Efficient Path Finding Using Haptic Interaction

Abstract –

Context/Background

Landscape navigation and route planning are two tasks which have been extremely important for humans as we explored our natural world. Combining these activities with new computer technologies, such as haptic devices, may speed up these processes and allow for more efficient paths to be found across landscapes.

Aims

This project aims to discover if haptic interaction is more beneficial than just purely visual feedback in the area of landscape path finding. It also aims to discover if learning speeds can be improved through this kind of interaction over just visual interaction.

Method

Digital maps will be created and displayed graphically showing the relief of the landscape and different environments, such as snow-capped mountains and forests. Users then have the ability to use a haptic device to control an avatar on the map and "feel" the landscape through the haptic feedback which the device gives to the user. Using this information the user should be able to determine efficient paths across the landscape as they will be able to understand the effort involved by traversing a particular path.

Proposed Solution

A solution will be created for Sensable's Phantom Omni haptic device, and will be written in C++ and make use of the OpenGL graphics library. Different environments on the map will provide varied kinds of haptic feedback to indicate the difficulties of passing through that environment; hills will be hard to move up and passing through forests will cause the haptic device to rumble. User tests will be performed near the end of the project to determine if path finding with haptic feedback is more beneficial than just through visual feedback.

Keywords – haptic technology, human-computer interaction, virtual environment, Phantom Omni

I. Introduction

A. Introduction to Haptics

Haptics refers to manual interactions with environments and haptic devices allow greater interaction between a user and a computer by allowing physical feedback to be given to the user as well as providing more advanced controls, particularly in 3D spaces (Srinivasan & Basdogan, 1997). As technology has advanced devices featuring haptics have entered our lives more and more in an effort to make our technological interaction more immersive. A commonly known feature such as this is the vibration function in mobile phones which is used to relay messages back to the user.

B. Project Overview

Finding an efficient path across a landscape can be a difficult task if one is not familiar with the terrain and if only a map is available. There are a few tools available, including route-finding services such as a car GPS or Google Maps, but the routes these services find are restricted to roads. As of yet, there is no suitable product to aid path finding in wild, off-road environments.

The purpose of this project is to investigate if a solution to this navigation problem can be found by using force-feedback from a haptic device, and if this interaction leads to better paths being found than by only studying a map visually. The project will discover if one method is superior at allowing a user to analyse the difficulty of a path more accurately.

The haptic device will let a user explore, navigate and manipulate a virtual 3D map which will contain physical features such as hills, forests and water. These different areas of the map will be known as *environments* and the actual shape of the land will be the *terrain*. The user will move their *avatar* across the map to "feel" the landscape. The haptic device will provide feedback according to what the user is walking on or through. For example, the device will apply resistance to movement when going up hills, and will rumble when going through forests. The purpose of this is to allow the user to gauge the effort it would take to travel through these environments in real life.

The perspective will be top-down third person, and the map will be able to be rotated and translated to allow better viewing angles. This viewing perspective is preferred over first person as it allows the program to retain some skeuomorphism as it will seem more like a regular paper map and will allow users to relate to it quicker.

In this project the conjecture that such haptic feedback would allow a user to understand the presented landscape to a higher degree than just through visual feedback is proposed. This conjecture is testable and, as such, user tests will need to be incorporated into the project.

Users will be asked to perform tasks, such as finding the quickest path between two points on a landscape in a certain time limit. This particular example can help determine the validity of the conjecture because it can be performed twice: once with haptics turned on and once with haptics turned off. The results can then be analysed and conclusions can be made about the effectiveness of path finding by using haptic feedback.

C. Stakeholders

There are a few possible direct stakeholders for this project. These would generally consist of individuals who are concerned about journeying across unknown landscapes, such as hikers. It could also be used as a learning tool for children in the classroom, to learn about geography and safe and efficient travel. Another area where this could be of use is in the video game area, where players could use a device such as the Phantom Omni or the Novint Falcon to create a more in-depth experience for the player by allowing them to interact with their virtual worlds in new ways.

However, the results of the project could have wider consequences. If the conjecture is found to be true then there are more applications that could make use of adding haptic feedback along with visual feedback. The results of this project's visual versus haptic feedback investigation could be of use to haptics researchers in general as well as haptic device manufacturers.

D. Deliverables

1) Minimum Objectives

1. Implementation of basic terrain creation
2. Implementation of basic terrain visualisation
3. Implementation of basic haptic interaction, including the ability for a user to move a marker around on the surface of the terrain and use the device as a travel mechanism

2) Intermediate Objectives

1. Support environments, such as forests, water and grassland.
2. Haptic feedback, such as shaking the device when moving through forests and needing more force applied when going up slopes.
3. Enhanced haptic interaction allowing the user to rotate and move the view of the terrain.

