La compétition canadienne d'ingénierie
À contre-courant - University of Manitoba

Canadian Engineering Competition
Against the Grain - University of Manitoba

Programming/Programmation

Aislinn Livingston

Competition Details

1. Programming teams will be comprised of a maximum of 4 competitors.
2. The entire design team must be representing an accredited undergraduate engineering program at an active CFES-member school. Students who have completed their undergraduate degree are only eligible to compete as long as their graduation was no greater than six (6) months before the commencement of CEC.
3. The presentation and any presentation materials can be done in either English or French, but must be consistent in language.
4. Teams will be given 8 hours to develop their solutions, produce all required deliverables and prepare their presentations. The design time will be from 9am to 5pm.
5. All deliverables shall be submitted prior to the end of the provided time (at 5pm), this must include the code used to evaluate the team's solution and their presentation.
6. Competitors will have a maximum of 20 minutes to present their solution to the judges. After this, the judges will have a maximum of 10 minutes to ask questions.
7. The time remaining will be announced 2 hours, 1 hour, 30 minutes, 15 minutes and 5 minutes before the deadline through the CEC 2020 Slack.

Competition Background

We live in an age where 3D visualization is of incredible importance, not only to engineers but to society in general. There are virtual reality technologies, 3D model software, 3D printers and so many more technological advances. The main focus of 3D visualization is to be able to interact with technology in the same way that we interact with our world on an everyday basis.

You work for an engineering company that specializes in creating art by 3D printing images and models. Clients come to you with an image in mind and your company helps them make it a reality. The way that your product works is by individually printing 3D 'pixels', and gluing them into their proper places. Once the model is complete, it is shipped to your clients as one big piece. Unfortunately, the glue that was used for your last shipment of models was not strong enough and has left several clients with scrambled models, with all the pixels in the wrong location!

Luckily, your boss has been able to get his hands on some drones that will be able to help apply glue back to the 3D pixels. Instead of travelling to the client headquarters and rearranging the blocks yourself, these drones will be deployed to all the clients as required. As these drones must be capable of visiting multiple clients, they must be able to unscramble multiple different models.

Your team has been tasked to write the control software for the drone. This includes the software to pickup, drop off, scan a location, etc. Your team is also responsible for creating an algorithm that will unscramble the models as efficiently as possible. Your boss will provide you with the size, unscrambled block locations and scrambled block locations for each model.

Competition Challenge

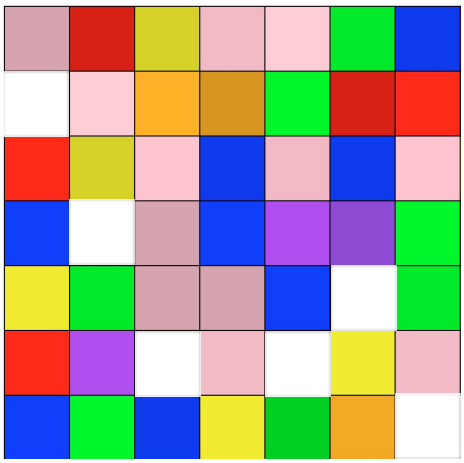
You will be given the scrambled and unscrambled model along with the size of each in a .txt file (see the **"Additional Information"** section for more info). You will be writing the software to control the drone, the algorithm that it will follow to do the unscrambling and a visualization to see it unscramble the models. Your drone can only see in the '*x*' and '*y*' dimensions. Due to the fact that this is a 3D model, the colour of some blocks may be hidden underneath others and it is your job to tell the drone to stack and unstack them as needed. In addition, you may only put a block in a location that is touching another block or touching the base of the model (i.e. you can not have blocks floating in mid-air, they must be connected to something).

The drone can only pick up a block if at least one side of it is not touching other blocks. Because of this, there may be instances where your drone has to 'dig' to get to a block. Additionally, due to the colour identification software in the drone, there is a larger time delay between picking up/dropping off a block of a different colour than the previous one your drone just touched. This should be considered when trying to

make the drone more efficient. The drone also has a storage hopper to keep the blocks in while moving around.

Size of storage hopper = $\left\lfloor \frac{\sqrt{n}}{2} \right\rfloor$ *where n = number of pixels in grid*

The amount of empty spaces upon initialization is based on the scrambled image file that is given.



