PUZLOK: Final Meeting Minutes

Meeting: Puzlock Meeting #2
Date: Friday, 12/4/2019, 15h00-15h40
In attendance: Prof. James Gain (JG); Dominic Ngoetjana (DN); Nkosi Gumede (NG)

Feedback on lit review drafts:
DN: 2d and 3d interlocking objects. Title too broad -> interlocking objects/ structures -> metrics are more than just comparison
NG: Speed. Grouping of sections by algortihmic approaches. Table sections: Computation. Ease of use. Outer surface. perfomance: computation; shape generation; burr puzzles

Abstract from the main features of the algorithm. Use images. Figure caption with source. 8 pages max excl. references.

Implementation of project:
NG output: voxel grid/ representation
DN output: printable triangle mesh from voxel grid

Next meeting:
Next week Thursday @ 2pm. Going over the algorithm. Prioretize Song et al. Algorithm.

Responsibilities for next week:
JG: Monday (share software) | IDE: Cmake. Qt creator (cross-platform)
JG: Email sea-monster to discuss the internship during vac (10 june – 14 july)
DN: 15-20 relevant papers (about 2 papers per day)
NG: 15-20 relevant papers (about 2 papers per day)

JG leaves 1st of May to 31st of July 2019.

Meeting: Puzlock Meeting #3
Date: Thursday, 18/4/2019, 14h00-14h45
In attendance: Prof. James Gain (JG); Dominic Ngoetjana (DN); Nkosi Gumede (NG)

Feedback on drafts:
Possible categories: exhaustive search, burr puzzles, layered approach, joint approach?, outer surface approaches?, recursive vs non-recursive?, 2D?, furniture?
Table categories: Nature of algorithm, applicability, big-O

Minimum 15 papers

NG implementation phase: Puzzle algorithms producing grid
DN implementation phase: Voxelization of grids (grids blocks on/ off)

Skype presentation: Week of 13th of May
Draft of the proposal: Week of 13th of May

Communicate via E-mail: until JG is back

Next meeting: Via Skype Wednesday the 15th May, 5pm

JG leaves 1st of May to 31st of July 2019.

Meeting: Puzlock Meeting #4
Date: Tuesday, 11/6/2019, 09h00-09h30
In attendance: Prof. James Gain (JG); Dominic Ngoetjana (DN); Nkosi Gumede (NG)

Meeting Discussion Points:

1. Work split
    a. Clear definition
    b. Highly dependent
2. Roles
    a. Existing tools/packages
    b. Pipeline
        i. One person
        ii. Suggestion: compare implementations
3. Workload
    a. Focus: algorithm development
        i. Code sample from authors. Has it been done before?
        ii. How do we visualize the fragmentation?
4. Implementation language
    a. C++ vs. Java | comfortable language. C# via Unity for visualization?
    b. Problems of integrating different languages
5. Challenges
    a. What happens if we can't reproduce the algorithm?
    b. What's the minimum amount of completed work that can be accomplished?
    c. What are we optimizing? Evaluation plan for output?
    d. Parallelization and cluster usage must be investigated

Key Points Addressed:

DN: Acknowledge work flow as a risk. This should not be a problem for Nkosi as cubes can be gathered online.
DN: Share James' renderer with Nkosi for the purpose of visualizing the fragmentation efficiently

- Much too much work for one person to do the entire project
- We can collaborate on the implementation of the 2012 algorithm as it poses a major risk
- The algorithm is not too complex (I.e. medium complexity)

NG: Get James' renderer (and other relevant resources) from Dominic. C# can be used for visualization
DN: can rewrite James' renderer (C++) in C# (if necessary, indicating a slight change in scope)
NG: Project could rather be in written entirely in C# (with Unity) to save development time
NG: Investigate C# parallelization environment (and support in openMP)

- Standalone processes in the pipeline (independent) can be written in different languages
- Parallelization on multicore architectures is key to evaluation. Compare to original (sequential) algorithm.

NG: Focus only on the 2012 paper
DN: Can focus on the 2015 paper with regards to adding outer surface

- Improved access to HPC cluster is required. We should speak to Michelle about this. Start with openMP (instead of MPI) on single machines. Single node of *hex* (I.e. cluster).
- Sea Monster internship begins on Monday 24th of June. Speak to them about the possibility of doing evening Skype meetings from their offices.
- Revised proposal due Sunday 30th of June (if necessary). Changes in scope and timeline should be accounted for as per James' feedback.

JG: Feedback on proposal by the end of the week. Necessity of revised proposal?