4. Test system with participants.

3) Advanced Objectives

1. Landscape haptic-based manipulation. Allows height to be changed and terrain features to be added or removed to create custom tests.
2. Haptic feedback whilst manipulating terrain.
3. Input real geo-science data

E. Related work

Research into haptic interaction is ongoing and new and novel ideas are constantly being created to allow humans to have a closer more "natural" feeling relationship with technology. Tactus has recently demonstrated a new touchscreen at 2013's Consumer Electronics Show (CES) where physical shapes rise out of the screen to allow for tactile button feedback (Tactus, 2013). This helps to solve the long-standing issue some people have with the lack of a tactile interface when typing on touchscreens.

In the field of Geographic Information Systems (GIS) digital representations of geographic data have been used for a many years and have many applications such as for visualising the view from a strategic military position or to better understand geological structures (Kraak & Ormeling, 1996: 104). Two standards for storing terrain shape have arisen, one based on a regular grid and one based on triangulation. While they both have significant advantages and disadvantages the grid method is preferred by such institutions as the British Ordinance Survey and the United States Geological Survey.

II. DESIGN

A. Requirements

This section details the system requirements which have been determined through analysis of the deliverables. Table 1 describes the key functional requirements while Table 2 details the non-functional considerations for the project. Reading Carroll (2002) helped to define the non-functional requirements.

Table 1. Functional Requirements

ID	Requirement	Priority
FR1	Terrain landscape can be generated and have realistic slopes.	High
FR2	The program can display suitable graphics to allow the user to understand the shape of the landscape.	High
FR3	The haptic device can connect to the program and move a marker on the landscape to signify the user's movements.	High
FR4	The haptic device can give force-feedback for the terrain's landscape slopes. I.e. You have to push harder to go up a slope.	High
FR5	The program can be set or generate tests for the user.	High
FR6	Terrain features, such as forests, bodies of water and sand, can be generated on the terrain and be displayed.	Medium
FR7	The haptic device gives the user force-feedback depending on the type of feature the user is moving through.	Medium
FR8	The terrain file should be formatted in a similar way to actual geo-science data to ease data importing later.	Low
FR9	Real geo-science data, can be imported to create actual terrain within the program.	Low
FR10	A terrain can be altered by the user using the haptic device. E.g. Changing heights on the terrain or adding/removing features.	Low

Table 2. Non-Functional Requirements

ID	Requirement
NF1	The processing power needed from the computer for the graphics and haptic device should not cause the computer to have no or very little lag when using the program.
NF2	The graphics and colours must be such that a colour-blind individual would have no issue interacting with the program.
NF3	Words and instructions for the user which are displayed by the program should be easily readable by picking font and colours suitably.
NF4	The device must not cause any harm to come to users. This is a possible issue as the device is able to give strong force feedback.

B. Hardware and Software Tools

The basic hardware for the project will be a computer and a haptic device for interaction with the user and the programming language will be C++. A laptop will be used for both the development of the application and the machine for running the user tests because it allows the flexibility of being able to work on the project in different places and it is more powerful than the standard CIS machines which make it more suitable to run graphics-intensive applications.

There were two possible options for the haptic device: Novint's Falcon and Sensable's Phantom Omni, as seen in Figure 1(a) and Figure 1(b), respectively.

The Falcon features 3 degrees of freedom and a couple buttons on a control stick. The three degrees of freedom allow the device to be moved in the x, y and z axis. The device can apply forces on all 3 of its axes via motors, which is how haptic feedback is delivered to the user. The Falcon's target demographic is the computer gaming community and there are attachments available for it, such as the pistol grip, which makes the device "the most immersive way to play video games" (Novint, 2013).

The Phantom Omni is a haptic device with 6 degrees of freedom and two buttons on the stylus. The additional degrees of freedom come from the stylus as it is capable of pitch, roll and yaw movements. It sends force feedback in a similar manner to the Falcon (Sensable, 2013).

The Phantom Omni will be used as it has more functionality with its 6 degrees of freedom, which will allow for a wider variety of actions. It also has a higher position resolution which will allow for finer selections. Although the Falcon can apply more force to the user the other specifications are more important for this project so the Phantom Omni is the preferred device.

