere. Utgangspunktet var noen 3D-modeller som var beskrevet med «Visualization Toolkit» (VTK). 3D-modellene defineres med parametere fra simulering og optimalisering gjennom modeller utviklet på SINTEF, FlexHX og FlexCS. Å se modellen fysisk kan bli mer aktuelt for mer komplekse geometrier. For rask iterasjon, vil det nok fortsatt være enklest å se på en 3D-modell på skjerm, men til dette trengs fortsatt en 3D-modell generert fra parametrene.

Å gjøre modellene leselige for en 3D-skriver, viste seg å være enkelt. Resten av prosjekttiden ble brukt til å utvikle verktøy for å gjøre det lettere å eksportere nye modeller til 3D-skrivere, samt å utvikle et visningsverktøy for å velge utsnitt, sette sammen modeller og skalere dem. Resultatet ligger ikke så mye i det endelige produktet, men i kodebasen som utgangspunkt til videre arbeid.

3D-skrivere har de siste årene blitt billige nok til at de kan kjøpes til hobbybruk. Flere kjøpere har gitt billigere og bedre skrivere. De må fortsatt regnes som verktøy som krever litt kunnskap, men med ferdige modeller kan man enkelt få relativt gode resultater. Dette har gjort dem mer aktuelle for å lage prototyper.

2 Verktøy

Utgangspunktet for prosjektet var modellbeskrivelser skrevet til VTK, basert på parametere fra simulering i FlexHX. VTK er et bibliotek for visualisering av datasett, og er her brukt for å visualisere flater satt sammen til 3D-modeller. Det ble valgt å fortsette å bruke dette biblioteket. Ulempen er at det er en relativt stor avhengighet som gjør oppsett av utviklingsverktøy samt distribusjon av modellbeskrivelser og visningsprogram vanskeligere. Fordelen er at rammeverket gjør det relativt greit å definere nye modeller. Problemet med størrelsen kan behjelpes ved å bare bruke rammeverket til prosessering av geometrien, og til konvertering mellom formater. En begrenset del av biblioteket kan distribueres sammen med programmet eller kompileres inn statisk. Visualisering kan gjøres i eksisterende 3D-modelleringsprogrammer.

En viktig faktor ved valg av verktøy har vært å endre minst mulig på eksisterende systemer.

Et alternativ til VTK som ble vurdert, var å bruke et «Computer Aided Design»-verktøy (CAD) som kan gjøre parametrisk modellering. Siden målet med modellene ikke er produksjonsklare modellbeskrivelser, og det kreves en del opplæring for å bruke dem, ble denne ideen kastet. CAD egner seg godt til å beskrive dagens modeller, men siden det kan være interessant med mer komplekse geometrier, gir en løsning bygd på direkte programmering av modellen mer fleksibilitet. Muligheter her diskuteres noe videre under «7 Videre arbeid».

Selve modellene er skrevet i scriptespråket TCL. Det har blitt utviklet et bibliotek for å gjøre eksport av modeller enklere. Dette er beskrevet under «4 Generering av STL». Fordelen med TCL, og tilsvarende med andre script-språk, er at man kan ha rask iterasjon i utviklingen. Hvis man allerede har installert VTK, kreves det ikke noe annet oppsett for å kjøre modellene. Man kan få tilsvarende iterasjonstid ved utvikling i kompilerte språk, men det vil kreve større endringer enn fortsatt bruk av TCL. Ulempen med TCL, er at det nok vil bli vanskelig å finne utviklere som kan vedlikeholde koden etterhvert. Det er likevel ikke et stort problem, siden mengde kode er relativt begrenset, og i stor grad består av kall direkte til VTK. Det vil derfor kreve lite arbeid å oversette til et annet script-språk som støttes av VTK, for eksempel Python, eller til C++ for å komme nærmere resten av koden til FlexHX, som er skrevet i C. Siden grensesnittet til VTK er skrevet i C++, vil det være mindre hensiktsmessig å skrive mot det i C.