Next Skype meeting: Tuesday 25th of June, 5pm

Meeting: Puzlock Meeting #5

Date: Tuesday, 25/6/2019, 17h00-18h00

In attendance: Prof. James Gain (JG); Dominic Ngoetjana (DN); Nkosi Gumede (NG)

Meeting discussion points

- How should Nkosi process the .stl model? (Unity vs adding code to renderer)
- Data structure to use to represent voxelized input
- Use Unity for implementation of Song et al. (or is it not necessary)?
- File format compatibility (.stl vs .obj in Unity) --> to be included in overall Puzlok system
- Algorithm: generate and output each piece as a separate STL file; framework: input each piece and triangularize
- Does the framework take in voxel meshes? Does it output voxel meshes?
- Framework not outputting viewable STL files correctly online (correct?)
- Output of voxelization is octahedrons (not cubes)
- Very slow voxelization speed (bunny took > 1 hour)
- Parallelization strategy?
- What is j supposed to be initially in computing the accessibility value?
- Proposal & literature review (Dominic) feedback

Key Points Addressed:

NG: Integrated rendered in needed for debugging. Try to find renderer on the web (C#).
NG: Data structure to use is a 3D grid with 1s and 0s (see JG' demonstration). Unity is not necessary for visualization. Stick to using .stl files.
DN: Dominic must write renderer with cube mesh in C++ if not already freely available online.

- Render cube and scale in right position for right position.

DN: Problem: voxelization algorithm is too slow because voxel grid too detailed --> dimensions should be lowered. Each voxel is treated as a point. Don't' use marching cubes as it treats the voxel as points.

- Each piece is a separate file converted into a triangle mesh
- Voxel to mesh converter for cuboids

JG: Test and debugging rendered software. The writeSLT method
DN: Print out what writeSLT outputs to debug
NG: J is the distance from the current voxel. I.e. J is the neighbours (see JGs' demonstration)
JG: Proposal feedback due tonight

- Useful tweaks to the proposal due in 5 days (30/6/2019)
- For initial demo: Section 1 of algorithm: removal of the first key piece and a .stl file triangle mesh of the first key piece

JG & DN: Chat about triangularization algorithm

- ♦ Note: Marching cubes algorithm cuts corners so do not use it.

JG: Write up an explanation of the mesh algorithm by the end of the week (30/6/2019)

Next meeting: Thursday 4/7/2019 @ 16h30

Meeting: Puzlock Meeting #6
Date: Thursday, 4/7/2019, 16h30-17h30

In attendance: Prof. James Gain (JG); Dominic Ngoetjana (DN); Nkosi Gumede (NG)

Meeting Discussion Points

- Converting mesh to voxel; output file not correct (previewed on Unity)
- "Load and save test on CGP framework" email
- Input and render grid file
- Section 3. Point 3.1) Ensure blocking and mobility: Which shortest path algorithm to use?
- Section 4. Point 3) What to use at Beta (−β) in the sum equation "where β is a parameter ranged from 1 to 6"?
- Section 5) Which flooding algorithm to use?

Key Points Addressed

- **DN:** Look at notes for CGP course. Model must be watertight, but some models have holes, for example the Bunny is a polygon mesh (which is not watertight). Solution ensure all models are watertight.
- **DN:** Any watertight models off a 3D printing repo. Easy to find. Google search for such stl files
- **DN:** Test for watertightness within James 'framework. Extract voxels. Run marching cubes algorithm. After voxelization it becomes a grid.
- **DN:** Voxelize a sphere and use the points of the grid as an example. Load mesh in as stl file and extract the voxels. Framework allows to visualize. Try it with the sphere.
- **DN:** Look to see how the renderer works and implements visualization in renderer as opposed to visualizing it in Unity.
- **NG:** Any shortest path algorithm
- **NG:** The equation is actually Sum of –Beta: take sum raised to the power of p. All pi sum up to 1. Multiple by 100. Generate a number 1-100. Array with 100 elements. Assign ranges with probability. This generates a random choice of puzzle piece. Sort of like the same thing is being done with the accessibiltiy values.
- **NG:** Flooding algorithm: adds neighbours. Wiki search: Flood fill algorithm.
- **NG:** By next week. Try to finish section 5.1.
- **DN:** By next week. Try to save in as a triangle mesh. Run marching cubes algorithm. Goes from grid to triangle mesh. Use James' algorithm.

Next meeting: Thursday 11th of July. 16H30.

