Functional Specification

Antarctic Chasm One

Client

Matt Polaine

British Antarctic Survey

Team Bravo

Julia Bibik, Andreea Deac, Matthew Else,

Nand Kishore, James Lomax, Valeria Staneva

1 Project Description

1.1 Project Background

The Halley Research Station, run by the British Antarctic Survey (BAS) [1], was designed to carry out scientific research in the field of Atmospheric Sciences, Space Weather and Glaciology. The station collects meteorological and atmospheric data every day that is used to create weather forecasts and analyze changes in space weather, ozone layer, and sea level, among others.

However, the Halley Research Station is located near a growing chasm, which could cut the station off the rest of the Brunt Ice Shelf and leave it on the resulting iceberg [2, 3]. This has lead to concerns about the future of research in the base. Predicting the rate and direction of growth of the chasm is particularly difficult, which makes the future of the research station especially troublesome.

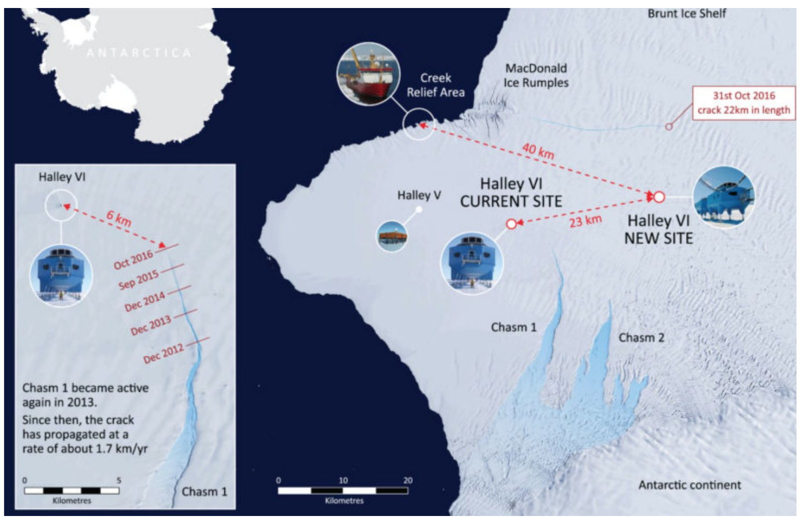


Image of Halley Research station and Brunt Ice Shelf from https://www.bas.ac.uk/project/moving-halley/

1.2 Goals

This project aims to facilitate the understanding of the growth of the chasm and how it would affect the Halley Research Station. The project's goal is to produce a three-dimensional immersive visualization of the chasm that allow viewers to descend into the fissure. This would be achieved by analyzing the data provided by the British Antarctic Survey: aerial scans, photographs, video and satellite imagery.

Using data processing methods, as well as concepts from machine learning, the project would visualise the growth of the chasm so far, and attempt to evaluate and predict its future growth. This is likely to be helpful in understanding how the chasm affects the research base.

1.3 Obstacles

There are a number of technical obstacles that this project faces.

First of all, the propagation trajectory of the chasm has not been determined. Even though there is past data that we can rely on to understand the growth of the fissure, we would still need to make significant assumptions to predict the future of the chasm. We could come up with some likely scenarios of the propagation; however, these may not be reliable.

Another challenge is the availability of footage and its usability: as Antarctica’s surface is ice-covered and mostly flat, it might be quite difficult to analyze the aerial images as light reflection of the surface could make it difficult to capture and recognize shadows [4, 5]. This would directly influence the outcomes of the visualization this project is aiming to achieve.

What is more, photography recorded via drones might not be of high quality and/or cover a large enough area. This would be due to the fact that electronics found in drones, such as batteries, might not perform well enough at low temperatures.

These are all technicalities that need to be considered when thinking about the realistic outcomes of the project.

2 Facilities

The main objective of the project is to provide a 3D visualisation of the growing crack in the Antarctic. This visualisation should aim to be realistic in appearance, although due to limitations on data not necessarily entirely accurate. The user will be able to control the camera in much the same way as the recordings from drones, allowing users to get a sense of scale of Chasm one. The software will also facilitate viewing from within the crack. Since there isn’t enough data to demonstrate the entirety of the chasm, it will be limited to viewing various parts of the chasm, selecting from an overview map.