Size of grid and amount of colours may vary between models but the grid will always be a cube meaning there will be an equal number of x, y and z coordinate locations.

Your time will not be tested in real-time, instead using the absolute time units provided in **"Additional Information"**.

To the left is a sample of what your drone sees, it is a top-down interpretation of the 3D scrambled image. The white blocks represent empty spaces.

Note: There may be blocks hidden below others that your drone can not see. You will need to tell your drone to move to the highest z location for a given x and y location.

Specific Objectives

Level 1A: Drone Backend

- Storage hopper for the drone.
- Functions to move in the +/- x and y directions, within this create the ability for the drone to automatically move to the highest z location of a given (x, y) location.
- Functions to pick up, drop off and scan the location that your drone is currently at (restrictions on this can be found in the **"Rules"**).

Level 1B: System Backend

- Read in and store both the scrambled and unscrambled image.
- Keep track of locations that have been checked, blocks that have been moved or any other information that might be valuable to the algorithm.
- Create a system to track the time that will keep track of the time required for each move that your drone is doing (more info in **"Additional Information"**).

Level 2: Algorithm

- Create the algorithm, using your drone's control software, to unscramble the scrambled model.
- Once the drone has read in the scrambled and unscrambled models, the program should run autonomously without user interaction.
- Output the final information after it is done running (i.e time taken, original image, the now unscrambled image and any other info you think might be useful).

Level 3: Visualization

- Create a visualization to show the image being unscrambled.

  Note: You can still earn points for providing documentation about *how* you would implement this level of the program, without actually implementing it.

Rules

- Corner blocks and edge blocks always count as having an empty space beside them.
- Moving to the highest z location of an (x,y) location does not take any time (i.e it does not count as a move).
- Your drone can pick up a block if (i) there is a block to pick up at its location, (ii)the drones storage hopper is not full and (iii) there is an empty side around it.
- Your drone can drop off a block if: your drone is in a valid location that a block could be placed in and the storage hopper is not empty.
- Blocks can only be placed in locations that are either on the z=0 plane or that are touching at least one other block.
- Even though the entire model has been loaded into your program, keep in mind that your drone can only see one location at a time.
- Some locations will have the correct blocks upon the instance creation, blocks

are never "locked in" and can be moved as needed.

- Your drone can remember locations that it has already checked.
- Your visualization should display block colours and all 6 faces of the 3D model. It should also show each intermediate step/state as the algorithm is happening and not require any additional input.
- Your finished code should take one argument as input (the path to a .txt file that will be in the format of the sample ones provided to you).
- You must include a README file in your submission, with instructions on how to run/test your code.
- You *must* include myself, the programming competition lead, in your chosen version control tool (i.e. GitHub, GitLab, BitBucket), if your team chooses to use version control. My usernames are underneath the **"Submissions"** tab. Add me to your projects.
- Using code obtained from the internet, or any outside source, is forbidden. If you are caught plagiarizing someone else's code, you will be disqualified. The use of libraries are allowed as long as they are open source. (See more info in **"Materials"**).
    - **Note**: The competition lead will be testing your code against codes found on the internet, to detect plagiarism.

Judging Rubric

Design Quality	**/70**
Solution Performance	/30
Program Design	/20
Technical innovation and creativity	/10
Code structure and efficiency	/10

Presentation	**/30**
Presentation Organization	/10
Design Justification	/5
Visual Aids	/5
Question Responses	/5

Addresses the 7 principles of ECL (descriptions can be found in Delegate package) 1. Seek Purpose 2. Take Responsibility 3. Expand Involvement 4. Widen Approaches 5. Advance Understanding 6. Realize Diversity 7. Deliberate Values	/5

Penalties	
Reported bugs/issues	-5 per bug/issue
Non-reported bugs/issues	-20 per bug/issue
Verbal disclosure of school during presentation	-10
Disclosure of school in presentation files/documents	-10
Disclosure of school by other delegates	-10
Absent Team Member	-25 per member
Plagiarism	Disqualification

Bonus	**+10**
Address competition theme: "Against the Grain"	/10

Total	**/100**

Additional Information

Time Measurements

As mentioned above, your code is responsible for tracking and reporting the time taken by your algorithm from start until finish.