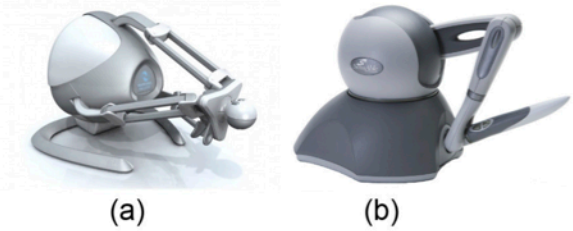

Figure 1. The two choices for the haptic device, Novint's Falcon (a) and Sensable's Phantom Omni (b). The Falcon features 3 degrees of freedom while the Phantom Omni boasts 6 degrees of freedom.

The application will be written in C++ in Visual Studio 2012. This language choice was made because the OpenHaptics SDK available for the Sensable Phantom Omni is written in C++. In addition to this, the examples provided by Sensable have multiple compiled versions and solutions made for Visual Studio. Thus, using Visual Studio will allow for faster development since there are already many resources available for it. The Visual Studio 2012

version will be used as it is the latest version and is most compatible with the Windows 8 operating system running on the laptop.

The OpenHaptics SDK is produced by Sensable for their range of haptic devices, including the Phantom Omni. It allows developers to take fine control of a haptic device and have access to all of its functions. These functions include reading the angles and positions of the different parts of the device as well as controlling the motors inside of the machine to dictate the feedback which is then perceived by the operator. The API for the SDK comes in two main flavours: Haptic Device API (HDAPI) and Haptic Library API (HLAPI).

The HDAPI allows much more low-level device control than HLAPI but at a much higher level of complexity. Amongst other things it "enables haptics programmers to render forces directly, offers control over configuring the runtime behavior of the drivers" (Sensable, 2012).

Conversely, the HLAPI provides high-level haptic rendering and large amounts of integration with OpenGL. This version of the API is still flexible in that it can output many types of haptic rendering, but instead of the programmer specifying exactly what should happen in terms of actual forces, as with the HDAPI, the programmer may use pre-made simple functions which can be applied to OpenGL shapes. It "simplifies synchronization of the haptics and graphics threads" (Sensable, 2012).

HLAPI will be used because, for this project, it is not necessary to have as much control over the device as the HDAPI allows. However, if a feature which is only available in the HDAPI both API systems are completely compatible and I can simply use that feature alongside the rest of the HLAPI features.

Due to HLAPI's strong integration with OpenGL it became the obvious choice for the graphics library. There is also a large amount of support for OpenGL in Visual Studio. Indeed, it comes pre-installed with the IDE and it requires no additional configuration. There are also large amounts of books related to OpenGL and online community support is strong which will aid the development.

C. Architecture and System Design

The software architecture of the system can be broken down into three sections, as seen in Figure 2. In this diagram each part of the system has been highlighted to show how much code reuse there will be. Green with solid border denotes parts which will be written from scratch for the project. Red with dashed border shows parts sourced from other parties which will remain untouched. Yellow with dotted border denotes code from elsewhere which has been reused and modified.

The arrows on the diagram show which parts are used by others. The arrows can be seen to progress up the diagram, from the low-level APIs to the high-level user interaction parts.

In the User Interaction Level are parts which the user will interact with directly. It consists of the user interface, graphics, and the haptic feedback which will be received through the Phantom Omni. Code fragments for haptic feedback will be used from the OpenHaptics example programs to speed the development.

The Backend Level has different parts of the program which the user cannot view directly. It consists of various functions, such as for map generation and loading, keeping track of the application and user state, and for generating the haptics which will be relayed to the user. Again, the haptics generation component will reuse example code as there is already a large amount of working demo code for haptic feedback.

The API Level shows all the 3rd party APIs which will be used. They are all in the red category by default because the source is not available for modification. The C++ Standard Library must be included for some functions such as *time* which is used for generating random terrains. The OpenHaptics SDK is shown split into its two separate API parts.