Meeting: Puzlock Meeting #7

Date: Thursday, 11/7/2019, 16h30-17h00

In attendance: Prof. James Gain (JG); Dominic Ngoetjana (DN); Nkosi Gumede (NG)

Meeting Discussion Points

- Section 5: m = [total # of voxels(N)/# of puzzle pieces(K)]. Ambiguous. How should m be determined by default? Give user choice of m or K?
- Section 5.1.4 (Expand the key piece) point 2: Ambiguity. "... {voxels} that are resided next to the key... For each ui, we identify also the voxels directly above it". Which takes priority; the neighbouring voxels or the voxels above each voxel (ui)?
- Section 5.1.4 (Expand the key piece) point 3: Ambiguity "we sum the accessibility of each ui and voxels above it". Do these include the voxels added as a result of point 2's implementation?
- Section 5.1.4 (Expand the key piece) point 3: "where Beta is a parameter ranged from 1 to 6". How is this supposed to be implemented as per the equation? Is Beta random or sequentially ranged between 1 and 6?
- Section 5.1.5 (Confirm the key piece): point 3 is not implemented but I can test with a reasonable sized puzzle piece (of 8 voxels from a 4x4 cube). Is this the cause of the rejection of the piece? Any recovery strategies available given that the key piece was not expanded correctly?
- How should I process the mesh to have it save properly?

Key Points Addressed

- **NG:** Explore having larger puzzle pieces m. This is also a parameter of the program.
- Put each adjacent into the set and the voxels above it first I.e. before visiting the neighbours.
- Beta is a parameter of the program. Set it to 3 by default.
- **DN:** Use James' renderer to create a complete voxelized mesh. Set the voxels in the voxel grid. There should be a 1 to 1 mapping. Voxel grid should work for setting individual voxels. Try using the same voxel grid as James'.
- Solving problem of saving the generated mesh: There a lot of separate meshes now. Combine mesh method combines 2 voxel grids. Load a new mesh over the same mesh. Process: create data structure, translate loaded cube, run voxelize, extract the surface. Remove shared faces which are internal to the data structure I.e. delete shared voxel faces. Separate meshes won't auto-combine. Create your own auto-combiner.
- Meshes are separate data structures. Look at how the data structure is set up. Reposition cubes to the right position.
- Demo: During next meeting via Skype (run program). **NG:** Extract key piece from 4x4x4 cube.
- Demo: Set up a date with the 2nd reader to run respective demos.

Next meeting (and demo) Thursdsay the 18th of July @ 4h30pm

Meeting: Puzlock Meeting #8 (feasibility demonstration)

Date: Thursday, 18/7/2019, 16h30-17h20

In attendance: Prof. James Gain (JG); Dominic Ngoetjana (DN); Nkosi Gumede (NG)

Meeting Discussion Points

- DNs demonstration
- NGs demonstration

Key Points Addressed

- DN: 1 mesh representing the entire object
- DN: Still to do... 1) figure out how to remove the shared voxel faces. 2) mapping the orientation of the cubes
- Code in the mesh class does merging of vertices (comes as triangle soup, do merge after the triangles are removed/relabeled)
- DN: Demo – visualizing the entire process
- Next step: shrinking the sides of the piece. Pieces will fit too tightly therefore they will need to be shrinked after the puzzle pieces are generated. Adding the outer surface
- Accessibility of all corners should be the same
- Unit test the entire thing with test cases
- Come to Michelle with an idea on how to multithread. Ways of doing high level parallelisation. Examine the lower level algorithms
- Example: parallelise the most expensive parts of the algorithm
- NG: 2 weeks other piece. 1 week (openMP) for parallelisation
- Demo on Monday: Show output and remainder within renderer (I.e. the combination of DNs and NGs parts)

Next meeting is on the 25th of July 2019 @ 16h30

Meeting: Puzlock Meeting #9

Date: Thursday, 25/7/2019, 16h30-17h20

In attendance: Prof. James Gain (JG); Dominic Ngoetjana (DN); Nkosi Gumede (NG)

Meeting Discussion Points:

- **NG:** Problem: set of nieghbours is still always fragmented. When gathering the set of neighbours, should we include neighbouring voxels on the top-right, top-left, forward-left, … backward-bottom-right, etc.?
- **NG:** Lack of time to implement section 2. What is the key focus between implementing section 2 and beginning to find ways to speed optimize the existing code?
- **DN:** Voxelisation problem (again?)
- **DN:** Vertex merge -> first two cubes merge successfully; third cube fails to merge with the second when along the same axis (all along x-axis, for instance)
- Both: Milestone (demo-able) project date and progress [mid-August]

Key Points Addressed:

- **NG:** No diagonal connection
- Try more runs until the set of neighbours is connected
- Show the remaining volume
- Rather have a complete algorithm instead of optimizing
- Complete testing as means of debugging
- **DN:** Draw out the cube and check for validation
- Check windings
- Merging triangles failure: debug print the the vertex list and faces. Eg. When combining 2 cubes we should get 12 vertices
- Must fix the shared faces problem, deleting the duplicate triangles which overlap
- Test case: combining 2 cubes. Compare actual result to expectation. If positions match it should work – shared vertices with shared faces

Next meeting is on the 2nd of August @ 2pm.