Visningsverktøyet for modeller, som beskrevet under «5 Visning av 3D-modeller», er skrevet i C++ og bruker VTK. C++ ble valgt framfor TCL siden forfatteren er relativt ny til TCL. I tillegg til VTK, er det hentet inn to avhengigheter. Begge disse er inkludert direkte i koden, og krever ikke noe ekstra kompileringssteg eller oppsett. ImGui er brukt for å lage grafiske grensesnitt. For å bruke ImGui med VTK som tegnemotor, måtte det skrives et lite lag mellom dem, med utgangspunkt i OpenGL-implementasjonen til ImGui. Hvis man ønsker å bruke VTK med andre tegnemotorer, kan dette laget generaliseres og bruke mer VTK-funksjonalitet for tegning, i bytte mot noe ytelse. «Native File Dialog» har et API for å bruke åpne/lagre-vinduer på tvers av plattformer (Windows, Linux og MacOS). Kravet til GTK for å kjøre på Linux gjør at biblioteket ikke er ideelt. Bilioteket kan fjernes helt og byttes ut med et mye enklere brukergrensesnitt for å oppgi filnavn, eller byttes ut hvis det finnes bedre alternativer.

3 3D-skrivere

3D-skrivere for plast leser vanligvis et filformat som kalles «gcode» som har litt ulike implementasjoner avhengig av skriveren. Kort beskrevet er denne koden et sett med instruksjoner som forteller hvor skriverhodet skal bevege seg, hvilke temperaturer som skal settes og når det skal sendes ut plast. For å komme fra en 3D-modell til gcode, bruker man en «slicer». Det er et program som deler opp modellen i tynne lag og finner ut hvordan skriverhodet skal bevege seg for å legge plast på de rette stedene. Når lagene skrives ut på hverandre, blir resultatet en 3D-modell.

3D-skriveren som er brukt i prosjektet er en Ultimaker 3, og den anbefalte «slicer»-en til denne skriveren heter Cura. Programmet importerer 3D-modeller i et lite utvalg formater og produserer gcode-filer som senere kan leses av skriveren. Programmet har mulighet for direkte oppkobling mot skriveren over et nettverk, eller man kan overføre filene med en USB-minnepinne. Siden programmet utvikles av produsenten av 3D-skriveren, gir standardinnstillingene gode resultater. Det hadde likevel vært nyttig å sammenligne resultatet fra andre «slicer»-e. Cura fungerer godt dersom man har en modell som kan importeres og deretter skrives ut direkte, og gir god kontroll over innstillingene til skriveren. Programmet kan sette sammen flere komponenter til en modell, men egner seg ikke like godt til mer sammensatte modeller. Hvis flere komponenter skal importeres, lønner det seg at de allerede er skalert og beskrevet relativt til et felles origo. Bedre valgmuligheter for å generere støttestrukturer vil også være ønskelig, og er tilgjengelig i andre, tilsvarende programmer, men ingen av alternativene har blitt testet i dette arbeidet.

3D-modeller bør være sammenhengende og ha lite nok overheng for å gi gode resultater. Cura forventer lukkede flater med definert innside og utside, og genererer feil gcode hvis modellen har flater som ikke er det. Ved å sette innstillingen «Surface mode» til «Surface» eller «Both», og å velge «Spiralize outer surface», kan modeller som bare består av enkeltflater skrives ut ved at skriveren følger disse flatene. Ulike kombinasjoner av disse to innstillingene gir litt ulike resultater. Overheng blir generelt bedre med lavere skrivehastighet, men uten støttestrukturer er overheng en viktig begrensning. I genererte modeller som ikke er veldig presise, hadde det vært gunstig med bedre, automatisk oppretting. En vurdering av verktøy som kan gjøre dette kan være interessant.

4 Generering av STL

Cura kan lese flere filformater som beskriver 3D-strukturer, og kan også tolke noen bildeformater som høydedata. STL («STereoLithography») er et av de vanligste formatene for å utveksle 3D-informasjon for 3D-skrivere, og VTK har støtte for å eksportere direkte til STL. Derfor ble dette filformatet valgt.