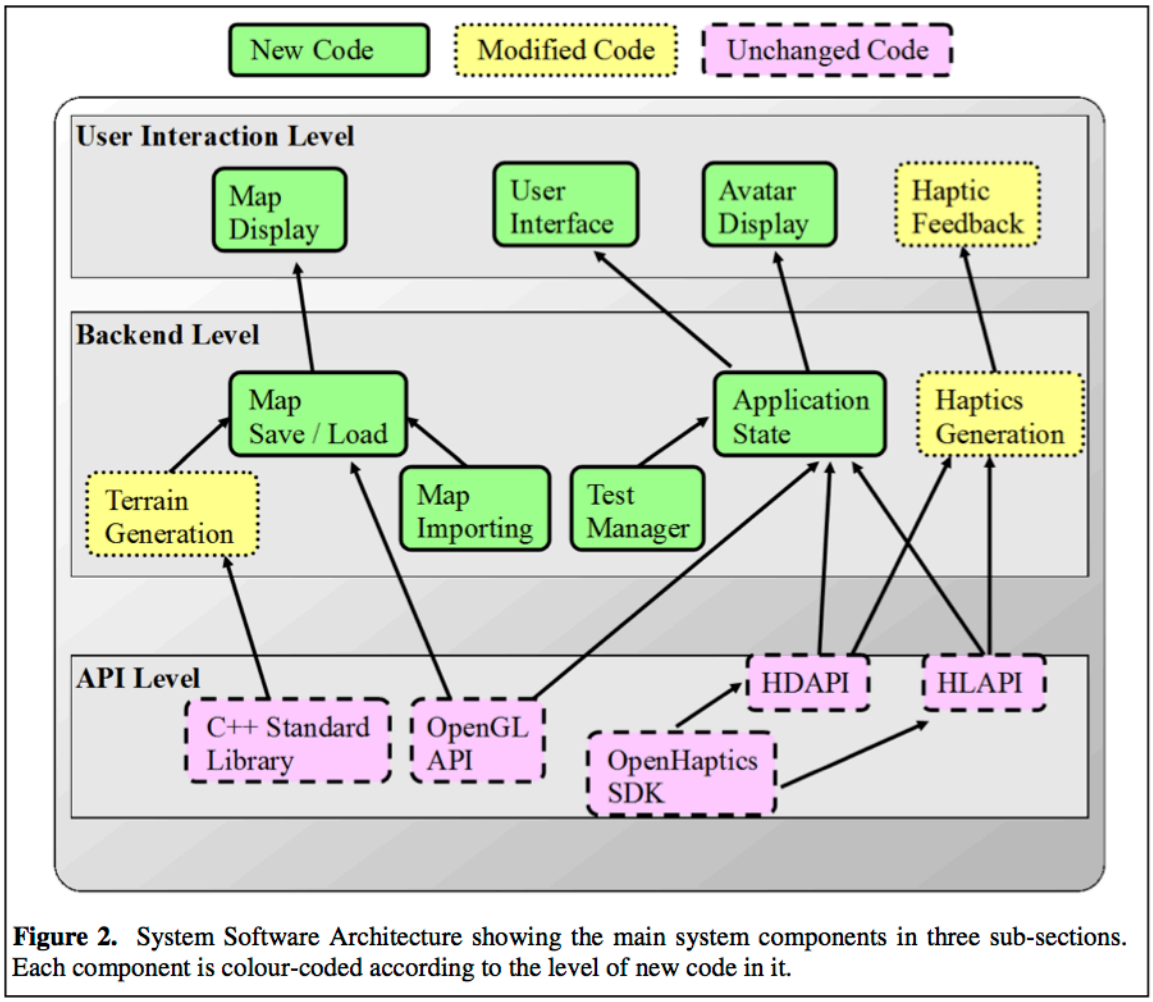

Figure 2. System Software Architecture showing the main system components in three sub-sections. Each component is colour-coded according to the level of new code in it.

D. Development Methodology

The Rapid Application Development (RAD) methodology will be used for the development of this project. RAD is based around the idea of producing prototypes in a faced-paced manner and designing the system on the go as requirements are altered and issues arise. This is particularly beneficial for a project such as this one because there are many new features and tools which cannot be learnt except through experimentation.

Although pre-design is minimal with this methodology the aims and specifications have been laid out so that there are clear targets to aim for. This leaves space for improvisation as new issues or ideas arise throughout development. The rapid prototyping also allows for user-centred development as trial users can test the software before its final release and suggest adjustments. This is especially important for this project as there are many facets which could be fine-tuned. For example, the strength and types of forces give to the user for different environments.

The application will be built modularly and in stages. The first stage is creating the map and displaying it. Basic user controls, such as rotating the map using the mouse, are also added here. This creates a framework for adding on additional features. The second stage involves applying the haptic interactions to the program. This will lead to a user being able to "feel" the landscape. The final stage is implementing the testing ability of the software. This will allow tests to be generated and results to be calculated.

Throughout the final two stages user will be consulted in pilot tests to

E. Graphics and User Interface

OpenGL will be used as the graphics library as it allows many advanced graphical features and is highly integrated with HLAPI. There will be two modes: Test Mode, which sets up and

runs tests, and Free Mode, in which a user may roam freely on the maps and have options such as to create new maps.