Below are the time units that your team should be using for the various movements performed by the drone. By "previously", it means directly before (this can be pickup-pickup, drop off-drop off, pickup-drop off or drop off-pickup). Use of any other time unit will result in penalties.

- Scan Location: 0
- Move in +/- x and y directions: 1
- Pick up same colour as previously touched block: 2
- Pick up different colour than previously touched block: 3
- Drop off same colour as previously touched block: 2
- Drop off different colour than previously touched block: 3

Submissions

You will submit your code in a zip file via a USB. The zip file must contain:

- Each level clearly indicated in its own folder. For example, if you are Team A, your folder directory would look like Figure 1, where "Team A", is the zip folder.
    - For level 1, you must have 2 subfolders, labelled 1A and 1B.
- Your design presentation, as a powerpoint file.
- Your README file

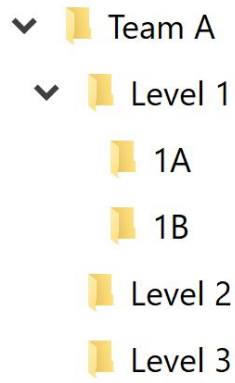


Figure 1. Submission layout

If your team has decided to use version control throughout the coding process, only submit your final solution.
As stated in the rulebook, your submission must have *no* references to your school. You must remain anonymous, and only refer to your team as the name you are designated.

Materials

A design room with at least one table, four chairs, papers and pencils/pens for writing

as well as access to internet connectivity, will be provided during the design phase. A presentation room with a digital projector, a computer containing the team's presentation file, a whiteboard or blackboard, and simultaneous translation equipment if required will be provided during the presentation phase.

Each competitor is allowed to bring any reference material, such as background research or course notes, as well as a computer and/or peripheral storage device with any electronic material stored onto it. If you are bringing electronic material on a peripheral without your own computer, please check with the competition director to make sure the format of your electronic information will be accessible using the computer provided.

It is expected that the teams participating in this competition have adequate knowledge in choosing the best tools to solve the given problem. Tools that are purchasable and do not offer free versions or trials are prohibited. There are no restrictions on coding languages that can be used.

Note: Since the use of the Internet and other external resources are permitted in this competition, all information used by competitors must be referenced very carefully.

Competitors are not permitted to submit work completed by anyone other than the members of their team. If they decide to recycle their own or someone else's code, it must be clearly cited in the presentation. In addition, the competitors also need to clearly explain why and where the recycled code was used in their software. The judges hold the right to ask any team member to describe what a particular section of the code does at any given point during the presentation. If there is any evidence that competitors are submitting plagiarized work, or that they have significantly edited their code since submission, the entire team will be eliminated from the

competition and their home schools will be notified. Volunteers will monitor each team during the design process to deter teams from cheating and to remind them to cite external resources. However, competitors are expected to act in good faith with the spirit of the competition.

Text Input Files

Your team will be given three .txt files; they will be labelled as 'easy', 'medium' and 'hard' to represent the various difficulties of the models within them, difficulty will be proportional to the size of the cube and amount of colours in the model. Each text file will contain the scrambled and unscrambled model information for each model. They will be in the form of a title, size, and coordinates with their respective block colours. Size specifies the amount of blocks for the coordinate axes (i.e. if the size is 3, that means that the entire 3D model is 3x3x3 for a total of 27 blocks). The unscrambled image will come first, after all it's coordinates have been listed there will be a blank line followed by 'scrambled_image'. The coordinates are laid out as x,y,z="R_G_B". "R_G_B"represents the red, green and blue values (between 0 and 255) of that particular colour. Locations that do not have a block in them (or do not require a block in the case of the unscrambled image) will have a ""in the place of the colour.

Here is an example of what a model might look like:

```
unscrambled_image
size=3
0,0,0="255_1_0
0,0,1=""
0,0,2="0_0_0"
0,1,0="255_0_0


. . .

scrambled_image
size=3
0,0,0="0_0_0"
0,0,1="0_0_0"
0,0,2="255_0_0"
```

0,1,0="0_0_255"