Filformatet er en beskrivelse av enkeltstående trekanter, og beholder ingen informasjon om sammenkoblingen av disse. Formatet er derfor relativt tungt å lese. Siden formatet ikke inneholder noe informasjon om sammenhengen til trekantene, bør ikke flere modeller lagres i samme fil. Dette ble løst ved å eksportere enkelt-deler i hver sin fil, og tilsvarende lese inn filene. Konvensjonen `<modellnavn>_i` ble brukt, der `i` er nummereringen til delen. Andre filformater som ble vurdert var «Wavefront OBJ» og «Polygon File Format» (PLY, også kalt Stanford Triangle

Format). Begge disse formatene er også relativt vanlige formater for utveksling av 3D-data, men er i prinsippet like og gir ingen større fordeler. Formatet kan defineres både binært og som tekst, der binært er mye mer kompakt og effektivt.

For modeller med mange av samme del, for eksempel et sett av rør, er det langt mer effektivt å definere én modell, og så et sett med punkter der kopier av delen skal plasseres. Dette kalles «glyphing» og VTK har støtte for dette ved hjelp av `vtkGlyph3D` eller `vtkGlyph3DMapper`. Avhengig av videre behandling, kan det være ønskelig å triangulere hele modellen og lagre den direkte til STL-filer, eller å ta vare på informasjonen om at samme komponent repeteres mange ganger. Mer om dette i «5 Visning av 3D-modeller».

VTK har funksjonalitet for å skrive direkte til flere filformater, hvorav STL er ett av dem. For å kreve minst mulig endringer i et TCL-fil for å lagre den beskrevne modellen til STL, ble det skrevet et bibliotek bestående av to funksjoner som kan inkluderes med **source** `stl_output.tcl`. Biblioteket brukes ved å legge til en eller flere `vtkActor`, tilsvarende måten man legger den til for tegning, og til slutt skriver man ut modellen. Biblioteket definerer funksjonene

```
proc save_actor {actor layer { glyping_enabled 1}}
proc output_stl {base_filename}
```

Et eksempel på bruk er

```
source stl_output.tcl

# Bygg opp modell og lag til actor
# ...

# Legg til for lagring, med actor1 og actor2 i lag 1, og actor3 i lag 2
save_actor actor1 1
save_actor actor2 1
save_actor actor3 2

renderer AddActor actor1
# ...

# Skriv ut til STL-filer, hvor actor1 og actor2 havner i my_model_name_1.stl,
# og actor3 i my_model_name_2.stl
output_stl my_model_name
```

`save_actor` kan også ta et tredje parameter (boolsk, 0 eller 1, med standard 0 hvis ikke angitt), som slår på glyphing slik at modellen kan importeres og bruke `vtkGlyph3D`.

Biblioteket kommer også i en C++-versjon, med samme grensesnitt og funksjonalitet. All koden er i filen `stl_output.h`, og brukes direkte ved å inkludere filen. Biblioteket har samme grensesnitt som TCL-versjonen, med en ekstra funksjon for å rydde opp allokert minne. Funksjonene er definert av

```
static void StlOutput_SaveActor(vtkActor *actor, unsigned int layer,
                                bool glyphingEnabled = 1);
static void StlOutput_OutputStl(char *baseFilename);
static void StlOutput_Destroy(void);
```

Modellene kan skaleres før lagring ved å **#define** `STLOUTPUT_SCALE <scale>` før filen importeres. STL er i utgangspunktet et enhetsløst format, men Cura tolker med standardinnstillinger størrelser som millimeter.

5 Visning av 3D-modeller

For å teste mulighetene med VTK, ble det implementert et program som bruker VTK for å visualisere 3D-modeller og behandle dem før utskrift. Resultatet er et program som kan laste

STL-filer og sette hvert enkelt importert modell til et lag. Flere modeller kan flyttes til samme lag. Disse lagene brukes for å kontrollere lagringen. Hvert lag lagres til en egen STL-fil.

Programmet har funksjonalitet for å skalere, flytte og rotere modellene. Man kan også definere utsnitt av modellen som skal tas med. Utsnittet defineres ved å plassere en boks, der de delene av modellene som er på innsiden beholdes. Ønsket utvidelse av denne funksjonaliteten er å kunne definere et utklipp per lag eller per modell. Det vil også være enkelt å definere utklippsområdet med enkeltplan.

