Speed Optimization of 3D Interlocking Puzzle Generation

Dominic Ngoetjana
Department of Computer Science
University of Cape Town
ngwkga001@myuct.ac.za

Nkosi Gumede
Department of Computer Science
University of Cape Town
gdmnko003@myuct.ac.za

ABSTRACT

In this paper, we present a proposal for our Computer Science Honours project, which is based on implementing algorithms to generate 3-Dimensional (3D) printable recursive interlocking puzzles—these being puzzles that use 3 dimensions to ensure that its component pieces are interlocked. Recursive interlocking is a property of interlocking puzzle pieces that enables them to form a puzzle such that the last piece inserted into the puzzle is always the first one that can be taken out and ensures that all other puzzle pieces are immobilized. While an implementation of this already exists, the algorithm that creates pieces is slow; so, we propose a solution to this. Our plan is to implement a program that does the following: 1) takes a triangulated object mesh as input. 2) Voxelizes the input object (converts into a voxel representation). 3) Generates voxelized interlocking puzzle pieces using an algorithm. 4) Triangulates the puzzle pieces using the software. 5) Add a triangulated outer surface. 6) Creates a file suitable to send to a 3D printer for 3D printing. This process, particularly step 3, will be done faster than the original implementation by adjusting it for parallelization. In this proposal, we distinguish between the respective paths of both team members (Dominic Ngoetjana and Nkosi Gumede) in creating the final system.

CCS Concepts

- **Computational geometry →3D printing**
- **Computing methodologies →Massively parallel and high-performance simulations**

Keywords

Computational geometry, 3D printing, interlocking puzzles

Figure 1: 40-piece home-made LEGO BUNNY (source: [3])

"3D puzzles are generally nontrivial geometric problems that challenge our ingenuity. They have been longstanding stimulating recreational artefacts, where the task is to put together the puzzle pieces to form a meaningful 3D shape" [3]. For our purpose, this is expanded to include interlocking puzzles - these being puzzles where assembled pieces are locked together and rendered immobilized, with one piece (known as a key) that is mobile, and which directly locks all adjacent pieces. This process is taken even further by making it recursive i.e., pieces must be assembled and disassembled in a deterministic order, with each subsequent piece unlocked (mobilized) by the key, and the remaining assembly of pieces can remain in an interlocking state (an example of this is shown in Figure 1). The problem with this approach is that it requires a blocky outer surface aligned on a regular grid and can be very slow (up to 10 hours). We propose a solution to this problem through optimizing the approach by speeding up the computation using multiple CPUs in a cluster.

1. PROJECT DESCRIPTION

The importance of our problem is found in the end-goal. We want to be able to (as fast as possible) 3D-print various types of interlocking puzzles and provide new, engaging, and challenging ways to solve such puzzles. We also want to contribute to the advancement of this field as it is low in the number of published resources over the years since its prevalence. Our solution will provide the means for faster mass production of interlocking puzzles, reducing the waiting time for computation. With that said, it is by no means an easy task and we anticipate some difficulties in the project - repurposing [3]'s algorithm for multiple CPUs will be a challenging task as we will have to determine the most efficient and data race-free techniques to use. In addition, there may be difficulties with integrating the various software systems used throughout the different phases of the project, as well as ensuring file type compatibility.

2. PROBLEM STATEMENT

As observed from our respective literature reviews, there are many unsolved problems with respect to creating 3D interlocking puzzles. Some of the most popular approaches are outdated, too slow or very difficult to implement. We aim to implement a selected algorithm to produce 3D printable interlocking puzzle pieces. A key requirement of interlocking is the assembly and dis-assembly of puzzles - puzzle pieces should immobilize each other.

Our main research question asks if an existing interlocking puzzle approach can be applied to a triangular mesh input object to produce 3D printable interlocking puzzle pieces faster than ever before using high-performance computing techniques such as parallelism. The main algorithm is by Song et al. [3]. It is described in section 5 of their paper titled 'Recursive Interlocking Puzzles'. This paper was published in 2012. In 2015, Peng Song and his colleagues went deeper into the topic of interlocking puzzles by addressing the requirement of the puzzle pieces being 3D printable in the paper entitled 'Printing 3D objects with interlocking parts' [4].

For the purpose of this project, our supervisor is the client. We are expected to produce 3D printable interlocking puzzle pieces given a triangle mesh object as input. The key requirement is that we speed up the implementation of the algorithm relative to what currently exists using efficient programming languages and algorithms. We will be using a collection of third-party software to help in various stages of the project and integrate them into the system. We aim to ensure that even the slowest of algorithms finish executing in a reasonable amount of time for the purpose of performance comparison.