The user interface for the Test Mode will be fairly minimal. Since it will be used in a test situation where distractions from the task are undesirable the fewer elements of information on the screen the better. In Free Mode the main screen will just have the map shown in a 3rd person view as well as some debugging information, such as the avatar's current location and speed.

In Test Mode there will be additional boxes on the screen which will show information relevant to the test. For example there might be a countdown timer and the total length of a currently drawn path. Another box could show details about the current drawing mode or test name. A possible mock-up design for this is shown in Figure 3.

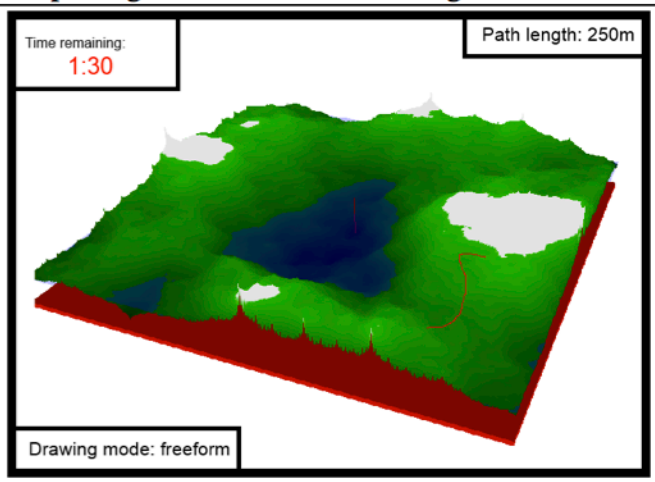

Figure 3. A possible layout for the user interface during a test in Test Mode. It displays certain information relevant to the user whilst not overloading the screen with distractions.

As the primary aim for the project is the haptic feedback the actual graphics of the map are not necessarily as important. In fact, a lower amount of polygons and complex textures will make the program run significantly faster. In Figure 3 only standard OpenGL colour shading is used, with some transparency for the water. However, if it is found that the program is able to run at a reasonable speed with some textures then they will be used.

There are several options for the graphical representation for the avatar. One option is to have a simple shape, such as a sphere act as the avatar. Another option is to have a more complex model, such as a human model. Pilot tests will be used to determine which is preferred.

F. Controls and Operation

The basic operation of the Phantom Omni is for the main arm to control the movement of the avatar on the screen. One of the two buttons on the stylus will act as a mode toggle. It will cycle between the default avatar mode and the camera control mode.

In camera control mode the map will be rotatable in the y axis and the x axis to allow easy visual exploration. The yaw and pitch of the stylus will be used to control the rotations. The map will also be able to be panned and zoomed via the Phantom Omni which will make use of the three dimensions of the arm. As MacEachren & Taylor (1994: 272) note, transforming the map is important because some map areas will disappear behind others as it is a 3D model represented on a flat screen.

In some tests the user will be required to move an avatar across the map. In this case the avatar will be "stuck" to the map. The stylus of the device will also stick to the avatar. The movement of the stylus will map directly to the height-map of the terrain, so that if the avatar is going up a hill the stylus will move upwards, away from the desk as you move it.

In tests where drawing is necessary one of the buttons will starting the drawing function. When the button is not pressed the avatar will move across the surface with the device. The avatar will "snap" to the end of the path when it moves near it to allow more drawing.

An advanced objective is for the terrain to be able to be manipulated directly by the user. This function will be available in Free Mode and pressing one of the stylus buttons will allow access to terraforming functions such as dragging on terrain to change the shape or "painting" new environments onto the landscape.

G. Terrain Generation

As the Ministry of Defence (1988: 31-2) notes, the factors most likely to affect travel by foot are unexpected terrain details, such as steep river banks and the type of vegetation and environment. Therefore it is important to be able to generate these two aspects of the map sufficiently.

Internally the terrain shape will be stored as a 2 dimensional array. The heights, relative to the minimum height of 0, are stored in each cell of the array. This makes the array into a height-map since the points are separated by a regular amount. The height-map can then be displayed as a 3D model when vertices are created above a grid and then joined up (see Figure 5(a)).

After the terrain is generated environments need to be applied to the map. Some environments are simple to calculate: heights under a certain value will be submerged in water and those over a certain value will be snow-capped. Other environments will be more specially generated. An area chosen to be a forest will need to make sure that it has obvious edges and borders.