The visualisation should also encompass a simulation of how the chasm grows over time (past and present). Users will be able to view the progression of the crevasse in the last 5 years; we will interpolate data obtained from the BAS. The program should also provide some means of extrapolating on past data, and/or predictions provided by the BAS to visualise the progression of the crack into the future.

Visualisations and simulations will be performed in as accurate a way as possible, and where not possible the simulation must be able to convey the accuracy so as not to be misinterpreted as scientific data. Where information is available on possible outcomes, we will demonstrate these outcomes.

The visualisation should be focused on demonstrating the chasm to the general public and so should be fairly easy to use. It will therefore provide a means of experiencing the progression of the chasm in a cinematic way.

To provide a more immersive experience, spatial audio could be used, generating sounds that one might hear when inside the chasm.

2.1 Possible extensions

Extending the project to different platforms will allow us to reach a wider audience. Demonstrating the simulator in a web browser would make it more accessible for the general public. The project could also be extended to smartphones/tablets and it would also expand its potential user base (but again would take a considerable amount of effort). Virtual Reality headsets could be supported to provide an immersive experience.

3 System Components

We are using the waterfall system, a development approach that implies the project is divided into sequential phases and that there will be an emphasis on planning and scheduling. This method is most appropriate for us because it is suitable for large projects that need to be implemented in a short time, by team members who may be inexperienced.

Database - Store data received from British Antarctic Survey

Future Modelling - Takes a sequence of images and uses machine learning/image processing techniques to extrapolate the future of the rift

Preprocessor - Utility functions to parse the raw data such as images/sound from files

Renderer - Transform data from preprocessor/database to 3D visualisation with OpenGL

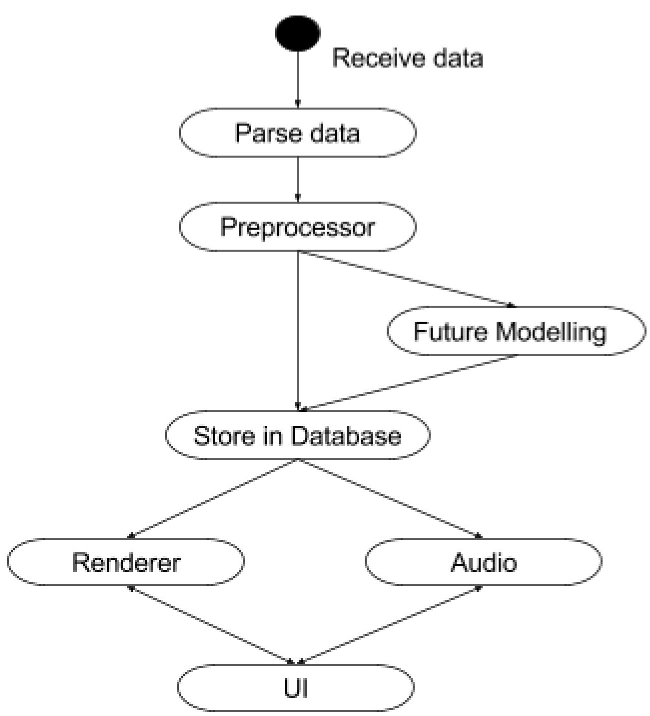
Audio - Encapsulates audio data and has methods for audio playback

UI - Allows users to control the scene.

Testing - test the components of the system and the system as a whole

Documentation

3.1 Activity Diagram



3.2 Database

Types of resources provided by the client will include aerial scans, photographs, video, multispectral satellite imagery, ground penetrating radar cross section of the crack and photogrammetric modelling of a ~300m section of the chasm. Two options to store these conveniently with a quick access are:

1. Storing the multimedia files in a database such as MongoDB.
2. Storing the files in a file system and the metadata (names, path, description, GPS coordinates) in a database.

The second option is probably more convenient.