3. PROCEDURES AND METHODS

Design features: Efficient programming language such as C++, C# and Java

Development platform: Qt, Visual Studio, Netbeans IDEs (on Ubuntu Linux)

Implementation strategy: Code the implementation of Song et al. [3] algorithm and use it to generate puzzle pieces when a voxelized object has been served as input. This code will be added to the sequence of a chain of algorithms used to fulfil the overall objective.

Expected challenges: Time, Complexity of algorithm, 3D printing, Not achieving speedup, Concurrency in multithreaded implementations, Testing, Debugging

Testing and Evaluation Plan

We plan to employ testing at every given stage of the system development phase. Every new implementation will be tested independently before being added to the full working system. We will test to check that the system works as expected. Testing will become increasingly valuable as we add components to the system. Testing should make it easier to debug the system and its use cases by process of elimination. We will evaluate our system by comparing its performance (time in seconds) to the existing algorithm and all other known and freely available implementations of generating recursive interlocking puzzles. We will ensure that speedup is accomplished by running our integrated system using a varying number of cores and selecting the most optimal number of cores.

4. ETHICAL, LEGAL AND PROFESSIONAL ISSUES

We will be testing all the components of our system on computers. Since this project does not require any third

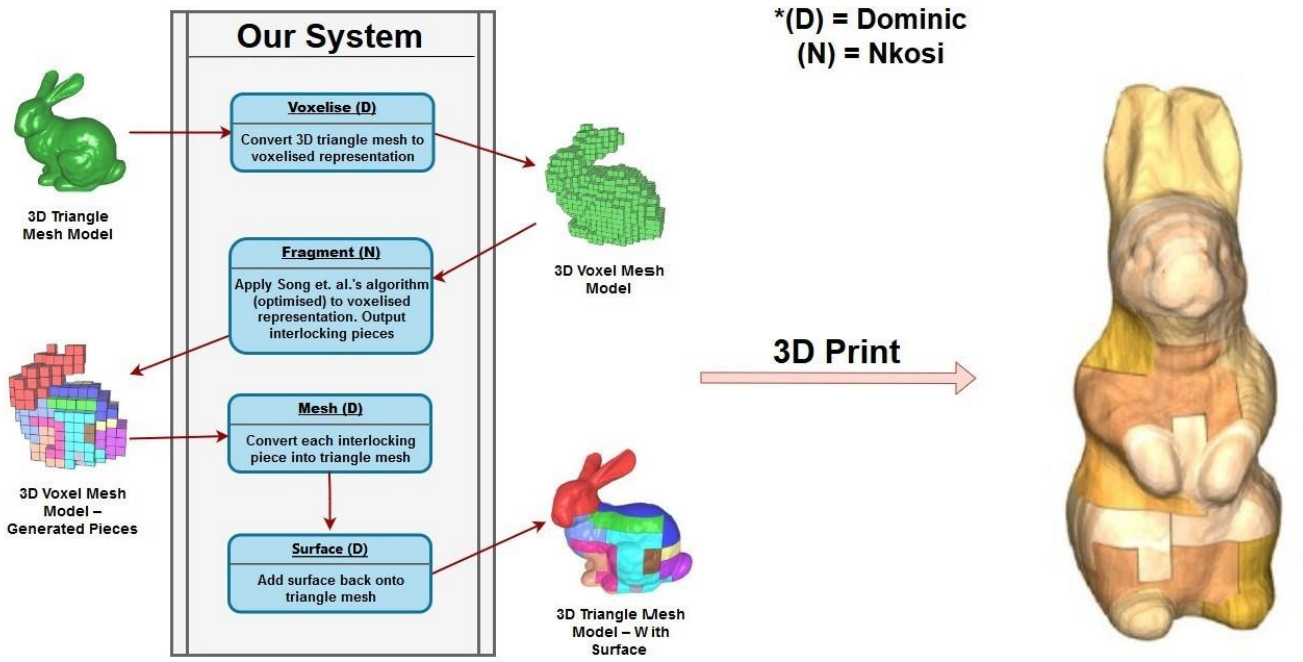

Figure 2: mesh data workflow (sources: [3,4])

parties or user experiments; there are no ethical, legal nor professional issues to worry about concerning people who are not involved in the project itself. Ethics, legal and professional issues will apply within our working relationship with our supervisor and the computer science department.

The Association for Computing Machinery (ACM) [1] asserts that general ethical principles include avoiding causing harm, honesty, and the consideration of stakeholders. As an ethical requirement, we will not plagiarize the work of others and cite our sources. We will report our findings and limitations, ensuring that they have a high degree of validity. As a legal requirement, we will not participate in any criminal activity. We will not allow ourselves to be unjustifiably enriched, harm others or causally misrepresent our system. We will only use the resources we are permitted to use. As a professional requirement, we will ensure to conduct ourselves in a professional manner throughout the duration of this project. We will arrive at meetings on time, communicate professionally, conform to deadlines, and maintain a high standard of work within our area of competence.