Når modellen kuttes, blir den ikke lengre lukket. VTK har et filter som kalles `vtkClipClosedSurface`, som gitt en lukket modell og et sett med plan, kutter modellen og lukker den igjen. Filteret fungerer for de fleste modellene det ble testet med. Ytelsen er dessverre ikke veldig god på store modeller, og minnebruken er særlig et problem.

All funksjonaliteten som er utviklet, finnes allerede i andre programmer, for eksempel 3D-modelleringsprogrammer som Blender eller Maja, eller CAD-er. Programmet har fungert for å teste ulik funksjonalitet i VTK. Se diskusjonen om 7 Videre arbeid. Samtidig er det et mer spesifikt verktøy, og dermed forhåpentligvis raskere å sette seg inn i enn generelle 3D-modelleringsprogrammer der bare noe av funksjonaliteten er nødvendig.

5.1 Gjennomgang av grensesnitt

Når programmet er åpnet, kan man importere modeller med `File > Open` (`CTRL+O`). Flere modeller kan velges samtidig. Modellene blir lagt til i hvert sitt lag. I samme meny kan man også lagre, `File > Save` (`CTRL+S`), eller lagre som, `File > Save As` (`CTRL+SHIFT+S`). Her kan man også angre (`CTRL+Z`) og gjenta (`CTRL+Y`) klipping, som beskrevet under.

`Models`-menyen har en liten liste med forhåndsdefinerte modeller som kan brukes for å sammenligne skalering. De har størrelser spesifisert i millimeter, og må eventuelt skaleres deretter hvis modellene har en annen skalering.

Under `Tools`-menyen kan man åpne tre vinduer. Vinduet `Layers` viser lagene og modellene som er importert. Her kan man flytte modeller mellom lagene (`\/` og `/\`), eller fjerne modeller (`X`). Modellene får farge avhengig av hvilket lag de er på. Vinduet `Model` viser informasjonen om den valgte modellen. Modeller kan velges ved å holde over dem og trykke `P`, eller ved å trykke på navnet deres i `Layers`-vinduet. Vinduet viser navnet, og har kontroller for å endre posisjon, skalering, rotasjon, gjennomsiktighet og gjennomsiktighet for deler av modellen utenfor klippet område. Mer om klipping under. Vinduet `Misc` har diverse funksjoner som ikke passer andre steder. Knappen `Reset Clipping` tilbakestiller utklippsboksen til å inkludere alle modellene. Nyttig etter importering av flere modeller. `Dimensions`-verdiene viser den totale størrelsen av modellen, i samme enheter som skrives ut til STL. Vinduet har også mulighet for å overstyre skaleringen til hver enkelte modell og skalere alle med samme skalering.

Navigering i 3D-miljøet er basert på `vtkInteractorStyleTrackballCamera`. Ved å dra mens venstre museknapp holdes inne, roteres kameraet rundt fokuspunktet. Fokuspunktet kan flyttes ved å dra rundt med musehjulet. Når man trykker `F`, flyttes fokuspunktet til der musepekeren er. Høyre museknapp, eller scrolling med musehjulet, zoomer. `R` flytter kameraet slik at alt er synlig. `W` bytter til «Wireframe»-modus, slik at bare kanten til flater synes. `S` bytter tilbake til å vise flater. `1, 3, 7, 9` snur kameraet slik at det ser i henholdsvis X, Y, Z og motsatt retning av nåværende. Disse er mest naturlige å bruke på en NUMPAD. `5` bytter mellom perspektivvisning og ortografisk/parallell visning.