Abstract Class (Python)

```
class Datastore:
    """Used for storing and querying data by location and time."""

    def get(location, time, radius=DEFAULT_RADIUS, type="IMAGE"):
        """Retrieves data for a certain location and time. Optional parameters include
radius from location and type of data to retrieve."""

    def put(data, location, time, type="IMAGE"):
        """Puts data at a certain location and time. Optional parameter includes type
of data to store."""
```

3.3 Future Modelling

In order to create models of the chasm that show how it will look in the future, an algorithm needs to be created that gets input images from the last 5 years and creates predictions for the next 50 years. The output would be the same format as the input (images).

Some possible techniques include calculating optical flow between pairs of images, running regressions to predict future optical flow, and running interpolation to generate the future images/models of the rift [6]. Other options include working directly in the pixel-space by employing convolutional neural networks + LSTMs. The state-of-the-art involves using Generative Adversarial Networks [7].

The component will likely depend on Python, NumPy [8], SciPy [9], Scikit-learn [10], and possibly TensorFlow [11].

Abstract class (Python)

Below is an abstract class for a basic Machine Learning module:

```
class MLModel:
    """A class used for machine learning models."""

    def train(images):
        "Takes a set of images and trains a machine learning model"

    def predict(images):
        "Takes a sequences of images and applies model to predict next image in
sequence."

    def save(filename):
        "Saves the machine learning model to filename."

    def load(filename):
        "Loads the machine learning model from filename."
```

3.4 Preprocessor

Each type of data needs to be pre-processed before it is used in building the 3D representation. Available forms of data such as aerial imagery need to be converted into a form which is useable for rendering, such as a depth maps or a list of points for the edge of the crack.

The preprocessor will likely use Python and numpy to process data.

1. Sort dataset based on GPS metadata. This information can then be delivered to Transform3D.
2. Turn videos into series of images and apply step 1.
3. Sort sound files based on temporality.
4. Sort sound files based on location.

Abstract class (Python)

```
class Preprocessor:
    "A class the preprocesses raw data into forms that are useful,"

    def read_images(filenames):
        "Reads images from different formats into numpy arrays."

    def read_video(filename):
        "Reads a video and converts it to a series of images."

    def read_location(filenames):
        "Reads location data that can be used for geo-location."

    def read_audio(filenames):
        "Reads raw audio files."
```

3.5 Renderer

Take data provided from the database and render it for the user. This will involve level of detail modelling, to produce a 3D scene while maintaining an acceptable framerate. The renderer will be responsible for performing rendering in OpenGL, including a visually appealing shading of the ice. It will also receive input information from the UI and process transformations to the scene the result (from panning, zooming etc.) A cinematic mode will be supported by the renderer, automatically panning across the ice and showing how the crack expands.

Abstract Class (C++)

```
// This class uses OpenGL to draw contents to the screen.
class Renderer {
public:
    // Draws the current scene.
    virtual void drawScene() = 0;
    // Called by OpenGL for drawing.
    virtual void display() = 0;
}
```

3.6 Audio (Optional)

After extracting the sound files from the file system or the database, an audio representation will be build. Similar to Transform3D, if data is missing, means of previous and following sounds from the ordered system can be used.

Abstract Class (C++)

```
// This class is used for controlling audio.
class Audio {
public:
    // Begins playing audio for scene.
    virtual void playAudio() = 0;
    // Stops playing audio for scene.
    virtual void stopAudio() = 0;
}
```

3.7 User Interface

3.7.1 Prototype

Image from BBC
http://www.bbc.co.uk/news/science-environment-36197657

3.7.2 Description

Processes input such as:

- mouse drags
- clicks on environment, zooming buttons and sliding bar
- arrow keys
- play button
- (optional) volume control

Mouse drags could be used in a similar way to Street View: drag left to rotate left and drag right to rotate right.

Clicks in front of you mean going forward. Clicks on zoom-in and zoom-out buttons respond accordingly.

Two options for the sliding bar:

1. Sliding bar with an automatic play button to view the progression of the rift over time.
2. By default, the user can see an overview of the map and, by clicking on an area, he can see inside the chasm as it is now. By clicking on the sliding bar, the scenery is replaced by how it would look at that time in the future and the user can look around.