There are several options when it comes to generating the actual terrain shape. Ignoring the extremely time-consuming option of creating terrains by hand the next best option is to randomise the heights of individual points based on the points around them. This ensures that a point's height is somewhat related to those around it instead of just being completely random. This method creates terrain which, while plausible, doesn't actually look overly realistic, see Figure 6(a).

While researching computer generated terrains the midpoint displacement algorithm and its improvement, the diamond-square algorithm, were found. Both of these algorithms are significant improvements over the random technique above.

The diamond-square algorithm is performed as follows, and references Figure 4:

- *The diamond step*: Taking a square of four points (a), generate a random value at the square midpoint, where the two diagonals meet (b). The midpoint value is calculated by averaging the four corner values, plus a random amount. This gives you diamonds when you have multiple squares arranged in a grid (c).
- *The square step*: Taking each diamond of four points, generate a random value at the centre of the diamond (d). Calculate the midpoint value by averaging the corner values, plus a random amount generated in the same range as used for the diamond step (e). This gives you squares again.

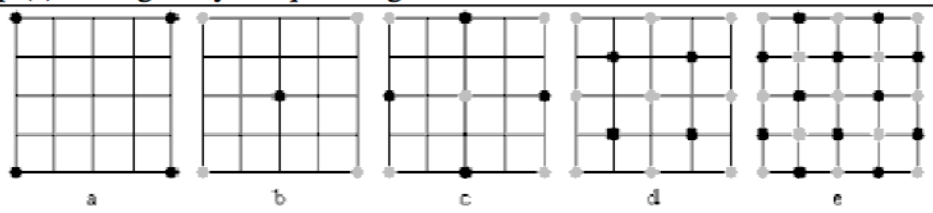

Figure 4. Diagram showing the process of using the diamond-square algorithm. Description and image taken from (Martz, 1997).

This produces height-maps which look much more like realistic terrain, see Figure 6(b). This algorithm initially starts from a completely flat land. Landscape features, such as mountains and valleys, can be forced into the algorithm so that specific landforms can be added according to the user's requirements.

The final option for creating terrain will be to import it from an outside source. Height-maps are already a staple in the field of GIS, but they are instead known by the name of Digital Elevation Models (DEMs) (Jones, 1997). DEMs exist for many real Earth locations and importing them into this project and letting users explore places they are familiar with would be very interesting and would engage the users. Ordinance Survey provides UK DEMs at a resolution of 2m on a grid size of 50m (Ordinance Survey, 2013). One drawback is that environment details are not kept in this data format, so the project program would have to artificially create the environments.

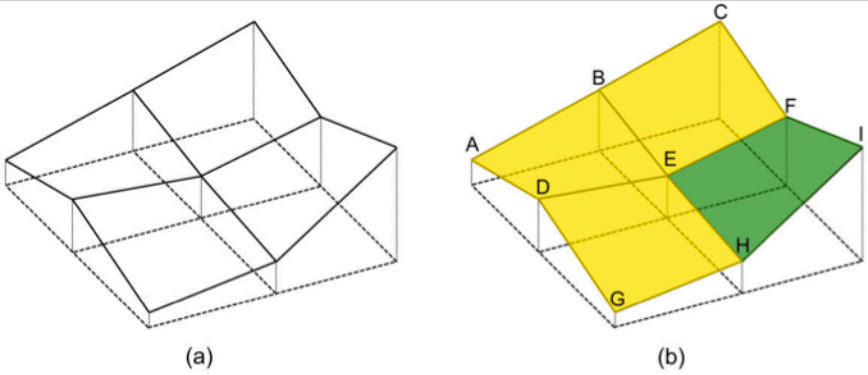

Figure 5. A wireframe of a small piece of terrain. (a) shows how terrain can be created by using a height-map on a regularly spaced grid. (b) demonstrates how environmental colours can be placed onto the terrain.