Etter å ha valgt en modell, kan man trykke `C` for å starte klipping av modellene. Utsnittet velges med en boks, der kulene på hver flate flytter flaten og kulen i midten flytter boksen. Kulene kan velges gjennom modeller hvis de er gjennomsiktige. Ved å dra på selve boksen roteres den, men det er vanskelig å få den rotasjonen man vil ha. En flate på boksen kan flyttes til en flate på modellen. Hold inne `CTRL`, og trykk på kulen til den flaten som skal flyttes. Boksen forsvinner. `CTRL`-trykk på en flate, og boksen roteres og flyttes slik at valgt flate på boksen ligger likt med valgt flate på modellen. Hvis man holder inne `CTRL+SHIFT` når flaten på modellen velges, roteres boksen til rett vinkel, og deretter flyttes bare den valgte flaten på boksen i stedet for hele boksen. Endringer i klipping kan angres med `CTRL+Z`, og gjentas med `CTRL+Y`.

5.2 Kompilering

All koden er samlet i en enkelt fil, `main.cpp` som kan kompileres og linkes direkte i ett steg, gitt riktige «includes»- og «libs»-flagg. For å følge oppsettet til andre, relaterte kodebaser (FlexHX, FlexCS, etc.), er det likevel inkludert en `Makefile`. Kompilering i windows er testet på Windows med MinGW og `g++`, og på Linux med `g++`.

Den største avhengigheten er VTK. I MinGW kan det installeres med `pacman -S mingw-w64-i686-vtk`. VTK kan også installeres ved å kompilere fra kildekode. VTK finnes i pakkearkivet til ulike Linux-distribusjoner, typisk med navnet `vtk`. I Windows med MSYS/MinGW, må variablene `VTKINCDIR` og `VTKLIBDIR` settes i `/c/progs/paths.mk` til henholdsvis VTK-mappen med header-filer og biblioteksfiler. Windows-versjonen bruker OpenGL-biblioteker fra operativsystemet samt glew fra VTK-installasjonen. Linux-versjonen krever separat installasjon av `glew` og `glu`. Linux-installasjoner krever også `gtk3`, for åpne/lukke-filvinduer. Denne avhengigheten kan relativt enkelt fjernes ved å ta inn filer som skal lastes fra kommandolinje ved start, og gi en enkel tekstboks for valg av filnavn for lagring.

6 Visualization Toolkit

En del funksjonalitet i VTK vil det være verdt å se litt videre på. `vtkBooleanOperationPolyDataFilter` skal kunne gjøre boolske operasjoner på geometriske objekter, men er ikke særlig god på komplekse objekter. Andre verktøy, både 3D-modelleringsprogrammer og CAD-er for geometri, gjør dette bedre, og kan eventuelt brukes før modellene importeres. Filteret kan være interessant både for å definere mer komplekse utklippsområder, og for å forenkle geometri der to modeller overlapper. Cura håndterer overlappende modeller til en viss grad, men for å få skarpe overganger, er det bedre med modeller som ikke overlapper. Områder med kompleks geometri i en modell kan for eksempel forenkles ved å overlappe med en boks som dekker deler av området, slik at man bare ser overflatekonturene.

Filteret `vtkClipClosedSurface` brukes for å klippe modeller og lukke kuttflaten. Filteret tar et sett med matematiske plan som den kutter langs. Det håndterer ikke glyphing, så før en modell klippes, må den trianguleres. Det har mye å si for ytelsen. En mulig løsning er å splitte opp modellen slik at glypher som blir påvirket av klippet flyttes til en egen modell og trianguleres, mens resten beholdes som med glyphing. Glyphing er en teknikk der et repeterende element representeres som modellen av en av enhetene samt et sett med punkter der de skal plassers.

`vtkClipPolyData` har samme funksjonalitet som `vtkClipClosedSurface`, men lukker ikke flatene igjen. Det kjører kjappere, og brukes derfor for å tegne deler av modellen som skal klippes bort. Områder som åpnes av klippet ligger mot flaten til modellen som beholdes, og trengs derfor ikke. Filteret tar en mer generell implisitt funksjon for klipping, og er ikke begrenset til plan.

Tegning av mange like modeller kan gjøres veldig mye mer effektivt på moderne skjermkort. I VTK kan man representere mange like modeller med `vtkGlyph3D`, som representerer samme modell på flere punkter. Denne representasjonen gir ikke nødvendigvis bedre ytelse for tegning. Dersom modellene bare skal tegnes, og ikke filtreres videre, kan man sende dataen direkte til `vtkGlyph3DMapper`, som fungerer på samme måte som `vtkGlyph3D`, men som blir spesialbehandlet av VTK for tegning.