Meeting: Puzlock Meeting #10

Date: Friday, 2/8/2019, 14h00-15h00

In attendance: Prof. James Gain (JG); Dominic Ngoetjana (DN); Nkosi Gumede (NG)

- **NG:** change recursive-intensive methods to loops to avoid stack overflow and heap space errors
- Finish debugging accessibility values and key piece correctness by Monday
- Continue with 4x4x4 instead of trying new input sizes. Finish the implementation of the entire paper by next Friday
- Report on whether key piece is blocked is blocked in every direction
- Report on correctness and speed in the final paper. Consider Song et. Al (2012) as an example
- **DN:** Debug the shared faces issue
- Do not worry about adding the outer surface as time does not permit
- Continue working on shrinking the puzzle pieces so that they fit into each other
- **JG:** Show NG and DN the 3D printer at the next meeting

Take note of the upcoming deadline dates as per the image on the board...

- **9/8:** Final code due
- **12/8:** Final paper structure due
- **16/8:** Draft of final paper due (submit on Vula)
- **19/9:** Feedback on draft returned by **JG** with comments
- **26/8:** Final paper due
- **2/9:** Code submission due (submit on Vula. Fix errors --> final fully functional version working)
- **NG and DN:** DO NOT HAND IN ANYTHING LATE!

Next meeting is on Thursday the 8th of August @ 2pm

Meeting: Puzlock Meeting #11

Date: Thursday, 8/8/2019, 14h00-15h00

In attendance: Prof. James Gain (JG); Dominic Ngoetjana (DN); Nkosi Gumede (NG)

- **DN:** New approach: average the normals of adjacent faces to find the direction (inverse: multiply by -1)
- If vector is not aligned, snap the vector to the grid @ 45 degree angle
- Note the notepad for pseudocode and explanation
- Create good test cases for the algorithm
- First test case is a single cube which can then be expanded
- Algorithm is O(n^2), correct but inefficient
- Map vertices to triangle list! Note top-right of notepad for explanation
- Build your own key piece and fit without jamming
- **NG and DN:** Finish code by Monday along with the paper structure
- **NG:** next week – writing final paper and finishing code (split over half of the day respectively)
- Paper structure: abstract; intro; previous work; body (design, implementation, etc); results (**NG** – speed, scope. **DN** – correctness, test cases, shrinkage); conclusion

Next meeting is on Tuesday the 13th of August @ 12:30pm

Meeting: Puzlock Meeting #12

Date: Tuesday, 13/8/2019, 12h30-13h20

In attendance: Prof. James Gain (JG); Dominic Ngoetjana (DN); Nkosi Gumede (NG)

- **DN and NG:** drafts and code due on Friday the 16th of August
- Drafts feedback returned on Monday the 19th of August
- Work on final paper the following week, due 26th of August
- Code deadline: 2nd of September
- **NG:** work on implementing final part of section 2 from tomorrow, testing and debugging the entire algorithm
- Write out test cases (not necessarily implement), compare expectations to actual code
- **DN:** shading problem; too many vertices
- Print out vertices to assist with debugging
- Length of hypotenuse is the square root of 2 for vectors of length 1 and breadth 1
- Snap to 0 for 0.005, +0D method has changed to ~0.01
- Implement normalization after the snapping. More code to before the previous method
- **DN:** paper structure – too general. Intro. Previous work. System framework (both – clarify which sections we have contributed towards). Merging. Shrinking. Rendering. Results (both). Conclusion (both)
- Possible title: "Preparing interlocking puzzle pieces"
- **NG:** Algorithm. Implementation. Key piece extraction. Secondary piece extraction. Limitations (possible subsection)
- Possible title: "Deriving interlocking puzzle piece"
- 12-15 references in previous work
- Each section the quarter of a page, nothing less
- **NG and DN:** Complete marking allocation- assign allocation marks by Friday as part of the draft

Next meeting (demo) is on Monday the 19th of August @ 15:00