Arrow keys would have the same functionalities as mouse drags and clicks: left and drag left, right as drag right and up as going forward.

The play button, when pressed, will show an overview of how the chasm will develop in the future.

A volume control is optional and would enable the user to hear inside the chasm in addition to seeing.

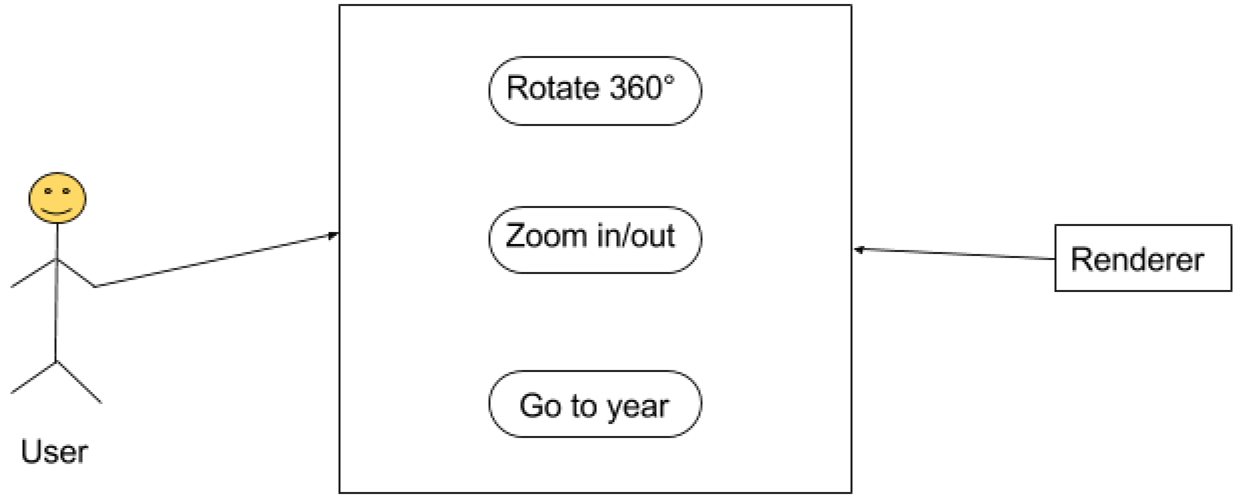
3.7.3 User characteristics

There are three groups of people who will be using the system: general public, specialists and people who maintain and improve the system.

The general public are people with basic knowledge of the field who might be using the tool for learning and entertainment.

The UI needs to be easy to use. There might be a benefit in having a legend or a short documentation on what the functionalities are.

3.7.4 Use Case Diagram



3.7.5 Abstract Class (C++)

```
// This class handles the user interface.
class UserInterface {
public:
    // Modifies the scene when arrow keys are pressed.
    virtual void keyPressed() = 0;
    // Modifies the scene when the mouse is clicked to zoom in/out.
    virtual void mouseClicked() = 0;
    // Modifies the scene when the mouse is dragged to rotate scene.
    virtual void mouseDragged() = 0;
    // Modifies the scene when the slider controlling time is moved.
    virtual void sliderDragged() = 0;
}
```

3.8 Testing

3.8.1 Unit testing

This will be done by each member individually for their respective pieces of code.

3.8.2 Integration testing

It is necessary to do this type of testing in order to make sure that the final product is working.

3.8.3 Think-aloud testing

The procedure will be to observe and record a system user performing sample tasks and to record and capture their understanding. Based on that, we will assess gulfs of execution and evaluation.

3.9 Documentation

Documentation will be done by each member of the team in the form of comments and additional Google Docs for more complex components and issues.

4 Acceptance Criteria

Database

- The database **should** be able to query data about the chasm shape around a point for the 3D visualisation within half a second.

Preprocessor

- The preprocessor **must** input the raw data, as received from the British Antarctic Survey.
- The preprocessor **must** be able to process heterogeneous input formats, including ground penetrating radar, satellite imagery, aerial scans, photography and videos.
- The preprocessor **must** output data in a format that can be used with minimal preprocessing by the 3D visualisation code.