5. RELATED WORKS

Our literature reviews focused explicitly on algorithms to generate 3D interlocking puzzles. We split the algorithms by approach. The exhaustive search approach describes the naive method of generating puzzle pieces. This approach was commonplace in the early days of puzzle generation and inspired the newer approaches. Bill Cutler's research paper on six-piece burr puzzles [2] shows that it is possible to find interlocking puzzle pieces by exhaustively searching for them.

Song et al. [3] and Song et al. [4] discuss algorithms to create 3D recursive interlocking puzzles via the construction approach. The construction approach refers to a method of generating interlocking puzzle pieces recursively from a source object. This can be achieved by dividing the source object into its constituent puzzle pieces (top-down approach) or by creating new puzzle pieces from the source object (bottom-up approach). Interlocking does not necessarily have to be produced with voxelized puzzle pieces. This is demonstrated by Lau et al. [5] who converted furniture models into parts and connectors using lexical and structural analysis. Wang et al. [7] presented a general framework for

designing interlocking structures which use direction blocking graphs and analysis tools.

Lo et al. [6] used the geometric design approach to generate 3D polyomino puzzles. They created shell-based 3D puzzles with polyominoes as the component shape of the puzzle pieces. Xin et al. [8] also explored the governing mechanics of interlocking puzzles using geometric methods. They replicated and connected pre-defined six-piece burr structures to create larger interlocking puzzles from 3D models. Zhang and Balkcom [9] explore a solution to assemble voxelized interlocking structures using joints (pairs of male-female connectors on adjacent voxels) to guarantee interlocking and assembly order. The geometric design approach refers to using lines and curves generated by a mathematical equation to generate puzzle pieces.

Once our system has generated the voxelized puzzle pieces, we will use [4] to add the outer surface back to the voxelized puzzle pieces before triangularization (the process of representing an object as a triangle mesh) as described in Figure 2. All the above-mentioned research papers are related to our project in various ways. They help us understand the complexities (such as time, spatial capacity and ease of implementation) involved in algorithms to generate 3D interlocking puzzles.

6. ANTICIPATED OUTCOMES

Outline. Put simply, the end-goal is for our system to take in a 3D triangle mesh (one compatible with [3]'s algorithm, generate and output a similar mesh, but one with interlocking pieces and a surface; the output mesh can then be fed into a 3D printer to manufacture a fully-realized interlocking puzzle. This process will involve not only using provided software for some of the intermediate steps but also requires implementing (and optimizing) [3]'s algorithm to make it run faster.

System. The optimization of [3]'s algorithm will be the most difficult aspect of the software implementation phase of the project. This is because [3] lays out the algorithm to work recursively, but not in parallel and not taking multicore architectures into consideration. Our final developed software must be able to reliably input any compatible 3D triangle mesh (regardless of dimensions) and fragment it into interlocking pieces. Another challenge with developing this software is determining how to parallelize the algorithm correctly, i.e., so that it speeds up the process rather than slowing it down. The final software must generate interlocking pieces faster than [3]'s. In terms of the system, the entire process of inputting and outputting a 3D triangle mesh model must be well integrated, automated, and not require the user's intervention. The challenge here will be with integrating the various software systems used (one for voxelizing the input model, triangulating it, and adding back its surface, and one for generating the interlocking puzzle) into one contiguous and autonomous system.

Impact and factors. Our ultimate goal with the project is to take an existing design of an algorithm that generates interlocking puzzles and optimize it to run faster by taking advantage of parallelizing techniques and multicore architectures; so we expect our implementation to be able to take in any 3D model (compatible with [3]'s algorithm) and generate interlocking puzzle pieces significantly faster. This difference in speed will help with mass production of interlocking puzzles and will allow for more time for geometric post-processing of the model on a 3D printer. There are few key factors to determine the success of the project—one of these is a favourable (increase) measurement of speedup from [3]'s algorithm and implementation to ours. Another factor is successfully fragmenting the input 3D mesh into interlocking pieces. Lastly, being able to 3D-print our fragmented model into a plastic object with pieces that interlock and physically fit into each other with minimal space between pieces. The project will be considered successful when the outline (above) is met.

7. PROJECT PLAN

7.1. Risks and Risk Management Strategies

The risks and risk management strategies for the project can be found in Appendix 1.

7.2. Timeline

The project runs from 11 March 2019 to 7 October 2019. The timeline can be found in Appendix 2 (Gantt Chart).