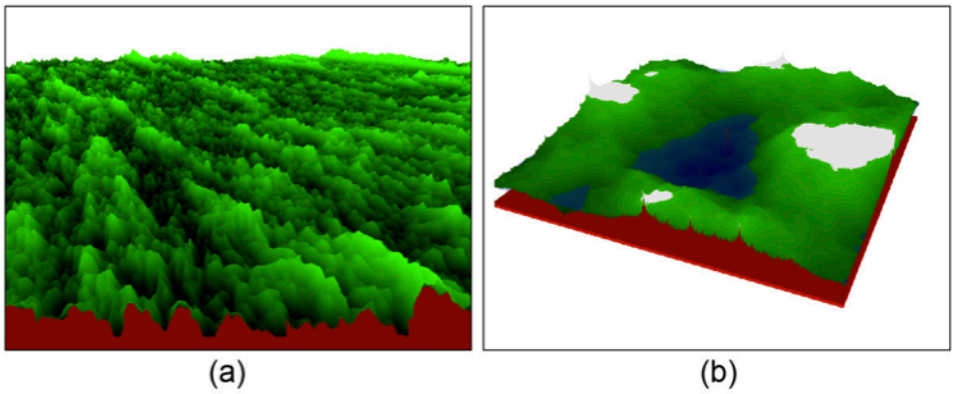

Figure 6. Generated maps using the random method (a) and the diamond-square algorithm (b).

H. Haptic Feedback

There are three main scenarios in which haptic feedback will be given back to the user when they are moving their avatar across the map and these combine to make the total haptic feedback experience.

The first effect is that the stylus moves up and down with the terrain as you move it. This is achieved simply by forcing the stylus to move in the y axis.

The second effect is related to the angle of the slope in the direction which the avatar is travelling in. If the avatar is climbing up a steep slope it should be harder to do that action. When going down a slope it should be easier. To achieve this frictional force will be applied to the stylus to make it more difficult to move.

The third aspect of the haptic feedback is from the type of environment. Different environments will behave very differently, as described in Table 3. Some of them only come

into effect when the avatar moves and some are constantly applied to the avatar, even when it is stationary.

Table 3. Different environments' haptic feedback according to the avatar's movement. This is a preliminary table and different haptics may be determined as the project progresses. NB: The y-axis is the axis which rises vertically out of the map.

Environment	Haptics while stationary	Haptics while moving
Grass	-	Normal movement friction as this is the normal easy environment type
Water	Wave motion in the y axis, probably following a sine wave	Lots of constantly changing frictional forces simulating random currents
Forest	-	Extra friction, slight y axis vibration as they break through branches
Sand	Stylus starts to lower in the y axis to indicate sinking into sand	Extra friction to indicate the difficulty of walking on sand

Travelling from one type of environment to another will require smoothing of the haptics so that it is not a jolting experience for the user as they transition from one area to another.

Using the HLAPI makes adding haptics to OpenGL elements relatively simple. A pair of HLAPI functions must be wrapped around OpenGL shape functions. The HLAPI functions take a parameter which defines the type of feedback to be given to the user when the avatar comes into contact with the shape.

I. Paths and Scoring

One of the most crucial parts of the system is having a way to score paths which are created either by the user of the application. This is essential to the testing phase, because if there isn't a suitable method for ranking paths in terms of difficulty then analysing the results would be impossible.

The score for each path will be the summation of all the scores for the individual segments of the entire path. A segment is simply a path of length 1 between two adjacent points. The lower the total path score the easier the path will be to travel along since the scores are analogous to the amount of energy expended to travel on the path.

There are a couple parts to the calculation of the score. The first is the steepness of the slope in the direction that the avatar is moving in. However, the actual angle is not needed. Since each point is the same distance apart along the X and Y axis just knowing the height difference is sufficient enough to know the difficulty in climbing or descending a sloped segment. The base score for the segment comes from this calculation.

An additional factor which could be used to score the path is the altitude which the avatar is at. This uses the idea that it is harder to travel at higher altitudes in real life, whether it be from a thinner atmosphere, lower temperature or increased wind strengths. This feature could be created by having certain thresholds which, when crossed, add a certain amount to the segment score. The thresholds will be based on the maximum height for a map. So, for instance, if the avatar is at 50% of that height then an altitude factor of 1.1 is introduced

The other factor which comes into deciding a score for a path is the type of terrain which is being moved through. This can be found by simply looking at the terrain matrix. Each type of terrain will have its own *terrain factor* which will be a number such as 1.2. This will be multiplied by the slope score. However, there is an issue when travelling along a segment on the border of two different terrain types, as shown in Figure 5(b). This arises from the fact that each point has an associated terrain type, but that terrain is displayed in a square which is offset to the point. If a user travels from E to F on the diagram then the terrain factor is the mean of the two terrain types. For example, if sand has a terrain factor of 1.8 and grass has a terrain factor of 1 then the resulting terrain factor will be 1.4.

Combining these three factors gives a score for a particular segment, as seen in Eq. (1):

$$segment\ score = height\ difference \times altitude\ factor \times terrain\ factor \quad (1)$$

J. Testing and Analysis

In order to test the conjecture given in the Introduction user tests have been devised. The tests aim to determine if haptics help or hinder calculating the difficulty of a path, which means that haptic test results should be compared against non-haptic test results. The user tests will be checked and approved by Ethics Committee of the Department of Computer Science, Durham University before the tests are conducted.