- The preprocessor **should** run offline, providing static data that can be used by the 3D visualisation.

3D Visualisation

- The 3D visualisation **must** allow the user to see the 3D visualisation over time.
- The 3D visualisation **must** allow the user to move in 3D space within fixed scenes along the crack.
- The 3D visualisation **must** run on macOS and Linux.

- The 3D visualisation **should** display the visualisation at at least 60fps (average).
- The 3D visualisation **should** run on Windows 10.
- The system **should** never drop below 60fps (for VR in the future).
- The system **should** show predicted progression of the crack in the future.
- Any predicted progression of the crack **should** agree with scientists' predictions of future progression of the crack.

Audio

- The audio system **should** provide the user with an immersive (3D audio) experience of their surroundings.

5 Management Strategy

5.1 Intro

Group Bravo is a six-people team consisting of students with diverse expertise. Different members of the group have different strengths, in terms of both computer science skills and organization skills: we are strong in working with data, graphics processing, logistics, among others. This diversity of skills makes it an interesting task to manage the team.

5.2 Goals and Roles

Our group has been successful in recognizing the common goal of the project: to construct a three-dimensional visualization for the purposes of the BAS. What is more, we strive for transparency and clear communication. No opinion should be disregarded and any disagreements will be resolved through presenting arguments to the team. To ensure that the project goals are unambiguous and individual responsibilities are accounted for, transparent communication with the project organizers is essential.

5.3 Challenges

5.3.1 Time management

A major challenge that our team faces is related to time management. This is due to the fact that the project is quite broadly defined. There are some discrepancies in the time allocation for the project from the point of view of the project organizers and the client. According to the Group Project course page, our group should spend approximately 360 human-hours spread over 6 weeks; the expectations of the client is much higher: 600 human-hours.

The time estimation is closely related to the requirements of the project: how much work we do is tightly linked with the functionalities we would be able to provide. Thus, to resolve this challenge, we would need to communicate clearly with the client to establish what the main, most important features that we should be targeting are, and what are the stretch goals.

Related, it would be a challenge to split the work in sensible chunks. It would be very easy to get caught up and spend a lot of time on one specific feature of the project, such as the machine learning-based prediction for the growth of the chasm. This would come on the expense of other features of the project that would be in turn underdeveloped. We are going to overcome this challenge by constructing an in-depth plan with clear milestones. In case parts of the project take more time than anticipated, we would modify the plan and make sure that these modifications are well-communicated within the team as well as with the client and the project organizers.

5.3.2 Communication

Another challenge in the management of the team is communication [12]. Members of Bravo have significant responsibilities outside of class, and we live in different colleges. There are clear discrepancies between our schedules: setting up meetings that fit everyone's timetable is especially difficult. It is very important to acknowledge that, in this environment, it could be challenging to keep track of the decisions we are taking and the work we are doing.

There are a number of ways we tackle this problem. We rely significantly on face-to-face communication via the group project meetings as well as meetings in informal setting. However, we make sure to document thoroughly the meetings, and we post the notes in a shared Google Drive. We also use Google Drive for keeping track of writing documents and goals.

We rely on a variety of communication channels that are decentralized in order to improve the team creativity [13]. Here are some of the other communication/synchronization channels that we are using or planning to use:

- Facebook Messenger for day-to-day communication. We picked it as a common platform that we shared and are easily engaged with. Moreover, using a platform that is associated with friends and family could well make the team closer with each other and encourage us to work together more.
- Git for version control and GitHub to host our code and parts of our documentation.
- Reference/citation collection tool for groups such as Zotero, which is well-suited for gathering and organizing knowledge on different topics that relate to the specifics of the project.
- A documentation tool, such as a Wiki for the project, to document our code and overall project (e.g. a user guide and a simple example of how the product works).
- Good in-code documentation via clear class, method and variable names and well-formatted, well-commented concise code.
- If needed, we could switch to a team communication product such as Slack, so that it's easier to keep track of day-to-day responsibilities and search in conversations.
- If needed, we could switch our minute and note taking to Dropbox Paper. Even though it is in its Beta version right now, its group editing capabilities, including introducing diverse media into a document, and being able to switch to/from presentation mode, make it a powerful tool that can be easily incorporated into our communication flow.