7.3. Required Resources

The most important of our required resources (particularly in terms of equipment) is a multicore CPU, access to a High-Performance cluster, and a 3D printer with materials—all of which will be provided/granted by the project supervisor. On the software side, we require specialized software for manipulating triangle meshes (also provided by supervisor), a Linux distribution environment (personal computer), and C++/Java IDEs for modifying the framework and developing the puzzle creation algorithm respectively (personal computers). 3D models can easily be sourced from the internet for testing purposes.

7.4. Deliverables

The main deliverable for the project is the fully integrated system that inputs a 3D mesh, outputs it in pieces, and can be fed into a 3D printer, as described in previous sections. Other deliverables, in no particular order, include:

- Literature review
- Project proposal
- Proposal presentation
- Software feasibility demonstration
- Project paper (draft and final)
- Project demonstration
- Poster
- Project website
- Reflection paper
- Fragmentation algorithm implementation, including prototypes

7.5. Milestones

We have several milestones throughout the project—with our biggest being the successful configuration of our framework to handle puzzle pieces/parts, successful implementation of the [3] algorithm and subsequently adjusting it for parallelization, and the integration of our respective sections into one system.

7.6. Work Allocation

Work is split as indicated in Figure 2. Dominic (D) will be converting the 3D triangle mesh into a voxelized representation, converting each generated interlocking piece back into a triangle mesh, and adding the surface back on. Nkosi (N) will be reproducing Song et. al.'s original algorithm, then once successful, adjusting the algorithm for parallelization on a high-performance cluster, and finally applying the algorithm to the voxelized representation. The voxelized representation is used to output a suitable file to send to a 3D printer. The culmination of both works is a 3D-printable and interlocking puzzle (as shown on the right of Figure 2).

8. REFERENCES

[1] ASSOCIATION FOR COMPUTING MACHINERY. ACM Code of Ethics and Professional Conduct. Available online: https://www.acm.org/code-of-ethics [Accessed 10 May 2019]

[2] CUTLER, B. 2007. A Computer Analysis of All 6-Piece Burrs. Available online: http://billcutlerpuzzles.com/docs/CA6PB/index.html [Accessed 23 April 2019]

[3] SONG, P., FU, C., AND COHEN-OR, D. 2012. Recursive Interlocking Puzzles. ACM Transactions on Graphics, Vol. 31, No. Article 128.

[4] SONG, P., FU, Z., LIU, L., AND FU, C. 2015. Printing 3D object with interlocking parts. Computer Aided Geometric Design, Vol. 35 Issue C, 137-148.

[5] LAU, M., OHGAWARA, A., MITANI, J., AND IGARASHI, T. 2011. Converting 3d furniture models to fabricatable parts and connectors. ACM Tran. on Graphics (SIGGRAPH) 30, 4. Article 85.

[6] LO, K.-Y., FU, C.-W., AND LI, H. 2009. 3D Polyomino puzzle. ACM Tran. on Graphics (SIGGRAPH Asia) 28, 5. Article 157.

[7] WANG, Z., SONG, P., AND PAULY, M. 2018. DESIA: a general framework for designing interlocking assemblies. ACM Transactions on Graphics, Vol. 37 Issue 6, No. 191.

[8] XIN, S.-Q., LAI, C.-F., FU, C.-W., WONG, T.-T., HE, Y., AND COHEN-OR, D. 2011. Making burr puzzles from 3D models. ACM Tran. on Graphics (SIGGRAPH) 30, 4. Article 97.

[9] ZHANG, Y., AND BALKCOM, D. 2016. Interlocking structure assembly with voxels. 2173-2180. 10.1109/IROS.2016.7759341.

APPENDIX

1. RISKS AND RISK MANAGEMENT

	RISK	LIKELIHOOD	IMPACT	MITIGATIONS / WARNINGS / REMEDIES
1	No/revoked access to university high-performance cluster	LOW	HIGH	1. Book access to resources in advance 2. Comply with cluster rules and requirements
2	Member/s unable to finish sections on time	LOW	MEDIUM	1. Ensure there's a clear separation between allocated project sections 2. Ensure each part can be evaluated a stand-alone project
3	Unable to code [3]'s algorithm	LOW	HIGH	1. Ensure complete understanding of the algorithm 2. Communicate difficulties with supervisor
4	Unable to adjust [3]'s algorithm for parallelization	MEDIUM	LOW	Use a serial (unadjusted) algorithm
5	Development takes longer than expected due to inexperience or other reasons	MEDIUM	MEDIUM	1. Start all tasks (including learning skills and tools) early 2. Communicate skill shortages with supervisor and make further provisions for gaining them
6	Unable to integrate members' sections into contiguous system	LOW	HIGH	Ensure project outputs are clear, including file and object formats

2. GANTT CHART