VTK har funksjonalitet for lesing og skriving av flere forskjellige filformater. STL (`vtkSTLReader` og `vtkSTLWriter`) og PLY er filformatene som støttes for skriving av 3D-formater. I tillegg støttes lesing av OBJ. For STL kan man velge mellom binært format og tekstformat. Det binære formatet er mer kompakt.

Andre filtre som er aktuelle for automatisk forbedring av modeller, eller som utgangspunkt for egne algoritmer, er `vtkCleanPolyData`, `vtkFeatureEdge` og `vtkSelectPolyData`. `vtkCleanPolyData` kan fjerne overlappende punkter, men ikke flater. `vtkFeatureEdge` kan finne blant annet ikke-manifolde kanter, og er veldig aktuelt for automatisk forbedring av modeller. Deretter kan kanskje `vtkSelectPolyData` velge flatene som er definert av de ikke-manifolde linjene, men første runde med testing har problemer med litt mer komplekse flater.

7 Videre arbeid

For å få mer ut av dette prosjektet, er sannsynligvis neste steg å ser mer på genereringen av modeller. Abstraksjonslaget definert i FlexHX, sammen med noen generiske objekter, bør kunne brukes til å generere geometrier mer direkte enn å bare skrive ut størrelsesparametere. Et godt utgangspunkt kan være å gå gjennom eksisterende modeller i FlexHX (`flexhx/models/`) og FlexCS (`flexcs/models`) for å finne felles komponenter, og se på hvordan de settes sammen. Deretter kan man se på generering av arbitrære polygonflater («3D mesh») basert på matematiske flatefunksjoner eller fraktalgeometrier.

I tillegg til forslag nevnt tidligere, følger her en punktliste med forbedringer til STL-viseren, uten noen bestemt rekkefølge

- Forenkle veldig detaljerte deler av modeller ved å legge en boks over dem, og eventuelt la konturer stikke ut. Generert fra `vtkCubeSource`
- Krasj i noen tilfeller ved sletting av modell
- Fjern valg (trykker `P` over en av dem) av usynlig modell, bør velge gjennom. Gjelder også for flytting av plan med `CTRL`-trykking.
- Omdefinering av origo, slik at alle figurer kan eksporteres med felles origo. Gjør det lettere å rotere/skalere i programmer som Cura
- Test boolean-filter med intersect, der det defineres en kube.
- Translasjonshastighet relativt til størrelse på modellen
- Import/eksport-enheter (les inn-data som meter, og skriv ut som millimeter etc.)
- Vis størrelse med enheter i modell (faktisk og skalert)
- Isolert test av glyph
- Eget valg av klipping for hvert lag (hvert lag har allerede individuell klipping, men settes basert på de samme planene)
- Mulighet for lagring av tynne plater til enkeltlag for «Surface mode»-utskrift
- Advarsel om for tynne områder
- Copy-paste tekst i ImGui
- Last på nytt fra fil
- Automatisk bygging av modell-menyen, les tilgjengelige filnavn
- Hvis kutteområdet krympes, ta utgangspunkt i allerede kuttet modell i stedet for å kutte fra full modell
- Prosjekt-format for lagring (lagre lag, transformasjoner, kutt)?
- Auto-repair med slic3r eller andre verktøy, hvis det er lettere enn selvdefinert forbedring av modell? (`exec "c:/Program Files/Slic3r/slic3r-console.exe" --repair $filename`)

8 Konklusjon

Sommerprosjektet har kartlagt mulighetene til bruk av 3D-skrivere for å visualisere varmevekslere. Nytten av prosjektet er sannsynligvis større for mer komplekse modeller. Gitt gode nok geometribeskrivelser er det relativt enkelt å konvertere til kommandoer som kan kjøres av en skriver. Begrensninger ved utskriften er oppløsning på detaljer og overheng (fritthengende områder) i modellen. Nå som kravene til geometribeskrivelsene i større grad er kjent, vil et naturlig videre arbeid involvere å forenkle generering av disse.