6 Project Plan

6.1 Member Responsibilities

	Primary Person Responsible	**Secondary Person(s) Responsible**
Datastore	Andreea	Matt
Machine Learning	Nand	Andreea, Matt
Preprocessor	Matt, Val	Nand
Audio[1]	-	-
Renderer	James	Nand, Matt
UI	Julia	Andreea
Integration Testing[2]		Val, Julia
Documentation	Split	Split

6.2 Key dates

February 1: Specification and project plan review due

February 2: First meeting with the client

February 16: Progress report on implementation and testing

February 17: Second meeting

March 2: Group report due

[1] To be decided at a later date
[2] Unit testing will be the responsibility of each subproject leader

March 3: Third meeting

March 6: Code completion deadline

March: 8 Public presentation of results

6.3 Time division

Given we are expecting to spend 40-60 hours on the project each, so 240 people-hours in total, the time for the project will be split up roughly as follows:

Database ~25 hours

1. Store data and metadata - **by week 4**
2. Improve the structure for more efficiency - by middle of **week 4**

Machine Learning ~ 55 hours

1. Take an image and detect the two edges representing the rift - **by week 4**
2. Input a series of images and train basic machine learning model - **by week 5**
3. Apply basic machine learning model on images to predict future - **by week 6**
4. Look into improving machine learning model if time remaining - **by week 7**

Preprocessor ~ 45 hours

Audio (optional) ~20 hours

Renderer ~55 hours

1. Produce a basic static rendering of (simulated) crack data - **by week 4**
2. Non-static rendering from simulated crack data - **by week 5**
3. Visually realistic rendering rendering - **by week 6**
4. Interoperability with database - **by week 6**

UI ~25 hours

1. Provide an overlay to control the renderer scene - **by week 5**
2. Provide complete controls - **by week 6**

Integration Testing ~20 hours

Documentation ~15 hours

References

1. Halley VI Research Station. (n.d.). Retrieved January 31, 2017, from https://www.bas.ac.uk/polar-operations/sites-and-facilities/facility/halley/

2. Knapton, S. (2017, January 16). British Antarctic Survey abandons polar base as worrying crack grows in ice. Retrieved January 31, 2017, from http://www.telegraph.co.uk/science/2017/01/16/british-antarctic-survey-abandons-polar-base-worrying-crack/

3. Gibbs, P. (2016, May 04). The chasm cutting an Antarctic base adrift. Retrieved January 31, 2017, from http://www.bbc.co.uk/news/science-environment-36197657

4. Fedorov, R., Camerada, A., Fraternali, P., & Tagliasacchi, M. (2016). Estimating snow cover from publicly available images. IEEE Transactions on Multimedia, 18(6), 1187-1200.

5. Computer Laboratory Group Design Projects Wiki: British Antarctic Survey. (n.d.). Retrieved January 31, 2017, from https://wiki.cam.ac.uk/cl-design-projects/British_Antarctic_Survey

6. Walker, J., Gupta, A., & Hebert, M. (2015). Dense optical flow prediction from a static image. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2443-2451).

7. Mathieu, M., Couprie, C., & LeCun, Y. (2015). Deep multi-scale video prediction beyond mean square error. arXiv preprint arXiv:1511.05440.

8. Walt, S. V., Colbert, S. C., & Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science & Engineering,13(2), 22-30. doi:10.1109/mcse.2011.37

9. Jones E, Oliphant E, Peterson P, et al. SciPy: Open Source Scientific Tools for Python, 2001-, http://www.scipy.org/ [Online; accessed 2017-01-31].

10. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825-2830.

11. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Ghemawat, S. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

12. Pinto, M. B., & Pinto, J. K. (1990). Project team communication and cross-functional cooperation in new program development. Journal of product innovation management, 7(3), 200-212.

13. Leenders, R. T. A., Van Engelen, J. M., & Kratzer, J. (2003). Virtuality, communication, and new product team creativity: a social network perspective. Journal of Engineering and Technology Management, 20(1), 69-92.