However, before the main tests are carried out pilot tests must be done to ensure that the program is calibrated correctly and that the experience is positive and not frustrating. The pilot tests will consist of asking a few users to perform a couple tasks on an early version of the software. The tasks will be similar to those in the main tests but much more feedback will be gathered from the user so that the software can be improved and refined for the final version.

The participants for the user tests will need to have a varied demographic, mainly with respect to gender and computer proficiency. Equal numbers of men and women will be sought for the tests and users will answer a pre-test questionnaire to determine their previous computer experience. It is important to ensure a wide range of abilities for the analysis of the results. Due to the tests being carried out at a university the age range for most of the participants will most likely be between 18 and 22. However, this is not seen as being of great consequence for the study.

Before the tests users will be given a haptic device tutorial program so that they can familiarise themselves with the sensations and feedback given from the device. This is a very important step as most of the participants will not have used a haptic device such as the Phantom Omni before and it will be detrimental to the study if users are focussed on trying to understand how the device functions rather than on the tests themselves. The tutorial will consist of an activity which will be unrelated to the actual tests. A possibility is for one of the demo programs supplied with the OpenHaptics SDK to be modified and given to the users for the training.

The specifics for the user tests will be determined at a later date. However, they will come in two main versions:

1) Pre-Drawn Paths Test

The users will be shown a map with several paths pre-drawn onto it. 50% of users will use visuals only to rank the paths in order of difficulty. The other 50% of users will use the stylus of the haptic device to "feel" along the paths and will rank the paths in order of difficulty. This will be repeated for different maps.

All of their rankings will be compared to the real rankings of the paths, as described in section II.I, Paths and Scoring. It is expected that each user will improve their ranking accuracy as they rank more paths. However, if the group using the haptic device improve quicker it may indicate that the haptics are helping those users to learn quicker.

2) Path Drawing Test

The users are shown points on the map which they must find the path of least effort between. This is achieved by "drawing" onto the map to create a path by using the stylus of the device. Some optimal paths will be determined before the tests.

50% of users will only use visual information to determine the paths. That is, haptic feedback will be turned off on the device. The other 50% of users will have full haptic feedback so that they may "feel" the map. One idea to increase the distinction between the two groups is to decrease the amount of visual information provided to the second group. For example, the map may be represented on the monitor simply as having flat terrain. Users will perform this process on multiple maps.

The scores for the user-drawn paths will be determined and then compared to the previously found optimal paths. The results can be analysed to see which method was better for finding more efficient paths and also which method saw the most improvement as more maps were used.

REFERENCES

Carroll, J. M., (2002). "Human-Computer Interaction in the New Millennium". s.l.:ACM Press Books.

Jones, C., (1997). "Geographical Information Systems and Computer Cartography". Singapore: Longman.

Kraak, M. J. & Ormeling, F. J., (1996). "Cartography: Visualization of Spacial Data". Singapore: Longman.

MacEachren, A. M. & Taylor, F., (1994). "Visualization in Modern Cartography". Great Yarmouth: Pergamon.

Martz, P., (1997). "Generating Random Fractal Terrain".
Available at: http://www.gameprogrammer.com/fractal.html [Accessed 24 January 2013].

Ministry of Defence, (1988). "Manual of Map Reading and Land Navigation". s.l.:HMSO.

Novint, (2013). "Novint Falcon".
Available at: http://www.novint.com/index.php/products/novintfalcon [Accessed 24 January 2013].

Ordinance Survey, (2013). "Land-Form Panorama".
Available at: http://edina.ac.uk/digimap/description/products/panorama.shtml [Accessed 24 January 2013].

Sensable, (2012). "OpenHaptics Toolkit version 3.1 Programmer's Guide". s.l.:s.n.

Sensable, (2013). "PHANTOM OMNI – Sensable".
Available at: http://www.sensable.com/haptic-phantom-omni.htm [Accessed 24 January 2013].

Srinivasan, M. A. & Basdogan, C., (1997). "Haptics in virtual environments: Taxonomy, research status, and challenges". *Computers & Graphics,* 21(4), pp. 393-404.

Tactus, (2013). "Tactus Technology Announces Technical Excellence Award from PCMag.com & New Integrated 7" Tablet Demonstration at CES 2013".
Available at: http://www.tactustechnology.com/release_130107.html [Accessed 24 January 2013].